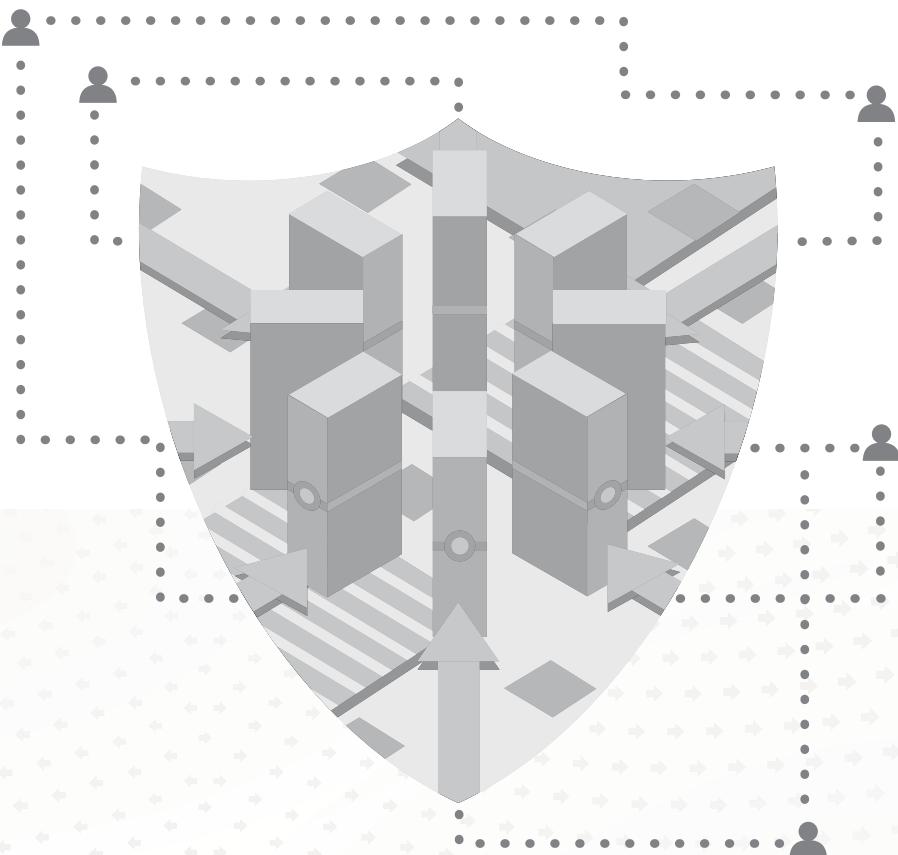


# Sourcefire 3D System Remediation API Guide

Version 5.2



## **Legal Notices**

Cisco, the Cisco logo, Sourcefire, the Sourcefire logo, Snort, the Snort and Pig logo, and certain other trademarks and logos are trademarks or registered trademarks of Cisco and/or its affiliates in the United States and other countries. To view a list of Cisco trademarks, go to this URL: [www.cisco.com/go/trademarks](http://www.cisco.com/go/trademarks). Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company.

The legal notices, disclaimers, terms of use, and other information contained herein (the "terms") apply only to the information discussed in this documentation (the "Documentation") and your use of it. These terms do not apply to or govern the use of websites controlled by Cisco or its subsidiaries (collectively, "Cisco") or any Sourcefire-provided or Cisco-provided products. Sourcefire and Cisco products are available for purchase and subject to a separate license agreement and/or terms of use containing very different terms and conditions.

The copyright in the Documentation is owned by Cisco and is protected by copyright and other intellectual property laws of the United States and other countries. You may use, print out, save on a retrieval system, and otherwise copy and distribute the Documentation solely for non-commercial use, provided that you (i) do not modify the Documentation in any way and (ii) always include Cisco's copyright, trademark, and other proprietary notices, as well as a link to, or print out of, the full contents of this page and its terms.

No part of the Documentation may be used in a compilation or otherwise incorporated into another work or with or into any other documentation or user manuals, or be used to create derivative works, without the express prior written permission of Cisco. Cisco reserves the right to change the terms at any time, and your continued use of the Documentation shall be deemed an acceptance of those terms.

© 2004 - 2014 Cisco and/or its affiliates. All rights reserved.

## **Disclaimers**

THE DOCUMENTATION AND ANY INFORMATION AVAILABLE FROM IT MAY INCLUDE INACCURACIES OR TYPOGRAPHICAL ERRORS. CISCO MAY CHANGE THE DOCUMENTATION FROM TIME TO TIME. CISCO MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE ACCURACY OR SUITABILITY OF ANY CISCO-CONTROLLED WEBSITE, THE DOCUMENTATION AND/OR ANY PRODUCT INFORMATION. CISCO-CONTROLLED WEBSITES, THE DOCUMENTATION AND ALL PRODUCT INFORMATION ARE PROVIDED "AS IS" AND CISCO DISCLAIMS ANY AND ALL EXPRESS AND IMPLIED WARRANTIES, INCLUDING BUT NOT LIMITED TO WARRANTIES OF TITLE AND THE IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL CISCO BE LIABLE TO YOU FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, PUNITIVE, OR CONSEQUENTIAL DAMAGES (INCLUDING BUT NOT LIMITED TO PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, LOSS OF DATA, LOSS OF PROFITS, AND/OR BUSINESS INTERRUPTIONS), ARISING OUT OF OR IN ANY WAY RELATED TO CISCO-CONTROLLED WEBSITES OR THE DOCUMENTATION, NO MATTER HOW CAUSED AND/OR WHETHER BASED ON CONTRACT, STRICT LIABILITY, NEGLIGENCE OR OTHER TORTIOUS ACTIVITY, OR ANY OTHER THEORY OF LIABILITY, EVEN IF CISCO IS ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. BECAUSE SOME STATES/JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES, THE ABOVE LIMITATIONS MAY NOT APPLY TO YOU.

2014-Mar-25 12:54

# Table of Contents

<b>Chapter 1:</b>	<b>Understanding the Remediation Subsystem.....</b>	<b>5</b>
	Prerequisites .....	6
	Sourcefire 3D System.....	6
	Programming Requirements and Support .....	6
	Sourcefire-Provided Remediation Modules .....	7
	The Remediation Subsystem.....	8
	Understanding Remediation Subsystem Architecture .....	8
	Remediation Subsystem Components .....	9
	Remediation Module Architecture .....	9
	Using the Remediation Subsystem .....	10
	Remediation Resources.....	11
<b>Chapter 2:</b>	<b>Planning and Packaging Your Remediation Module.....</b>	<b>13</b>
	Data Available from the Remediation Subsystem.....	14
	Event Data .....	14
	Instance Configuration Data .....	22
	The config Element.....	23
	The remediation Element .....	25
	Data Returned by Modules .....	27
	Packaging and Installing Your Module.....	27
	Packaging Your Module.....	28
	Installing Your Module .....	28

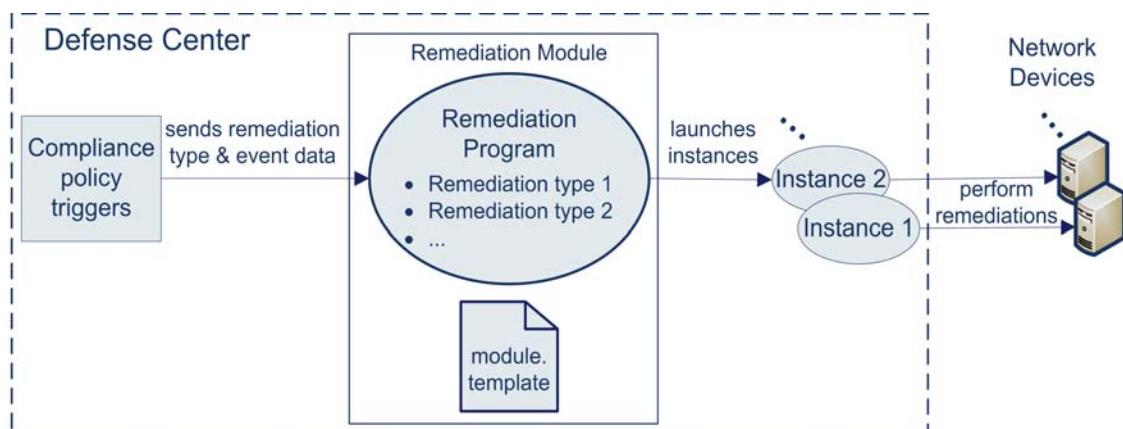
## Table of Contents

<b>Chapter 3:</b>	<b>Communicating with the Remediation Subsystem.....</b>	<b>30</b>
	Defining the Global Configuration.....	32
	Defining the Configuration Template .....	34
	The boolean Element.....	36
	The integer Element .....	37
	The string Element .....	39
	The password Element.....	41
	The ipaddress Element .....	43
	The netmask Element.....	45
	The host Element .....	46
	The network Element .....	48
	The enumeration Element .....	50
	The list Element.....	51
	Sample Configuration Template.....	54
	Defining Remediation Types .....	56
	Defining Exit Statuses .....	59
<b>Chapter 4:</b>	<b>Working with the Remediation SDK.....</b>	<b>61</b>
	Understanding the Remediation SDK.....	61
	Purpose of the SDK .....	61
	Description of the SDK .....	62
	Downloading the SDK.....	63
	Overview of the Development and Installation Process.....	63
	Notes for Remediation Program Developers .....	64
	Implementing Remediation Types in a Remediation Program .....	65
	Understanding the Remediation Subsystem File Structure .....	65
	Understanding the Remediation Program Workflow .....	66
	The Order of Command Line Parameters.....	67
	Handling Undefined Data Elements.....	67
	Handling Return Codes.....	68
	Important Global Configuration Elements.....	68
<b>Index .....</b>		<b>69</b>

# CHAPTER 1

## UNDERSTANDING THE REMEDIATION SUBSYSTEM

The Sourcefire 3D System® remediation API allows you to create remediations that your Defense Center can automatically launch when conditions on your network violate the associated correlation policy. A *remediation* is the response your software program executes to mitigate the detected condition. For example, you can block traffic at a router on the source or destination IP address, or initiate a host Nmap scan to assess the host status. If multiple rules in a policy trigger, the Defense Center can launch responses for each rule. A *remediation module* is the package of files you install on the Defense Center to perform the response. A remediation module can incorporate several *remediation types* as shown in the graphic below.



For example, one of the Sourcefire-provided remediation modules, the Cisco PIX router module, performs two remediation types: it either blocks packets by source IP address or blocks them by destination IP address.

If a remediation module targets multiple devices on your network (routers, hosts, and so forth), you configure your remediation module to perform multiple *instances*, one per device, when the correlation policy triggers. An instance is an instantiation of the remediation module, with one or more remediation types that correspond to functions in the remediation module code, and with a set of variables needed to run on the target device. For each instance, you specify the remediation type or types it executes and the instance-specific information such as the device's IP address and password for the remediation to access the target device on your network.

## Prerequisites

Before using the remediation API for custom remediations, you should be familiar with information in the following categories:

- [Sourcefire 3D System](#) on page 6
- [Programming Requirements and Support](#) on page 6
- [Sourcefire-Provided Remediation Modules](#) on page 7

## Sourcefire 3D System

To understand the information in this guide, you should be familiar with the features and nomenclature of the Sourcefire 3D System, and the functions of certain components:

- the Defense Center role in the Sourcefire 3D System architecture
- correlation policy management module on the Defense Center
- remediation management module on the Defense Center

See the *Sourcefire 3D System User Guide* for further information.

## Programming Requirements and Support

You must be able to code your custom remediation in Perl or shell script, or as a precompiled, statically-linked C program (with the exception of links to routines in glibc).

In addition, you must be able to produce a configuration file in XML for each remediation module. This file is called `module.template`. See the Sourcefire-provided remediation modules for samples of this file. For module locations on the Defense Center, see [Understanding the Remediation Subsystem File Structure](#) on page 65.

For each instance you add, the Defense Center generates an instance-specific XML configuration file called `instance.conf`. Your code must parse this file each time a remediation instance executes.

The [Additional Packages](#) table lists the packages available on the Defense Center as resources for writing and executing your remediation program.

#### Additional Packages

ADDITIONAL PACKAGES	LOCATION
GNU bash, version 3.2.33(1)-release	/bin/bash
tcsh 6.17.00	/bin/tcsh
glibc 2.7	/lib/libc-2.7.so
perl v5.10.1	/usr/bin/perl
Net::Telnet	N/A
Net::SSH::Perl	N/A
XML::Smart	N/A

## Sourcefire-Provided Remediation Modules

The [Sourcefire-Provided Remediation Modules](#) table describes the predefined remediation modules included with the Defense Center. You should use these modules for reference when designing your remediation programs.

The Sourcefire-provided modules are already installed on the Defense Center and include both the remediation executable (in Perl and C) and completed module template configuration file for each module. For information on the easy steps to deploy Sourcefire-provided remediation modules, see the “Configuring Responses for Correlation Policies” chapter in the *Sourcefire 3D System User Guide*.

#### Sourcefire-Provided Remediation Modules

MODULE NAME	FUNCTION
Cisco IOS Null Route	if you are running Cisco routers that use Cisco IOS® Version 12.0 or higher, allows you to dynamically block traffic sent to an IP address or network that violates a correlation policy
Cisco PIX Shun	if you are running Cisco PIX® Firewall Version 6.0 or higher, allows you to dynamically block traffic sent from an IP address that violates a correlation policy

#### Sourcefire-Provided Remediation Modules (Continued)

MODULE NAME	FUNCTION
Nmap Scanning	allows you to actively scan specific targets to determine operating systems and servers running on those hosts
Set Attribute Value	allows you to set a host attribute on a host where a correlation event occurs

## The Remediation Subsystem

The remediation subsystem consists of the following components:

- the Defense Center's web interface, which you use to set up correlation policies and associate them with remediations, and to track the status of remediation processing
- the remediation API, which enables you to define the data that will be provided to your remediation modules
- the remediation daemon, which passes data to the remediation modules at run time and collects execution status information
- remediation modules, which perform specific responses to correlation policy violations

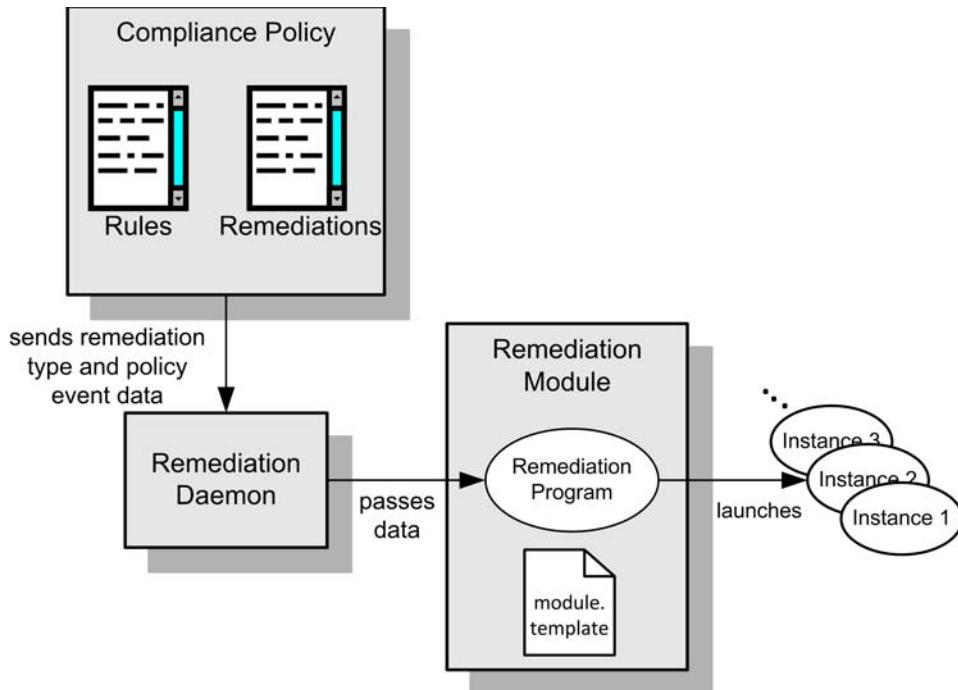
## Understanding Remediation Subsystem Architecture

The remediation subsystem has a two-part architecture that is diagrammed in the figure below. The architecture consists of:

- infrastructure components such as the web interface and the remediation daemon which support all remediation modules. The infrastructure components allow you to create and manage all the remediation modules on your Defense Center. The remediation daemon manages the execution of the remediations. See [Remediation Subsystem Components](#) on page 9 for more details.
- the individual remediation modules which you develop to respond to specific correlation policy violations. See [Remediation Module Architecture](#) on page 9 for more details.

## Remediation Subsystem Components

The following diagram illustrates the main functions of the remediation subsystem and their interactions.



You create remediations in order to respond to rule violations on your network in an automated mode. The Defense Center web interface allows you to define and activate your correlation policies and associate them with remediations. When a policy violation occurs, the remediation subsystem passes the name of the remediation and the event data specified in the module template configuration file to the remediation daemon.

The remediation daemon launches the remediation and passes the correlation event data and instance-specific parameters to your remediation program. It also accepts return codes from the remediation program. The Defense Center uses the return codes for status displays.

The remediation program launches a set of *instances* of the remediation when the associated policy rule triggers. Each instance targets a particular network device. You create instances on the Instance Detail page of the Defense Center web interface. For each instance you provide the necessary instance-specific configuration details such as IP address and password of the target device.

## Remediation Module Architecture

Each remediation module that you install on your Defense Center includes one or more remediation types. You assign one or more remediation types to each

instance. For information on configuring remediations as responses to policy violations, see the “Configuring Responses for Correlation Policies” chapter in the *Sourcefire 3D System User Guide*.

Remediation modules include the following components:

- the remediation program, included in the remediation module package at installation. See [Planning and Packaging Your Remediation Module](#) on page 13.
- a required XML `module.template` file, also included in the remediation module package at installation. This file provides module-level information about your module and its data requirements that the remediation subsystem references each time it launches one of the remediation module’s instances. See [Communicating with the Remediation Subsystem](#) on page 30.
- one XML `instance.conf` file per instance. The Defense Center auto-generates this file each time you configure a new instance of your remediation module.

## Using the Remediation Subsystem

You deploy remediations by adding them as responses to specific rules in correlation policies on your Defense Center. You define the associations of correlation policies and remediations using the Defense Center web interface.

**To deploy a remediation module, you must:**

1. Identify the condition you want to mitigate and the actions that appropriately resolve that condition in your environment. These actions are the main functions your custom remediation program must implement.
2. If you can use a Sourcefire-provided remediation module, skip directly to step 7.
3. If you need to produce a custom remediation module, familiarize yourself with the data elements obtainable from the remediation subsystem. See [Data Available from the Remediation Subsystem](#) on page 14.
4. If you develop a custom remediation module you must also create a `module.template` file to be included in your module package. See [Communicating with the Remediation Subsystem](#) on page 30 for the format and syntax of the `module.template` file.
5. Write your remediation program so that it addresses all the functions necessary for the desired remediations. You can write your remediation module programs in bash, tsch, Perl or C. Develop your program using the technical guidance in [Notes for Remediation Program Developers](#) on page 64.
6. Package your remediation module as described in [Packaging Your Module](#) on page 28.

7. Install the module on the Defense Center using the web interface as described in [Installing Your Module](#) on page 28.
8. Ensure that the individual remediation types in your remediation module are assigned as responses to the correct correlation rules in your active correlation policies. See the *Sourcefire 3D System User Guide* for procedure details.

## Remediation Resources

The [Remediation Resources](#) table describes some of the topics explained in the documentation and where to look for more information.

In addition to this document, other resources you can use to create your remediation modules include:

- a remediation SDK with sample program code in C or Perl that generates syslog alerts and demonstrates how a module can interact with your network. See [Working with the Remediation SDK](#) chapter of this document for detailed information. The SDK can be downloaded from the Sourcefire Support web site at <https://support.sourcefire.com/downloads>.
- the `modul e. templ ate` schema (`modul e. templ ate. xsd`), which is located on the Defense Center at `/etc/sf/remedi ati on/modul e. templ ate. xsd`.

The [Remediation Resources](#) table describes some of the topics explained in the documentation and where to look for more information.

### Remediation Resources

TO LEARN MORE ABOUT...	SEE ...
the sample remediation module and the general procedure for creating, installing, and configuring one	<a href="#">Working with the Remediation SDK</a> on page 61
writing your remediation program	<a href="#">Planning and Packaging Your Remediation Module</a> on page 13
creating the <code>modul e. templ ate</code> file	<a href="#">Communicating with the Remediation Subsystem</a> on page 30
packaging your remediation module so you can install it on the Defense Center	<a href="#">Packaging Your Module</a> on page 28

Remediation Resources (Continued)

TO LEARN MORE ABOUT...	SEE ...
installing your remediation module	<a href="#">Installing Your Module</a> on page 28
configuring your remediations as responses to security policy violations	the “Configuring Responses for Correlation Policies” chapter in the <i>Sourcefire 3D System User Guide</i>

# CHAPTER 2

## PLANNING AND PACKAGING YOUR REMEDIATION MODULE

Planning the development of a custom remediation module consists of the tasks listed in the [Remediation Module Planning Tasks](#) table below, which indicates where to find information and guidance on each task area.

### Remediation Module Planning Tasks

FOR GUIDANCE ON ...	LOOK IN...
performing a functional analysis and the importance of understanding the remediation subsystem concept of operations	<a href="#">Overview of the Development and Installation Process</a> on page 63
reviewing the data available from the remediation subsystem	<a href="#">Data Available from the Remediation Subsystem</a> on page 14
using the return code function of the remediation subsystem	<a href="#">Data Returned by Modules</a> on page 27
coordinating your software development and generating the module template file	<a href="#">Communicating with the Remediation Subsystem</a> on page 30
packaging the remediation module and installing it	<a href="#">Packaging and Installing Your Module</a> on page 27

## Data Available from the Remediation Subsystem

Custom remediation modules can receive two kinds of data from the remediation subsystem:

- event data, which includes a variety of data about the correlation policy that was violated and about the original triggering event that caused the policy violation
- instance configuration data, which includes values entered in the web interface when an instance of a remediation is configured

These two types of data incorporate both the data about the network traffic or change that triggered the rules in the violated policy, and the configured instance of the remediation that runs in response to that policy violation. See “Configuring Correlation Policies and Rules” and “Configuring Responses for Correlation Policies” in the *Sourcefire 3D System User Guide* for more information about creating, configuring and using correlation policies and remediations.

See the following sections for more information:

- [Event Data](#) on page 14 describes how event data is provided to your remediation module and lists the correlation event data available to your module.
- [Instance Configuration Data](#) on page 22 explains how `instance.config` files are made available to your remediation module and describes the types of data they may include.

### Event Data

Event data is one type of information available to your remediation module. Event is information about intrusion, correlation, and other event types that the Defense Center generates when rules in a correlation policy trigger. You specify the event data fields to be sent for each remediation type in your module using the `pe_item` element in the `module.template` file.

When the remediation daemon sends event data to your remediation module, it passes the name of the remediation first, followed by the `pe_item` fields in the order in which they appear in `module.template`.

The remediation daemon handles any undefined `pe_item` fields from the database differently depending on whether they field is marked as optional or required in `module.template`. See [Handling Undefined Data Elements](#) on page 67.

For details on specifying event data for remediations, see [Defining Remediation Types](#) on page 56. When specifying the `pe_item` element, you must use the field names provided in the tables below.

The [Triggering Event Data](#) table describes data available about the original event that triggered the correlation policy violation. Note that some fields in this table

are event specific. These fields are set to zero when not applicable for the specific type of triggering event.

**Triggering Event Data**

<b>NAME</b>	<b>DESCRIPTION</b>	<b>FIELD</b>	<b>TYPE</b>	<b>BYTES</b>
Transport Protocol	The transport protocol (TCP, UDP, IP, ICMP) of the packet that triggered the intrusion or discovery event that caused the policy violation.	ip_protocol	uint8_t	1
Network Protocol	The network protocol (for example, ethernet) of the packet that triggered the intrusion or discovery event that caused the policy violation.	net_protocol	uint16_t	2
Triggering Event Type	A numeric identifier for the type of event that triggered the correlation event. Values are:  1 = intrusion 2 = network discovery, connection, or connection summary 3 = user awareness 4 = white list	event_type	uint8_t	1
Triggering Event ID	An internal identifier for the event that triggered the correlation event. Set only for intrusion events. Set to 0 for other event types.	event_id	uint32_t	4
Triggering Event Time	Content varies by event type: <ul style="list-style-type: none"><li>• for intrusion, network discovery, connection, and user awareness events: UNIX timestamp of the triggering event</li><li>• for connection summaries: correlation event time (that is, policy_tv_sec)</li><li>• for white list events: set to 0</li></ul>	tv_sec	uint32_t	4
Triggering Event Time (usec)	The microsecond increment of the event time. Set to 0 if granularity is not available.	tv_usec	uint32_t	4
Triggering Event Description	A text description of the original event that triggered the correlation event. Content varies by event type.	description	char *	Max 1024
Triggering Event Sensor ID	The internal identifier of the sensor where the triggering event occurred.  Primarily for Sourcefire internal use, not typically used for remediations.	sensor_id	uint32_t	4

Triggering Event Data (Continued)

NAME	DESCRIPTION	FIELD	TYPE	BYTES
Triggering Event Generator ID	<p>Content varies by event type:</p> <ul style="list-style-type: none"> <li>• for intrusion events: the generator ID (GID) for the event. See the <i>Sourcefire 3D System User Guide</i> for a complete list of GIDs.</li> <li>• for network discovery and connection events: the network discovery event type.</li> <li>• for connection summaries: set to 4 for all.</li> <li>• for user awareness events: the user awareness event type.</li> <li>• for white list events: set to 0.</li> </ul> <p>Primarily for Sourcefire internal use and not typically used for remediations.</p>	sig_gen	uint32_t	4
Triggering Event Signature ID	<p>Content varies by event type:</p> <ul style="list-style-type: none"> <li>• for intrusion events: the signature ID (SID) for the event. May not match the SID displayed in the user interface.</li> <li>• for network discovery and connection events: network discovery event subtype.</li> <li>• for connection summaries: set to 17 for all.</li> <li>• for user awareness events: user awareness event subtype</li> <li>• for white list events: set to 0.</li> </ul> <p>Primarily for Sourcefire internal use and not typically used for remediations.</p>	sig_id	uint32_t	4

Triggering Event Data (Continued)

NAME	DESCRIPTION	FIELD	TYPE	BYTES
Impact Flags	<p>Impact flag value of the event. The low-order eight bits indicate the impact level. Values are:</p> <ul style="list-style-type: none"> <li>• 0x01 (bit 0) - Source or destination host is in a network monitored by the system.</li> <li>• 0x02 (bit 1) - Source or destination host exists in the network map.</li> <li>• 0x04 (bit 2) - Source or destination host is running a server on the port in the event (if TCP or UDP) or uses the IP protocol.</li> <li>• 0x08 (bit 3) - There is a vulnerability mapped to the operating system of the source or destination host in the event.</li> <li>• 0x10 (bit 4) - There is a vulnerability mapped to the server detected in the event.</li> <li>• 0x20 (bit 5) - The event caused the managed device to drop the session (used only when the device is running in inline, switched, or routed deployment). Corresponds to blocked status in the Sourcefire 3D System web interface.</li> <li>• 0x40 (bit 6) - The rule that generated this event contains rule metadata setting the impact flag to red. The source or destination host is potentially compromised by a virus, trojan, or other piece of malicious software.</li> <li>• 0x80 (bit 7) - There is a vulnerability mapped to the client detected in the event. (version 5.0+ only)</li> </ul> <p>The following impact level values map to specific priorities on the Defense Center. An X indicates the value can be 0 or 1:</p> <ul style="list-style-type: none"> <li>• gray (0, unknown): 00X00000</li> <li>• red (1, vulnerable): XXXX1XXX, XXX1XXXX, X1XXXXXX, 1XXXXXXX (version 5.0+ only)</li> <li>• orange (2, potentially vulnerable): 00X0011X</li> <li>• yellow (3, currently not vulnerable): 00X0001X</li> <li>• blue (4, unknown target): 00X00001</li> </ul>	impact_flags	uint32_t	4

The [Correlation Event Data](#) table describes the data available about each correlation event. Note that some of the data elements are not populated for certain event types.

#### Correlation Event Data

<b>NAME</b>	<b>DESCRIPTION</b>	<b>FIELD</b>	<b>TYPE</b>	<b>BYTES</b>
Correlation Event Time	UNIX timestamp of when the correlation event was generated.	policy_tv_sec	uint32_t	4
Correlation Event ID	The internal identification number of the event generated by the sensor. Set only for intrusion events.  Primarily for Sourcefire internal use and not typically used for remediations.	policy_event_id	uint32_t	4
Correlation Appliance ID	The internal identification number of the Defense Center that generated the correlation event.  Primarily for Sourcefire internal use and not typically used for remediations.	policy_sensor_id	uint32_t	4
Correlation Policy ID	The internal identification number of the of the correlation policy that was violated by the triggering event.  Primarily for Sourcefire internal use and not typically used for remediations.	policy_id	uint32_t	4
Correlation Rule ID	The internal identification number of the correlation rule that triggered the correlation event.  Primarily for Sourcefire internal use and not typically used for remediations.	rule_id	uint32_t	4
Correlation Rule Priority	The priority assigned to the rule for the correlation policy that generated the event. The rule may have a different priority in another policy. Value: 0 - 5 (0 = no priority)	priority	uint32_t	4
Event-Defined Mask	A bit field in the correlation event message that indicates which of the fields that follow the mask are valid. See the <a href="#">Event Defined Values table</a> on page 19 for the values.  Primarily for Sourcefire internal use and not typically used for remediations.	defined_mask	uint32_t	4

The [Event Defined Values](#) table defines the mask values for the correlation event message fields. These values are used in the correlation event message to indicate which of the fields that follow the mask are valid.

Event Defined Values

CORRELATION EVENT FIELD	MASK VALUE
Event Impact Flags	0x00000001
IP Protocol	0x00000002
Network Protocol	0x00000004
Source IP	0x00000008
Source Host Type	0x00000010
Source VLAN ID	0x00000020
Source Fingerprint ID	0x00000040
Source Criticality	0x00000080
Source Port	0x00000100
Source Server	0x00000200
Destination IP	0x00000400
Destination Host Type	0x00000800
Destination VLAN ID	0x00001000
Destination Fingerprint ID	0x00002000
Destination Criticality	0x00004000
Destination Port	0x00008000
Destination Server	0x00010000
Source User	0x00020000
Destination User	0x00040000

The [Source Host Data](#) table describes the data available about the source host involved in the intrusion event, or the only host involved in any other discovery event that caused the correlation policy violation. Note that only the source IP address is guaranteed to be populated.

#### Source Host Data

<b>NAME</b>	<b>DESCRIPTION</b>	<b>FIELD</b>	<b>TYPE</b>	<b>BYTES</b>
IP Address	The IP address of the source host in the event that triggered the policy violation. For a discovery event, the host or initiator host's IP address.	src_ip_addr	uint32_t	4
Host Type ID	The host's recognized type (for example, router, bridge); discovery events only.	src_host_type	uint8_t	1
VLAN ID	The host's VLAN ID; discovery events only.	scr_vlan_id	uint16_t	2
OS Vendor	The vendor of the host's identified operating system; discovery events only.	src_os_vendor	char*	max 255
OS Product	The host's identified operating system; discovery events only.	src_os_product	char*	max 255
OS Version	The version number of the host's identified operating system; discovery events only.	src_os_version	char*	max 255
Host Criticality	A user-defined value in host and connection events.	src_criticality	uint16_t	2

The [Source Server Data](#) table describes the data available about the source host's server, or only server identified in the event that caused the correlation event. Note that only the transport protocol is guaranteed to be populated

#### Source Server Data

<b>NAME</b>	<b>DESCRIPTION</b>	<b>FIELD</b>	<b>TYPE</b>	<b>BYTES</b>
Port	Port on which the identified server is running. For intrusion events, port is populated only if the protocol is TCP or UDP.	src_port	uint16_t	2
Server	Server (for example, HTTP, SMTP) identified in the event that caused the policy violation.	src_service	char	max 255

The [Destination Host Data](#) table describes the data available about the destination host. This data is only available for intrusion events.

#### Destination Host Data

<b>NAME</b>	<b>DESCRIPTION</b>	<b>FIELD</b>	<b>TYPE</b>	<b>BYTES</b>
IP Address	The IP address of the destination host in the event that triggered the policy violation.	dest_ip_addr	uint32_t	4
Host Type ID	The destination host's recognized type (for example, router, bridge).	dest_host_type	uint8_t	1
VLAN ID	The destination host's VLAN ID.	dest_vlan_id	uint16_t	2
OS Vendor	The vendor of the host's identified operating system; discovery events only.	dest_os_vendor	char*	max 255
OS Product	The host's identified operating system; discovery events only.	dest_os_product	char*	max 255
OS Version	The version number of the host's identified operating system; discovery events only.	dest_os_version	char*	max 255
Host Criticality	A user-defined value in; discovery host and connection events.	dest_critically	uint16_t	2

The [Destination Server Data](#) table describes the data available about the destination host's server, or the only server identified in the event that caused the

correlation event. Note that only the transport protocol is guaranteed to be populated.

#### Destination Server Data

NAME	DESCRIPTION	FIELD	TYPE	BYTES
Destination Port	Port on which the identified server is running. In the case of intrusion events, the port is populated only if the protocol is identified as TCP or UDP.	dest_port	uint16_t	2
Destination Server	Server (for example, HTTP, SMTP) identified in the event that caused the policy violation.	dest_service	char	max 255

#### Instance Configuration Data

When a user configures a new instance of your module, they provide data requested in your module template document. The values provided by the user are then written into the `instance.conf` document for use by your remediation program.

For each configured instance of a remediation, the remediation subsystem places an `instance.conf` document in a directory with the same name as the instance. This directory is created in the directory where your module was uploaded and installed. For example, if your module is called Firewall, it is uploaded into a directory called `firewall`. If you then configure an instance called `block_tokyo`, the remediation subsystem creates a directory called `block_tokyo` in your `firewall` directory and places the `instance.conf` there. The directory path appears as follows:

```
/var/sf/remediation/firewall/block_tokyo/instance.conf
```

See [Packaging Your Module](#) on page 28 for more information on the directories where your module files reside.

Your module must be able to open, read, parse, and close the `instance.conf` file.

Each `instance.conf` document contains a top level element called `instance`. The `instance` element has two child elements: `config` and `remediation`. The

[instance Attributes and Child Elements](#) table describes the attributes and elements available to the instance element.

#### instance Attributes and Child Elements

NAME	TYPE	DESCRIPTION
name	attribute	Ties the data in the document to the named, configured instance and reflects the name of the instance specified by the configuring user.
config	element	Contains the data entered into the instance configuration fields on the web interface at configuration.
remediation	element	Contains the data entered into the web interface when configuring the remediation for an instance.

For more information about the data provided in the config and remediation elements, see the following:

- [The config Element](#) on page 23
- [The remediation Element](#) on page 25

## The config Element

The config element contains the data entered into the fields rendered on the web interface in response to the config\_template element in that remediation module's module\_template document. These fields are translated back into the elements used to specify them in the module\_template document, and further specified using the name provided as an attribute of the element rather than a child element. They can include the following types of fields:

- boolean
- string
- integer
- password
- host
- netmask
- network
- ipaddress
- enumeration
- list

See [Defining the Configuration Template](#) on page 34 for more details on how these fields are specified in the module template file.

For example, if the module template document contains the following configuration template element definition:

```
<configuration>
    <ipaddress>
        <name>host_ip</name>
        <display_name>Host IP</display_name>
    </ipaddress>
    <string>
        <name>user_name</name>
        <display_name>Username</display_name>
        <constraints>
            <pcre>\S+</pcre>
        </constraints>
    </string>
    <password>
        <name>login_password</name>
        <display_name>Login Password</display_name>
    </password>
</configuration>
```

The Instance Configuration screen for that element contains the following three fields:

- Host IP, which takes an IP address value.
- Username, which takes a string value that may not contain white space characters.
- Login Password, which takes a string value identified as a password.

Suppose a user configures an instance, named AdminInstance, of the remediation module and provides the following values:

#### Sample Values

FIELD	VALUE
Host IP	192.1.1.1
Username	adminuser
Login Password	3admin3

The `instance.conf` will contain the following:

```
<instance name="AdminInstance">
  <config>
    <ipaddress name="host_ip">192.1.1.1</ipaddress>
    <string name="user_name">adminuser</string>
    <password name="Login_password">3admin3</password>
  </config>
```

Note that the above example does not include `</instance>`. This is because the `instance.conf` document for this example instance would go on to include the `remediation` element discussed next in this section. If you do not require additional remediation configuration in your module, the `instance.conf` returned for that module does **not** include remediation elements.

## The remediation Element

The `instance` element contains a `remediation` element for each remediation configured for that instance. Each `remediation` element has, as an attribute, the name of the remediation instance (entered into the web interface at the time the instance is configured) and the type of the remediation, which was initially provided by the `remediation_type` element in the module template document. For more information about the module template file, see [Communicating with the Remediation Subsystem](#) on page 30.

In addition, `remediation` elements can contain `config` elements. These function in the same way as `config` elements that are child elements of `instance`, but use data originally specified in the `config_template` element that is a child of `remediation_type` in the module template document. The [remediation Attributes and Child Elements](#) table describes these attributes and elements.

remediation Attributes and Child Elements

NAME	TYPE	DESCRIPTION
name	attribute	Ties the data in the document to the named, configured remediation and reflects the name specified by the configuring user.
type	attribute	Provides the type of remediation configured in this instance.
config	element	Contains the data entered into the remediation configuration fields on the web interface at configuration.

For example, suppose the module template document in the example provided in [The config Element](#) on page 23 continues with the following:

```
<remediation_type name="acl_insert">
  <display_name>ACL Insertion</display_name>
  <policy_event_data>
    <item>src_ip_addr</item>
    <item>src_port</item>
    <item>src_protocol</item>
    <item>dest_ip_addr</item>
    <item>dest_port</item>
    <item>dest_protocol</item>
  </policy_event_data>
  <config_template>
    <integer>
      <name>acl_num</name>
      <display_name>ACL Number</display_name>
    </integer>
  </config_template>
</remediation_type>
```

The Instance Detail page that allows you to add remediations to a created instance contains the remediation type “ACL Insertion.” Adding “ACL Insertion” to the instance takes the user to a page that includes a name field, which populates the name attribute value for that remediation element in the `instance.conf`, and a field labelled ACL Number, which accepts an integer value.

Suppose a user adds this remediation to the `AdminInstance` instance and provides the following values:

**Sample Values**

FIELD	VALUE
Remediation Name	AdminRemediation
ACL Number	55

The `instance.conf` document written when the user saved the example configuration values would, after the section provided in the example in [The](#)

[config Element](#) on page 23, continue as follows:

```
<remediation name="AdminRemediation" type="acl_insert">
  <config>
    <integer="acl_num">55</integer>
  </config>
</remediation>
```

Note that if no more remediations were added to the instance, the `instance.conf` should be terminated with `</instance>` at this point.

## Data Returned by Modules

Remediation modules must return exit status codes, known as return codes, to the Defense Center. The Table View of Remediations in the Defense Center web interface displays a result message for each remediation launched. The return code from the remediation program determines the result message displayed.

Return codes must be integers in the 0 to 255 range inclusive, as defined in the [Return Code Ranges](#) table below.

Return Code Ranges

RANGE	USE
0 - 128	Reserved for Sourcefire predefined return codes
129 - 255	Available for custom remediations

See [Defining Exit Statuses](#) on page 59 for the list of predefined codes and for directions on creating custom codes.

## Packaging and Installing Your Module

The remediation API requires that you package your remediation modules. The files that make up your module must be provided in a gzipped tar file.

See the following sections for more information:

- [Packaging Your Module](#) on page 28 provides helpful tips for packaging your binaries and module template files for upload and installation.
- [Installing Your Module](#) on page 28 explains how to install your remediation module on the Defense Center.

## Packaging Your Module

When packaging your remediation files for installation, keep in mind the following:

- Remediation modules must be packaged in a gzipped tarball (.tar.gz or .tgz) before you install them.
- When you install the module, the package is extracted into /var/sf/remediation/remediation\_directory where remediation\_directory is a combination of the name attribute of the module's module element and the data in the version element.  
For example, one of the default remediation modules shipped with the Defense Center is the Cisco PIX Shun module. That module resides in /var/sf/remediation/cisco\_pix\_1.0.
- When extracted, your remediation module's module.template document must reside in the top level of the directory created to contain that module package.
- As instances of remediations are created, they are saved in a directory created in your module directory and named for the instance.

For example, instances of the Cisco PIX Shun module might reside in /var/sf/remediation/cisco\_pix\_1.0/PIX\_01 and /var/sf/remediation/cisco\_pix\_1.0/PIX\_02.

For example, you upload and install a module that is packaged in firewall.tgz and is named in the module.template as firewall with a version value of 1.0. The system installs the module in the following directory:

/var/sf/remediation/firewall\_1.0. That directory contains your module.template file and your program binary. When you add an instance to the remediation module and name it block\_tokyo, the system creates the following directory:

/var/sf/remediation/firewall\_1.0/block\_tokyo

and places the instance.conf file for block\_tokyo in it.

## Installing Your Module

Once you have correctly packaged your remediation module, use the Modules page to install it.

To install a new module on the Remediation API:

1. Select **Policies > Actions > Modules**.

The Installed Remediation Modules page appears.

Installed Remediation Modules		
Module Name	Version	Description
Cisco IOS Null Route	1.0	Block an IP address in a Cisco IOS router
Cisco PIX Shun	1.0	Shun an IP address in the PIX firewall
Nmap Remediation	2.0	Perform an Nmap Scan
Set Attribute Value	1.0	Set an Attribute Value

Install a new module

2. Click **Browse** to navigate to the location where you saved the tar.gz file that contains the custom remediation module.

3. Click **Install**.

The custom remediation module installs.

4. Select **Policies > Actions > Modules**.

The Installed Remediation Modules table lists the module just installed. The Module Name, Version, and Description columns match the information defined in the module template file.

5. Add instances of your new module and associate remediations to each instance, as described in the *Sourcefire 3D System User Guide*.

You can use the Modules page to view the remediation modules installed on the Defense Center. The list displays custom remediation modules and Sourcefire-provided ones. You can also delete your custom modules.

To view or delete a module from the Remediation API:

1. Select **Policies > Actions > Modules**.

The Installed Remediation Modules page appears.

2. Perform one of the following actions:

- Click the View icon to view the module.

The Module Detail page appears.

- Click the Delete icon next to the module you want to delete. You **cannot** delete default modules provided by Sourcefire.

The remediation module is deleted from the Remediation API.

# CHAPTER 3

## COMMUNICATING WITH THE REMEDIATION SUBSYSTEM

Your remediation module must receive information from the Defense Center remediation subsystem to successfully perform its function. You configure the information that your module receives in an XML file called `modul e. templ ate`. Without it, the remediation subsystem cannot interact with your remediation module.

The `modul e. templ ate` XML file allows you to specify:

- a set of module-level declarations such as the name and version of your remediation module, a short descriptive text, and the name of the binary file for your remediation program
- the information the module requires from the user when the user configures remediation instances in the Defense Center user interface
- the specific remediation actions, known as remediation types, that the module can perform and the correlation event data each remediation type requires
- any custom return codes and exit status messages that your remediation program returns to the Defense Center

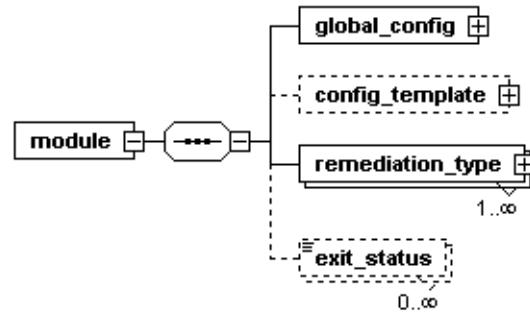
Before writing a `modul e. templ ate` for your remediation module, you should understand the `modul e. templ ate` schema (`modul e. templ ate. xsd`). The schema defines the elements (or tags used to contain data) and attributes (or data used to modify the data contained in an element) you can use to provide information to the remediation subsystem. The `modul e. templ ate` schema is located on the DC at `/etc/sf/remedi ati on/modul e. templ ate. vsd`.

The top-level element in `modul e.templ ate` is `modul e`, in which you specify the name of the remediation module using the `name` attribute. The `name` attribute is required and accepts a string value between 1 and 64 alphabetic characters.

---

**WARNING!** You cannot use white space in the module's `name` attribute value. In addition, you cannot use punctuation marks except for underscore (`_`) or dash (`-`).

---



Some XML editors can read the `modul e.templ ate` schema and automatically generate a `modul e.templ ate` file with a namespace and schema declaration, with the top level element and child elements and attributes. If you choose not to use such an editor, you must include the child elements manually.

---

**WARNING!** If you set your XML editor to auto-generate the namespace and schema location, you must delete those lines before including the final version of `modul e.templ ate` in your installation package.

---

The following example illustrates the `modul e` element with only the `name` attribute defined.

```
<modul e name="example_modul e">
  <gl obal _confi g>
    <di spl ay_name/>
    <versi on/>
    <bi nary/>
  </gl obal _confi g>
  <remedi ati on_type name="">
    <di spl ay_name/>
  </remedi ati on_type>
</modul e>
```

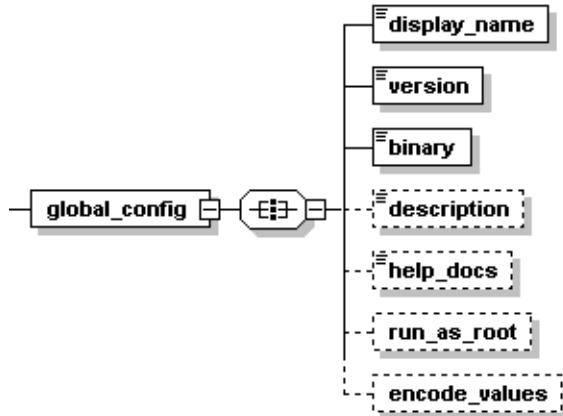
See the following sections for details about writing the rest of `module.template`.

- [Defining the Global Configuration](#) on page 32 explains how to use the `global_config` element to define the name that appears for your module on the Modules page, as well as the module's version, binary location, and its description.
- [Defining the Configuration Template](#) on page 34 explains how to use the `config_template` element to define the configuration information that your module requires the user to specify from the web interface.
- [Defining Remediation Types](#) on page 56 explains how to use the `remediation_type` element to define the remediations the module can launch and the correlation event data that each remediation requires.
- [Defining Exit Statuses](#) on page 59 explains how to use the `exit_status` element define the custom exit statuses your module returns to the remediation subsystem.

## Defining the Global Configuration

The first required section of `module.template` uses the `global_config` element to define global configuration information. These attributes include the module's name and description, which appear in the list of remediation modules displayed on the Modules page of the Defense Center user interface. The global information also includes the module's version and the location of the executable program that runs when a remediation is triggered.

The following portion of the `module.template` schema diagram illustrates the child elements of the `global_config` element.



The [global\\_config Child Elements](#) table describes the child elements available to the global\_config element.

#### global\_config Child Elements

NAME	DESCRIPTION	REQUIRED?
display_name	Specifies the name that appears for this remediation module on the Modules page. The display name can contain only alphanumeric characters and white spaces and must be between 1 and 127 characters long. It must be unique across remediation modules.	yes
version	Specifies the version of the remediation module. This value appears on the Modules page. The value for the version element must begin and end with numeric characters, but may contain period (.) characters.	yes
	<b>IMPORTANT!</b> The combination of the name attribute of the module element and the data in the version element must be unique across remediation modules.	
binary	Specifies the UNIX filename of the binary that makes up your remediation module	yes
description	Provides a description of the remediation module and its available remediations. The description element appears on the Modules page. Descriptions with more than 255 characters are truncated.	yes
run_as_root	Sets a flag that allows the remediation module to run as root on the Sourcefire appliance where it is installed.	no
	<b>WARNING!</b> Sourcefire recommends that you use this element only if absolutely necessary.	
encode_values	Sets a flag that HTML-encodes user input. This allows users to enter input that might otherwise be unintentionally interpreted by the XML processor.	no
	<b>IMPORTANT!</b> If you use this element, your remediation module <b>must</b> handle HTML decoding as part of its input handling.	

Consider the following XML code, which illustrates the global configuration portion of a `module.templ` ate file.

```
<global_config>
    <display_name>My Firewall </display_name>
    <binary>firewall_block.pl </binary>
    <description>Dynamically apply firewall rules to my
firewall. </description>
    <version>1.0</version>
    <run_as_root/>
</global_config>
```

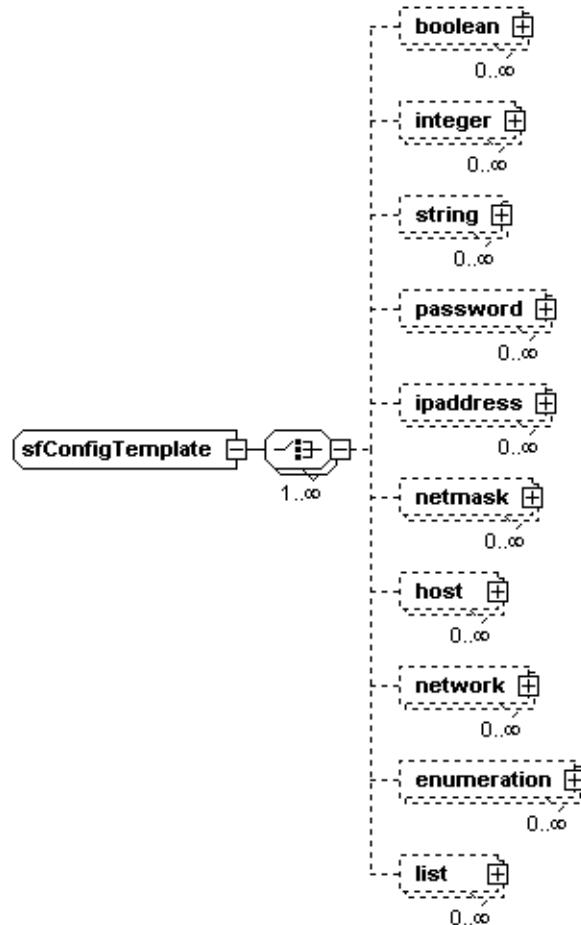
In this example, the remediation module is represented by the name My Firewall in the web interface. It runs version 1.0 of a program called `firewall_block.pl`, which you install using the Defense Center (see [Packaging and Installing Your Module](#) on page 27 for more information). The program dynamically applies firewall rules to a specific firewall and runs as root on the Defense Center.

## Defining the Configuration Template

The `config_template` child element of the `module` element specifies the types of information the user must provide when configuring the instances that this remediation module executes (see [Instance Configuration Data](#) on page 22). The user provides the information specified in this element via the Defense Center user interface. Each `module` element may contain only one direct child `config_template` element and this element applies to all instances that are configured.

Note, however, that each `remediation_type` element in `module.templ` ate can also contain a child `config_template` element. The `config_template` child element under `remediation_type` allows you to define information that the user must provide for each of the different remediation types. So a user will have to configure general instance-level fields using the `config_template` element in the `module` portion, and then, optionally, an additional set of `config_template` fields specific to the remediation type being executed by the instance. For more information, see [Defining Remediation Types](#) on page 56.

The following diagram illustrates the child elements available to the config\_template element.



The config\_template element allows you to render several basic field types in the web interface. You choose which config\_template child elements to use depending on the data you need to collect from the user for the remediation module. All child elements of config\_template are optional and can be used as many times as needed within a config\_template element. Fields are rendered on the web interface in the order in which they are included in the config\_template element.

See the following sections for more information on the child elements that represent the fields you can use to collect configuration information on the instance configuration and remediation configuration pages in the web interface:

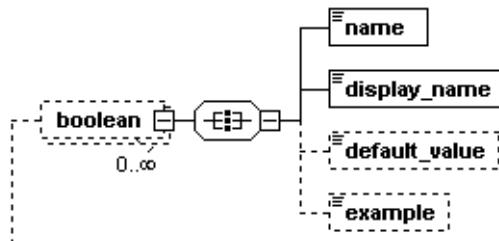
- [The boolean Element](#) on page 36
- [The integer Element](#) on page 37
- [The string Element](#) on page 39

- The password Element on page 41
- The ipaddress Element on page 43
- The netmask Element on page 45
- The host Element on page 46
- The network Element on page 48
- The enumeration Element on page 50
- The list Element on page 51

## The boolean Element

Each boolean element you use in a configuration template represents a true/false choice, which appears as a set of radio buttons labeled **On** or **Off**; users can make in the web interface. If you set the element's required attribute to false, an additional radio button is available, labeled **Not Selected**.

The following portion of the module template schema diagram illustrates the boolean element's child elements.



When configuring child elements for an occurrence of a boolean element, you may only use each available child element once. The [boolean Attributes and Child Elements](#) table describes the child elements available to the boolean element.

#### boolean Attributes and Child Elements

NAME	TYPE	DESCRIPTION	REQUIRED?
required	attribute	Indicates whether specifying a value in the field is optional.  This attribute defaults to true. You are not required to use this attribute. Therefore, if you do not use it (or if you explicitly set its value to true), users must select either <b>On</b> or <b>Off</b> . If you set the value of the attribute to false, the web interface indicates that the choice is optional.	no
name	element	Provides context to the remediation module for the value entered in the field. Names may not contain white space and may only contain alphanumeric characters and the underscore (_) and dash (-) character. Names should be unique within a module.	yes
display_name	element	Specifies the web interface label for this field.	yes
default_value	element	Specifies the default value for this field. If the web interface user does not specify a value, the remediation program uses this value by default.	no

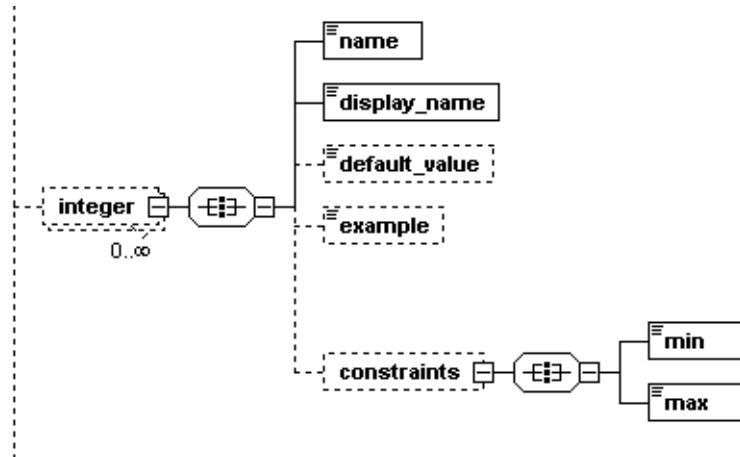
The following portion of a config\_template element definition indicates that the web interface displays a field labeled "Enabled?" that provides user with two choices: **On** or **Off**. The choice defaults to true, that is, the radio button labeled **On** is preselected.

```
<boolean>
  <name>process_enabled</name>
  <display_name>Enabled?</display_name>
  <default_value>true</default_value>
</boolean>
```

## The integer Element

Each integer element you use in a config\_template represents a field in the web interface that accepts an integer value.

The following diagram illustrates the child and grandchild elements of the `integer` element.



The [integer Attributes, Child Elements, and Grandchild Elements](#) table describes the child elements available to the `integer` element.

#### integer Attributes, Child Elements, and Grandchild Elements

NAME	TYPE	DESCRIPTION	REQUIRED?
required	attribute	Indicates whether users must provide a value in the field.  This attribute defaults to <code>true</code> . You are not required to use this attribute. Therefore, if you do not use it (or if you explicitly set its value to <code>false</code> ), users must provide a value. If you set the value of the attribute to <code>false</code> , the web interface indicates that providing a value is optional.	no
name	element	Provides context to the remediation module for the value entered in the field. Names may not contain white space and may only contain alphanumeric characters and the underscore ( <code>_</code> ) and dash ( <code>-</code> ) character. Names should be unique within a module.	yes
display_name	element	Specifies the web interface label for this field.	yes

integer Attributes, Child Elements, and Grandchild Elements (Continued)

NAME	TYPE	DESCRIPTION	REQUIRED?
default_value	element	Specifies the default value for this field. If the web interface user does not specify a value, the remediation program uses this value by default.	no
example	element	Provides an example of the input that the remediation module expects to receive.  <b>IMPORTANT!</b> This value is not displayed in the web interface.	no
constraints	element	Constrains the values that the user can enter in this field to fall between specified minimum and maximum values, inclusive.  The <code>constraints</code> element has two child elements: <code>min</code> and <code>max</code> . Each is an optional, single-occurrence child element that accepts an integer value.	no

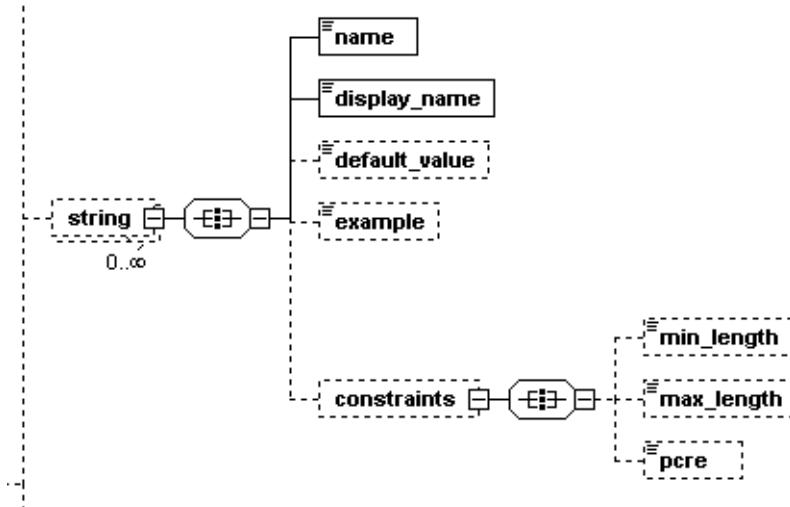
The following portion of a `config_template` element definition indicates that the web interface displays a field labeled “Rate,” which accepts an integer value between 0 and 500 but defaults to 430.

```
<integer>
  <name>rate</name>
  <display_name>Rate</display_name>
  <default_value>430</default_value>
  <constraints>
    <min>0</min>
    <max>500</max>
  </constraints>
</integer>
```

## The string Element

Each `string` element you use in a `config_template` represents a field in the web interface that accepts a string value.

The following diagram illustrates the child elements of the string element instance.



The [string Attributes, Child Elements, and Grandchild Elements](#) table describes child elements available to the string element.

#### string Attributes, Child Elements, and Grandchild Elements

NAME	TYPE	DESCRIPTION	REQUIRED?
required	attribute	Indicates whether users must provide a value in the field.  This attribute defaults to true. You are not required to use this attribute. Therefore, if you do not use it (or if you explicitly set its value to false), users must provide a value. If you set the value of the attribute to false, the web interface indicates that providing a value is optional.	no
name	element	Provides context to the remediation module for the value entered in the field. Names may not contain white space and may only contain alphanumeric characters and the underscore (_) and dash (-) character. Names should be unique within a module.	yes
display_name	element	Specifies the web interface label for this field.	yes

string Attributes, Child Elements, and Grandchild Elements (Continued)

NAME	TYPE	DESCRIPTION	REQUIRED?
default_value	element	Specifies the default value for this field. If the web interface user does not specify a value, the remediation program uses this value by default.	no
example	element	Provides an example of the input that the remediation module expects to receive.  <b>IMPORTANT!</b> This value is not displayed in the web interface.	no
constraints	element	Constrains the values that the user can enter in this field.  The constraints element has three child elements: min_Length, max_Length and pcre. The min_Length and max_Length elements are optional, single-occurrence child elements that accept integer values and specify a range for the acceptable length of string values. The pcre element is optional; use it to specify a Perl-compatible regular expression that provides additional constraints.	no

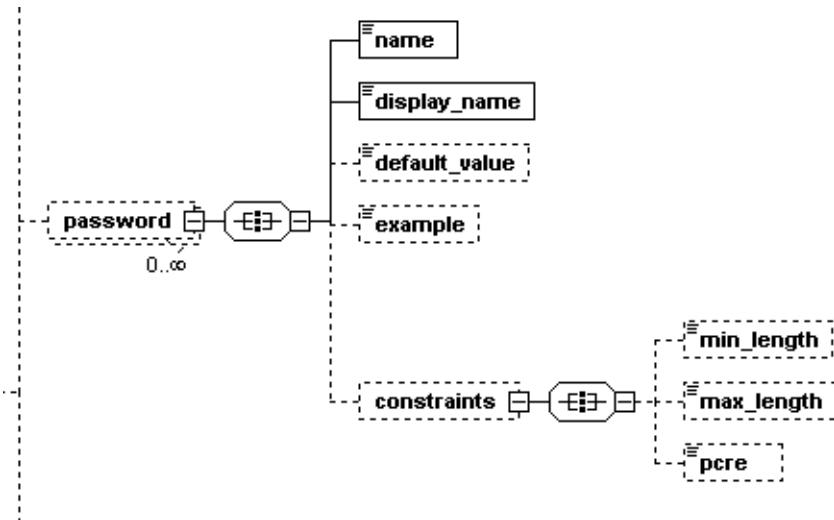
The following portion of a config\_template element definition indicates that the web interface displays a field labeled “Username,” which accepts a string value that is at least eight characters long and does not use white spaces.

```
<string>
  <name>user_name</name>
  <display_name>Username</display_name>
  <constraints>
    <min_length>8</min_length>
    <pcre>\S+</pcre>
  </constraints>
</string>
```

## The password Element

Each password element you use in a config\_template represents a field in the web interface that accepts a string comprised of alphanumeric characters.

The following diagram illustrates the child and grandchild elements of the password element instance.



The [password Attributes, Child Elements, and Grandchild Elements](#) table describes the child elements available to the password element.

#### password Attributes, Child Elements, and Grandchild Elements

NAME	TYPE	DESCRIPTION	REQUIRED?
required	attribute	<p>Indicates whether users must provide a value in the field.</p> <p>This attribute defaults to <code>true</code>. You are not required to use this attribute. Therefore, if you do not use it (or if you explicitly set its value to <code>true</code>), users must provide a value. If you set the value of the attribute to <code>false</code>, the web interface indicates that providing a value is optional.</p>	no
name	element	<p>Provides context to the remediation module for the value entered in the field. Names may not contain white space and may only contain alphanumeric characters and the underscore (<code>_</code>) and dash (<code>-</code>) character. Names should be unique within modules.</p>	yes
display_name	element	Specifies the web interface label for this field.	yes

password Attributes, Child Elements, and Grandchild Elements (Continued)

NAME	TYPE	DESCRIPTION	REQUIRED?
default_value	element	Specifies the default value for this field. If the web interface user does not specify a value, the remediation program uses this value by default.	no
example	element	Provides an example of the input that the remediation module expects to receive.  <b>IMPORTANT!</b> This value is not displayed in the web interface.	no
constraints	element	Constrains the values that the user can enter in this field.  The constraints element has three child elements: min_Length, max_Length and pcre. The min_Length and max_Length elements are optional, single-occurrence child elements that accept integer values and specify a range for the acceptable length of password values. The pcre element is optional; use it to specify a Perl-compatible regular expression that provides additional constraints.	no

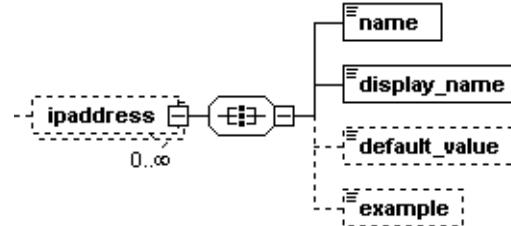
The following portion of a config\_template element definition indicates that the web interface displays a field labeled “Login Password”, which accepts an alphanumeric string between 6 and 12 characters long.

```
<password>
  <name>Login_Password</name>
  <display_name>Login Password</display_name>
  <constraints>
    <min_length>6</min_length>
    <max_length>12</max_length>
  </constraints>
</password>
```

## The ipaddress Element

Each ipaddress element you use in a config\_template represents a field in the web interface that accepts a single IP address. IP addresses may be entered in the form of a fully formed dotted quad (for example, 1.1.1.1).

The following diagram illustrates the child elements of the `i_paddress` element.



When configuring child elements for an occurrence of an `i_paddress` element, you may only use each available child element once. The [ipaddress Attributes and Child Elements](#) table describes the child elements available to the `i_paddress` element.

#### ipaddress Attributes and Child Elements

NAME	TYPE	DESCRIPTION	REQUIRED?
required	attribute	Indicates whether users must provide a value in the field.  This attribute defaults to <code>true</code> . You are not required to use this attribute. Therefore, if you do not use it (or if you explicitly set its value to <code>true</code> ), users must provide a value. If you set the value of the attribute to <code>false</code> , the web interface indicates that providing a value is optional.	no
name	element	Provides context to the remediation module for the value entered in the field. Names may not contain white space and may only contain alphanumeric characters and the underscore ( <code>_</code> ) and dash ( <code>-</code> ) character. Names should be unique within modules.	yes
display_name	element	Specifies the web interface label for this field.	yes
default_value	element	Specifies the default value for this field.	no
example	element	Provides an example of the input that the remediation module expects to receive.  <b>IMPORTANT!</b> This value is not displayed in the web interface.	no

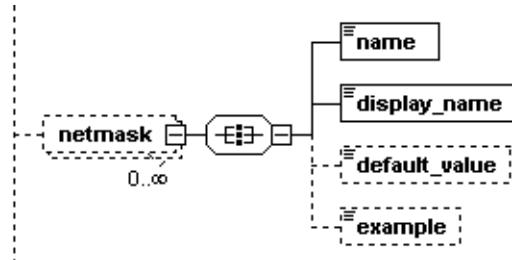
The following portion of a config\_template element definition indicates that the web interface displays a field labeled “Mail Server,” which accepts a single IP address.

```
<i paddress>
  <name>mail_server</name>
  <display_name>Mail Server</display_name>
</i paddress>
```

## The netmask Element

Each netmask element you use in a config\_template represents a field in the web interface that accepts netmask values. Netmask values can be denoted by a dotted quad (255.255.255.255) or a CIDR mask (/8).

The diagram illustrates the child elements of the netmask element.



When configuring child elements for an occurrence of a netmask element, you may only use each available child element once. The [netmask Attributes and Child Elements](#) table describes the child elements available to the netmask element.

#### netmask Attributes and Child Elements

NAME	TYPE	DESCRIPTION	REQUIRED?
required	attribute	Indicates whether users must provide a value in the field.  This attribute defaults to <code>true</code> . You are not required to use this attribute. Therefore, if you do not use it (or if you explicitly set its value to <code>true</code> ), users must provide a value. If you set the value of the attribute to <code>false</code> , the web interface indicates that providing a value is optional.	no
name	element	Provides context to the remediation module for the value entered in the field. Names may not contain white space and may only contain alphanumeric characters and the underscore ( <code>_</code> ) and dash ( <code>-</code> ) character. Names should be unique within modules.	yes
display_name	element	Specifies the web interface label for this field.	yes
default_value	element	Specifies the default value for this field. If the web interface user does not specify a value, the remediation program uses this value by default.	no
example	element	Provides an example of the input that the remediation module expects to receive.  <b>IMPORTANT!</b> This value is not displayed in the web interface.	no

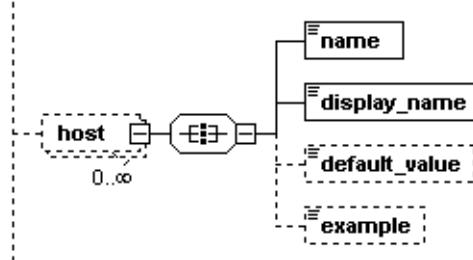
The following portion of a `config_template` element definition indicates that the web interface displays a field labeled "Netmask," which accepts netmask values denoted by a dotted quad or CIDR mask and defaults to 255.255.255.255.

```
<netmask>
    <name>netmask</name>
    <display_name>Netmask</display_name>
    <default_value>255.255.255.0</default_value>
</netmask>
```

## The host Element

Each host element you use in a `config_template` represents a field in the web interface that accepts a single IP address or string.

The following diagram illustrates the child elements of the host element.



When configuring child elements for an occurrence of a host element, you may only use each available child element once. The [host Attributes and Child Elements](#) table describes the child elements and attributes available to the host element.

#### host Attributes and Child Elements

NAME	TYPE	DESCRIPTION	REQUIRED?
required	attribute	Indicates whether users must provide a value in the field.  This attribute defaults to true. You are not required to use this attribute. Therefore, if you do not use it (or if you explicitly set its value to true), users must provide a value. If you set the value of the attribute to false, the web interface indicates that providing a value is optional.	no
name	element	Provides context to the remediation module for the value entered in the field. Names may not contain white space and may only contain alphanumeric characters and the underscore (_) and dash (-) character. Names should be unique within modules.	yes
display_name	element	Specifies the web interface label for this field.	yes
default_value	element	Specifies the default value for this field. If the web interface user does not specify a value, the remediation program uses this value by default.	no
example	element	Provides an example of the input that the remediation module expects to receive.  <b>IMPORTANT!</b> This value is not displayed in the web interface.	no

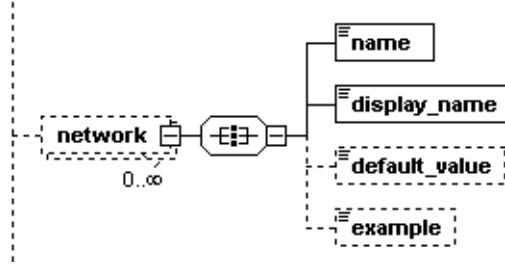
The following portion of a config\_template element definition indicates that the web interface displays a field labeled “Host Name,” which accepts an IP address or string. The web interface also provides example text of “192.10.1.3.”

```
<host>
    <name>hostname</name>
    <display_name>Host Name</display_name>
    <example>192.10.1.3</example>
</host>
```

## The network Element

Each network element you use within a config\_template represents a field in the web interface. A network field accepts an IP address (assumed to be a single IP address, that is, an IP address with /32 netmask) or a CIDR block.

The following diagram illustrates the child elements of the network element.



When configuring child elements for an occurrence of a network element, you may only use each available child element once. The [network Attributes and Child](#)

[Elements](#) table describes the child elements and attributes available to the network element.

#### network Attributes and Child Elements

NAME	TYPE	DESCRIPTION	REQUIRED?
required	attribute	<p>Indicates whether users must provide a value in the field.</p> <p>This attribute defaults to <code>true</code>. You are not required to use this attribute. Therefore, if you do not use it (or if you explicitly set its value to <code>true</code>), users must provide a value. If you set the value of the attribute to <code>false</code>, the web interface indicates that providing a value is optional.</p>	no
name	element	Provides context to the remediation module for the value entered in the field. Names may not contain white space and may only contain alphanumeric characters and the underscore (_) and dash (-) character. Names should be unique within modules.	yes
display_name	element	Specifies the web interface label for this field.	yes
default_value	element	Specifies the default value for this field. If the web interface user does not specify a value, the remediation program uses this value by default.	no
example	element	<p>Provides an example of the input that the remediation module expects to receive.</p> <p><b>IMPORTANT!</b> This value is not displayed in the web interface.</p>	no

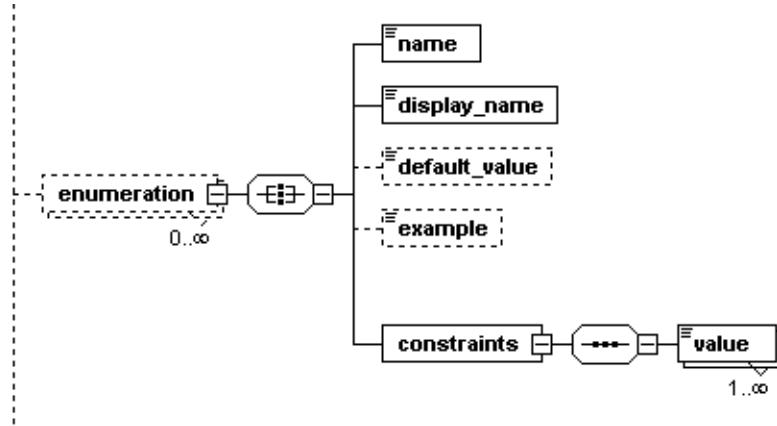
The following portion of a `config_template` element definition indicates that the web interface displays a field labeled "Monitored Network," which accepts either a /32 IP address or an IP address and netmask value, and which has a default value of 192.168.1.0/24.

```
<network>
    <name>monitored_network</name>
    <display_name>Monitored Network</display_name>
    <default_value>192.168.1.0/24</default_value>
</network>
```

## The enumeration Element

Each enumeration element you use in a config\_template represents a drop-down list of strings displayed in the web interface. Users can select a single value from this list.

The following diagram illustrates the child and grandchild elements of the enumeration element.



The [enumeration Attributes, Child Elements, and Grandchild Elements](#) table describes the child elements and attributes available to the enumeration element.

### enumeration Attributes, Child Elements, and Grandchild Elements

NAME	TYPE	DESCRIPTION	REQUIRED?
required	attribute	Indicates whether users must provide a value in the field.  This attribute defaults to true. You are not required to use this attribute. Therefore, if you do not use it (or if you explicitly set its value to false), users must provide a value. If you set the value of the attribute to false, the web interface indicates that providing a value is optional.	no
name	element	Provides context to the remediation module for the value entered in the field. Names may not contain white space and may only contain alphanumeric characters and the underscore (_) and dash (-) character. Names should be unique within modules.	yes
display_name	element	Specifies the web interface label for this field.	yes

enumeration Attributes, Child Elements, and Grandchild Elements (Continued)

NAME	TYPE	DESCRIPTION	REQUIRED?
default_value	element	Specifies the default value for this field. If the web interface user does not specify a value, the remediation program uses this value by default.	no
example	element	Provides an example of the input that the remediation module expects to receive.  <b>IMPORTANT!</b> This value is not displayed in the web interface.	no
constraints	element	Specifies the values that the user can enter in this field.  The constraints element has one required child element, value, that accepts a string that represents one choice for the users. Use multiple value elements to provide multiple choices to the user.	yes

The following portion of a config\_template element definition indicates that the web interface displays a field labeled "Day", which allows users to select one of the values provided (Monday, Tuesday, Wednesday, Thursday, and Friday).

```

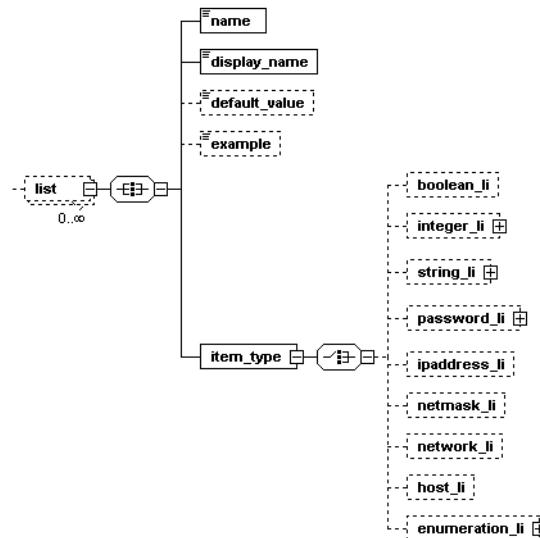
<enumeration>
    <name>day</name>
    <display_name>Day</display_name>
    <constraints>
        <value>Monday</value>
        <value>Tuesday</value>
        <value>Wednesday</value>
        <value>Thursday</value>
        <value>Friday</value>
    </constraints>
</enumeration>

```

## The list Element

Each list element you use in a config\_template represents a field in the web interface that allows users to enter a list of values, one per line, whose type is specified by the required item\_type child element.

The following diagram illustrates the child and grandchild elements of the `list` element.



The [list Attribute and Child Elements](#) table describes the child elements available to the `list` element.

#### list Attribute and Child Elements

NAME	TYPE	DESCRIPTION	REQUIRED?
required	attribute	Indicates whether users must provide a value in the field.  This attribute defaults to <code>true</code> . You are not required to use this attribute. Therefore, if you do not use it (or if you explicitly set its value to <code>true</code> ), users must provide a value. If you set the value of the attribute to <code>false</code> , the web interface indicates that providing a value is optional.	no
name	element	Provides context to the remediation module for the value entered in the field. Names may not contain white space and may only contain alphanumeric characters and the underscore ( <code>_</code> ) and dash ( <code>-</code> ) character. Names should be unique within modules.	yes
display_name	element	Specifies the web interface label for this field.	yes

list Attribute and Child Elements (Continued)

NAME	TYPE	DESCRIPTION	REQUIRED?
default_value	element	Specifies the default value for this field. If the web interface user does not specify a value, the remediation program uses this value by default.	no
example	element	Provides an example of the input that the remediation module expects to receive.  <b>IMPORTANT!</b> This value is not displayed in the web interface.	no
item_type	element	Specifies the type of value that can appear in this field. The value type is specified by a child element. Valid child elements are listed below.	no

The following list describes the child elements available to the `item_type` element, which are similar to the child elements of the `config_template` element; the only difference is that `item_type` child elements do not use the `required` attribute. Each instance of the `item_type` element can use only one child element:

- `boolean_list` indicates that the list accepts multiple Boolean values (see [The boolean Element](#) on page 36).
- `integer_list` indicates that the list accepts multiple integer values (see [The integer Element](#) on page 37).
- `string_list` indicates that the list accepts multiple string values (see [The string Element](#) on page 39).
- `password_list` indicates that the list accepts multiple password values (see [The password Element](#) on page 41).
- `ipaddress_list` indicates that the list accepts multiple ipaddress values (see [The ipaddress Element](#) on page 43).
- `network_list` indicates that the list accepts multiple network values (see [The network Element](#) on page 48).
- `netmask_list` indicates that the list accepts multiple netmask values (see [The netmask Element](#) on page 45).
- `host_list` indicates that the list accepts multiple host values (see [The host Element](#) on page 46).
- `enumeration_list` indicates that the list accepts multiple values as defined by the `value` child elements of the `enumeration_list` element's `constraints` child element (see [The enumeration Element](#) on page 50).

The following portion of a config\_template element definition indicates that the web interface should allow the user to provide a list of integers between zero and 500 inclusive, one per line, in a field labeled "Integer List".

```
<list>
    <name>List_integer</name>
    <display_name>Integer List</display_name>
    <example>Constrained value [0-500]</example>
    <item_type>
        <integer_list>
            <constraints>
                <min>0</min>
                <max>500</max>
            </constraints>
        </integer_list>
    </item_type>
</list>
```

## Sample Configuration Template

This section provides a sample config\_template element definition, which governs both the web interface appearance and the types of information the remediation module must receive from the user.

```
<config_template>
    <ipaddress>
        <name>host_ip</name>
        <display_name>Host IP</display_name>
    </ipaddress>
    <string>
        <name>user_name</name>
        <display_name>Username</display_name>
    </string>
    <password>
        <name>login_password</name>
        <display_name>Connection Password</display_name>
    </password>
    <password>
        <name>root_password</name>
        <display_name>Enable Password</display_name>
    </password>
</config_template>
```

The above template renders four fields on the web interface. The [Fields Created by the Sample Configuration Template](#) table describes each field.

**Fields Created by the Sample Configuration Template**

FIELD	DESCRIPTION
Host IP	Accepts an IP address that the remediation module identifies as <code>host_ip</code> .
Username	Accepts a string that the remediation module identifies as <code>user_name</code> .
Connection Password	Accepts an alphanumeric password string that the remediation module identifies as <code>login_password</code> .
Enable Password	Accepts an alphanumeric password string that the remediation module identifies as <code>root_password</code> .

The following screen illustrates how these fields appear on the web interface. Users must provide the data requested by these fields to configure the remediation module from the web interface.

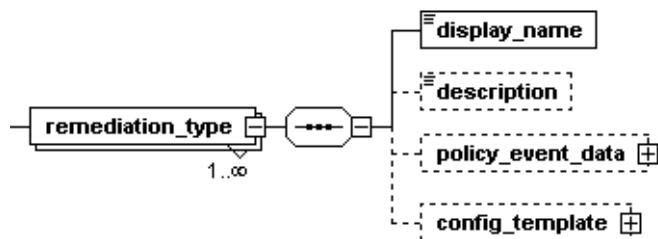
Edit Instance

Instance Name	<input type="text"/>
Module	Module Name (v1.0)
Description	<input type="text"/>
Host IP	<input type="text"/>
Server Port	<input type="text"/>
Username	<input type="text"/>
Connection Password <small>Retype to confirm</small>	<input type="password"/>
Enable Password <small>Retype to confirm</small>	<input type="password"/>
	<input type="button" value="Create"/> <input type="button" value="Cancel"/>

## Defining Remediation Types

Remediation types describe the actions, or *remediations*, taken by the device that is governed by the remediation module. Each `remediation_type` element you use in `module.templ ate` represents one of those remediations. Remediations are triggered by correlation event data from the remediation subsystem. For more information see [Event Data](#) on page 14.

The following diagram illustrates the child elements of the `remediation_type` element.



The [remediation\\_type Attributes and Child Elements](#) table describes the attributes and child elements available to the `remediation_type` element.

### remediation\_type Attributes and Child Elements

NAME	TYPE	DESCRIPTION	REQUIRED?
name	attribute	Provides context to the remediation module for the remediation type.  This attribute is required and accepts a string between 1 and 64 characters, inclusive. Names may not contain white space and may only contain alphanumeric characters and the underscore (_) and dash (-) character. <code>remediation_type</code> names must be unique within each module.	yes

remediation\_type Attributes and Child Elements (Continued)

NAME	TYPE	DESCRIPTION	REQUIRED?
display_name	element	Labels the remediation type on the web interface.	yes
policy_event_data	element	<p>Specifies the correlation event data that the remediation module needs to receive from the remediation subsystem.</p> <p>The policy_event_data has one child element, pe_item, that represents a specific correlation event data item. Use multiple pe_item elements to provide multiple correlation event data items. For more information on appropriate correlation event data values, see <a href="#">Event Data</a> on page 14.</p>	no
config_template	element	Specifies the information the user must provide when configuring an instance of this remediation module. For more information, see <a href="#">Defining the Configuration Template</a> on page 34.	no

The following portion of a module template file illustrates several remediation\_type element definitions.

```
<remediation_type name="block_src">
    <display_name>Block Source</display_name>
    <policy_event_data>
        <item>src_ip_addr</item>
        <item>src_port</item>
        <item>src_protocol </item>
    </policy_event_data>
</remediation_type>
<remediation_type name="block_dest">
    <display_name>Block Destination</display_name>
    <policy_event_data>
        <item>dest_ip_addr</item>
        <item>dest_port</item>
        <item>dest_protocol </item>
    </policy_event_data>
</remediation_type>
<remediation_type name="acl_insert">
    <display_name>ACL Insertion</display_name>
    <policy_event_data>
        <item>src_ip_addr</item>
        <item>src_port</item>
        <item>src_protocol </item>
        <item>dest_ip_addr</item>
        <item>dest_port</item>
        <item>dest_protocol </item>
    </policy_event_data>
    <config_template>
        <integer>
            <name>acl_num</name>
            <display_name>ACL Number</display_name>
        </integer>
    </config_template>
</remediation_type>
```

The example above contains 3 remediation types: block\_src, block\_dest, and acl\_insert. Each of these requires specific correlation event (pe\_item) data. The acl\_insert remediation type also requires configuration data, which is specified in its config\_template child element; users must provide an ACL number when they configure instances of that type.

## Defining Exit Statuses

The remediation subsystem expects to receive an exit status, or return code, in the form of an integer from your remediation module.

Sourcefire provides a set of predefined exit status messages your remediation module can return. You can return predefined exit statuses, which correspond to integer values between 1 and 128, inclusive. The [Predefined Exit Statuses](#) table lists and describes these predefined exit status codes.

### Predefined Exit Statuses

Exit Status	Description
0	Successful completion of remediation.
1	Error in the input provided to the remediation module.
2	Error in the remediation module configuration.
3	Error logging into the remote device or server.
4	Unable to gain required privileges on remote device or server.
5	Timeout logging into remote device or server.
6	Timeout executing remote commands or servers.
7	The remote device or server was unreachable.
8	The remediation was attempted but failed.
10	A white-list match was found.
11	Failed to execute remediation program
20	Unknown/unexpected error.

Alternatively, your module may return integers between 129 and 254, inclusive, as custom exit statuses. If your remediation module returns custom exit statuses, you must define the set of exit statuses it can return. Each `exit_status` element you use in `module.template` represents a custom exit status that your remediation module can return. For more information, see [Data Returned by Modules](#) on page 27.

The `exit_status` element accepts a string that describes a return code. In addition, the element requires an attribute, `value`, that accepts a unique integer

between 129 and 255. This attribute associates remediation module return codes with their descriptions, which the user can see in remediation status event views.

The following example illustrates valid custom exi t\_status elements.

```
<exi t_status val ue="138">syslog error</exi t_status>
<exi t_status val ue="139">unknown error</exi t_status>
```

# CHAPTER 4

# WORKING WITH THE REMEDIATION SDK

## Understanding the Remediation SDK

In addition to deploying Sourcefire-provided remediation modules, you can install and run your own custom remediations to automate responses to violations of associated correlation policies. Sourcefire provides a software developer kit (SDK) that you can download from the Support Site to help you get started.

### Purpose of the SDK

Using the SDK and the information in this chapter of the Sourcefire *Remediation API Guide*, you can:

- Practice deploying a simple remediation module to gain familiarity with the process. Installation, configuration, and removal are easy.
- Inspect the source code of a remediation program to see one way to use the API to interact with the remediation subsystem and perform multiple remediation functions.

---

**IMPORTANT!** The syslog module in the SDK is not intended for production use.

---

Please note that you can use the Sourcefire-provided modules already loaded on the Defense Center as a reference resource while you develop. All of these modules are accessible at `/var/sf/remediation_modules` on the Defense Center. Each installed module has a `.tgz` package in this directory. For information on the modules, see [Sourcefire-Provided Remediation Modules](#) on page 7.

## Description of the SDK

The remediation SDK has a syslog alert remediation module in two versions, Perl and C. To use it, you need a syslog server running and receiving remote traffic.

The module provides two remediation types:

- **Si mpl e\_Noti fi cati on** - generates syslog alerts with the source IP address, source port (if available), and IP protocol (if available) for the triggering event.
- **Compl ete\_Noti fi cati on** - generates a syslog alert with the same fields as the simple notification, and also includes the destination IP address, destination port, and a severity indicator for the triggering event.

As with all remediation modules, you enter a small amount of configuration in the web interface to add instances of the module. Each instance targets a particular device on your network (in this case a syslog server) and runs the remediation for the instance. To run the **Compl ete\_Noti fi cati on** remediation type, you select a syslog facility level not required for the **Si mpl e\_Noti fi cati on** remediation type.

See the following table for a list of the Perl version files.

### Sample Perl Module

INCLUDED FILES	DESCRIPTION
syslog.pl	The program that executes the syslog alert when the correlation policy associated it with is violated.
module.template	Module configuration file. Defines required event data, required information to collect in the web interface when users create instances, and other essential setup parameters.
Makefile	Sample makefile to package the files in a remediation module for installation on the Defense Center.

See the [Sample C Module table](#) on page 62 for a list of the C version files.

### Sample C Module

INCLUDED FILES	DESCRIPTION
syslogc.c	The program that executes the syslog alert when the correlation policy associated it with is violated.
module.template	Module configuration file. Defines required event data, required information to collect in the web interface when users create instances, and other essential setup parameters.

## Downloading the SDK

To download the remediation SDK:

1. Access the support website at <https://support.sourceforge.com/downloads>.
2. Select a software version, then under Product Category, select **Software**. The download link for the remediation SDK is in the **api** portion of the page.
3. Unpack the .zip file in a convenient folder on your client machine.

## Overview of the Development and Installation Process

The steps below form a checklist of tasks that need to be performed to create, install, and configure a custom remediation module. Some of the steps involve procedural and descriptive details that are explained in cross-referenced sections of the *Remediation API Guide* or the *Sourcefire 3D System User Guide*.

To develop, install, and configure a custom remediation module, you must:

1. Identify the condition you want to mitigate and the actions that appropriately resolve the detected condition in your environment.
2. Familiarize yourself with data elements that can be obtained from the remediation subsystem. See [Data Available from the Remediation Subsystem](#) on page 14 for definitions of all available fields that the Defense Center can provide for your remediation.  
You should also understand the return code functionality built into the remediation subsystem. See [Defining Exit Statuses](#) on page 59 for information.
3. Generate a high-level design that identifies all the remediation actions (remediation types) that your program needs to address.
4. Write your remediation program so that it addresses all the functions necessary for the desired remediations. Remediation module programs may be written in bash, tsch, Perl or C. Develop your program using the technical guidance provided in [Notes for Remediation Program Developers](#) on page 64.
5. Create the `modul.e.templ ate` file for your remediation module. For an understanding of the data elements and syntax of `modul.e.templ ate`, see the chapter [Communicating with the Remediation Subsystem](#) on page 30.  
You can save time by editing an existing `modul.e.templ ate` file to start with.
6. Package your remediation module as described in [Packaging Your Module](#) on page 28.
7. Install the module on the Defense Center using the Policy and Response component as described in [Installing Your Module](#) on page 28. You will load the package on the Defense Center and proceed as if you were configuring one of the Sourcefire-provided modules.

8. Ensure that the individual remediation types in your remediation module are assigned as responses to the correct correlation rules in your defined correlation policies. See the *Sourcefire 3D System User Guide* for procedure details.

## Notes for Remediation Program Developers

When you have defined the required scope and functionality of your remediation program and understood the data elements available for your remediation actions, you can write the remediation program.

Remediation module programs may be written in bash, tsch, Perl or C.

The [Programmer Notes](#) table indicates where to find information on topics of interest.

### Programmer Notes

TO LEARN MORE ABOUT...	LOOK IN...
the file structure and workflow environment of the remediation subsystem	<a href="#">Understanding the Remediation Subsystem File Structure</a> on page 65
implementing multiple remediation types in a remediation program	<a href="#">Implementing Remediation Types in a Remediation Program</a> on page 65
the remediation subsystem file structure	<a href="#">Understanding the Remediation Subsystem File Structure</a> on page 65
the interactions of the remediation program and the Defense Center remediation subsystem	<a href="#">Understanding the Remediation Program Workflow</a> on page 66
the order in which parameters are passed from the Defense Center to the remediation module	<a href="#">The Order of Command Line Parameters</a> on page 67
how the remediation daemon handles undefined data elements	<a href="#">Handling Undefined Data Elements</a> on page 67
return codes from the remediation program	<a href="#">Handling Return Codes</a> on page 68
runtime modes for the remediation program	<a href="#">Important Global Configuration Elements</a> on page 68
alternative encoding of user input	<a href="#">Important Global Configuration Elements</a> on page 68

## Implementing Remediation Types in a Remediation Program

The remediation daemon on the Defense Center specifies the remediation name as the first argument on the command line when it launches the remediation program. The code snippet below from the SDK Perl program, `syslog.pl`, shows one way your program can branch to the appropriate remediation function. The program runs either `SimpleNotification()` or `ComprehensiveNotification()`, based on the content of `$remediation_config`, which is set by the first field from the remediation daemon. The sample also shows the use of return codes which are discussed in [Handling Return Codes](#) on page 68.

```
# Call the appropriate function for the remediation type
my $rval = 0;
if($remediation_config->{type} eq "Simple_Notification")
{
    $rval = SimpleNotification($instance_config,
        $remediation_config,
        \@pe_event_data);
}
elsif($remediation_config->{type} eq
    "Comprehensive_Notification")
{
    $rval =
    ComprehensiveNotification($instance_config, $remediation_config,
        \@pe_event_data);
}
else
{
    warn "Invalid remediation type. Check your instance.conf\n";
    exit(CONFIG_ERR);
}
exit($rval);
```

You declare the names of all remediation types in the `module.template` file, and associate remediation types with each instance as you add the instance via the web interface. The remediation type that is executed by the instance is recorded in the `instance.config` file which is stored in the `instance.config` subdirectory described in [Understanding the Remediation Subsystem File Structure](#) on page 65.

## Understanding the Remediation Subsystem File Structure

The root directory of each remediation module is derived from the remediation module name and version number, both of which are declared in the `module.template` file. See [The config Element](#) on page 23 for details on the elements of `module.template`.

If you install a module packaged in syslog.tgz with the name syslog and version 1.0 in module\_template, the system puts the module in the following directory: /var/sf/remediation/syslog\_1.0. That directory contains the module\_template file and the remediation program binary for the module.

When you add an instance of the remediation and name the instance log\_tokyo, the system creates the following directory:

/var/sf/remediation/syslog\_1.0/log\_tokyo

and places a file named instance.conf in it. The instance.conf file, which is in XML format, contains the configuration information for the log\_tokyo instance.

The following Linux command sequence illustrates the directory structure described above.

```
# cd /var/sf/remediations
# ls
NMap_perl_2.0 SetAttribute_1.0           ci_sco_px_1.0
ci_sco_ios_router_1.0 syslog_perl_0.1
# cd syslog_perl_0.1
# ls
log_chicago log_tokyo module_template syslog.pl
# cd log_tokyo
# ls
# instance.conf
```

Note that the instance.conf file contains the name of the remediation type that the log\_tokyo instance runs. In the above example, the user who added the log\_tokyo instance could have configured it to run either remediation type defined for the syslog remediation module: Simple\_Notification or Complete\_Notification.

For details on the elements in the instance.conf XML file, see [Instance Configuration Data](#) on page 22.

## Understanding the Remediation Program Workflow

When the Defense Center executes a remediation instance, the remediation daemon launches the remediation program from the instance subdirectory and supplies data from the instance.conf file to the remediation program as command line arguments.

An example will illustrate the process. If a policy violation launches a syslog instance named log\_tokyo, which calls the remediation named Simple\_Notification with a source IP address of 1.1.1.1 and a destination IP address of 2.2.2.2, the Defense Center sets the working directory to /var/sf/remediations/Syslog\_1.0/log\_tokyo (that is, the instance.conf

subdirectory) and executes the remediation binary, `syslog.pl`. The daemon's command line syntax will be as follows:

```
.../syslog.pl Simple_Notification 1.1.1.1 2.2.2.2
```

Note in particular that the `syslog.pl` executable is in the parent directory of the `instance.conf` subdirectory.

When the command runs in this way, the `syslog.pl` binary can load the information in `instance.conf` file because it is in the current directory. If the binary needs to load any modules or other files in the parent directory (`/var/sf/remediations/Syslog_1.0` in this case), the code must explicitly load them from the parent directory; that is, it must provide a path starting with "...". Otherwise the binary will not be able to find the files that it needs.

In Perl, you can also deal with this issue using the `lib()` function as follows:

```
use lib("../../../");
```

Your program must be able to open, read, parse, and close the `instance.conf` file.

## The Order of Command Line Parameters

When the remediation daemon passes event data to your remediation module, it passes the name of the remediation followed by the correlation event data in the order in which the fields are specified in `module.templete`. In `module.templete`, each field to be passed to your module is declared using the `<pe_item>` tag.

If a `pe_item` is set to optional in `module.templete` and is undefined (meaning there is no value for the specific `pe_item`), the remediation daemon passes "undefined" or null to your module. If `pe_item` is set to required in `module.templete` but is undefined, the remediation daemon logs a message to the remediation log stating that no value is available, and does not execute your remediation module binary. You can view the remediation log in the web interface where it is called the Table View of Remediations. See the *Sourcefire 3D System User Guide* for details on how to access and use this view.

## Handling Undefined Data Elements

The remediation daemon handles undefined data items differently, depending on whether an item is marked as optional or required in `module.templete`. Undefined means that the Defense Center database has no value for the item. The daemon's processing is as follows:

- If the undefined `pe_item` is set to optional in `module.templete`, the daemon passes "undefined" or null to your module.
- If the undefined `pe_item` is set to required in `module.templete`, the daemon does not execute the remediation and logs a message to the remediation log stating that no value is available.

## Handling Return Codes

The Defense Center waits for a return code for each instance and records the code in the remediation log. For information on predefined and custom return codes, see [Defining Exit Statuses](#) on page 59.

The Table View of Remediations in the web interface of the Defense Center displays the results of each launched remediation. See the *Sourcefire 3D System User Guide* for information on accessing and using the Table View of Remediations.

## Important Global Configuration Elements

You can enable the remediation API features described in the table below by setting their corresponding elements in the module.template file. For configuration details, see [Defining the Global Configuration](#) on page 32.

Features Enabled in Global Configurations of module.template	
TO ENABLE THIS FEATURE...	SET THIS MODULE.TEMPLATE PARAMETER ...
run remediation program as root	<code>run_as_root</code> <b>WARNING!</b> Sourcefire recommends that you use this element only if absolutely necessary.
HTML-encoding of user input	<code>encode_values</code> <b>IMPORTANT!</b> If you use this element, your remediation module must handle HTML decoding as part of its input handling.

# Index

## A

architecture  
  remediation module 9  
  remediation subsystem 8  
attribute  
  name 56  
  required 40, 42, 44, 46, 47, 49, 50, 52

## B

binary element 33  
boolean element  
  default\_value element 37  
  display\_name element 37  
  name element 37  
  required attribute 37

## C

compliance event data 14  
config element 23  
config\_template element 34, 57

boolean element 36  
enumeration element 50  
host element 46  
integer element 37  
ipaddress element 43  
list element 51  
netmask element 45  
network element 48  
password element 41  
sample definition 54  
string element 39  
configuration template 34  
constraints element 39, 41, 43, 51  
  max element 39  
  max\_length element 41, 43  
  min element 39  
  min\_length element 41, 43  
  pcre element 41, 43  
  value element 51

## D

data  
  compliance event 14  
  from remediation subsystem 14  
  instance configuration 22

# Index

return codes 27  
returned from remediation modules 27  
default\_value element 37, 39, 41, 43, 44, 46, 47, 49, 51, 53  
description element 33  
display\_name element 33, 37, 38, 40, 42, 44, 46, 47, 49, 50, 52, 57

**E**

element  
binary 33  
boolean 36  
config 23  
config\_template 34, 57  
constraints 39, 41, 43, 51  
default\_value 37, 39, 41, 43, 44, 46, 47, 49, 51, 53  
description 33  
display\_name 33, 37, 38, 40, 42, 44, 46, 47, 49, 50, 52, 57  
enumeration 50  
example 39, 41, 43, 44, 46, 47, 49, 51, 53  
global\_config 32  
host 46  
integer 37  
ipaddress 43  
item\_type 53  
list 51  
max 39  
max\_length 41, 43  
min 39  
min\_length 41, 43  
name 37, 38, 40, 42, 44, 46, 47, 49, 50, 52  
netmask 45  
network 48  
password 41  
pcre 41, 43  
policy\_event\_data 57  
remediation 25  
remediation\_type 56  
run\_as\_root 33  
string 39  
value 51  
version 33  
enumeration element 50  
constraints element 51  
default\_value element 51  
display\_name element 50  
example element 51

name element 50  
required attribute 50  
example element 39, 41, 43, 44, 46, 47, 49, 51, 53

## F

file structure, remediation subsystem 65

## G

global configuration 32  
global\_config element 32  
binary element 33  
description element 33  
display\_name element 33  
run\_as\_root element 33  
version element 33

## H

host element 46  
default\_value element 47  
display\_name element 47  
example element 47  
name element 47  
required attribute 47

## I

installing remediation modules 28  
instance configuration data 22  
config element 23  
remediation element 25  
instances  
definition 6  
integer element 37  
constraints element 39  
default\_value element 39  
display\_name element 38  
example element 39

# Index

name element 38  
required attribute 38  
ipaddress element 43  
  default\_value element 44  
  display\_name element 44  
  example element 44  
  name element 44  
  required attribute 44  
item\_type element 53

## L

list element 51  
  default\_value element 53  
  display\_name element 52  
  example element 53  
  item\_type element 53  
  name element 52  
  required attribute 52

## M

max element 39  
max\_length element 41, 43  
min element 39  
min\_length element 41, 43  
module.template  
  config\_template element 34  
  function 30  
  global\_config element 32

## N

name attribute 56  
name element 37, 38, 40, 42, 44, 46, 47, 49, 50, 52  
netmask element 45  
  default\_value element 46  
  display\_name element 46  
  example element 46  
  name element 46  
  required attribute 46  
network element 48  
  default\_value element 49

display\_name element 49  
example element 49  
name element 49  
required attribute 49

## P

packaging remediation modules 28  
password element 41  
  constraints element 43  
  default\_value element 43  
  display\_name element 42  
  example element 43  
  name element 42  
  required attribute 42  
pcre element 41, 43  
policy\_event\_data element 57

## R

remediation element 25  
remediation module  
  architecture 9  
  definition 5  
  installing 28  
  notes for programmers 64  
  packaging 28  
  planning 13  
  pre-defined 7  
  sourcefire-provided 7  
remediation modules  
  how to develop 63  
remediation program 63  
  command line parameters 67  
  option to run as root 68  
  return codes 67  
  sample workflow 66  
Remediation subsystem  
  data provided by 14  
remediation subsystem  
  architecture 8  
  components 8  
  file structure 65  
remediation types 56  
  definition 5  
  sample code 65

## Index

remediation\_type element 56  
  config\_template element 57  
  display\_name element 57  
  name attribute 56  
  policy\_event\_data element 57  
required attribute 37, 38, 40, 42, 44, 46, 47, 49, 50, 52  
resources 11  
return codes 27  
run\_as\_root element 33

## S

sdk 61  
  description 62  
  how to obtain 63  
  notes for programmers 64  
  sample remediation modules 62  
  sourcefire remediation modules as samples 7  
string element 39  
  constraints element 41  
  default\_value element 41  
  display\_name element 40  
  example element 41  
  name element 40  
  required attribute 40

## V

value element 51  
version element 33