



Firefly Suite

Firefly Host Cloud Security SDK

Release

6.0



Published: 2014-04-21

Juniper Networks, Inc.
1194 North Mathilda Avenue
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

Copyright © 2014, Juniper Networks, Inc. All rights reserved.

Juniper Networks, Junos, Steel-Belted Radius, NetScreen, and ScreenOS are registered trademarks of Juniper Networks, Inc. in the United States and other countries. The Juniper Networks Logo, the Junos logo, and JunosE are trademarks of Juniper Networks, Inc. All other trademarks, service marks, registered trademarks, or registered service marks are the property of their respective owners.

Juniper Networks assumes no responsibility for any inaccuracies in this document. Juniper Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice.

Firefly Suite Firefly Host Cloud Security SDK

Copyright © 2014, Juniper Networks, Inc.
All rights reserved.

The information in this document is current as of the date on the title page.

YEAR 2000 NOTICE

Juniper Networks hardware and software products are Year 2000 compliant. Junos OS has no known time-related limitations through the year 2038. However, the NTP application is known to have some difficulty in the year 2036.

END USER LICENSE AGREEMENT

The Juniper Networks product that is the subject of this technical documentation consists of (or is intended for use with) Juniper Networks software. Use of such software is subject to the terms and conditions of the End User License Agreement ("EULA") posted at <http://www.juniper.net/support/eula.html>. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

Table of Contents

	About the Documentation	ix
	Documentation and Release Notes	ix
	Documentation Conventions	ix
	Documentation Feedback	xi
	Requesting Technical Support	xii
	Self-Help Online Tools and Resources	xii
	Opening a Case with JTAC	xii
Part 1	Introduction	
Chapter 1	Overview	3
	About Firefly Host and Its SDK	3
	Why You Might Want to Use the Firefly Host SDK	3
	Firefly Host SDK Conventions	4
Part 2	Firefly Host APIs	
Chapter 2	User Authentication Method	7
	Authenticating the User (center.authenticateUser)	7
Chapter 3	Methods for Managing Virtual Machines	9
	Getting Information About a Virtual Machine Based on Its Host ID or Virtual Infrastructure ID (center.getMachine)	9
	Get Information About a Machine by Specifying Its VMware BIOS UUID (center.getMachineByUuid)	10
	Get Information About All Machines (center.getMachines)	11
	Add a Machine's Information to the Firefly Host Dashboard Management Center Inventory or Edit It (center.setMachine)	12
	Remove the Specified Machine from the Firefly Host Dashboard Management Center Inventory (center.setMachine)	13
Chapter 4	Methods for Securing and Unsecuring a Virtual Machine	15
	Secure a Virtual Machine (center.secureMachine)	15
	Unsecure a Virtual Machine (center.unsecureMachine)	17
	Check the Status of a Task Used to Secure or Unsecure a Virtual Machine (center.secureTaskStatus)	17
Chapter 5	Methods for Managing Policies and Policy Groups	19
	Get the Firewall Policy From the Firefly Host Dashboard Management Center (center.getPolicy)	20
	Define a Policy and Save It (center.setPolicy)	22
	Install Policies for VMs and Auto-Pushed Groups (center.instPolicy)	23

	Get Status on the Policy Install Process (center.chkPolicyInstall)	24
	Check If Policy Definitions Have Been Changed (center.chkPolicy)	24
	Remove a Policy Group (center.rmPolicyGroup)	26
Chapter 6	Methods for Managing Groups	27
	Get Group Information Based on Group ID (center.getGroup)	27
	Get Information About Built-In, User-Defined, and Policy Groups (center.getGroups)	28
	Set Information About a Group (center.setGroup)	29
	Remove a User-Defined Group from the Firefly Host Dashboard Management Center (center.rmGroup)	30
	Remove a User-Defined Group (center.rmUserDefinedGroup)	30
Chapter 7	Methods for Managing Protocols	31
	Get Protocol Information Based On Protocol Number (center.getProtocol)	31
	Get Information About All Defined Protocols (center.getProtocols)	32
	Set Protocol Information (center.setProtocol)	32
	Remove a Protocol (center.rmProtocol)	33
	Get Information About Protocol Groups (center.getProtocolGroups)	33
	Setting Protocol Group Information (center.setProtocolGroup)	34
	Remove a Protocol Group (center.rmProtocolGroup)	34
Chapter 8	Methods for Managing Compliance Rules	35
	Get All Compliance Rules (center.getComplianceRules)	35
	Get Compliance Rules State in Relation to VMs (center.getComplianceState)	36
	Add or Edit a Compliance Rule (center.setComplianceRule)	37
	Add or Edit Compliances Rules (center.setComplianceRules) [deprecated]	38
	Remove a Compliance Rule (center.rmComplianceRule)	38
Chapter 9	Methods for Managing Network Information in the Firefly Host Dashboard Management Center	39
	Get Information About All Known Networks From the Firefly Host Dashboard Management Center Inventory (center.getNetworks)	39
	Get Information from the Firefly Host Dashboard Management Center About a Specific Network (center.getNetwork)	40
	Save Information About a Network (center.setNetwork)	40
	Remove a Network from the Firefly Host Dashboard Management Center Inventory (center.rmNetwork)	41
Chapter 10	Method for Managing IDS Signatures	43
	Get IDS Signatures (center.getIdsSignatures)	43
	Set IDS Signatures (center.setIdsSignatures)	44
Chapter 11	Synchronizing VMs	47
	Resynchronize the Specified VM in the Firefly Host Dashboard Management Center with VMware vCenter (center.updateVm)	47

Chapter 12	Monitoring the Firefly Host Dashboard VM and the Firefly Host Security VM	49
	Get Status Information On the Firefly Host Dashboard VM (center.getCenterStatus)	49
	Get Status Information and Configuration Information On All Firefly Host VMs (center.getSVMStatus)	49
Chapter 13	Configuring the AutoDeploy Feature and Obtaining the Configuration	51
	Configure AutoDeploy (center.setAutoDeployConfig)	51
	Get the AutoDeploy Configuration (getAutoDeployConfig)	52
Chapter 14	Setting Configuration Parameters	55
	Set a Configuration Parameter Value (center.setConfig)	55
Chapter 15	Getting Information About Supported Smart Attributes and Alerts and Events	57
	Get a List of the System's Alerts and Events (center.listAlertsEvents)	57
	Get a List of Attributes for Smart Groups (center.listSmartAttributes)	57
Chapter 16	Getting Information About AJAX APIs And Supported Methods	59
	List AJAX APIs Accessible Through JsrServlet (listAjaxApis)	59
Chapter 17	XML-RPC Methods Information	61
	Get Information About All Supported XML-RPC Methods	61
	Get Help on an XML-RPC Method (system.methodHelp)	61
Part 3	Index	
	Index	65

List of Tables

About the Documentation	ix
Table 1: Notice Icons	x
Table 2: Text and Syntax Conventions	x

About the Documentation

- Documentation and Release Notes on page ix
- Documentation Conventions on page ix
- Documentation Feedback on page xi
- Requesting Technical Support on page xii

Documentation and Release Notes

To obtain the most current version of all Juniper Networks® technical documentation, see the product documentation page on the Juniper Networks website at <http://www.juniper.net/techpubs/>.

If the information in the latest release notes differs from the information in the documentation, follow the product Release Notes.

Juniper Networks Books publishes books by Juniper Networks engineers and subject matter experts. These books go beyond the technical documentation to explore the nuances of network architecture, deployment, and administration. The current list can be viewed at <http://www.juniper.net/books>.

Documentation Conventions

Table 1 on page x defines notice icons used in this guide.

Table 1: Notice Icons

Icon	Meaning	Description
	Informational note	Indicates important features or instructions.
	Caution	Indicates a situation that might result in loss of data or hardware damage.
	Warning	Alerts you to the risk of personal injury or death.
	Laser warning	Alerts you to the risk of personal injury from a laser.

Table 2 on page x defines the text and syntax conventions used in this guide.

Table 2: Text and Syntax Conventions

Convention	Description	Examples
Bold text like this	Represents text that you type.	To enter configuration mode, type the configure command: <code>user@host> configure</code>
Fixed-width text like this	Represents output that appears on the terminal screen.	<code>user@host> show chassis alarms</code> <code>No alarms currently active</code>
<i>Italic text like this</i>	<ul style="list-style-type: none"> Introduces or emphasizes important new terms. Identifies guide names. Identifies RFC and Internet draft titles. 	<ul style="list-style-type: none"> A policy <i>term</i> is a named structure that defines match conditions and actions. <i>Junos OS CLI User Guide</i> RFC 1997, <i>BGP Communities Attribute</i>
<i>Italic text like this</i>	Represents variables (options for which you substitute a value) in commands or configuration statements.	Configure the machine's domain name: <code>[edit]</code> <code>root@# set system domain-name <i>domain-name</i></code>
Text like this	Represents names of configuration statements, commands, files, and directories; configuration hierarchy levels; or labels on routing platform components.	<ul style="list-style-type: none"> To configure a stub area, include the stub statement at the <code>[edit protocols ospf area area-id]</code> hierarchy level. The console port is labeled CONSOLE.
< > (angle brackets)	Encloses optional keywords or variables.	<code>stub <default-metric <i>metric</i>>;</code>

Table 2: Text and Syntax Conventions (*continued*)

Convention	Description	Examples
(pipe symbol)	Indicates a choice between the mutually exclusive keywords or variables on either side of the symbol. The set of choices is often enclosed in parentheses for clarity.	broadcast multicast <i>(string1 string2 string3)</i>
# (pound sign)	Indicates a comment specified on the same line as the configuration statement to which it applies.	rsvp { # Required for dynamic MPLS only
[] (square brackets)	Encloses a variable for which you can substitute one or more values.	community name members [community-ids]
Indentation and braces ({ })	Identifies a level in the configuration hierarchy.	[edit] routing-options { static { route default { nexthop address; retain; } } }
;(semicolon)	Identifies a leaf statement at a configuration hierarchy level.	
GUI Conventions		
Bold text like this	Represents graphical user interface (GUI) items you click or select.	<ul style="list-style-type: none"> In the Logical Interfaces box, select All Interfaces. To cancel the configuration, click Cancel.
> (bold right angle bracket)	Separates levels in a hierarchy of menu selections.	In the configuration editor hierarchy, select Protocols>Ospf .

Documentation Feedback

We encourage you to provide feedback, comments, and suggestions so that we can improve the documentation. You can provide feedback by using either of the following methods:

- Online feedback rating system—On any page at the Juniper Networks Technical Documentation site at <http://www.juniper.net/techpubs/index.html>, simply click the stars to rate the content, and use the pop-up form to provide us with information about your experience. Alternately, you can use the online feedback form at <https://www.juniper.net/cgi-bin/docbugreport/>.
- E-mail—Send your comments to techpubs-comments@juniper.net. Include the document or topic name, URL or page number, and software version (if applicable).

Requesting Technical Support

Technical product support is available through the Juniper Networks Technical Assistance Center (JTAC). If you are a customer with an active J-Care or JNASC support contract, or are covered under warranty, and need post-sales technical support, you can access our tools and resources online or open a case with JTAC.

- JTAC policies—For a complete understanding of our JTAC procedures and policies, review the *JTAC User Guide* located at <http://www.juniper.net/us/en/local/pdf/resource-guides/7100059-en.pdf>.
- Product warranties—For product warranty information, visit <http://www.juniper.net/support/warranty/>.
- JTAC hours of operation—The JTAC centers have resources available 24 hours a day, 7 days a week, 365 days a year.

Self-Help Online Tools and Resources

For quick and easy problem resolution, Juniper Networks has designed an online self-service portal called the Customer Support Center (CSC) that provides you with the following features:

- Find CSC offerings: <http://www.juniper.net/customers/support/>
- Search for known bugs: <http://www2.juniper.net/kb/>
- Find product documentation: <http://www.juniper.net/techpubs/>
- Find solutions and answer questions using our Knowledge Base: <http://kb.juniper.net/>
- Download the latest versions of software and review release notes: <http://www.juniper.net/customers/csc/software/>
- Search technical bulletins for relevant hardware and software notifications: <http://kb.juniper.net/InfoCenter/>
- Join and participate in the Juniper Networks Community Forum: <http://www.juniper.net/company/communities/>
- Open a case online in the CSC Case Management tool: <http://www.juniper.net/cm/>

To verify service entitlement by product serial number, use our Serial Number Entitlement (SNE) Tool: <https://tools.juniper.net/SerialNumberEntitlementSearch/>

Opening a Case with JTAC

You can open a case with JTAC on the Web or by telephone.

- Use the Case Management tool in the CSC at <http://www.juniper.net/cm/>.
- Call 1-888-314-JTAC (1-888-314-5822 toll-free in the USA, Canada, and Mexico).

For international or direct-dial options in countries without toll-free numbers, see <http://www.juniper.net/support/requesting-support.html>.

PART 1

Introduction

- [Overview on page 3](#)

CHAPTER 1

Overview

This chapter includes the following topics:

- [About Firefly Host and Its SDK on page 3](#)
- [Why You Might Want to Use the Firefly Host SDK on page 3](#)
- [Firefly Host SDK Conventions on page 4](#)

About Firefly Host and Its SDK

Firefly Host is a fault-tolerant service provider and enterprise grade security solution that is purpose-built for the virtualized environment. Not only does it secure VMs but it also protects the hypervisor. Firefly Host delivers complete virtualization security for multitenant public and private clouds and clouds that are a hybrid of the two.

Firefly Host provides a set of application programming interfaces (APIs) that give you programmatic control over its core features after you install its components. The Firefly Host Cloud Security SDK is intended for service providers, large enterprises, universities, and other customers who create scripts or modify existing applications that control or report on the security of virtual machines (VMs). The APIs allow you to automate provisioning of security in the virtual environment.

Firefly Host uses the XML-RPC scripting language.

Related Documentation

- [Firefly Host SDK Conventions on page 4](#)
- [Why You Might Want to Use the Firefly Host SDK on page 3](#)

Why You Might Want to Use the Firefly Host SDK

The Firefly Host Cloud Security SDK provides you with a way to provision and maintain security for a VM without having to use the Firefly Host Dashboard management center. However, it is assumed that you have already performed the initial installation of the Firefly Host Dashboard and installed a Firefly Host VM on each ESX/ESXi host to be protected before you use the SDK APIs. Afterward, you can use them to instantiate security on specific VMs or to manipulate security policy on previously secured VMs.

The Firefly Host Cloud Security SDK provides support for the following types of actions:

- Retrieving information (machines, networks, policies, protocols, groups)
- Securing and unsecuring VMs
- Performing the following groups of operations:
 - Basic (setting and deleting objects via read-modify-write operations)
 - Policy (saving policies, checking policy status, and installing policies)
 - Compliance rule (setting and retrieving compliance rules)
 - Smart Groups (creating and using Smart Groups)



NOTE: If there are Firefly Host features that you want to control through the API that are not covered in this document, please contact your Juniper Networks representative to request that the functionality be added.

**Related
Documentation**

- [Firefly Host SDK Conventions on page 4](#)
- [About Firefly Host and Its SDK on page 3](#)

Firefly Host SDK Conventions

This section provides a brief description of the various types of IDs used in API methods.

The Firefly Host Cloud Security SDK uses the term *ID* in the following ways:

- *id* or *ID*—This ID usage is also known as the machine ID. The Firefly Host Dashboard management center uses the machine ID to track systems. A machine ID is assigned to all VMs, hosts, and external machines.
- *viid*—This ID is assigned by VMware vCenter to all ManagedObjects including datastores, clusters, hosts, VMs, and other objects. For example, it follows the form datacenter-1, host-123, and vm-33. The VI ID is not guaranteed to be persistent. For example, when an ESXi host is disconnected or reconnected and a vCenter is rebuilt it is assigned a new value. If *vmid* is specified, the value refers specifically to a VM, for example, vm-33, and not the other types of objects managed by VMware.
- *uuid*—This ID is the BIOS Universally Unique Identifier (*uuid*) for VMs assigned by VMware. It is used not only for VMs but also for ESX/ESXi hosts. Note that the UUID value can be duplicated.

Vcid, also referred to as the VC UUID—This ID is a unique value assigned by VMware. It is the recommended ID to use to identify VMs.

**Related
Documentation**

- [Why You Might Want to Use the Firefly Host SDK on page 3](#)
- [About Firefly Host and Its SDK on page 3](#)

PART 2

Firefly Host APIs

This part covers the Firefly Host methods. They are organized in the following categories.

- [User Authentication Method on page 7](#)
- [Methods for Managing Virtual Machines on page 9](#)
- [Methods for Securing and Unsecuring a Virtual Machine on page 15](#)
- [Methods for Managing Policies and Policy Groups on page 19](#)
- [Methods for Managing Groups on page 27](#)
- [Methods for Managing Protocols on page 31](#)
- [Methods for Managing Compliance Rules on page 35](#)
- [Methods for Managing Network Information in the Firefly Host Dashboard Management Center on page 39](#)
- [Method for Managing IDS Signatures on page 43](#)
- [Synchronizing VMs on page 47](#)
- [Monitoring the Firefly Host Dashboard VM and the Firefly Host Security VM on page 49](#)
- [Configuring the AutoDeploy Feature and Obtaining the Configuration on page 51](#)
- [Setting Configuration Parameters on page 55](#)
- [Getting Information About Supported Smart Attributes and Alerts and Events on page 57](#)
- [Getting Information About AJAX APIs And Supported Methods on page 59](#)
- [XML-RPC Methods Information on page 61](#)

CHAPTER 2

User Authentication Method

This chapter includes the following topic:

- [Authenticating the User \(center.authenticateUser\)](#) on page 7

Authenticating the User (center.authenticateUser)

center.authenticateUser

Specify the authentication username and password.

Parameters

- `userName`: (string) username to verify
- `password`: (string) password to verify

Returns

- boolean value of true if success, otherwise false.

CHAPTER 3

Methods for Managing Virtual Machines

Using the machines page of the Firefly Host Dashboard management center, you can view information about virtual machines (VMs) that are already defined. You can view the machine type and name, DNS name, IP address, a description of the machine, whether it belongs to a monitoring group, if it is secured, and other information specified by its log tag. You can also add a machine to the Firefly Host Dashboard's inventory by entering this information about the machine to be added.

The methods for managing VMs in this group allow you to use APIs to get this information about defined machines or set it to add a machine to the Firefly Host Dashboard's inventory.

You can get information for an individual machine or for all defined machines. You can use different ID types to identify the machine you are interested in. Getting a machine's information by specifying its VMware BIOS UUID returns complete information.

Just as you can remove a machine from the Firefly Host Dashboard inventory using the GUI, you can use an API in this group for the same purpose.

- [Getting Information About a Virtual Machine Based on Its Host ID or Virtual Infrastructure ID \(center.getMachine\) on page 9](#)
- [Get Information About a Machine by Specifying Its VMware BIOS UUID \(center.getMachineByUuid\) on page 10](#)
- [Get Information About All Machines \(center.getMachines\) on page 11](#)
- [Add a Machine's Information to the Firefly Host Dashboard Management Center Inventory or Edit It \(center.setMachine\) on page 12](#)
- [Remove the Specified Machine from the Firefly Host Dashboard Management Center Inventory \(center.setMachine\) on page 13](#)

Getting Information About a Virtual Machine Based on Its Host ID or Virtual Infrastructure ID (center.getMachine)

center.getMachine

Get information about the specified machine from the Firefly Host Dashboard management center's inventory using its Firefly Host ID or its VMware vCenter ID.

Parameters

- id: (int) Firefly Host Dashboard's ID for the machine

Or

- viid: (string) virtual infrastructure (VI) ID for the machine

Returns

struct representation of the machine containing:

- id: (int) machine's ID
- name: (string) name of the machine
- desc: (string) description of the machine
- type: (int) type of machine:
 - 0 - VM
 - 1 - Altor agent
 - 2 - ESX server
 - 3 - external machine
- dnsName: (string) DNS name for the machine
- dnsNameSame: (boolean) If the value is true, the name of the machine and its DNS name are the same.
- ipAddresses: (string) machine's IP addresses
- macAddresses: (string) machine's MAC addresses

Get Information About a Machine by Specifying Its VMware BIOS UUID ([center.getMachineByUuid](#))

center.getMachineByUuid

Get information about the machine whose VMware Universally Unique Identifier (vc.uuid) you specify.

Parameters

- uuid: (string) vc.uuid for the machine

Returns

- struct containing the following information about the machine:
 - id: (int) machine's ID
 - name: (string) machine's name
 - desc: (string) machine's description
 - type: (int) machine's type:

- 0 - VM
- 1 - Altor agent
- 2 - ESX server
- 3 - external machine
- `dnsName`: (string) machine's DNS name
- `dnsNameSame`: (boolean) machine's name and DNS name are the same.
- `macAddresses`: (string) machine's MAC addresses
- `ipAddresses`: (string) machine's IP addresses
- `vild`: (string) machine's VMware VI ID
- `viName`: (string) machine's VI name
- `uuid`: (string) machine's VI uuid
- `isMonitored`: (Boolean) true if the machine is monitored by an Altor agent
- `isSecured`: (Boolean) true if the machine is secured by an Altor agent
- `tags`: (string) smart tag(s).

The syntax for a smart tag is of the form `attribute=value`, such as `qowner=finance`. You can define multiple tags separated by semicolons, such as `finance;pci=true;audited=true`.

- `logTags`: (string) log tag(s). The value is added to Syslog entries relevant to this machine to assist in log filtering.
- `revision`: (string) machine's revision

Get Information About All Machines (`center.getMachines`)

`center.getMachines`

Get information about all machines in the Firefly Host Dashboard management center's inventory. If a parameter is not specified, the method returns information only about machines that were not deleted.

Parameters

- `m` – struct of optional parameters:
 - `deleted`: (boolean) whether to also show VMs that are deleted in Firefly Host Dashboard inventory. If omitted, the default is false.
 - `none`: If no parameter is specified, the default applies and information on only non-deleted machines is returned.

Returns

array of struct containing:

- id: (int) machine's ID
- name: (string) machine's name
- desc: (string) description of machine
- type: (int) machine's type:
 - 0 - VM
 - 1 - Altor Agent
 - 2 - ESX Server
 - 3 - external machine
- dnsName: (string) machine's DNS name
- dnsNameSame: (boolean) true if machine's name and DNS name are the same
- macAddresses: (string) machine's MAC addresses
- ipAddresses: (string) machine's IP addresses
- vild: (string) machine's VI ID
- viName: (string) machine's VI name
- uuid: (string) machine's VI uuid
- isMonitored: (boolean) true if the machine is monitored by an Altor agent
- isSecured: (boolean) true if the machine is secured by an Altor agent
- deleted: (boolean) true if the machine is VI deleted
- tags: (string) smart tag(s)

The syntax for a smart tag is of the form attribute=value, such as qowner=finance. Multiple tags separated by semicolons can be defined, such as finance;pci=true;audited=true
- logTags: (string) log tag(s).

This value is added to Syslog entries relevant to this machine. It is used to assist in log filtering.
- revision: (string) machine's revision.

Add a Machine's Information to the Firefly Host Dashboard Management Center Inventory or Edit It (`center.setMachine`)

center.setMachine

Set or update information about the specified machine in the Firefly Host Dashboard management center's inventory. This method allows you to manually add a VM to the Firefly Host Dashboard management center inventory.

Parameters

m - struct containing machine representation:

- id: (int) ID of the machine; if id is -1, a machine is created
- name: (string) machine name (required)
- revision: (string) machine's revision. If a revision exists and it is not equal to -1, the method validates that the revision is the latest version.
- type: (int) type of the machine:
 - 0 - VM
 - 1 - Altor agent
 - 2 - ESX/ESXi server
 - 3 - external machine
- desc: (string) description of the machine
- dnsName: (string) machine's DNS name
- dnsNameSame: (boolean) true if name and DNS name are the same
- ipAddresses: (string) machine's IP addresses
- replaceIpAddresses: (boolean) whether to replace the machine's IP addresses or add to the existing one(s)
- tags: (string) smart tag(s).

Syntax for a smart tag is of the form attribute=value, such as qowner=finance. Multiple tags separated by semicolons can be defined such as finance;pci=true;audited=true.
- logTags: (string) log tag(s). This value is added to Syslog entries relevant to this machine to assist in log filtering.

Returns

struct containing:

- id: (int) ID of the machine
- errorMsg: (string) error message

Remove the Specified Machine from the Firefly Host Dashboard Management Center Inventory (`center.setMachine`)

center.rmMachine

Remove the specified machine.

Parameters

- id: (int) ID of the machine to remove

Returns

A struct containing the operating status. If status is "Error", the related error message is also returned.

- status: (string) "Error" or "OK"
- ErrorMsg: (string) error message

CHAPTER 4

Methods for Securing and Unsecuring a Virtual Machine

After you install the main Firefly Host components—the Firefly Host Dashboard management center and the Firefly Host VM—you can begin to secure the virtual machines (VMs) in your virtualized environment. This group of APIs allows you to secure VMs on ESX/ESXi hosts that are protected by a deployed Firefly Host VM. You can use these APIs to provision security for VMs as part of your own service or system instead of relying on the Firefly Host Dashboard management center GUI.

You can use these method to secure a VM, unsecure one, and check the status of tasks that run to perform these actions.



NOTE: You can call these methods for only one VM at a time

Because securing and unsecuring an individual VM can be done using either the XML-RPC API or the Firefly Host Dashboard GUI, results can be unpredictable if you issue multiple actions on a VM at the same time. It is as if you were attempting to secure and unsecure a VM through multiple Web sessions concurrently.

This chapter includes the following topics:

- [Secure a Virtual Machine \(center.secureMachine\) on page 15](#)
- [Unsecure a Virtual Machine \(center.unsecureMachine\) on page 17](#)
- [Check the Status of a Task Used to Secure or Unsecure a Virtual Machine \(center.secureTaskStatus\) on page 17](#)

Secure a Virtual Machine (center.secureMachine)

center.secureMachine

Secure a VM that is on an ESX/ESXi host on which the Firefly Host VM is installed and protected by it. You can secure only one VM per method call. That is, you must issue a separate method call for each VM to be secured.

After you secure a VM and the API secure task completes, the VM appears in the Firefly Host Dashboard management center as secured.

The `secureMachine` method exits with an error if any of the following conditions exist:

- If the ID that you specify does not match any machine or if it matches a machine but the machine is not a VM. For example, the ID might belong to a template, an ESXi host, or an external machine.
- If the ID matches a machine that is a VM but the VM is a Firefly Host VM.
- If `SecurePervNIC` is disabled and a string value is provided for `portGroup`.

The `SecurePervNIC` option is controlled through the Settings module Install Settings > Policy Per vNIC pane, "Enable opt-out of firewalling per vNIC" option. You can not control its setting using a method.

- If a value for `portGroup` is specified, but the VM is not connected to a port group with that name.

When you attempt to secure a VM using the `secureMachine` method, the Firefly Host Dashboard management center does not check to determine if the VM is already secured. The Firefly Host Dashboard secures it again in response to the method call. The Firefly Host Dashboard does not allow you to secure a VM again that is already secured using it.

However, the API method allows it. The method call results in an error condition, but the VM itself is not affected negatively. Note that this process might entail suspending and resuming the VM.

Parameters

- `id`: (int) machine ID

ID representation for the VM used by Firefly Host Dashboard. It is not its VMID, UUID, or VCUUID. To obtain a list of machines that gives you all IDs for each machine, call the `getMachines` method.

- `portGroup`: (string) port group to secure on. An empty string if all port groups.

This is a required parameter. You can specify a port group only if `SecurePervNIC` is enabled.

Important

- If `SecurePervNIC` is disabled, an empty string is interpreted as not having set a value.
- If `SecurePervNIC` is enabled, an empty string is interpreted as a request to secure the VM on all its port groups.

Returns

struct containing:

- `status`: (string) "Error" or "OK"
- `ErrorMSG`: (string) error message

Unsecure a Virtual Machine (`center.unsecureMachine`)

center.unsecureMachine

Unsecure a VM that is secured on an ESX/ESXi host. You can remove the security for only one secured VM per method call. That is, you must issue a separate method call for each VM that you want to unsecure.

This method exits with an error if any of the following conditions exist:

- If the ID that you specify does not match any machine, or if it matches a machine but the machine is not a VM. For example, the ID might be that of a template, an ESXi host, or an external machine, in which case an error condition exists.
- If the ID matches a machine that is a VM, but the VM is a Firefly Host VM.
- If SecurePervNIC is disabled and a string value is provided for portGroup.
- If a value for portGroup is specified, but the VM is not connected to a port group with that name.

Parameters

- `id`: (int) machine ID

This ID is the Firefly Host Dashboard ID representation for the VM, not its VMID, UUID, or VCUUID.

To obtain a list of machines and their IDs, call the `getMachines` method.

- `portGroup`: (string) name of the port group that the VM is secured on.

This is a required parameter, but the port group name that you specify is valid only if the SecurePervNIC field—"Enable opt-out of firewalling per vNIC option" on the Firefly Host Dashboard—is enabled.

- If SecurePervNIC is disabled, an empty string is interpreted as not having set a value.
- If SecurePervNIC is enabled, an empty string is interpreted as a request to unsecure the VM on all its port groups.

Returns

struct containing:

- `status`: (string) "Error" or "OK"
- `ErrorMSG`: (string) error message

Check the Status of a Task Used to Secure or Unsecure a Virtual Machine (`center.secureTaskStatus`)

center.secureTaskStatus

Check the status of a task initiated through a `secureMachine` or `unsecureMachine` method call.

This method exits with an error if any of the following conditions exist:

- If the ID that you specify does not match a secure or unsecure task for a VM.

Details for a task are removed 15 minutes after the most recent status request. The counter starts when the secure or unsecure task begins, not with the status check request.

- For any state that requires user intervention through the equivalent Firefly Host Dashboard management center interface.
 - For example, if a host is disconnected the user would need to reconnect it. This condition applies when a task is cancelled. The status for the task is stored under the 15 minute counter mentioned previously. For example, if a host is disconnected the user would need to reconnect it.

Parameters

- Id: (int) ID of the machine to remove

Returns

- struct containing:
 - action: (string) "secure" or "unsecure". The action reported is the one that was most recently requested for the specified VM.
 - completed: (boolean) set to true if the task has completed
 - failed: (boolean) set to true if failed.
 - msg: (string) additional messages, if any.
 - status: (string) "Error" or "OK"
 - ErrorMSG: (string) error message



NOTE: Important: You can use the XML-RPC API to query the status for a secure or unsecure task only for tasks that were initiated by secure and unsecure method calls using XML-RPC. That is, you can not use XML-RPC to check the status for securing and unsecuring tasks initiated through the Firefly Host Dashboard VM (GUI).

CHAPTER 5

Methods for Managing Policies and Policy Groups

The Firefly Host Firewall module allows you to create policy rules to use in building policies for all VMs, groups of VMs, and individual VMs for protection against inbound and outbound traffic. The Firefly Host Firewall policy used to secure the virtualized environment is modeled on the data center infrastructure overall, and it is purpose-built to meet its requirements. Policies are reusable and tiered to allow you to construct global, group, and individual policy rules. You define a policy and save it which is separate from the process of installing the policy. This separation of definition and installation allows for policy reusability. You can create a default policy and a policy to apply to VMs that have been quarantined.

The Firefly Host SDK provides an alternative way to execute and integrate with your application the same Firewall procedures that the Firefly Host Dashboard management center provides.

The methods in this group allow you to:

- Get the current policy for a VM, a group of VMs, or vNICs of a VM. (If a VM has multiple vNICs, each can have its own unique policy). The method returns complete information that makes up the policy including its version number. For example, the information that this method returns is similar to that which you can view on the Firefly Host Dashboard Firewall module page when you select a VM in the VM tree navigation pane, and click show-all.
- Define a policy and save it, just as you would do using the Firefly Host Dashboard Firewall module Manage Policy tab. When you define a rule, you can specify its precedence among other rules in the policy.
- Use the Firewall module's Apply Policy tab to select the VMs to install the policy on and apply it to them. You can identify policies for VMs and for auto-pushed groups.
- You can check the status on the installation process of a policy.
- You can check to determine whether a policy for a VM, group of VMs, vNIC, or a node has been changed or ever applied

This chapter includes the following topics:

- [Get the Firewall Policy From the Firefly Host Dashboard Management Center \(center.getPolicy\)](#) on page 20
- [Define a Policy and Save It \(center.setPolicy\)](#) on page 22
- [Install Policies for VMs and Auto-Pushed Groups \(center.instPolicy\)](#) on page 23
- [Get Status on the Policy Install Process \(center.chkPolicyInstall\)](#) on page 24
- [Check If Policy Definitions Have Been Changed \(center.chkPolicy\)](#) on page 24
- [Remove a Policy Group \(center.rmPolicyGroup\)](#) on page 26

Get the Firewall Policy From the Firefly Host Dashboard Management Center (center.getPolicy)

center.getPolicy

Retrieve a firewall policy from the Firefly Host Dashboard inventory based on the policy ID, the policy ID number, the group ID, the machine ID, or a virtual NIC (vNIC) .

Parameters

- id: (int) policy ID
- Or
- id: (int) ID of the policy owner, whether the policy belongs to a machine or to a group
- isVm: (boolean):
 - true if the policy belongs to a VM
 - false if the policy belongs to a group
- Or
- struct containing the following parameters:
 - groupId: (int) ID of the group policy to retrieve
 - machineId: (int) ID of the VM or the vNIC whose policy you want to retrieve
 - nicNum: (int) ID of the vNIC whose policy you want to retrieve

Returns

struct containing:

- name: (string) policy name
- id: (int) ID of the policy
- rev: (string) policy revision
- revision: DB revision of the policy
- type: policy type
 - G: global

- R: group
- V: VM
- groupId: (int) group ID associated with this policy (-1 if none)
- machineId: (int) VM ID associated with this policy (-1 if none)
- grpPols: array of integers containing the VM's policy groups, including the global policy, provided only for VM policies.
- vmPols: (int) ID of the VM policy if this is a vNIC policy
- nicPols: (array of int) IDs of vNIC policies if this is a VM policy
- Inbound: struct representation of the inbound policy containing:
 - pre: struct representing the policy rules applied before the next level of policy rules is applied. All rules in a VM policy are considered pre (first).
 - rules array containing struct representation of rules:
 - id: (int) ID of the rule
 - srcsDsts: (string) one of the following:
 - "any"
 - encoding of the source or destinations, for example: "g|1,2|n|2,4|m|4"
 - protocols: (string) one of the following:
 - "any"
 - encoding of the protocols, for example, "g|1,2|p|2,4"
 - action: (string) rule's action:
 - A: allow
 - R: reject
 - D: drop
 - C: mirror
 - targetId: (int) target used for mirroring
 - logAction: (string) rule's log action:
 - -: no log
 - L: log
 - !: alert

- mailAlertEntId: (int) mail alert target (is missing or -1 for none)
- desc: (string) description of the rule
- post: struct representing the policy rules applied after the next level policy rules
- outbound: struct representing the outbound policy (similar to the struct for the inbound policy)

Define a Policy and Save It (`center.setPolicy`)

center.setPolicy

Define a new policy and save it. This method allows you to create and save a new policy but not install it. To install a policy, call the `instPolicy` method. (See “[Install Policies for VMs and Auto-Pushed Groups \(center.instPolicy\)](#)” on page 23.)

If the policy map that you define contains the revision keyword, this method first checks to ensure that the existing policy object is the latest version before it modifies the policy.

Parameters

struct containing

- id: (int) ID of the policy
- revision: (string) database revision of the policy. If the policy exists and its version is not equal to -1, the method validates that the revision is the latest one.
- inbound: struct representation of the inbound policy containing the following information.
- pre: struct representing the policy rules that are applied before the next level of rules. All VM policy rules are treated as “pre” rules. They are the most specific rules in a policy.



NOTE: A policy includes at least global rules but it can also include group rules and rules for an individual VM.

- rules: array containing struct representation of the policy rules. For each rule:
 - id: (int) ID of the rule. If the policy rule exists and you are modifying it, this is its ID. For a new rule specify -1 for this parameter or leave out the parameter.
 - srcsDsts: (string) *any* or encoding of the source or destination, for example, "g|1,2|n|2,4|m|4"
 - protocols: (string) *any* or encoding of the protocols, for example, "g|1,2|p|2,4" *
 - action: (string) the rule's action:
 - A: allow
 - R: reject

- D: drop
- C: mirror
- targetId: (int) target used for mirroring
 - target IDs 1 to-4 are defined in Settings -> Global
 - target ID 5 is used for internal IDS inspection. Target ID -1 - when target is not in use
 - target ID -1 indicates that the target is not in use
- logAction: (string) specifying the log action of the rule:
 - - : no log
 - L: log
 - ! : alert
- mailAlertEntId: (int) mail alert target (missing or -1 for none)
- desc: (string) description of the rule
- post: struct similar to the pre struct representing the policy rules applied after the next level rules that follow the “pre” rules.



NOTE: Do not use this element in a VM policy.

- Outbound: struct similar to the Inbound struct representation of the inbound policy

Returns

- If the method is successful, an array of int containing the IDs of the rules and the revision number of the new policy
- If the method fails, throws a runtime exception (RuntimeException)

Install Policies for VMs and Auto-Pushed Groups (center.instPolicy)

center.instPolicy

Install the policy for the specified VMs and auto-pushed groups.

Parameters

- array of type string containing machine IDs or group IDs
 - VM IDs: ID of VMs for which to install policies prefaced with “m”
 - Group IDs: IDs of auto-push groups for which to install policies prefaced with “g”

Returns

Returns a string specifying the key for checking the status (to use with the `chkPolicyInstall` method).

Get Status on the Policy Install Process (`center.chkPolicyInstall`)

center.chkPolicyInstall

Poll the install process status.

Parameters

- (string) process key received from `instPolicy` method. See [“Install Policies for VMs and Auto-Pushed Groups \(`center.instPolicy`\)” on page 23](#)

Returns

struct representation of the policy install status containing the following information.

- status: (string):
 - PENDING
 - RUNNING
 - COMPLETED
 - SKIPPED
 - FAILED
 - SUSPENDED
- stepCount: (int) total number of steps in the install process
- processed: (int) number of steps processed
- running: (array of string) VM for which policies are currently being installed.
- successful: (string) VMs for which the policy install process completed successfully
- migrated: (string) VMs migrated during the install process
- errors: (string) errors encountered during the install process
- warnings: (string) warnings encountered during the install process

Check If Policy Definitions Have Been Changed (`center.chkPolicy`)

center.chkPolicy

Parameters

Check whether the policies of a VM or VMs belonging to a group have been changed.

- id - (int) ID of the VM or group
- isVM - (boolean) true if the ID belongs to a VM

OR

Checks the policy install status of a given node for changes. Struct containing:

- groupId: (int) group ID whose policy to check for changes
- machineId: (int) ID of a machine to check its VM or vNIC policies for changes
- nicNum: (int) ID of a vNIC whose policy to check for changes

Returns

For the check on policies for VMs and groups:

array of struct containing the policy status for each VM

- id: (int) VM ID
- groupId: (int) group ID (valid when id is -1, the status if for a dynamic group)
- nicNum: (int) vNIC number if this policy is for a vNIC, otherwise -1
- name: (string) VM name
- installDate: (string) date of last policy installation
- globalChanged: (string) global policy status
 - unchanged
 - changed
 - never installed
 - none
- groupChanged: (string) group policy status
 - unchanged
 - changed
 - never installed
 - none
- vmChanged: (string) VM policy status
 - unchanged
 - changed
 - never installed
 - none
- vnicChanged: (string) vNIC policy status
 - unchanged
 - changed
 - never installed
 - none

If the policy of a specific node is checked for changes, the method returns the policy status for all machines and vNICs in the node.

Remove a Policy Group (`center.rmPolicyGroup`)

`center.rmPolicyGroup`

Parameters

- `id`: (int) ID of the policy group to remove

Returns

- struct containing operation status and error message:
 - `status`: (string) "Error" or "OK"
 - `ErrorMSG`: (string) error message

CHAPTER 6

Methods for Managing Groups

This chapter includes the following topics:

- [Get Group Information Based on Group ID \(center.getGroup\)](#) on page 27
- [Get Information About Built-In, User-Defined, and Policy Groups \(center.getGroups\)](#) on page 28
- [Set Information About a Group \(center.setGroup\)](#) on page 29
- [Remove a User-Defined Group from the Firefly Host Dashboard Management Center \(center.rmGroup\)](#) on page 30
- [Remove a User-Defined Group \(center.rmUserDefinedGroup\)](#) on page 30

Get Group Information Based on Group ID (center.getGroup)

center.getGroup

Query the Firefly Host Dashboard management center's inventory for a single group given the group ID.

Parameters

- id: (int) ID of the group

Returns

- struct containing
 - id: (int) group ID
 - name: (string) group name
 - type: (string) group type:
 - B: built-in group
 - U: user-defined group
 - P: policy group
 - A: application group
 - groups: array of int containing sub-group IDs
 - machines: array of int containing the IDs of the machines in this group

- `networkIds`: Array of int containing the IDs of the networks in this group
- `order`: number showing precedence within the priority level of the group
- `num`: (int) order in the tree of the group in case its parent has more than one group child
- `policy`: (true or false) whether the group has the policy attribute
- `push`: (true or false) whether the group has the push attribute
- `smart`: (true or false) whether the group has the smart attribute
- `smartDef`: string representation of the smart group rules
- `machineVnics`: map from ints (machineId) to list of ints (Numbers of NICs in the group)

Get Information About Built-In, User-Defined, and Policy Groups (`center.getGroups`)

center.getGroups

Query the Firefly Host Dashboard management center's inventory for built-in, user-defined, and policy groups.

Parameters

- none

Returns

array of structs representing groups containing:

- `id`: (int) group ID
- `name`: (string) group name
- `type`: (string) group type
 - B: built-in group
 - U: user-defined group
 - E: partially editable groups created through the compliance module for which you can update only the group policy attributes
 - A: application group
- `groups`: array of type int containing subgroup IDs
- `machines`: array of type int containing the IDs of the machines in the group
- `machineVnics`: a map from the machine ID to a list of vNIC numbers for members of a vNIC-level group
- `networkIds`: array of type int containing the IDs of the networks that belong to the group
- `order`: the precedence of the policy within the priority level
- `num`: (int) number assigned to the rule on creation

- policy: (true or false) whether the group has the policy attribute
- priority: priority level of the policy (h=high, m=medium, l=low, =NA)
- push: (true or false) whether the group has the push attribute
- smart: (true or false) whether the group has the smart attribute
- smartDef: string representation of the smart group rules

Set Information About a Group (`center.setGroup`)

`center.setGroup`

Save a group information in the Firefly Host Dashboard management center's inventory.

Parameters

map representation of the group to save

- id: (int) group ID. If it is missing or is -1, a new protocol group will be created (not mandatory)
- revision: If a revision exists and it is not equal to -1, the method validates that it is the latest one.
- policy:
 - true
 - false (default)
- push
 - true
 - false (default)
- smart:
 - true
 - false (default)
- smartDef: (string) representation of the smart group rules. Default: empty string.
- priorityL: priority level of the group (h=high, m=medium, l=low or -=NA)
- order: (int) precedence within the priority level of the group



NOTE: If the group has a policy attribute and the order is not specified, the next available one is assumed. If the group does not have a policy attribute order will be ignored.

Returns

struct representation containing:

- id: (int) group's ID
- ErrorMsg: (string) error message

Remove a User-Defined Group from the Firefly Host Dashboard Management Center (center.rmGroup)

center.rmGroup

Remove a user-defined group from the Firefly Host Dashboard's inventory.

Parameters

- id - ID of the group to remove

Returns

struct representation containing:

- id: (int) group's ID
- ErrorMsg: (string) error message

Remove a User-Defined Group (center.rmUserDefinedGroup)

center.rmUserDefinedGroup

Remove a user-defined group from the Firefly Host Dashboard management center's inventory

Parameters

WARNING: This method is deprecated. Use center.rmGroup instead.

- id: (int) ID of the user defined group to remove.

Returns

struct containing:

- id: (int) machine's ID
- status: (string) "Error" or "OK"
- ErrorMsg: (string) error message
- notResponding: (string) VMs not responding

CHAPTER 7

Methods for Managing Protocols

This chapter includes the following topics:

- [Get Protocol Information Based On Protocol Number \(center.getProtocol \) on page 31](#)
- [Get Information About All Defined Protocols \(center.getProtocols\) on page 32](#)
- [Set Protocol Information \(center.setProtocol\) on page 32](#)
- [Remove a Protocol \(center.rmProtocol\) on page 33](#)
- [Get Information About Protocol Groups \(center.getProtocolGroups\) on page 33](#)
- [Setting Protocol Group Information \(center.setProtocolGroup\) on page 34](#)
- [Remove a Protocol Group \(center.rmProtocolGroup\) on page 34](#)

Get Protocol Information Based On Protocol Number (center.getProtocol)

center.getProtocol

Query the Firefly Host Dashboard management center's inventory for a specific protocol given the protocol ID.

Parameters

- id: (int) protocol's ID

Returns

struct representation of the protocol containing:

- id: (int) protocol's ID
- ipProto: (string) IP protocol:
 - tcp
 - udp
 - other
- port: (int) port number
- name: (string) protocol name, for example "telnet"
- name1: (string) protocol name combining port and IP protocol, for example "telnet (23/tcp)"

- alert: (boolean): alert, if seen
- maxPort: (int) max port number of the range, for protocol ranges only (range is *port* to *maxPort*)
- revision: (string) protocol revision

Get Information About All Defined Protocols (`center.getProtocols`)

center.getProtocols

Query the Firefly Host Dashboard management center's inventory for all defined protocols.

Parameters

- none

Returns

array containing:

- struct representation of a protocol containing:
 - ipProto: (string) IP protocol:
 - tcp
 - udp
 - other
 - port: (int) port number
 - name: (string) protocol name, for example "telnet"
 - name1: (string) protocol name combining port and IP protocol, for example "telnet (23/tcp)"
 - alert: (boolean) alert, if seen
 - number: (int) a protocol ID calculated as a method of port number and IP protocol
 - maxPort: (int) for protocol ranges only, the max port number of the range (range is *port* to *maxPort*)
 - revision: (string) protocol revision

Set Protocol Information (`center.setProtocol`)

center.setProtocol

Save protocol information in the Firefly Host Dashboard management center's inventory. If the map contains a revision key, the method validates that the revision is the latest version before updating the information.

Parameters

protocolInfo: struct containing protocol representation, similar to the struct in `getProtocols` method but with the following differences:

- id: (int) If protocol id is missing or -1, a new protocol is created.
- name1 and number: not needed, are ignored
- revision: (string) If revision exists and is not equal to -1, validates that information is latest.
- port: (string) for a single port, the port number; for a range, *minPort* to *maxPort*
- alert: (boolean) alert, if seen

Returns

struct containing:

- id: (int) new protocol's ID
- ErrorMSG: (string) error message

Remove a Protocol (`center.rmProtocol`)

center.rmProtocol

Remove a protocol from the Firefly Host Dashboard management center's inventory.

Parameters

id: (int) ID of the protocol to remove

Returns

struct containing:

- status: (string) "Error" or "OK"
- ErrorMSG: (string) error message

Get Information About Protocol Groups (`center.getProtocolGroups`)

center.getProtocolGroups

Query the Firefly Host Dashboard management center's inventory for all defined protocol groups.

Parameters

- none

Returns

array containing:

- struct representation of a protocol group containing:

- id: (int) ID of the protocol group
- name: (string) name of the protocol group
- protocolIds: array of int containing the IDs for the protocols in the group
- revision: (string) revision of the protocol group

Setting Protocol Group Information (`center.setProtocolGroup`)

center.setProtocolGroup

Saves the provided protocol group information in the Firefly Host Dashboard management center's inventory. If the protocol group map contains revision key, the method validates that the revision is the latest version before updating the information.

Parameters

- protocolGroup: struct containing protocol group representation similar to the struct in `getProtocolGroups` with the following differences:
 - If id is missing or is -1, a new protocol group is created.
 - If revision exists and it isn't equal to -1, the method validates that it is the latest (string)

Returns

struct containing:

- id: (int) ID for the new protocol group
- ErrorMSG: (string) error message

Remove a Protocol Group (`center.rmProtocolGroup`)

center.rmProtocolGroup

Remove a protocol group from the Firefly Host Dashboard management center's inventory.

Parameters

- id: (int) ID of the protocol group to remove.
- lower case letter –desc

Returns

struct containing:

- status: (string) "Error" or "OK"
- ErrorMSG: (string) error message

CHAPTER 8

Methods for Managing Compliance Rules

This chapter includes the following topics:

- [Get All Compliance Rules \(center.getComplianceRules\)](#) on page 35
- [Get Compliance Rules State in Relation to VMs \(center.getComplianceState\)](#) on page 36
- [Add or Edit a Compliance Rule \(center.setComplianceRule\)](#) on page 37
- [Add or Edit Compliance Rules \(center.setComplianceRules\)](#) [deprecated] on page 38
- [Remove a Compliance Rule \(center.rmComplianceRule\)](#) on page 38

Get All Compliance Rules (center.getComplianceRules)

center.getComplianceRules

Query the Firefly Host Dashboard management center's inventory for all defined Compliance rules.

Parameters

none

Returns

an array containing:

- struct containing:
 - id: (int) rule ID
 - name: rule name
 - smartDef: smart definition of the rule
 - comment: rule description
 - weight: (int 1-5) rule weight
 - createCompGroup: (true or false) whether to create a group for the rule-compliant groups
 - createNonCompGroup: (true or false) whether to create a group for the rule-noncompliant groups
 - disabled: (true or false) whether the rule should be disabled
 - scope: name of the group that is the scope of this rule
 - alert: (true or false) whether to send an alert when a VM status changes from comply to non-comply (or vice versa) according to this rule
 - groupings: (string) comma-separated string of the rule labels
 - revision: (string) rule revision

Get Compliance Rules State in Relation to VMs (`center.getComplianceState`)

center.getComplianceState

Query the Firefly Host Dashboard management center to determine which compliance rules are applied to a VM and if they are being violated by the VM.

Parameters

- id: (int) the Firefly Host Dashboard management center's ID for the VM.

OR

viid: (string) the VI ID for the VM

Returns

array of struct containing:

- id: (int) rule ID
- smartDef: the smart definition of the rule
- comment: rule description
- weight: (int 1-5) rule weight
- createCompGroup: (true or false) whether to create a group for the rule compliant groups

- `createNonCompGroup`: (true or false) whether to create a group for the rule non-compliant groups
- `disabled`: (true or false) whether the rule should be disabled
- `scope`: name of the group which is the scope of this rule
- `alert`: (true or false) whether to send alert when a VM status changes from comply to non-comply (or vice versa) according to this rule
- `groupings`: (string) comma separated of the rule labels
- `violating`: {true or false} whether or not the VM is violating

Add or Edit a Compliance Rule (`center.setComplianceRule`)

center.setComplianceRule

Save a compliance rule in the Firefly Host Dashboard management center's inventory

Parameters

- a struct containing compliance rule parameters:
 - `id`: (int) ID of existing compliance rule to modify; -1 for a new one [mandatory]
 - `name`: (string) name for the compliance rule [mandatory]
 - `revision`: (string) revision of the rule. If a revision exists and it is not equal to -1, the method validates that it is the latest version.
 - `comment`: (string) comment for the compliance rule [mandatory]
 - `remediation`: (string) description of remediation steps
 - `groupings`: (string) groupings for the compliance rule, comma separated [mandatory]
 - `smartDef`: (string) smart definition of the compliance rule [mandatory]
 - `scope`: (int) ID of the scope group of the compliance rule; default=1 (all machines) [mandatory]
 - `weight`: (int) weight of compliance rule; default = 3 [mandatory]
 - `alert`: (boolean) whether to alert when VM compliance state changes:
 - default = false
 - true = if it appears
 - `disabled`: (boolean) whether the rule is disabled. disabled if it appears
 - `createCompGroup`: (boolean) whether to create a group for compliant VMs
 - default = false
 - true = if it appears
 - `createNonCompGroup`: (boolean) whether to create group for non-compliant VMs
 - default = false

- true = if it appears
- includeVnic: (boolean) whether this compliance rule is tested at the vNIC level

Returns

A struct representation of the machine containing:

- id: (int) machine's ID
- otherwise
- status: (string) "Error" or "OK"
- ErrorMsg: (string) error message

Add or Edit Compliances Rules (`center.setComplianceRules`) [deprecated]

center.setComplianceRules

WARNING: This method is deprecated. Use `setComplianceRule` instead.

Save compliance rules (add or edit).

Parameters

Same as `setComplianceRule`.

Returns

See `setComplianceRule`.

Remove a Compliance Rule (`center.rmComplianceRule`)

center.rmComplianceRule

Remove a compliance rule from the Firefly Host Dashboard management center's inventory.

Parameters

- id: (int) ID of the compliance rule to remove

Returns

struct containing:

- status: (string) "Error" or "OK"
- ErrorMsg: (string) error message

CHAPTER 9

Methods for Managing Network Information in the Firefly Host Dashboard Management Center

This chapter covers the following methods:

- [Get Information About All Known Networks From the Firefly Host Dashboard Management Center Inventory \(center.getNetworks\) on page 39](#)
- [Get Information from the Firefly Host Dashboard Management Center About a Specific Network \(center.getNetwork\) on page 40](#)
- [Save Information About a Network \(center.setNetwork\) on page 40](#)
- [Remove a Network from the Firefly Host Dashboard Management Center Inventory \(center.rmNetwork\) on page 41](#)

Get Information About All Known Networks From the Firefly Host Dashboard Management Center Inventory (center.getNetworks)

center.getNetworks

Get information about all of the networks in the Firefly Host Dashboard management center's inventory.

Parameters

- none

Returns

struct representation of a network containing:

- id: (int) ID for the network
- name: (string) name of the network
- desc: (string) description of the network
- type: (string) type of network
 - R: IP Range
 - S: Subnet Mask

- networkFirst: (string) first address in a range or a network address
- maskLast: (string) last address in a range or a sub-net mask
- revision: (string) revision of the network

Get Information from the Firefly Host Dashboard Management Center About a Specific Network (`center.getNetwork`)

center.getNetwork

Query the Firefly Host Dashboard management center's inventory for a specific network based on the network ID.

Parameters

- id: (int) Firefly Host Dashboard ID for the network

Returns

A struct representation of the network containing:

- id: (int) network's ID
- name: (string) network name
- desc: (string) description of the network
- type: (string) network type
 - R: IP range
 - S: subnet mask
- networkFirst: (string) first address in a range or a network address
- maskLast: (string) last address in a range or a subnet mask

Save Information About a Network (`center.setNetwork`)

center.setNetwork

Save the identified network's information in the Firefly Host Dashboard management center's inventory.

Parameters

struct containing network representation similar to the information in the `getNetworks` method's returned struct but with the following differences:

- If id is missing or is -1, a new network is created.
- If a revision exists and it is not equal to -1, the method validates that it is the latest version (string).

Returns

- struct containing:
 - id: (int) ID of the new network
 - ErrorMSG: (string) error message

Remove a Network from the Firefly Host Dashboard Management Center Inventory (center.rmNetwork)

center.rmNetwork

Remove a network from the Firefly Host Dashboard management center's inventory.

Parameters

- id: (int) ID of the network to remove

Returns

struct containing:

- status: (string) "Error" or "OK"
- ErrorMSG: (string) error message

CHAPTER 10

Method for Managing IDS Signatures

This chapter covers the following methods:

- [Get IDS Signatures \(center.getIdsSignatures\)](#) on page 43
- [Set IDS Signatures \(center.setIdsSignatures\)](#) on page 44

Get IDS Signatures (center.getIdsSignatures)

center.getIdsSignatures

Query the Firefly Host Dashboard management center for the intrusion detection system (IDS) signatures.

Parameters

- None

Returns

A struct representation of the signature containing:

- sources: array containing struct representations of signature sources:
 - id: (int) internal ID for the source
 - name: (string) name of source
 - lastUpdate: (string) last date of changes made to signatures from this source
 - rev: (string) version of signatures package (*may be empty*)
- sets: an array of struct representations of signature sets containing:
 - id: (int) internal ID of the signature set
 - name: (string) name of the set
 - enabled: (string) Is the set active:
 - T: true
 - F: false
 - D: use the default setting from source
 - enabledByDefault: (string): default, or recommended, setting for this set:

- T: true
 - F: false
 - srcId: (int) ID of the set's source
 - lastUpdate: (string) last date changes were made to signatures of this set
 - signatures: an array of struct representations of signatures containing:
 - sid: (int) signature ID (SID)
 - rev: (int) signature revision
 - setId: (int) ID of the signature's set
 - enabled: (string) Is the set active:
 - T: true
 - F: false
 - D: use the default setting from source
- enabledByDefault: (string):
- T: true
 - F: false
 - lastUpdate: (string) last date of changes to the signatures
 - msg: (string) signature's alert message
 - signature: (string) raw signature
 - revision: (string) revision of the IDS signatures

Set IDS Signatures (`center.setIdsSignatures`)

center.setIdsSignatures

Update the enabled flag for IDS rules. The enabled flag can be set to true (T) or false (F). (The default is false.)

Parameters

struct containing:

- signatures: (struct) map of signature IDs (sids) to use to enable flags
- revision: (string) revision of the IDS signatures. If revision exists and it is not equal to -1, the method validates that the revision is the latest version.

Returns

struct containing:

- rulesUpdated: (int) count of signatures whose enabled flags were changed

- ErrorMessage: (string) error message

CHAPTER 11

Synchronizing VMs

This chapter covers the following function:

- Resynchronize the Specified VM in the Firefly Host Dashboard Management Center with VMware vCenter (`center.updateVm`) on page 47

Resynchronize the Specified VM in the Firefly Host Dashboard Management Center with VMware vCenter (`center.updateVm`)

center.updateVm

Resync the specified VM in the Firefly Host Dashboard management center inventory with VMware vCenter.

Parameters

- `vild` - ID of the VM to resync

Returns

struct containing:

- `status`: (string) "Error" or "OK"
- `ErrorMSG`: (string) error message

CHAPTER 12

Monitoring the Firefly Host Dashboard VM and the Firefly Host Security VM

This chapter covers the following methods:

- Get Status Information On the Firefly Host Dashboard VM (`center.getCenterStatus`) on page 49
- Get Status Information and Configuration Information On All Firefly Host VMs (`center.getSVMStatus`) on page 49

Get Status Information On the Firefly Host Dashboard VM (`center.getCenterStatus`)

center.getCenterStatus

Get status on the Firefly Host Dashboard VM. This method returns information similar to that made available in the Main -> Status window.

Parameters

- `reCalc`: (boolean) whether the center should recalculate its status

Returns

array containing:

- `desc`: short description of entry (high availability, Firefly Host Dashboard management center, Virtual Center, Multi Center)
- `status`: (string) "Error" / "OK" / "Warning"
- `details`: (string) details on possible issues or "Not Applicable"

Get Status Information and Configuration Information On All Firefly Host VMs (`center.getSVMStatus`)

center.getSVMStatus

Get status and configuration information on the Firefly Host VMs.

Parameters

- none

Returns

array of struct representing status on Firefly Host VMs

- id: (int) ID of the Firefly Host VM
- name: (string) name of the Firefly Host VM
- mgmtAddr: (string) management address of the Firefly Host VM
- esxHost: (string) VMware ESX/ESXi host name
- status: (string) status description for the Firefly Host VM
- fastpathStatus: (string) status code for the fast-path module:
 - OK
 - WRONG VERSION
 - NOT LOADED
 - GENERAL ERROR
 - MISCONFIGURED
- isStandby: (boolean) is this Firefly Host VM the standby one
- standby: (int) the SVM standby's ID or (string) "none" if none is configured
- monitorHypervisorConsole: (String):
 - OFF
 - MONITOR_ONLY
 - MONITOR_AND_IDS
- trafficMonitor: (boolean) whether the Firefly Host VM is actively monitoring
- version: (string) version of the Firefly Host VM
- vms: (string) comma-separated list of names of VMs protected by the

CHAPTER 13

Configuring the AutoDeploy Feature and Obtaining the Configuration

This chapter covers the following methods:

- [Configure AutoDeploy \(center.setAutoDeployConfig\)](#) on page 51
- [Get the AutoDeploy Configuration \(getAutoDeployConfig\)](#) on page 52

Configure AutoDeploy (center.setAutoDeployConfig)

center.setAutoDeployConfig

Configure the AutoDeploy feature. This method allows you to do through an API what you can do using the Settings > Install Settings > Automatic Securing of Auto-Deployed Hosts.

Parameters

struct containing:

- selection: (string) AutoDeploy state. Whether it is:
 - none—disabled
 - all—enabled for all hosts
 - specific—enabled for one or more clusters
- clusterList: (string) comma-delimited list of cluster names for which AutoDeploy is configured. Applies only to "specific". Otherwise, has no effect.
- portGroup: (string) name of the port group on which installed Firefly Host VMs will be installed
- namePrefix: (string) Firefly Host VMs installed through AutoDeploy are named using this prefix + the last IP octet of the host
- datastore: (string) name of the datastore where Firefly Host VM installed through AutoDeploy will be stored
- ipMethod: (string) method for configuring Firefly Host VMs:
 - static—static IPs

- dhcp— DHCP
- ipPart1: (string) If the IP configuration is set to static, this is the first octet of the Firefly Host VM's IP address.
- ipPart2: (string) If the IP configuration is set to static, this is the second octet of the Firefly Host VM's IP address.
- ipPart3: (string) If the IP configuration is set to static, this is the third octet of the Firefly Host VM's IP address.
- gateway: (string) If the IP configuration is set to static, this is the configured gateway.
- netmask: (string) If the IP configuration is set to static, this is the configured netmask.
- forceRecheck: (string) If non-empty, forces a recheck on all hosts to see if they fit AutoDeploy configuration and therefore require a Firefly Host VM. This check also runs if any configuration was changed.

Returns

struct containing operation status and error message:

- status: (string) "OK" if completed without issue, "Error" if an error occurred
- changed: (string):
 - true—if configuration was changed
 - false—otherwise
- errorMsg: (string) error message(s), if applicable

Get the AutoDeploy Configuration (*getAutoDeployConfig*)

getAutoDeployConfig

Get the AutoDeploy Configuration from the Firefly Host Dashboard management center.

Parameters

none

Returns

struct containing:

- selection: (string) AutoDeploy state. Whether it is:
 - none—disabled
 - all—enabled for all hosts
 - specific—enabled for one or more clusters
- clusterList: (string) comma-delimited list of cluster names for which AutoDeploy is configured. If selection is not "specific", this value has no effect.

- portGroup: (string) name of the port group on which installed Firefly Host VMs will be installed
- namePrefix: (string) Firefly Host VM installed through AutoDeploy are named using this prefix + the last IP octet of the host
- datastore: (string) name of the datastore where Firefly Host VM installed through AutoDeploy will be stored
- ipMethod: (string) method for configuring Firefly Host VMs:
 - static—static IPs
 - dhcp—DHCP
- ipPart1: (string) If the IP configuration is set to static, this is the first octet of the Firefly Host VM's IP address.
- ipPart2: (string) If the IP configuration is set to static, this is the second octet of the Firefly Host VM's IP address.
- ipPart3: (string) If the IP configuration is set to static, this is the third octet of the Firefly Host VM's IP address.
- gateway: (string) If the IP configuration is set to static, this is the configured gateway.
- netmask: (string) If the IP configuration is set to static, this is the configured netmask.

CHAPTER 14

Setting Configuration Parameters

This chapter covers the following method:

- [Set a Configuration Parameter Value \(center.setConfig\)](#) on page 55

Set a Configuration Parameter Value (center.setConfig)

center.setConfig

Parameters

Set a configuration parameter.

- key: (string) key for the configuration parameter
- val: (string) value to assign to the configuration parameter

Returns

- (boolean): true, if successful

CHAPTER 15

Getting Information About Supported Smart Attributes and Alerts and Events

This chapter covers the following methods:

- [Get a List of the System's Alerts and Events \(center.listAlertsEvents\)](#) on page 57
- [Get a List of Attributes for Smart Groups \(center.listSmartAttributes\)](#) on page 57

Get a List of the System's Alerts and Events (center.listAlertsEvents)

center.listAlertsEvents

Parameters

- none

Returns

array of string:

- module name: (string) name of attribute
- subModuleName: (string) data type of attribute
- priority: (string) description of the attribute
- alert text
- alert meaning
- suggested action
- whether the alert is associated with an administrator

Get a List of Attributes for Smart Groups (center.listSmartAttributes)

center.listSmartAttributes

Get a list of attributes that can be used to define smart groups.

Parameters

- none

Returns

array of:

- name: (string) name of attribute
- data type: (string) data type of attribute
- description: (string) description of the attribute

CHAPTER 16

Getting Information About AJAX APIs And Supported Methods

This chapter covers the following introspection methods provided by the XML_RPC framework. They are not part of the Firefly Host SDK, but rather used by it. For more information about them, see:

<http://xmlrpc-c.sourceforge.net/introspection.html>.

- [List AJAX APIs Accessible Through JsrServlet \(listAjaxApis\)](#) on page 59

List AJAX APIs Accessible Through JsrServlet (listAjaxApis)

listAjaxApis

List all AJAX APIs that are accessible through JsrServlet.

Parameters

- none

Returns

array of:

- key: (string) Jsr key
- value: (string) description of the API

CHAPTER 17

XML-RPC Methods Information

This chapter covers the following methods that give you information about the XML-RPC methods supported by the server:

- [Get Information About All Supported XML-RPC Methods on page 61](#)
- [Get Help on an XML-RPC Method \(system.methodHelp\) on page 61](#)

Get Information About All Supported XML-RPC Methods

system.listMethods

Parameters

- none

Returns

- array of strings each of which is the name of an XML-RPC method implemented by the server.

Get Help on an XML-RPC Method (system.methodHelp)

system.methodHelp

This XML-RPC service that given a method name provides a description of it.

Parameters

- method name: (string) name of a method implemented by the XML-RPC server.

Returns

- description: (string) a description documenting the use of that method. Presently the descriptions are auto-generated and simply provide the method signature in string format.

PART 3

Index

- [Index on page 65](#)

T

technical support	
contacting JTAC.....	xii

Index

Symbols

#, comments in configuration statements.....	xi
(), in syntax descriptions.....	xi
< >, in syntax descriptions.....	x
[], in configuration statements.....	xi
{ }, in configuration statements.....	xi
(pipe), in syntax descriptions.....	xi

B

braces, in configuration statements.....	xi
brackets	
angle, in syntax descriptions.....	x
square, in configuration statements.....	xi

C

comments, in configuration statements.....	xi
conventions	
text and syntax.....	x
curly braces, in configuration statements.....	xi
customer support.....	xii
contacting JTAC.....	xii

D

documentation	
comments on.....	xi

F

font conventions.....	x
-----------------------	---

M

manuals	
comments on.....	xi

P

parentheses, in syntax descriptions.....	xi
--	----

S

support, technical See technical support	
syntax conventions.....	x

