FireSIGHT System Host Input API Guide

Version 5.3.1
June 23, 2014

Cisco Systems, Inc.
www.cisco.com
Cisco has more than 200 offices worldwide.
Addresses, phone numbers, and fax numbers
are listed on the Cisco website at
www.cisco.com/go/offices.

CONTENTS

Understanding Host Input

The FireSIGHT System provides two tools for importing data from other sources on your network to augment your network map: the host input API and the host input import tool.

If your organization has the expertise to create Perl scripts, the host input API allows you to script direct data transfer between a third-party application and the network map. For example, you might have a patch management application on your network that contains information about the current patch levels for the hosts on your network. You could import the third-party fix information for each host into the network map. If you set up a map of the names that the third-party application uses for each patch and invoke it before adding the fixes, the system can use that information to update the vulnerability list on each host to deactivate vulnerabilities addressed by the fix. The host input API allows you to create a script that maps third-party data structures to Cisco data structures, so you can re-run the script to import new data as needed, as long as the names of data elements do not change on either side.

If you do not have a programmer available to you, or if you want to import a set of data and do not need to re-run similar imports in the future, you can create a text file containing the data and use the host input import tool to perform the import on the Defense Center using the `nmimport.pl` script.

For example, if you are setting up a new installation of FireSIGHT, you might want to make sure that all the computers listed in your asset management software exist in the network map. You could export the host data from the asset management application, format the results into an appropriately formatted text file, and import the host data using the host input import tool. If the asset management system includes operating system information for each host, you could set up a third-party product map for the asset management system and map each third-party operating system label to the corresponding Cisco label. You can set that map before you run the import, and the system will associate the appropriate Cisco operating system definition with each host.

There are five major steps to setting up a host input API connection with the FireSIGHT System:

Step 1 If you want to perform impact correlation using third-party host data, you can configure third-party product maps to map service, operating system, or fix definitions to Cisco product or fix definitions, using the Defense Center web interface.

Step 2 If you want to import third-party vulnerabilities, you can configure third-party vulnerability maps to map third-party vulnerability identification strings to Cisco vulnerability IDs, using the Defense Center web interface. Note that you can also perform this mapping in your client using the SetCurrent3rdPartyMap API function with the appropriate vulnerability keys.

Step 3 Write a script that imports data to hosts in the network map using the host input API, including calls to invoke third-party product maps as needed.

Step 4 Log in as `admin` on your Defense Center.

Step 5 Run the script to import the data.

There are five major steps to using the host input import tool with the FireSIGHT System:

Step 1 If you want to perform impact correlation using third-party host data, you can configure third-party product maps to map service, operating system, or fix definitions to Cisco product or fix definitions, using the Defense Center web interface.

Step 2 If you want to import third-party vulnerabilities, you can configure third-party vulnerability maps to map third-party vulnerability identification strings to Cisco vulnerability IDs, using the Defense Center web interface. Note that you can also perform this mapping in your import file.

Step 3 Export data from a third-party application and format it to match the formatting guidelines provided in Using the Host Input Import Tool, page 3-1.

Step 4 Run the import tool.

Step 5 To use the import tool, log in as `admin` on your Defense Center. Use the import tool to set the third-party product map. Use the import tool to import data from the import file you created.

Prerequisites

To understand the information in this guide, you should be familiar with the features and nomenclature of the FireSIGHT System and the function of its components (in particular, the network map), and with the various related event data the system generates. Information about these functions, together with definitions of unfamiliar or product-specific terms, may be obtained from the *FireSIGHT System User Guide*. Additional information about the data fields documented in this guide may be obtained from the *User Guide* as well.

Product Version Compatibility

The following table describes the product version required for various host input functionality:

Table 1-1 Product Version Compatibility

Functionality	Product Version
Host input functionality	FireSIGHT System version 4.9+
`ScanUpdate` function, `AddHostAttribute` function, `DeleteHostAttribute` function, `SetSourceType` function, `DeleteScanResult` function, `AddScanResult` function, `ScanFlush` function, setting source types, setting source type IDS	FireSIGHT System version 4.10+
Host input external client functionality	FireSIGHT System version 5.0+
Host input mobile device identification functionality	FireSIGHT System version 5.1+
`DeleteClientApp` function	FireSIGHT System version 5.1.1+
IPv6 address support	FireSIGHT System version 5.2+

Document Conventions

The following table lists the names used in this book to describe the various data field formats employed in host input calls. Numeric constants used by the host input API or host input import tool are typically unsigned integer values. Bit fields use low order bits unless otherwise noted. For example, in a one-byte field that contains five bits of flag data, the low order five bits will contain the data.

Table 1-2 Key Value Data Type Conventions

Data Type	Description
uint	Unsigned integer
uint8	Unsigned 8-bit integer
uint32	Unsigned 32-bit integer
string	Variable length bytes containing character data.
[n]	Array subscript following any of the above data types to indicate n instances of the indicated data type, for example, uint8[4] is an array of four 8-bit elements.
variable	Collection of various data types.

Host Input Scripting Resources

The following describes some of the topics explained in the documentation and where to look for more information.

Table 1-3 Host Input Resources

To learn more about...	Look in...
the host input API	Using the Host Input API, page 2-1
the host input import tool	Using the Host Input Import Tool, page 3-1
writing a script to connect to the host input API	Writing Host Input API Scripts, page 2-1
writing a script to import data using the host input API	Running a Host Input API Script, page 2-3
calling a specific host input API function	Host Input API Functions, page 2-5
guidelines for writing an import file to use with the host input import tool	Writing Host Input Import Files, page 3-3
syntax for a specific host input function to include in an import file	Host Input Import Syntax, page 3-6
running the host input import tool	Running a Host Input Import, page 3-31
installing, configuring, and running the host input reference client	Using the Host Input Reference Client, page 4-2

Using the Host Input API

You can set up a custom Perl application to import data to the network map from a third-party application, using the host input API.

See the following sections for more information:

Writing Host Input API Scripts

This chapter provides details on the syntax used to call each of the functions available using the host input API. When writing your script, make sure you include the elements indicated in the following sections:

You must call the host input module, set the source type, and obtain an application ID in the order indicated above.

Calling the Host Input Module

You must include a `use` statement for the `SF::SFDataCorrelator::HostInput` module, installed on the FireSIGHT System, before calling any host input functions in your script.

Include the following code segment in your script:

```
use SF::SFDataCorrelator::HostInput;
```

See Example: Invoking the Host Input Module, page 2-40 for an example of this command used in a script.

Setting the Source Type

After you declare use of the `HostInput` module, you must identify the source application for the data you import as `"Application"` or `"Scanner"`. The system marks the source for data imported using this designation as `Scanner: source_id` or `Application: source_id`. If you import scan results using this function, be sure to edit the source definition for the input source in your network discovery policy to set the identity source type to Scanner.For more information on setting the application or scanner name, see Obtaining a Source ID, page 2-2.

Include the following code segment in your script:

```
# Set the Source Type
my $source_type_id =
SF::SFDataCorrelator::HostInput::GetSourceTypeIDByName('Application');
```

See Example: Setting the Source Type, page 2-40 for an example of this command used in a script.

Obtaining a Source ID

Applications must set the application (or source) ID using the `SetCurrentSource(name)` function.

Use this syntax for the `SetCurrentSource` function:

```
SF::SFDataCorrelator::HostInput::SetCurrentSource ($source_type_id,"CustomApp");
```

where `"CustomApp"` is the application identification string you want to use to identify the imported data.

Include a code segment similar to the following in your script (using your application name in place of "*CustomApp*"):

```
# Set the Application ID
SF::SFDataCorrelator::HostInput::SetCurrentSource ($source_type_id,"CustomApp");
# Retrieve the Application ID you set
my $source_id =
SF::SFDataCorrelator::HostInput::GetCurrentSource();
```

See Example: Setting the Source ID, page 2-40 for an example of this command used in a script.

Required Fields

Each host input function requires either an address string (for specifying hosts by IPv4 or IPv6 address), an attribute list (for specifying IP hosts by attribute value), or a MAC list (for specifying MAC only hosts). The documentation for each function call indicates any additional required fields for that function.

Note that fields are required only in that you must supply that information to make sure that the host input succeeds and adds meaningful data to the network map. For example, you can add a fix to the system without providing a fix identification number or fix name that matches an existing Cisco fix definition and without mapping the third-party fix to a Cisco fix. However, even if that fix addresses vulnerabilities on the host where you added it, those vulnerabilities cannot be marked invalid if the system cannot map the fix to the vulnerabilities using a Cisco fix definition.

In general, supply as much information as possible for any data you import to ensure that the data can be used for data correlation.

Running a Host Input API Script

When you run your script, take the following requirements into account:

Application Privileges

To connect to the Host Input channel, applications must run with `admin` privileges.

Setting a Third-Party Vulnerability Map

If you want to import data including third-party vulnerabilities and use that data for impact correlation, you must create a third-party vulnerability map set before importing the data.

You can create a map set in two ways: using the Defense Center web interface or using the `AddScanResult` function. If you import scan results using this function, be sure to edit the source definition for the input source in your network discovery policy to set the identity source type to Scanner. The third-party map set allows the system to translate the third-party vulnerability ID to the corresponding vulnerability in the database. If you do not map a third-party vulnerability before import, the vulnerability does not map to a vulnerability ID and cannot be used for impact correlation.

For more information on mapping third-party vulnerabilities, see the *FireSIGHT System User Guide*.

Setting a Third-Party Product Map

When you import operating system or server data to a host, you can map third-party product name details to a Cisco product definition. You can create a third-party product map through the Defense Center web interface.

The third-party product map set allows the system to translate the third-party vendor, product, and version to the corresponding Cisco definition. When you set a third-party product map containing a server definition or an operating system definition, within the same script you can then define only the display strings for a third-party server or operating system when you add or set it using the API.

If you map third-party fixes to Cisco fix definitions using a third-party product map, set the product map, and then add fixes to hosts using the third-party fix name, the system maps the fixes to the appropriate Cisco fix definitions and deactivates vulnerabilities addressed by the fix.

To map a third-party product to a Cisco product definition:

Access: Admin

Step 1 Select **Policies > Application Detectors**, then click **User Third-Party Mappings**.

The User Third-Party Mappings page appears.

Step 2 You have two choices:

- To edit an existing map set, click **Edit** next to the map set.

- To create a new map set, click **Create Product Map Set**.

The Edit Third-Party Product Mappings page appears.

Step 3 Type a name for the mapping set in the **Mapping Set Name** field.

Step 4 Type a description in the **Description** field.

Step 5 You have two choices:

- To map a third-party product, click **Add Product Map.**
- To edit an existing third-party product map, click **Edit** next to the map set.

The Add Product Map page appears.

Step 6 Type the vendor string used by the third-party product in the **Vendor String** field.

Step 7 Type the product string used by the third-party product in the **Product String** field.

Step 8 Type the version string used by the third-party product in the **Version String** field.

Step 9 In the Product Mappings section, select the operating system, product, and versions you want to use for vulnerability mapping from the following lists (if applicable):

- **Vendor**
- **Product**
- **Major Version**
- **Minor Version**
- **Revision Version**
- **Build**
- **Patch**
- **Extension**

For example, if you want a host running a product whose name consists of third-party strings to use the vulnerabilities from Red Hat Linux 9, select **Redhat, Inc.** as the vendor, **Redhat Linux** as the product, and **9** as the version.

Step 10 Click **Save**.

Once you have the third-party product map, you can import data using the `SetOS`, `SetService`, or `AddService` functions. Note the third-party product name details and Cisco product definition before importing data.

To locate third-party and Cisco product details:

Access: Admin

Step 1 Select **Policies > Application Detectors.**

The Application Detectors page appears.

Step 2 Select **User Third-Party Mappings**.

The Third-Party Product Mappings page appears.

Step 3 Click the edit icon () for your product map set.

The Edit Third-Party Product Mappings page appears.

Step 4 Click the edit icon () for your product map.

The Add Product Map pop-up window appears. Note the **Vendor String**, **Product String**, and **Version String** values.

For more information on mapping third-party products, see the *FireSIGHT System User Guide*.

Host Input API Functions

After you include the prerequisite calls required in a host input API script (as described in Writing Host Input API Scripts, page 2-1), you can call various host input functions to import the specific data you want to add to your network map. For more information, see the following sections:

Host Functions

You can use the host input API to add and remove hosts in the network map and to set operating system definitions for hosts.

For more information on host functions, see the following sections:

AddHost

You can use the `AddHost` function to add a host to the network map. You can add an IP host (a host with an IP address and optionally a MAC address) or a MAC-only host (a host with only a MAC address). Because hosts created using the API are not tracked by the system, these hosts are not subject to the normal host timeout.

Note that you cannot create a MAC-only host for a MAC address if the system detects traffic which indicates that the MAC address is already mapped as a primary MAC address for an IP host in the network map.

See Example: Adding a Host to the Network Map, page 2-41 for an example of this function used in a script.

Use this syntax:

```
AddHost($source_type_id, $source_id, $ip_address, $mac_address)
```

Table 2-1 AddHost Fields

Field	Description	Required	Allowed Values
$source_type_id	Indicates the type of the host input source.	Yes	`"Application"` or `"Scanner"` Note you should set the `$source_type_id` variable to contain the appropriate value before invoking the `AddHost` function, and then reference `$source_type_id` in your function call. For more information, see Setting the Source Type, page 2-2.
$source_id	Indicates the source ID for the source adding the host input.	Yes	*`"source_id"`* Set the `$source_id` variable to contain the source ID before invoking the `AddHost` function, and then reference `$source_id` in your function call. For more information, see Obtaining a Source ID, page 2-2.
$ip_address	Indicates the IP address for the added host.	Yes (unless a MAC address is provided)	A single IP address, enclosed in double quotes
$mac_address	Indicates the MAC address for the added host.	Yes (unless an IP address is provided)	A single MAC address, enclosed in double quotes

DeleteHost

You can use the `DeleteHost` function to remove a host (or hosts) from the network map. You can remove an IP host (a host with an IP address and optionally a MAC address) by specifying either the IP address or the MAC address for the host. To remove a MAC-only host (a host with only a MAC address), indicate the MAC address as the `$mac_list` value.

Use this syntax:

```
DeleteHost($source_type_id, $source_id, $addr_string, $attrib_list, $mac_list)
```

Table 2-2 DeleteHost Fields

Field	Description	Required	Allowed Values
$source_type_id	Indicates the type of the host input source.	Yes	`"Application"` or `"Scanner"` Note you should set the `$source_type_id` variable to contain the appropriate value before invoking the `DeleteHost` function, and then reference `$source_type` in your function call. For more information, see Setting the Source Type, page 2-2.
$source_id	Indicates the source ID for the source adding the host input.	Yes	*`"source_id"`* Note you should set the `$source_id` variable to contain the source ID before invoking the `DeleteHost` function, and then reference `$source_id` in your function call. For more information, see Obtaining a Source ID, page 2-2.
$addr_string	Indicates the string containing the IP address or addresses for the affected hosts.	Yes (unless attribute lists or MAC addresses are provided)	A comma-separated list of IP addresses, CIDR blocks, and ranges of IP addresses, enclosed in double quotes.

$attrib_list	Indicates the host attribute or attributes specifying the hosts affected by the host input.	Yes (unless IP addresses or MAC addresses are provided)	A list of attribute value hash pairs of the format:
			`{attribute => "Department",`
			`value => "Development"},`
			Note that `$attrib_list` must be an array or reference an array.

$mac_list	Indicates the list of MAC addresses for the affected hosts.	Yes (unless IP addresses or attribute lists are provided)	A list of MAC address strings, with or without separating colons. Note that `$mac_list` must be an array or reference an array.

SetOS

You can use the `SetOS` function to specify the vendor, product, version, and mobile device information for the operating system for specified hosts. When you import operating system information, you set the display strings for the vendor, product, version, and mobile device information.

You can also map the third-party vendor, product, and version strings to a Cisco product definition. If you map third-party operating system names to a Cisco definition, the vulnerabilities for that operating system in the Cisco database map to the host where the third-party data was imported. If you have already created a third-party product map set using the Defense Center web interface, you can use the `SetCurrent3rdPartyMap` function to use the values you specified in that map set for the third-party application strings and corresponding Cisco definitions, as described in SetCurrent3rdPartyMap, page 2-34.

The operating system identity displayed in a host profile is set by the highest priority source. Possible sources have the following priority order: user, scanner and application (set in the system policy), FireSIGHT, then NetFlow. Note that a new higher priority operating system identity will not override a current operating system identity if it has less detail than the current identity.

If you define a custom operating system for a host, the FireSIGHT System web interface indicates the source for the change in the Source Type field of the event view or the basic host information of the host profile.

See Example: Setting the Operating System on the Host, page 2-41 for an example of this function used in a script.

Use this syntax:

```
SetOS($source_type_id, $source_id, $addr_string, $attrib_list, $os)
```

Table 2-3 SetOS Fields

Field	Description	Required	Allowed Values
$source_type_id	Indicates the type of the host input source.	Yes	`"Application"` or `"Scanner"` Note you should set the `$source_type_id` variable to contain the appropriate value before invoking the `SetOS` function, and then reference `$source_type_id` in your function call. For more information, see Setting the Source Type, page 2-2.
$source_id	Indicates the source ID for the source adding the host input.	Yes	*`"source_id"`* Note you should set the `$source_id` variable to contain the source ID before invoking the `SetOS` function, and then reference `$source_id` in your function call. For more information, see Obtaining a Source ID, page 2-2.
$addr_string	Indicates the string containing the IP address or addresses for the affected hosts.	Yes (unless attribute lists are provided)	A comma-separated list of IP addresses, CIDR blocks, and ranges of IP addresses, enclosed in double quotes.

$attrib_list	Indicates the host attribute or attributes specifying the hosts affected by the host input.	Yes (unless IP addresses are provided)	A list of attribute value hash pairs of the format:
			`{attribute => "Department",`
			`value => "Development"},`
			Note that `$attrib_list` must be an array or reference an array.

$os	Contains a hash with keys describing the details of an operating system definition.	If you set a current third-party map before calling `SetOS`, only the rendering keys are required.	The `$os` variable is an OS definition hash that supports several keys. For more information, see Keys for the $os Variable, page 2-8.

Keys for the $os Variable

The `$os` variable is an OS definition hash that supports several keys. If you call the `SetCurrent3rdPartyMap` function before calling the `SetOS` function, note the third-party product name details and Cisco product definition when creating the third-party mapping. See Setting a Third-Party Product Map, page 2-3 for more information.

You need only specify the vendor, product, and version strings for this function. Otherwise, the system assigns the most focused set of vulnerabilities it can using each piece of Cisco product definition detail you provide. For example, you could set the `vendor_str`, `product_str`, and `version_str` keys to `Microsoft`, `Windows`, and `3.x`, respectively, then only set the `vendor_id`, `product_id`, and `major` keys to the identification numbers for the vendor, product, and version for `Microsoft`, `Windows`, and `3`, respectively. All hosts where you set the operating system to Microsoft Windows 3.x would have all vulnerabilities for both Microsoft Windows 3.1 and Microsoft Windows 3.11.

For more information, see SetCurrent3rdPartyMap, page 2-34.

For more information on individual keys, see the tables that follow.

Table 2-4 Keys for Rendering

Key	Data Type	Definition
vendor_str	string	Use this key to supply the operating system vendor display name used by the third-party application.
product_str	string	Use this key to supply the operating system product display name used by the third-party application.
version_str	string	Use this key to supply the operating system version display name used by the third-party application.
device_string	string	Use this key to supply the detected mobile device hardware information.
mobile	uint8	Use this key to indicate whether the operating system is running on a mobile device.
jailbroken	uint8	Use this key to indicate whether the mobile device operating system is jailbroken.

Table 2-5 Keys for Vulnerability Mapping

Key	Data Type	Definition
vendor_id	uint32	Use this key to supply the Cisco vendor definition.
product_id	uint32	Use this key to supply the Cisco product definition.
major	uint32	Use this key to supply the Cisco major version definition to map to.
minor	uint32	Use this key to supply the Cisco minor version definition to map to.
revision	uint32	Use this key to supply the Cisco revision string to map to.
to_major	uint32	Use this key to set the last version number of the Cisco major version range to map to.
to_minor	uint32	Use this key to set the last version number of the Cisco minor version range to map to.
to_revision	uint32	Use this key to set the last revision number of the Cisco revision range to map to.
build	string	Use this key to supply the Cisco build definition to map to.
patch	string	Use this key to supply the Cisco patch definition to map to.

Table 2-5 Keys for Vulnerability Mapping (continued)

Key	Data Type	Definition
extension	string	Use this key to supply the Cisco extension definition to map to.
fixes	variable	Use this key to supply a list of fix_ids or fix names to be applied to the operating system. If a fix id or fix name matches a fix in the Cisco database, the system looks up the ID for the matching fix and uses it.

Use the following key to delete the user OS definition:

- `drop_user_product`

If the `drop_user_product` value is set to 1, the `SetOS` function deletes the existing user operating system definition from the host.

UnsetOS

The `UnsetOS` function removes a user-added OS definition from the specified hosts. `UnsetOS` does not remove an OS definition from a host if it was detected through FireSIGHT.

Use this syntax:

```
UnsetOS($source_type_id, $source_id, $addr_string, $attrib_list)
```

Table 2-6 UnsetOS Fields

Field	Description	Required	Allowed Values
$source_type_id	Indicates the type of the host input source.	Yes	`"Application"` or `"Scanner"` Note you should set the `$source_type_id` variable to contain the appropriate value before invoking the `UnsetOS` function, and then reference `$source_type_id` in your function call. For more information, see Setting the Source Type, page 2-2.
$source_id	Indicates the source ID for the source adding the host input.	Yes	*`"source_id"`* Note you should set the `$source_id` variable to contain the source ID before invoking the `UnsetOS` function, and then reference `$source_id` in your function call. For more information, see Obtaining a Source ID, page 2-2.
$addr_string	Indicates the string containing the IP address or addresses for the affected hosts.	Yes (unless attribute lists are provided)	A comma-separated list of IP addresses, CIDR blocks, and ranges of IP addresses, enclosed in double quotes.

$attrib_list	Indicates the host attribute or attributes specifying the hosts affected by the host input.	No	A list of attribute value hash pairs of the format:
			`{attribute => "Department",`
			`value => "Development"},`
			Note that `$attrib_list` must be an array or reference an array.

Server Functions

You can update server information for hosts in the network map using the server functions.

For more information, see the following sections:

AddService

You can add a server to an existing host in the network map using the `AddService` function.

The server identity displayed in a host profile is set by the highest priority source. Possible sources have the following priority order: user, scanner and application (set in the system policy), FireSIGHT, then NetFlow. Note that a new higher priority operating server identity will not override a current operating server identity if it has less detail than the current identity.

See Example: Adding a Server to the Host, page 2-43 for an example of this function used in a script.

Use this syntax:

```
AddService($source_type_id, $source_id, $addr_string, $attrib_list, $service)
```

Table 2-7 AddService Fields

Field	Description	Required	Allowed Values
$source_type_id	Indicates the type of the host input source.	Yes	`"Application"` or `"Scanner"` Note you should set the `$source_type_id` variable to contain the appropriate value before invoking the `AddService` function, and then reference `$source_type_id` in your function call. For more information, see Setting the Source Type, page 2-2.
$source_id	Indicates the source ID for the source adding the host input.	Yes	*`"source_id"`* Note you should set the `$source_id` variable to contain the source ID before invoking the `AddService` function, and then reference `$source_id` in your function call. For more information, see Obtaining a Source ID, page 2-2.
$addr_string	Indicates the string containing the IP address or addresses for the affected hosts.	Yes (unless attribute lists are provided)	A comma-separated list of IP addresses, CIDR blocks, and ranges of IP addresses, enclosed in double quotes.

Table 2-7 AddService Fields (continued)

Field	Description	Required	Allowed Values

$attrib_list	Indicates the host attribute or attributes specifying the hosts affected by the host input.	Yes (unless IP addresses are provided)	A list of attribute value hash pairs of the format:
			`{attribute => "Department",`
			`value => "Development"},`
			Note that `$attrib_list` must be an array or reference an array.

$service	Contains a hash with keys describing the details of a server definition.	If you set a current third-party map before calling `AddService`, only the rendering and `service_name` or `service_id` keys are required.	The `$service` variable is a server definition hash that supports several keys. For more information, see Service Keys, page 2-15.

SetService

You can use the `SetService` function to specify the server protocol, vendor, product, and version for a specified server. You can set display strings for the server using `$service` keys. By mapping a third-party product in the Defense Center web interface or using the `SetCurrent3rdPartyMap` function (see SetCurrent3rdPartyMap, page 2-34), you can associate third-party server data with the vulnerability information for specific Cisco product definitions.

If the server protocol does not already exist, this call causes a new server identity to be created for the string. If the specified server does not already exist, the system creates it.

The server identity displayed in a host profile is set by the highest priority source. Possible sources have the following priority order: user, scanner and application (set in the system policy), FireSIGHT, then NetFlow. Note that a new higher priority server identity will not override a current server identity if it has less detail than the current identity.

If you create a custom server definition for a host, the FireSIGHT System web interface indicates the source for the change in the Source Type field of the Servers table view of events or the Servers section of the host profile.

Note If the number of servers stored in the network map for a specific host exceeds 100, new server information is ignored until servers are deleted from the host.

Use this syntax:

```
SetService($source_type_id, $source_id, $addr_string, $attrib_list, $service)
```

Table 2-8 SetService Fields

Field	Description	Required	Allowed Values
$source_type_id	Indicates the type of the host input source.	Yes	`"Application"` or `"Scanner"` Note you should set the `$source_type_id` variable to contain the appropriate value before invoking the `SetService` function, and then reference `$source_type_id` in your function call. For more information, see Setting the Source Type, page 2-2.
$source_id	Indicates the source ID for the source adding the host input.	Yes	*`"source_id"`* Note you should set the `$source_id` variable to contain the source ID before invoking the `SetService` function, and then reference `$source_id` in your function call. For more information, see Obtaining a Source ID, page 2-2.
$addr_string	Indicates the string containing the IP address or addresses for the affected hosts.	Yes (unless attribute lists are provided)	A comma-separated list of IP addresses, CIDR blocks, and ranges of IP addresses, enclosed in double quotes.

$attrib_list	Indicates the host attribute or attributes specifying the hosts affected by the host input.	Yes (unless IP addresses are provided)	A list of attribute value hash pairs of the format:
			`{attribute => "Department",`
			`value => "Development"},`
			Note that `$attrib_list` must be an array or reference an array.

$service	Contains a hash with keys describing the details of a server definition.	If you set a current third-party map before calling `SetService`, only the rendering and `service_name` or `service_id` keys are required.	The `$service` variable is a server definition hash that supports several keys. For more information, see Service Keys, page 2-15.

UnsetService

You can use the `UnsetService` function to remove user-added server definitions from a specified host. `UnsetService` does not remove any server definitions detected through FireSIGHT.

Note If the number of servers stored in the network map for a specific host exceeds 100, new server information is ignored until servers are deleted from the host.

Use this syntax:

```
UnsetService($source_type_id, $source_id, $addr_string, $port, $protocol)
```

Table 2-9 UnsetService Fields

Field	Description	Required	Allowed Values
$source_type_id	Indicates the type of the host input source.	Yes	`"Application"` or `"Scanner"` Note you should set the `$source_type_id` variable to contain the appropriate value before invoking the `UnsetService` function, and then reference `$source_type_id` in your function call. For more information, see Setting the Source Type, page 2-2.
$source_id	Indicates the source ID for the source adding the host input.	Yes	*`"source_id"`* Note you should set the `$source_id` variable to contain the source ID before invoking the `UnsetService` function, and then reference `$source_id` in your function call. For more information, see Obtaining a Source ID, page 2-2.
$addr_string	Indicates the string containing the IP address or addresses for the affected hosts.	Yes	A comma-separated list of IP addresses, CIDR blocks, and ranges of IP addresses, enclosed in double quotes.
$port	Indicates the port for the server to be unset.	Yes	Integers in the range of 1-65535.
$proto	Indicates the protocol for the server to be unset.	Yes	Either the strings `tcp` or `udp` or the appropriate protocol IDs `6` (tcp) or `17` (udp).

DeleteService

You can use the `DeleteService` function to remove a server from a specified host. You must specify the port and protocol of the server to be deleted.

Use this syntax:

```
DeleteService($source_type_id, $source_id, $addr_string, $attrib_list, $port, $proto)
```

Table 2-10 DeleteService Fields

Field	Description	Required	Allowed Values
$source_type_id	Indicates the type of the host input source.	Yes	`"Application"` or `"Scanner"` Note you should set the `$source_type_id` variable to contain the appropriate value before invoking the `DeleteService` function, and then reference `$source_type_id` in your function call. For more information, see Setting the Source Type, page 2-2.
$source_id	Indicates the source ID for the source adding the host input.	Yes	*`"source_id"`* Note you should set the `$source_id` variable to contain the source ID before invoking the `DeleteService` function, and then reference `$source_id` in your function call. For more information, see Obtaining a Source ID, page 2-2.
$addr_string	Indicates the string containing the IP address or addresses for the affected hosts.	Yes (unless attribute lists are provided)	A comma-separated list of IP addresses, CIDR blocks, and ranges of IP addresses, enclosed in double quotes.

$attrib_list	Indicates the host attribute or attributes specifying the hosts affected by the host input.	Yes (unless IP addresses are provided)	A list of attribute value hash pairs of the format:
			`{attribute => "Department",`
			`value => "Development"},`
			Note that `$attrib_list` must be an array or reference an array.

$port	Indicates the port for the server to be deleted.	Yes	Integers in the range of 1-65535.
$proto	Indicates the protocol for the server to be deleted.	Yes	Either the strings `tcp` or `udp` or the appropriate protocol IDs `6` (tcp) or `17` (udp).

Service Keys

The `$service` variable is a server definition hash that supports several keys.

If you do not set either the `service_name` or the `service_id` value, the server displays as `"unknown"` in the web interface.

If you call the `SetCurrent3rdPartyMap` function before calling the `SetOS` function, note the third-party product name details and Cisco product definition when creating the third-party mapping. See Setting a Third-Party Product Map, page 2-3 for more information.

You need only specify the vendor, product, and version strings for this function. Otherwise, the system assigns the most focused set of vulnerabilities it can using each piece of Cisco product definition detail you provide. For example, if you use this function to set a server definition on a host by setting the `vendor_str`, `product_str`, and `version_str` keys to Apache, Tomcat, and 4.x, respectively, then only set the `vendor_id`, `product_id`, and `major` keys to the identification numbers for Apache, Tomcat, and 4, respectively, that host will have all vulnerabilities for both Apache Tomcat 4.0 and Apache Tomcat 4.1.

Use the following key to delete an existing user server definition:

- `drop_user_product`

If the `drop_user_product` value is set to 1, the existing user server definition is deleted from the host.

When you set a string value for a key in the hash, enclose that value in single quotes.

The following tables provide information on the keys you can use with the `$service` field.

Table 2-11 Keys for Server Identity

Key	Data Type	Definition
port	uint	Use this key in combination with the `proto` key and the address or attribute specifications to specify the server on the hosts where you want to modify the vulnerability listings.
proto	string	Use this key in combination with the `port` key and the address or attribute specifications to specify the server on the hosts where you want to modify the vulnerability listings, using either the strings `tcp` or `udp` or the appropriate protocol IDs `6` (tcp) or `17` (udp).

Table 2-12 Keys for Rendering

Key	Data Type	Definition
vendor_str	string	Use this key to supply the server vendor display name used by the third-party application.
version_str	string	Use this key to supply the server version display name used by the third-party application.

Table 2-13 Keys for Vulnerability Mapping

Key	Data Type	Definition
vendor_id	uint32	Use this key to supply the third-party server vendor definition.
product_id	uint32	Use this key to supply the third-party server product definition.
service_name	uint32	The name of the server in the Cisco database. To identify the server, you must include a value for either `service_name` or `service_id`. If neither is provided the server will be listed as `unknown`. If a service_name is provided, the system looks up the server ID. If no ID exists for the `service_name` value, the system creates an ID.
service_id	uint32	The ID of the server in the Cisco database.
major	uint32	Use this key to supply the Cisco server major version definition to map to.
minor	uint32	Use this key to supply the Cisco server minor version definition to map to.
revision	uint32	Use this key to supply the Cisco server revision string to map to.
to_major	uint32	Use this key to set the last version number of the Cisco server major version range to map to.

Table 2-13 Keys for Vulnerability Mapping (continued)

Key	Data Type	Definition
to_minor	uint32	Use this key to set the last version number of the Cisco server minor version range to map to.
to_revision	uint32	Use this key to set the last revision number of the Cisco server revision range to map to.
build	string	Use this key to supply the Cisco server build definition to map to.
patch	string	Use this key to supply the Cisco server patch definition to map to.
extension	string	Use this key to supply the Cisco server extension definition to map to.
fixes	variable	Use this key to supply a comma-separated list of `fix_ids` or fix names to be applied to the server. If a fix id or fix name matches a fix in the Cisco database, the system looks up the ID for the matching fix and uses it.

Client Application Functions

You can use the client application functions to modify client application data for hosts in the network map. You can locate client application names in the Application Filters page.

To locate client application names and types:

Access: Admin/Access Admin/Network Admin

Step 1 Select **Objects > Object Management**.

The Object Management page appears.

Step 2 Click **Application Filters**.

The Application Filters section appears.

Step 3 Click **Add Application Filter**.

The Application Filter pop-up window appears.

Step 4 Select Client Application in the **Application Filters** list to retrieve the list of client applications.

The Available Applications list displays client application names.

For more information, see the following sections:

- AddClientApp, page 2-17
- DeleteClientApp, page 2-19
- DeleteClientAppPayload, page 2-19

AddClientApp

You can use the `AddClientApp` function to add client applications to existing hosts in the network map. If the client application name does not already exist in the Cisco database, the system creates a new entry for the client application.

The client application identity displayed in a host profile is set by the highest priority source. Possible sources have the following priority order: user, scanner and application (set in the system policy), FireSIGHT, then NetFlow. Note that a new higher priority client application identity will not override a current client application identity if it has less detail than the current identity.

See Example: Adding a Client Application to Multiple Hosts, page 2-43 for an example of this function used in a script.

Use this syntax:

```
AddClientApp($source_type_id, $source_id, $addr_string, $attrib_list, $id, $type,
$version)
```

Table 2-14 AddClientApp Fields

Field	Description	Required	Allowed Values
$source_type_id	Indicates the type of the host input source.	Yes	`"Application"` or `"Scanner"` Note you should set the `$source_type_id` variable to contain the appropriate value before invoking the `AddClientApp` function, and then reference `$source_type_id` in your function call. For more information, see Setting the Source Type, page 2-2.
$source_id	Indicates the source ID for the source adding the host input.	Yes	*`"source_id"`* Note you should set the `$source_id` variable to contain the source ID before invoking the `AddClientApp` function, and then reference `$source_id` in your function call. For more information, see Obtaining a Source ID, page 2-2.
$addr_string	Indicates the string containing the IP address or addresses for the affected hosts.	Yes (unless attribute lists are provided)	A comma-separated list of IP addresses, CIDR blocks, and ranges of IP addresses, enclosed in double quotes.

$attrib_list	Indicates the host attribute or attributes specifying the hosts affected by the host input.	Yes (unless IP addresses are provided)	A list of attribute value hash pairs of the format:
			`{attribute => "Department",`
			`value => "Development"},`
			Note that `$attrib_list` must be an array or reference an array.

$id	Indicates the client application name.	Yes	A string consisting of alphanumeric characters or spaces, enclosed in double quotes. For existing applications, corresponds to ID values in the database. The system looks up the ID to see if it matches an existing client application ID. If it does not, a new ID is created.
$type	This field is deprecated.	No	A null value.
$version	Indicates the application version.	No	A string consisting of alphanumeric characters or spaces, enclosed in double quotes.

DeleteClientApp

You can use the `DeleteClientApp` function to remove a client application from the specified host.

Use this syntax:

```
DeleteClientApp($source_type_id, $source_id, $addr_string, $attrib_list, $id, $type,
$version)
```

Table 2-15 DeleteClientApp Fields

Field	Description	Required	Allowed Values
$source_type_id	Indicates the type of the host input source.	Yes	`"Application"` or `"Scanner"` Note you should set the `$source_type_id` variable to contain the appropriate value before invoking the `DeleteClientApp` function, and then reference `$source_type_id` in your function call. For more information, see Setting the Source Type, page 2-2.
$source_id	Indicates the source ID for the source adding the host input.	Yes	*`"source_id"`* Note you should set the `$source_id` variable to contain the source ID before invoking the `DeleteClientApp` function, and then reference `$source_id` in your function call. For more information, see Obtaining a Source ID, page 2-2.
$addr_string	Indicates the string containing the IP address or addresses for the affected hosts.	Yes (unless attribute lists are provided)	A comma-separated list of IP addresses, CIDR blocks, and ranges of IP addresses, enclosed in double quotes.

$attrib_list	Indicates the host attribute or attributes specifying the hosts affected by the host input.	Yes (unless IP addresses are provided)	A list of attribute value hash pairs of the format:
			`{attribute => "Department",`
			`value => "Development"},`
			Note that `$attrib_list` must be an array or reference an array.

$id	Indicates the client application name.	Yes	A string consisting of alphanumeric characters or spaces, enclosed in double quotes. For existing applications, corresponds to ID values in the database. The system looks up the ID to see if it matches an existing client application ID. If it does not, a new ID is created.
$type	This field is deprecated.	No	A null value.
$version	Indicates the application version.	No	A string consisting of alphanumeric characters or spaces, enclosed in double quotes.

DeleteClientAppPayload

You can use the `DeleteClientAppPayload` function to remove a web application from the specified host.

Use this syntax:

```
DeleteClientAppPayload($source_type_id, $source_id, $addr_string, $attrib_list, $id,
$type, $version, $payload_type, $payload_id)
```

Table 2-16 DeleteClientAppPayload Fields

Field	Description	Required	Allowed Values
$source_type_id	Indicates the type of the host input source.	Yes	`"Application"` or `"Scanner"` Note you should set the `$source_type_id` variable to contain the appropriate value before invoking the `DeleteClientAppPayload` function, and then reference `$source_type_id` in your function call. For more information, see Setting the Source Type, page 2-2.
$source_id	Indicates the source ID for the source adding the host input.	Yes	*`"source_id"`* Note you should set the `$source_id` variable to contain the source ID before invoking the `DeleteClientAppPayload` function, and then reference `$source_id` in your function call. For more information, see Obtaining a Source ID, page 2-2.
$addr_string	Indicates the string containing the IP address or addresses for the affected hosts.	Yes (unless attribute lists are provided)	A comma-separated list of IP addresses, CIDR blocks, and ranges of IP addresses, enclosed in double quotes.

$attrib_list	Indicates the host attribute or attributes specifying the hosts affected by the host input.	Yes (unless IP addresses are provided)	A list of attribute value hash pairs of the format:
			`{attribute => "Department",`
			`value => "Development"},`
			Note that `$attrib_list` must be an array or reference an array.

$id	Indicates the client application name.	Yes	A string consisting of alphanumeric characters or spaces, enclosed in double quotes. For existing applications, corresponds to ID values in the database. The system looks up the ID to see if it matches an existing client application ID. If it does not, a new ID is created.
$type	This field is deprecated.	No	A null value.
$version	Indicates the application version.	No	A string consisting of alphanumeric characters or spaces, enclosed in double quotes.
$payload_type	Indicates the web application category.	Yes	The number `0`.
$payload_id	Indicates the web application name.	Yes	A string consisting of alphanumeric characters or spaces, enclosed in double quotes. For existing applications, corresponds to ID values in the database. The system looks up the ID to see if it matches an existing client application ID. If it does not, a new ID is created.

Protocol Functions

You can use the protocol functions to update protocol information for hosts in the network map.

For more information, see the following sections:

DeleteProtocol

You can use the `DeleteProtocol` function to remove a protocol from the specified IP or MAC host.

Use this syntax:

```
DeleteProtocol($source_type_id, $source_id, $addr_string, $attrib_list, $mac_list,
$proto, $type)
```

Table 2-17 DeleteProtocol Fields

Field	Description	Required	Allowed Values
$source_type_id	Indicates the type of the host input source.	Yes	`"Application"` or `"Scanner"` Note you should set the `$source_type_id` variable to contain the appropriate value before invoking the `DeleteProtocol` function, and then reference `$source_type_id` in your function call. For more information, see Setting the Source Type, page 2-2.
$source_id	Indicates the source ID for the source adding the host input.	Yes	*`"source_id"`* Note you should set the `$source_id` variable to contain the source ID before invoking the `DeleteProtocol` function, and then reference `$source_id` in your function call. For more information, see Obtaining a Source ID, page 2-2.
$addr_string	Indicates the string containing the IP address or addresses for the affected hosts.	Yes (unless attribute lists or MAC addresses are provided)	A comma-separated list of IP addresses, CIDR blocks, and ranges of IP addresses, with each address, block, or range enclosed in double quotes.

$attrib_list	Indicates the host attribute or attributes specifying the hosts affected by the host input.	Yes (unless IP addresses or MAC addresses are provided)	A list of attribute value hash pairs of the format:
			`{attribute => "Department",`
			`value => "Development"},`
			Note that `$attrib_list` must be an array or reference an array.

$mac_list	Indicates the list of MAC addresses for the affected hosts.	Yes (unless IP addresses or attribute lists are provided)	A list of MAC address strings, with or without separating colons. Note that `$mac_list` must be an array or reference an array.

Table 2-17 DeleteProtocol Fields (continued)

Field	Description	Required	Allowed Values
$proto	Indicates the identification string or name of the protocol to be deleted.	Yes	Valid protocol names consisting of alphanumeric characters or spaces, enclosed in double quotes. For transport protocols (“`xport`”), protocols listed in the /etc/protocols file are acceptable. For network protocols (“`net`”), see Network Protocol Values, page A-1.
$type	Indicates the type of protocol to be deleted.	Yes	`"xport"` or `"net"`

AddProtocol

You can use the `AddProtocol` function to add either a network or transport protocol to an existing host in the network map. You can supply either a protocol ID, a transport protocol name that exists in the `/etc/protocols` file on your Defense Center, or a network protocol name from Network Protocol Values, page A-1.

Note You cannot add transport protocols to MAC-only hosts.

See Example: Adding a Protocol to the Host, page 2-43 for an example of this function used in a script.

Use this syntax:

```
AddProtocol($source_type_id, $source_id, $addr_string, $attrib_list, $mac_list,
$proto, $type)
```

Table 2-18 AddProtocol Fields

Field	Description	Required	Allowed Values
$source_type_id	Indicates the type of the host input source.	Yes	`"Application"` or `"Scanner"` Note you should set the `$source_type_id` variable to contain the appropriate value before invoking the `AddProtocol` function, and then reference `$source_type_id` in your function call. For more information, see Setting the Source Type, page 2-2.
$source_id	Indicates the source ID for the source adding the host input.	Yes	*`"source_id"`* Note you should set the `$source_id` variable to contain the source ID before invoking the `AddProtocol` function, and then reference `$source_id` in your function call. For more information, see Obtaining a Source ID, page 2-2.
$addr_string	Indicates the string containing the IP address or addresses for the affected hosts.	Yes (unless attribute lists or MAC addresses are provided)	A comma-separated list of IP addresses, CIDR blocks, and ranges of IP addresses, with each address, block, or range enclosed in double quotes.

Table 2-18 AddProtocol Fields (continued)

Field	Description	Required	Allowed Values

$attrib_list	Indicates the host attribute or attributes specifying the hosts affected by the host input.	Yes (unless IP addresses or MAC addresses are provided)	A list of attribute value hash pairs of the format:
			`{attribute => "Department",`
			`value => "Development"},`
			Note that `$attrib_list` must be an array or reference an array.

$mac_list	Indicates the list of MAC addresses for the affected hosts.	Yes (unless IP addresses or attribute lists are provided)	A list of MAC address strings, with or without separating colons. Note that `$mac_list` must be an array or reference an array.
$proto	Indicates the identification string or name of the protocol to be added.	Yes	Valid protocol names consisting of alphanumeric characters or spaces, enclosed in double quotes. For transport protocols (“`xport`”), protocols listed in the /etc/protocols file are acceptable. For network protocols (“`net`”), see Network Protocol Values, page A-1.
$type	Indicates the type of protocol to be added.	Yes	`"xport"` or `"net"`

Package Fix Functions

You can use the Package Fix functions to apply or remove fixes for hosts in your network map.

For more information, see the following sections:

- AddFix, page 2-23
- RemoveFix, page 2-24

AddFix

You can use the `AddFix` function to map a fix to a specified host or server. You can map a fix using a fix ID or a fix name from the Cisco vulnerability database (VDB), or using a third-party fix that you map to a fix in the VDB using the Defense Center web interface.

Note You can also specify fixes with the `SetOS` and `SetService` functions. If a fix list is supplied using one of these functions the supplied fix list replaces the existing fix list for the host or server.

When you apply a fix to a host or server, the vulnerability mappings for the system are adjusted and the fixed vulnerabilities are marked as Invalid in the web interface and are not used for impact assessment. However, note that if the applied fix is not applicable to the operating system or server identity the fix has no effect.

Use the following syntax:

```
AddFix($source_type_id, $source_id, $addr_string, $attrib_list, $port, $proto, $fix)
```

Table 2-19 AddFix Fields

Field	Description	Required	Allowed Values
$source_type_id	Indicates the type of the host input source.	Yes	`"Application"` or `"Scanner"` Note you should set the `$source_type_id` variable to contain a value before invoking the `AddFix` function, and then reference `$source_type_id` in your function call. For more information, see Setting the Source Type, page 2-2.
$source_id	Indicates the source ID for the source adding the host input.	Yes	*`"source_id"`* Note you should set the `$source_id` variable to contain the source ID before invoking the `AddFix` function, and then reference `$source_id` in your function call. For more information, see Obtaining a Source ID, page 2-2.
$addr_string	Indicates the string containing the IP address or addresses for the affected hosts.	Yes (unless attribute lists or MAC addresses are provided)	A comma-separated list of IP addresses, CIDR blocks, and ranges of IP addresses, with each address, block, or range enclosed in double quotes.

$attrib_list	Indicates the host attribute or attributes specifying the hosts affected by the host input.	Yes (unless IP addresses or MAC addresses are provided)	A list of attribute value hash pairs of the format:
			`{attribute => "Department",`
			`value => "Development"},`
			Note that `$attrib_list` must be an array or reference an array.

$mac_list	Indicates the list of MAC addresses for the affected hosts.	Yes (unless IP addresses or attribute lists are provided)	A list of MAC address strings, with or without separating colons. Note that `$mac_list` must be an array or reference an array.
$port	With the `$proto` field, indicates the server affected by the fix.	Yes, if the fix applies to a server	Integers in the range of 1-65535, enclosed in double quotes.
$proto	With the `$port` field, indicates the server affected by the fix.	No	Either the strings `tcp` or `udp` or the appropriate protocol IDs `6` (tcp) or `17` (udp).
$fix	Indicates the identification string for the fix.	**Yes**	A Cisco fix identification number or a third-party fix name, enclosed in double quotes, defined in a third-party product map that you use by calling the `SetCurrent3rdPartyMap` function before invoking the `AddFix` function. For more information, see SetCurrent3rdPartyMap, page 2-34.

RemoveFix

You can use the `RemoveFix` function to remove a fix mapping from the specified host or server. When you remove a fix, vulnerability mappings are updated accordingly.

Tip You can also specify fixes using the `SetOS` and `SetService` functions.

Use this syntax:

```
RemoveFix($source_type_id, $source_id, $addr_string, $attrib_list, $port, $proto,
$fix)
```

Table 2-20 RemoveFix Fields

Field	Description	Required	Allowed Values
$source_type_id	Indicates the type of the host input source.	Yes	`"Application" or "Scanner"` Note you should set the `$source_type_id` variable to contain a value before invoking the `RemoveFix` function, and then reference `$source_type_id` in your function call. For more information, see Setting the Source Type, page 2-2.
$source_id	Indicates the source ID for the source adding the host input.	Yes	*`"source_id"`* Note you should set the `$source_id` variable to contain the source ID before invoking the `RemoveFix` function, and then reference `$source_id` in your function call. For more information, see Obtaining a Source ID, page 2-2.
$addr_string	Indicates the string containing the IP address or addresses for the affected hosts.	Yes (unless attribute lists or MAC addresses are provided)	A comma-separated list of IP addresses, CIDR blocks, and ranges of IP addresses, with each address, block, or range enclosed in double quotes.

$attrib_list	Indicates the host attribute or attributes specifying the hosts affected by the host input.	Yes (unless IP addresses or MAC addresses are provided)	A list of attribute value hash pairs of the format:
			`{attribute => "Department",`
			`value => "Development"},`
			Note that `$attrib_list` must be an array or reference an array.

$port	With the `$proto` field, indicates the server affected by the fix.	Yes, if the fix applies to a server	Integers in the range of 1-65535, enclosed in double quotes.
$proto	With the `$port` field, indicates the server affected by the fix.	No	Either the strings `tcp` or `udp` or the appropriate protocol IDs `6` (tcp) or `17` (udp).
$fix	Indicates the identification string for the fix.	Yes	A Cisco fix identification number or a third-party fix name, enclosed in double quotes, defined in a third-party product map that you use by calling the `SetCurrent3rdPartyMap` function before invoking the `RemoveFix` function. For more information, see SetCurrent3rdPartyMap, page 2-34.

Host Attribute Functions

For more information, see the following sections:

AddHostAttribute

You can use the `AddHostAttribute` function to add text or URL attributes.

Note that adding a host attribute does not add a value for the attribute. For more information on setting an attribute value, see SetAttributeValue, page 2-27.

Use this syntax:

```
AddHostAttribute($source_type_id, $source_id, $attrib_name, $attrib_type)
```

where *`attributename`* is the name of the attribute (consisting of alphanumeric characters and spaces) and *`attributetype`* is the type of attribute (`text` or `URL`).

Table 2-21 AddHostAttribute Fields

Field	Description	Required	Allowed Values
$source_type_id	Indicates the type of the host input source.	Yes	`"Application" or "Scanner"` Note you should set the `$source_type_id` variable to contain a value before invoking the `AddHostAttribute` function, and then reference `$source_type_id` in your function call. For more information, see Setting the Source Type, page 2-2.
$source_id	Indicates the source ID for the source adding the host input.	Yes	*`"source_id"`* Note you should set the `$source_id` variable to contain the source ID before invoking the `AddHostAttribute` function, and then reference `$source_id` in your function call. For more information, see Obtaining a Source ID, page 2-2.
$attrib_name	Indicates the host attribute or attributes specifying the hosts affected by the host input.	Yes	"`Department`"
$attrib_type	Indicates the host attribute or attributes specifying the hosts affected by the host input.	Yes	"`Text`" or "`URL`"

DeleteHostAttribute

You can use the `DeleteHostAttribute` function to delete attributes.

Use this syntax:

```
DeleteHostAttribute($source_type_id, $source_id, $attrib_name)
```

where *`attributename`* is the name of the attribute (consisting of alphanumeric characters and spaces.)

Table 2-22 DeleteHostAttribute Fields

Field	Description	Required	Allowed Values
$source_type_id	Indicates the type of the host input source.	Yes	`"Application" or "Scanner"` Note you should set the `$source_type_id` variable to contain a value before invoking the `AddHostAttribute` function, and then reference `$source_type_id` in your function call. For more information, see Setting the Source Type, page 2-2.
$source_id	Indicates the source ID for the source adding the host input.	Yes	*`"source_id"`* Note you should set the `$source_id` variable to contain the source ID before invoking the `DeleteHostAttribute` function, and then reference `$source_id` in your function call. For more information, see Obtaining a Source ID, page 2-2.
$attrib_name	Indicates the host attribute or attributes specifying the hosts affected by the host input.	Yes	“`Department"`

SetAttributeValue

You can use the `SetAttributeValue` function to set the value of an existing attribute to the specified value for the specified hosts. This function can set the value of user-defined host attributes and the Criticality attribute. You can use this function to set the host criticality by using `"criticality"` as the attribute `$id`.

See Example: Setting the Host Criticality, page 2-43 for an example of this function used in a script.

Use this syntax:

```
SetAttributeValue($source_type_id, $source_id, $addr_string, $attrib_list, $id,
$value)
```

Table 2-23 SetAttributeValue Fields

Field	Description	Required	Allowed Values
$source_type_id	Indicates the type of the host input source.	Yes	`"Application" or "Scanner"` Note you should set the `$source_type_id` variable to contain a value before invoking the `SetAttributeValue` function, and then reference `$source_type_id` in your function call. For more information, see Setting the Source Type, page 2-2.
$source_id	Indicates the source ID for the source adding the host input.	Yes	*`"source_id"`* Note you should set the `$source_id` variable to contain the source ID before invoking the `SetAttributeValue` function, and then reference `$source_id` in your function call. For more information, see Obtaining a Source ID, page 2-2.
$addr_string	Indicates the string containing the IP address or addresses for the affected hosts.	Yes (unless attribute lists are provided)	A comma-separated list of IP addresses, CIDR blocks, and ranges of IP addresses, with each address, block, or range enclosed in double quotes.

$attrib_list	Indicates the host attribute or attributes specifying the hosts affected by the host input.	Yes (unless IP addresses are provided)	A list of attribute value hash pairs of the format:
			`{attribute => "Department",`
			`value => "Development"},`
			Note that `$attrib_list` must be an array or reference an array.

$id	Indicates the host attribute name.	Yes	Valid attribute names consisting of alphanumeric characters or spaces, enclosed in double quotes.
$value	Indicates the host attribute value.	Yes	Valid attribute values for the named attribute, consisting of alphanumeric characters or spaces, enclosed in double quotes. If a value is passed in for a list attribute, `$value` must be an existing named value for the list attribute.

DeleteAttributeValue

You can use the `DeleteAttributeValue` function to remove an attribute value for a host.

Use this syntax:

```
DeleteAttributeValue($source_type_id, $source_id, $addr_string, $attrib_list, $id)
```

Table 2-24 DeleteAttributeValue Fields

Field	Description	Required	Allowed Values
$source_type_id	Indicates the type of the host input source.	Yes	`"Application" or "Scanner"` Note you should set the `$source_type_id` variable to contain a value before invoking the `DeleteAttributeValue` function, and then reference `$source_type` in your function call. For more information, see Setting the Source Type, page 2-2.
$source_id	Indicates the source ID for the source adding the host input.	Yes	*`"source_id"`* Note you should set the `$source_id` variable to contain the source ID before invoking the `DeleteAttributeValue` function, and then reference `$source_id` in your function call. For more information, see Obtaining a Source ID, page 2-2.
$addr_string	Indicates the string containing the IP address or addresses for the affected hosts.	Yes (unless attribute lists are provided)	A comma-separated list of IP addresses, CIDR blocks, and ranges of IP addresses, with each address, block, or range enclosed in double quotes.

$attrib_list	Indicates the host attribute or attributes specifying the hosts affected by the host input.	Yes (unless IP addresses are provided)	A list of attribute value hash pairs of the format:
			`{attribute => "Department",`
			`value => "Development"},`
			Note that `$attrib_list` must be an array or reference an array.

$id	Indicates the host attribute name.	Yes	Valid attribute names consisting of alphanumeric characters or spaces, enclosed in double quotes.

SetCriticality

You can use the `SetCriticality` function to set the criticality level for a host.

Use this syntax:

```
SetCriticality($source_type_id, $source_id, $addr_string, $attrib_list, $criticality)
```

Table 2-25 SetCriticality Fields

Field	Description	Required	Allowed Values
$source_type_id	Indicates the type of the host input source.	Yes	`"Application" or "Scanner"` Note you should set the `$source_type_id` variable to contain a value before invoking the `SetCriticality` function, and then reference `$source_type_id` in your function call. For more information, see Setting the Source Type, page 2-2.
$source_id	Indicates the source ID for the source adding the host input.	Yes	*`"source_id"`* Note you should set the `$source_id` variable to contain the source ID before invoking the `SetCriticality` function, and then reference `$source_id` in your function call. For more information, see Obtaining a Source ID, page 2-2.
$addr_string	Indicates the string containing the IP address or addresses for the affected hosts.	Yes (unless attribute lists are provided)	A comma-separated list of IP addresses, CIDR blocks, and ranges of IP addresses, with each address, block, or range enclosed in double quotes.

$attrib_list	Indicates the host attribute or attributes specifying the hosts affected by the host input.	Yes (unless IP addresses are provided)	A list of attribute value hash pairs of the format:
			`{attribute => "Department",`
			`value => "Development"},`
			Note that `$attrib_list` must be an array or reference an array.

$criticality	Indicates the criticality level for the host.	Yes	The identification number or string for the criticality level: • `"0"` or `"None"` • `"1"` or `"Low"` • `"2"` or `"Medium"` • `"3"` or `"High"`

Vulnerabilities Functions

You can use the vulnerabilities functions to update the status of vulnerabilities on a host.

For more information, see the following sections:

- SetInvalidVulns, page 2-31
- SetValidVulns, page 2-32

SetInvalidVulns

You can use the `SetInvalidVulns` function to deactivate vulnerabilities on a host or set of hosts. For the function call to be effective, the vulnerability must exist on the host and be set to valid. When you use `SetInvalidVulns` to deactivate a third-party vulnerability for a host, it deletes the vulnerability from the host.

Use this syntax:

```
SetInvalidVulns($source_type, $source_id, $addr_string, $attrib_list, $vulns,
$vuln_type)
```

Table 2-26 SetInvalidVulns Fields

Field	Description	Required	Allowed Values
$source_type_id	Indicates the type of the host input source.	Yes	`"Application" or "Scanner"` Note you should set the `$source_type_id` variable to contain a value before invoking the `SetInvalidVulns` function, and then reference `$source_type_id` in your function call. For more information, see Setting the Source Type, page 2-2.
$source_id	Indicates the source ID for the source adding the host input.	Yes	*`"source_id"`* Note you should set the `$source_id` variable to contain the source ID before invoking the `SetInvalidVulns` function, and then reference `$source_id` in your function call. For more information, see Obtaining a Source ID, page 2-2.
$addr_string	Indicates the string containing the IP address or addresses for the affected hosts.	Yes (unless attribute lists are provided)	A comma-separated list of IP addresses, CIDR blocks, and ranges of IP addresses, with each address, block, or range enclosed in double quotes.

$attrib_list	Indicates the host attribute or attributes specifying the hosts affected by the host input.	Yes (unless IP addresses are provided)	A list of attribute value hash pairs of the format:
			`{attribute => "Department",`
			`value => "Development"},`
			Note that `$attrib_list` must be an array or reference an array.

$vulns	Supplies information about the vulnerability to be set to invalid.	Yes	Uses a hash of vulnerability keys to set vulnerability information. For more information, see Vulnerability Keys, page 2-33.
$vuln_type	Indicates the type of the vulnerability.	Yes	Any of the following: • `rna` • name of custom third-party vulnerability map set For more information on mapping third-party vulnerabilities, see the *FireSIGHT System User Guide* or see SetCurrent3rdPartyMap, page 2-34.

SetValidVulns

You can use the `SetValidVulns` function to activate vulnerabilities on a host or set of hosts. Once you set a vulnerability as Valid for a host, Defense Center assigns a red impact to the event if the SID in the event is mapped to the valid vulnerability. For the function call to be effective for a Cisco vulnerability, it must exist on the host and be set to invalid. When you use `SetValidVulns` to activate a third-party vulnerability for a host, it adds the vulnerability to the host.

Use this syntax:

```
SetValidVulns($source_type_id, $source_id, $addr_string, $attrib_list, $vulns,
$vuln_type)
```

Table 2-27 SetValidVulns Fields

Field	Description	Required	Allowed Values
$source_type_id	Indicates the type of the host input source.	Yes	`"Application" or "Scanner"` Note you should set the `$source_type_id` variable to contain a value before invoking the `SetValidVulns` function, and then reference `$source_type_id` in your function call. For more information, see Setting the Source Type, page 2-2.
$source_id	Indicates the source ID for the source adding the host input.	Yes	*`"source_id"`* Note you should set the `$source_id` variable to contain the source ID before invoking the `SetValidVulns` function, and then reference `$source_id` in your function call. For more information, see Obtaining a Source ID, page 2-2.
$addr_string	Indicates the string containing the IP address or addresses for the affected hosts.	Yes (unless attribute lists are provided)	A comma-separated list of IP addresses, CIDR blocks, and ranges of IP addresses, with each address, block, or range enclosed in double quotes.

$attrib_list	Indicates the host attribute or attributes specifying the hosts affected by the host input.	Yes (unless IP addresses are provided)	A list of attribute value hash pairs of the format:
			`{attribute => "Department",`
			`value => "Development"},`
			Note that `$attrib_list` must be an array or reference an array.

$vulns	Supplies information about the vulnerability to be activated.	Yes	Uses a hash of vulnerability keys to set vulnerability information. For more information, see Vulnerability Keys, page 2-33.
$vuln_type	Indicates the type of the vulnerability.	Yes	Any of the following: • `rna` • name of custom third-party vulnerability map set For more information on mapping third-party vulnerabilities, see the *FireSIGHT System User Guide* or see SetCurrent3rdPartyMap, page 2-34.

Vulnerability Keys

The `$vulns` field for the `SetValidVulns` and the `SetInvalidVulns` functions and the `$mapping_vuln_list` field for the `AddScanResult` function use a vulnerability definition hash with some or all of the keys defined in the following tables.

Because you can map vulnerabilities to multiple servers running on a system, the `port` and `proto` information must be provided in order to mark server vulnerabilities.

The following tables provide information on the keys you can use with the `$vulns` and `$mapping_vuln_list` fields.

Table 2-28 Keys for Vulnerability Mapping

Key	Data Type	Used by	Definition
cve_ids	string	`$mapping_vuln_list`	A comma-separated list of CVE IDs, with each ID enclosed in single quotes. If this field, `vuln_id`, and `bugtraq_ids` are empty, this is a generic scan result Use this key to specify the CVE ID for any vulnerabilities on the hosts.
bugtraq_ids	uint	`$mapping_vuln_list`	A comma-separated list of BugTraq IDs, with each ID enclosed in single quotes. If this field, `vuln_id`, and `cve_ids` are empty, this is a generic scan result. Use this key to specify the BugTraq ID for any vulnerabilities on the hosts.
vuln_id	string	`$vulns` and `$mapping_vuln_list`	A string, enclosed in single quotes. If this field, bugtraq_ids, and cve_ids are empty, this is a generic scan result. Use this key to indicate the vulnerability ID for the vulnerability. For third-party vulnerabilities, note that you must map the third-party vulnerability ID and reference the vulnerability map set in the `vuln_type` field. For more information, see Setting a Third-Party Vulnerability Map, page 2-3.

Table 2-29 Keys for Server Identity

Key	Data Type	Applies to	Definition
port	uint	`$vulns` and `$mapping_vuln_list`	With the `proto` key, use this key to specify the server that may be affected by this vulnerability.
proto	string	`$vulns` and `$mapping_vuln_list`	With the `port` key, use this key to specify the server that may be affected by this vulnerability, using either the strings `tcp` or `udp` or the appropriate protocol IDs `6` (tcp) or `17` (udp).

Table 2-30 Keys for Rendering

Key	Data Type	Applies to	Definition
name	string	`$mapping_vuln_list`	Use this key to supply the vulnerability name used by the third-party application.
desc	string	`$mapping_vuln_list`	Use this key to supply the vulnerability description used by the third-party application.

Third-Party Mapping Functions

You can use the third-party mapping functions to invoke a set of product mappings on a host. When you invoke a map set, mappings from third-party application names to Cisco product definitions apply for hosts affected by any following function calls in your script. When you unset a product map, the settings on the host revert to `Unidentified`.

For more information, see the following sections:

- SetCurrent3rdPartyMap, page 2-34
- UnsetCurrent3rdPartyMap, page 2-35

SetCurrent3rdPartyMap

You can use this function to set the current third-party map for the current session. You create third-party mappings using the Defense Center web interface to set up a reusable map between each third-party vendor, product, and version combination and the corresponding Cisco product definition. If you set a third-party map and then add or set host operating system or server data that includes third-party application names included in the map, the system uses the mappings to map the Cisco product definition, and associated vulnerabilities, to each host where the input occurs.

For instance, you could create a map set called `"Custom Utility"`, in which you could define the third-party strings as follows:

- **Vendor String** - `Microsoft`
- **Product String** - `Win2k`

You could select the following Cisco product mapping in the map set:

- **Vendor** - `Microsoft, Corp.`
- **Product** - `Windows 2000`
- **Patch** - `SP3`

If you set this product map by calling `SetCurrent3rdPartyMap("Custom Utility")`, it maps `"Microsoft Win2k"` to the VDB entry for the `"Microsoft Windows 2000 SP3"` product.

If you want to import host data for a host operating system, you can then call the `SetOS` function and only specify the vendor, product, and version string. The host input API processor automatically converts the strings specified in the product map into the VDB parameters mapped to those strings. See Setting a Third-Party Product Map, page 2-3 for more information on creating 3rd party mapping sets.

See Example: Setting the Operating System on the Host, page 2-41 for an example of this function used in a script.

Use this syntax:

```
SetCurrent3rdPartyMap($map_name)
```

where `$map_name` is the name of the third-party product map, enclosed in double quotes, that you created using the Defense Center web interface.

UnsetCurrent3rdPartyMap

This function unsets the current active third-party map.

Use this syntax:

```
UnsetCurrent3rdPartyMap()
```

AddScanResult Function

This function adds scan results from a third-party vulnerability scanner and maps each vulnerability to a BugTraq or CVE ID.

If you import a scan result with a vulnerability for a server on a host, but do not use `AddService` to import the server to the host, the application protocol for the server will show a value of `unknown` in the host profile. If you import scan results using this function, be sure to edit the source definition for the input source in your network discovery policy to set the identity source type to Scanner.

For examples of how to use `AddScanResult` in a script, see Example: Adding a Scan Result to a Host, page 2-44, Example: Adding a Generic Scan Result to a Host, page 2-44, and Full Example Script, page 2-45.

Use this syntax:

```
AddScanResult($scanner_id,$ipaddr,$mapping_vuln_list,$generic_item_list,$flag)
```

Table 2-31 AddScanResult Fields

Field	Description	Required	Allowed Values
$scanner_id	Indicates the scanner ID for the scanner that obtained the scan results.	Yes	*"scanner_id"* where `scanner_id` is a string indicating the name of the scanner that is the source of the vulnerability data you add. To add scan results from a previously used scanner, indicate the specific scanner name listed in system policies on the Defense Center where you added the results. Adding results from a new scanner ID adds that scanner to the system policy. New scanners are added as the lowest priority by default. If you want to change the priority of the scanner, you can do so in the system policy. For more information, see the *FireSIGHT System User Guide*.
$ipaddr	Indicates the IP address of the scanned hosts.	Yes	A comma-separated list of IP addresses, CIDR blocks, and ranges of IP addresses, with each address, block, or range enclosed in double quotes.

Table 2-31 AddScanResult Fields (continued)

Field	Description	Required	Allowed Values

$mapping_vuln_list	Indicates scan results with vulnerability IDs for the affected hosts.	Yes	A list of vulnerability hash values of the format:
			`{`
			`'cve_ids' => [ '2003-0988' ],`
			`'bugtraq_ids' => [ 9506,9507,9508 ],`
			'vuln_id' => 10150,
			'port' => 107,
			'proto' => 17,
			'name' => 'Using NetBIOS to retrieve info from a Windows host',
			'desc' => 'The following 2 NetBIOS names have been gathered...',
			`}`
			Note that `$mapping_vuln_list` must be an array or reference an array.
			Uses a hash of vulnerability keys to set vulnerability information. For more information, see Vulnerability Keys, page 2-33.

Table 2-31 AddScanResult Fields (continued)

Field	Description	Required	Allowed Values

$generic_item_list	Indicates scan results without vulnerability IDs for the affected hosts.	Yes	A list of vulnerability hash values of the format:
			`{`
			`'port' => 107,`
			`'proto' => 17,`
			`'name' => 'Using NetBIOS to retrieve info from a Windows host',`
			`'desc' => 'The following 2 NetBIOS names have been gathered...',`
			`}`
			Note that `$generic_item_list` must be an array or reference an array.
			Uses a hash of vulnerability keys to set vulnerability information. For more information, see Vulnerability Keys, page 2-33.

$flag	Indicates how the scan result should be processed.	Yes	The number for the action to be performed: • `1` – update scan result: send this flag to append the scan result to existing scan results on the host • `2` – delete scan result: send this flag to delete the specific scan result indicated by the values you supply • `3` – delete all vulnerability scan results: send this flag to delete all scan results with vulnerability IDs on the specified hosts (The `$mapping_vuln_list` field should be empty when using this flag.) • `4` – delete all generic scan results: send this flag to delete all scan results without vulnerability IDs on the specified hosts (The $generic_item_list field should be empty when using this flag.) • `5` – delete all scan results: send this flag to delete all scan results on the specified hosts (The `$mapping_vuln_list` and $generic_item_list field should be empty when using this flag.)

DeleteScanResult

This function deletes scan results from a third-party vulnerability scanner and maps each vulnerability to a BugTraq or CVE ID.

For examples of how to use `DeleteScanResult` in a script, see Example: Deleting a Scan Result from a Host, page 2-45 and Full Example Script, page 2-45.

Use this syntax:

```
DeleteScanResult($ipaddr,$scanner_id,$mapping_vuln_list, $generic_item_list,$flag)
```

Table 2-32 DeleteScanResult Fields

Field	Description	Required	Allowed Values
$scanner_id	Indicates the scanner ID for the scanner that obtained the scan results.	Yes	*"scanner_id"* where `scanner_id` is a string indicating the name of the scanner that is the source of the vulnerability data you add. To add scan results from a previously used scanner, indicate the specific scanner name listed in system policies on the Defense Center where you added the results. Adding results from a new scanner ID adds that scanner to the system policy. New scanners are added as the lowest priority by default. If you want to change the priority of the scanner, you can do so in the system policy. For more information, see the *FireSIGHT System User Guide*.
$ipaddr	Indicates the IP address of the scanned hosts.	Yes	A comma-separated list of IP addresses, CIDR blocks, and ranges of IP addresses, with each address, block, or range enclosed in double quotes.

$mapping_vuln_list	Indicates scan results with vulnerability IDs for the affected hosts.	No	A list of vulnerability hash values of the format:
			`{`
			`'vuln_id' => 10150,`
			`'port' => 107,`
			`'proto' => 17,`
			`}`
			Note that `$mapping_vuln_list` must be an array or reference an array.
			Uses a hash of vulnerability keys to set vulnerability information. For more information, see Vulnerability Keys, page 2-33.
			If vuln_id is used, only the specified vuln_id is removed.

Table 2-32 DeleteScanResult Fields (continued)

Field	Description	Required	Allowed Values

$generic_item_list	Indicates scan results without vulnerability IDs for the affected hosts.	Yes	A list of vulnerability hash values of the format:
			`{`
			`'port' => 107,`
			`'proto' => 17,`
			`'name' => 'Using NetBIOS to retrieve info from a Windows host',`
			`'desc' => 'The following 2 NetBIOS names have been gathered...',`
			`}`
			Note that `$generic_item_list` must be an array or reference an array.
			Uses a hash of vulnerability keys to set vulnerability information. For more information, see Vulnerability Keys, page 2-33.

$flag	Indicates how the scan result should be processed.	Yes	The number for the action to be performed: • `1` - update scan result: send this flag to append the scan result to existing scan results on the host • `2` - delete scan result: send this flag to delete the specific scan result indicated by the values you supply • `3` - delete all vulnerability scan results: send this flag to delete all scan results with vulnerability IDs on the specified hosts (The `$mapping_vuln_list` field should be empty when using this flag.) • `4` - delete all generic scan results: send this flag to delete all scan results without vulnerability IDs on the specified hosts (The $generic_item_list field should be empty when using this flag.) • `5` - delete all scan results: send this flag to delete all scan results on the specified hosts (The `$mapping_vuln_list` and $generic_item_list field should be empty when using this flag.)

Example Host Input API Scripts

The following code samples illustrate how you might construct a script to import data using the host input API.

The following sections, in sequential order, show each portion of the script:

Example: Invoking the Host Input Module

The example script starts by declaring the use of the HostInput Perl module:

```
#!/usr/bin/perl
use SF::SFDataCorrelator::HostInput;
```

Example: Setting the Source Type

The example script next initiates the `$source_type_id` variable and sets it to `Scanner`:

```
# Set the Source Type
my $source_type_id = SF::SFDataCorrelator::HostInput::GetSourceTypeIDByName
('Scanner');
```

This source type value is used in most of the host input function calls. In host input events resulting from this input function, the Source Type lists as `Application`.

Example: Setting the Source ID

After the `$source_type_id` value is set, the example script uses the `GetSourceAppIDByName` function to set the `$source_id` value to `AssetManageApp`:

```
# Set the Application ID
SF::SFDataCorrelator::HostInput::SetCurrentSource ($source_type_id,"CustomApp");
# Retrieve the Application ID you set
my $source_id =
SF::SFDataCorrelator::HostInput::GetCurrentSource();
```

This source ID value is used in most of the host input function calls. In host input events resulting from this input function, the Source Type lists as `Application: AssetManageApp`.

Example: Adding a Host to the Network Map

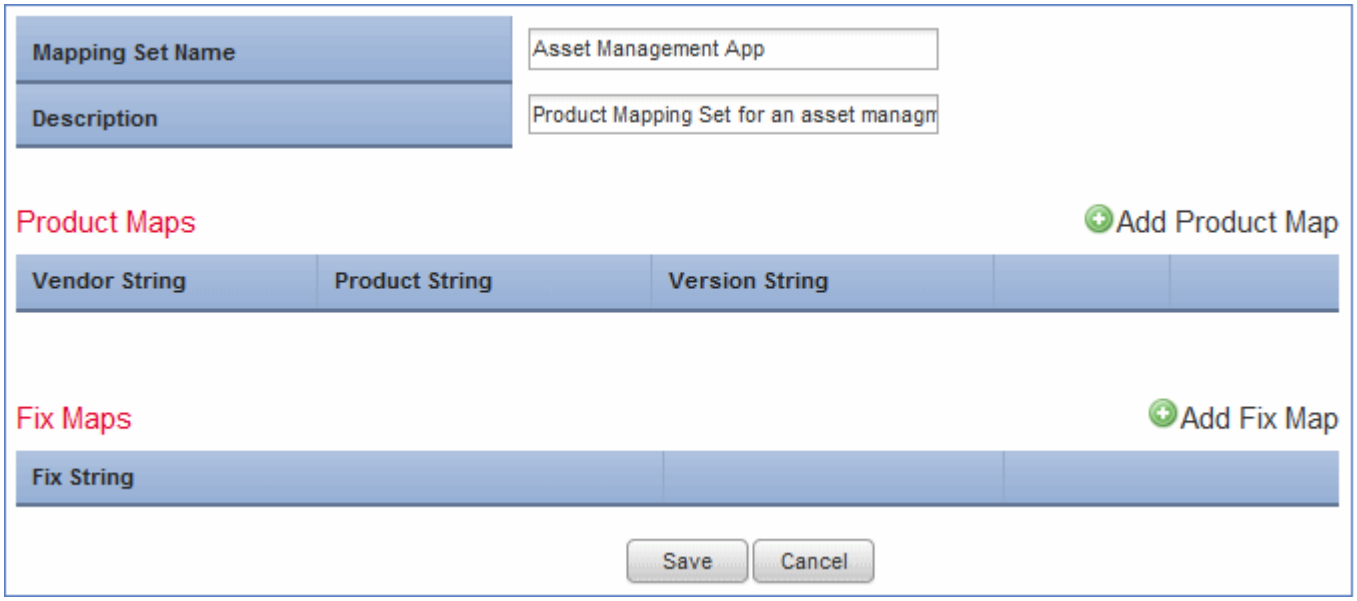
After the script establishes use of the `HostInput` module and sets the source type and source id values, it can begin to import data. The first import function called is the `AddHost` function.

```
# Add an IP host with a Primary MAC address
if ($retval = SF::SFDataCorrelator::HostInput::AddHost(
$source_type_id, $source_id, "1.2.3.4", "01:02:03:04:05:06"))
{
warn "AddHost Failed with error $retval";
exit;
}
```

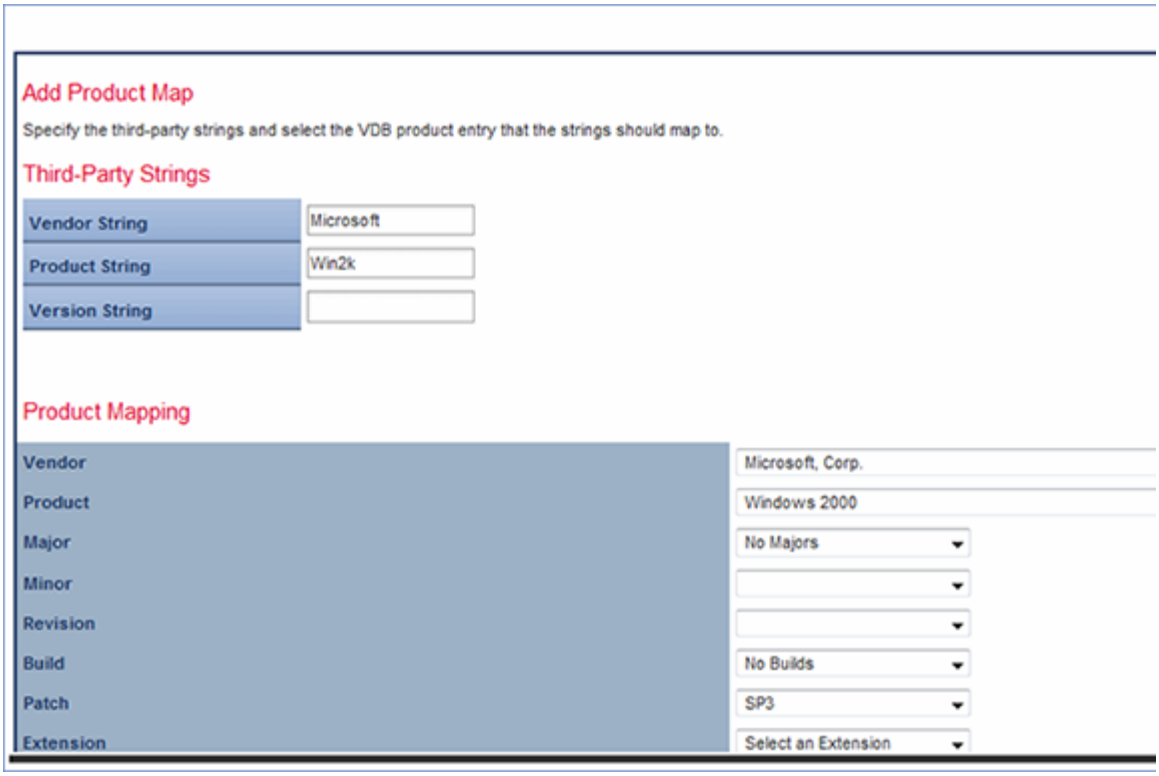
Note that in addition to the IP address, the function sets a primary MAC address of 01:02:03:04:05:06.

Example: Setting the Operating System on the Host

After the script adds the host, it sets the operating system value for the host. To simplify the `SetOS` call, a product map called `Asset Management App` is created using the Defense Center web interface:



The Asset Management App map set contains a product map mapping the third-party product name `Microsoft Win2K` to the Cisco product definition for Microsoft Windows 2000 SP3:



The script sets the product map to `"Asset Management App"`:

```
# Set the current product map set to "Asset Management App"
if ($retval = SF::SFDataCorrelator::HostInput::SetCurrent3rdPartyMap ("Asset
Management App"))
{
warn "SetCurrent3rdPartyMap Failed with error $retval";
exit;
}
```

The script then uses the `vendor_str` and `product_str` keys to set the operating system display name to `Microsoft Windows 2000`, mapping that third-party operating system name to the Cisco product definition as defined in the `Asset Management App` product map set because the product map set is already in effect:

```
# Set the operating system on the newly created host
if ($retval = SF::SFDataCorrelator::HostInput::SetOS(
$source_type_id, $source_id, "1.2.3.4", [],
{
vendor_str => 'Microsoft',
product_str => 'Windows 2000',
}))
{
warn "SetOS Failed with error $retval";
exit;
}
```

Example: Adding a Protocol to the Host

The script next adds the `ospf` protocol to the `1.2.3.4` host. Note that the protocol type for the protocol is `"xport"`.

```
# Add the transport protocol "ospf" to the newly created host
if ($retval = SF::SFDataCorrelator::HostInput::AddProtocol
($source_type_id, $source_id, "1.2.3.4", [], [],
"ospf", "xport" ))
{
warn "AddProtocol Failed with error $retval";
exit;
}
```

Example: Adding a Server to the Host

The script then uses the `AddService` function to add the `OpenSSH` server to the `1.2.3.4` host:

```
# Add the OpenSSH server to the host
if ($retval = SF::SFDataCorrelator::HostInput::AddService(
$source_type_id, $source_id, "1.2.3.4", [],
{
port => 22,
proto => 'tcp',
vendor_str => 'OpenSSH',
version_str => '4.1',
service_name => 'ssh'
}))
{
warn "AddService Failed with error $retval";
exit;
}
```

Note that the `$service` hash is used to set the port to `22`, the protocol to `tcp`, the vendor display string to `OpenSSH`, the version display string to `4.1`, and the server name to `ssh`.

Example: Setting the Host Criticality

Next, the `SetAttributeValue` function is used to set the host criticality for the `1.2.3.4` host to Medium:

```
# Set the Criticality of the host to "Medium"
if ($retval = SF::SFDataCorrelator::HostInput::SetAttributeValue
($source_type_id, $source_id, "1.2.3.4", [],
"Criticality", "medium" ))
{
warn "SetAttributeValue Failed with error $retval";
exit;
}
```

Note that the attribute name is set to "`Criticality`" and the attribute value is set to `"medium"`.

Example: Adding a Client Application to Multiple Hosts

Finally, the script adds a client application named `BMC Remedy` to every host with a Medium criticality.

```
# Add a Client Application to all hosts with a Criticality Value of "Medium"
if ($retval = SF::SFDataCorrelator::HostInput::AddClientApp(
$source_type_id, $source_id, "",
[ { attribute => "Criticality", value => "medium"} ],
```

```
"BMC Remedy", "Asset Manager", "0.0" ))
{
warn "AddClientApp Failed with error $retval";
exit;
}
```

Note that the client application ID is set to `BMC Remedy`, the client application type is set to `Asset Manager`, and the version is set to `0.0`.

Example: Adding a Scan Result to a Host

The script adds the scan results from a third-party scanner that scanned host `1.2.3.4` to the network map.

```
$params=
{
'scanner_id' => 'Scanner_ID',
'ip_address' => '1.2.3.4'
};
$mapping_vuln_list = [
{
'cve_ids' => [ '2003-0988' ],
'vuln_id' => '10150A',
# 3rd party scanner vuln id
'port' => 107,
'proto' => 17,
'name' => 'Using NetBIOS to retrieve info from a Windows host',
'desc' => 'The following 2 NetBIOS names have been gathered ...',
},
{
'cve_ids' => [],
'bugtraq_ids' => [ 29506,29507,29508 ],
'vuln_id' => '10159B',
# 3rd party scanner vuln id
'port' => 109,
'proto' => 17,
'name' => 'Name 2',
'desc' => 'description 2',
},
];
$generic_item_list = [];
$flag =
getPkgVar("SF::SFDataCorrelator::UserMessage",'$UPDATE_SCAN_RESULT');
# Send message indicating that you are updating scan result and set
# flag to append the scan result to existing scan results on the host
SF::SFDataCorrelator::HostInput::AddScanResult($params,
$mapping_vuln_list,$generic_item_list,$flag);
```

Example: Adding a Generic Scan Result to a Host

The script adds a generic scan result to the network map.

```
my ($scanner_id,$vuln_id,$mapping_vuln_id);
my $ip = '1.2.3.4';
$scanner_id = 'Scanner_ID';
$mapping_vuln_list = [];
$generic_item_list = [
{
'port' => 107,
'proto' => 17,
```

```
'name' => 'Using NetBIOS to retrieve info from a Windows host',
'desc' => 'The following 2 NetBIOS names have been gathered ...',
} ];
my $rval = SF::SFDataCorrelator::HostInput::AddScanResult($params,
$mapping_vuln_list,$generic_item_list,$flag);
```

Example: Deleting a Scan Result from a Host

The script deletes a scan result from the network map.

```
my ($scanner_id,$vuln_id,$mapping_vuln_id);
my $ip = '1.2.3.4';
$scanner_id = 'Scanner_ID';
$mapping_vuln_list = [
{
'cve_ids' => [ '2003-0988' ],
'vuln_id' => '10150A',
# 3rd party scanner vuln id
'port' => 107,
'proto' => 17,
'name' => 'Using NetBIOS to retrieve info from a Windows host',
'desc' => 'The following 2 NetBIOS names have been gathered ...',
},
{
'cve_ids' => [],
'bugtraq_ids' => [ 29506,29507,29508 ],
'vuln_id' => '10159B',
# 3rd party scanner vuln id
'port' => 109,
'proto' => 17,
'name' => 'Name 2',
'desc' => 'description 2',
},
];
my $rval = SF::SFDataCorrelator::HostInput::DeleteScanResult($params,
$mapping_vuln_list,$generic_item_list,$flag);
```

Full Example Script

The full script explained in the sections above looks like this:

```
#!/usr/bin/perl

use FlyLoader;
use SF::SFDataCorrelator::HostInput;

# Set the Source Type
my $source_type_id = SF::SFDataCorrelator::HostInput::GetSourceTypeIDByName
('Scanner');

# Set the Application ID
SF::SFDataCorrelator::HostInput::SetCurrentSource ($source_type_id,"CustomApp");
# Retrieve the Application ID you set
my $source_id =
SF::SFDataCorrelator::HostInput::GetCurrentSource();

# Add an IP host with a Primary MAC address
if ($retval = SF::SFDataCorrelator::HostInput::AddHost(
$source_type_id, $source_id, "1.2.3.4", "01:02:03:04:05:06" ))
{
```

```
warn "AddHost Failed with error $retval";
exit;
}

# From UI: Policies > Application Detectors > User Third-Party Mappings, Set the
current product map to
# 'Asset Management App' before running SetCurrent3rdPartyMap()
if ($retval = SF::SFDataCorrelator::HostInput::SetCurrent3rdPartyMap
("Asset Management App"))
{
warn "SetCurrent3rdPartyMap Failed with error $retval";
exit;
}

# Set the operating system on the newly created host
if ($retval = SF::SFDataCorrelator::HostInput::SetOS(
$source_type_id, $source_id, "1.2.3.4", [],
{
vendor_str => 'Microsoft',
product_str => 'Windows 2000',
}))
{
warn "SetOS Failed with error $retval";
exit;
}

# Add the transport protocol "ospf" to the newly created host
if ($retval = SF::SFDataCorrelator::HostInput::AddProtocol
($source_type_id, $source_id, "1.2.3.4", [], [],
"ospf", "xport" ))
{
warn "AddProtocol Failed with error $retval";
exit;
}

# Add the OpenSSH server to the host
if ($retval = SF::SFDataCorrelator::HostInput::AddService(
$source_type_id, $source_id, "1.2.3.4", [],
{
port => 22,
proto => 'tcp',
vendor_str => 'OpenSSH',
version_str => '4.1',
service_name => 'ssh'
}))
{
warn "AddService Failed with error $retval";
exit;
}

# Set the Criticality of the host to "Medium"
if ($retval = SF::SFDataCorrelator::HostInput::SetAttributeValue(
$source_type_id, $source_id, "1.2.3.4", [],
"Criticality", "medium" ))
{
warn "SetAttributeValue Failed with error $retval";
exit;
}

# Add a Client Application to all hosts with a Criticality Value of "Medium"
if ($retval = SF::SFDataCorrelator::HostInput::AddClientApp(
$source_type_id, $source_id, "",
[ { attribute => "Criticality", value => "medium"} ],
"BMC Remedy", "Asset Manager", "0.0" ))
```

```
{
warn "AddClientApp Failed with error $retval";
exit;}

# Update an existing scan result on the host to reflect a new scan result
$params=
{
'scanner_id' => 'Scanner_ID',
'ip_address' => '1.2.3.4'
};
$mapping_vuln_list = [
{
'cve_ids' => [ '2003-0988' ],
'vuln_id' => '10150A',
# 3rd party scanner vuln id
'port' => 107,
'proto' => 17,
'name' => 'Using NetBIOS to retrieve info from a Windows host',
'desc' => 'The following 2 NetBIOS names have been gathered ...',
},
{
'cve_ids' => [],
'bugtraq_ids' => [ 29506,29507,29508 ],
'vuln_id' => '10159B',
# 3rd party scanner vuln id
'port' => 109,
'proto' => 17,
'name' => 'Name 2',
'desc' => 'description 2',
},
];
$generic_item_list = [];
$flag =
getPkgVar("SF::SFDataCorrelator::UserMessage",'$UPDATE_SCAN_RESULT');
# Send message indicating that you are updating scan result and set
# flag to append the scan result to existing scan results on the host
SF::SFDataCorrelator::HostInput::AddScanResult($params,
$mapping_vuln_list,$generic_item_list,$flag);

# Add a generic scan result
my ($scanner_id,$vuln_id,$mapping_vuln_id);
my $ip = '1.2.3.4';
$scanner_id = 'Scanner_ID';
$mapping_vuln_list = [];
$generic_item_list = [
{
'port' => 107,
'proto' => 17,
'name' => 'Using NetBIOS to retrieve info from a Windows host',
'desc' => 'The following 2 NetBIOS names have been gathered ...',
} ];
my $rval = SF::SFDataCorrelator::HostInput::AddScanResult($params,
$mapping_vuln_list,$generic_item_list,$flag);

#Delete the scan result
my ($scanner_id,$vuln_id,$mapping_vuln_id);
my $ip = '1.2.3.4';
$scanner_id = 'Scanner_ID';
$mapping_vuln_list = [
{
'cve_ids' => [ '2003-0988' ],
'vuln_id' => '10150A',
# 3rd party scanner vuln id
'port' => 107,
```

```
'proto' => 17,
'name' => 'Using NetBIOS to retrieve info from a Windows host',
'desc' => 'The following 2 NetBIOS names have been gathered ...',
},
{
'cve_ids' => [],
'bugtraq_ids' => [ 29506,29507,29508 ],
'vuln_id' => '10159B',
# 3rd party scanner vuln id
'port' => 109,
'proto' => 17,
'name' => 'Name 2',
'desc' => 'description 2',
},
];
my $rval = SF::SFDataCorrelator::HostInput::DeleteScanResult($params,
$mapping_vuln_list,$generic_item_list,$flag);
```

Using the Host Input Import Tool

You can import data to the network map by creating an import file and processing it with the host input import tool.

See the following sections for more information:

- Writing Host Input Import Files, page 3-3
- Host Input Import Syntax, page 3-6
- Running a Host Input Import, page 3-31

Preparing to Run Host Input Imports

The host input import tool depends on product mapping information you supply using the Defense Center web interface to map third-party product, fix, and vulnerability names and IDs to definitions in the Cisco database. Depending on the data you plan to import, you may need to perform the configuration steps described in the following sections before you run your import:

- Creating a Third-Party Vulnerability Map, page 3-1
- Creating a Third-Party Product Map, page 3-2

Creating a Third-Party Vulnerability Map

If you want to import data including third-party vulnerabilities and use that data for impact correlation, you must create a third-party vulnerability map set before importing the data. The third-party map set allows the system to translate the third-party vulnerability ID to the corresponding Cisco vulnerability ID. If you do not map a third-party vulnerability before import, the vulnerability does not map to a Cisco vulnerability ID and cannot be used for impact correlation. You can create a map set in two ways: using the Defense Center web interface or using the AddScanResult function. If you import scan results using this function, be sure to edit the source definition for the input source in your network discovery policy to set the identity source type to Scanner.

For more information on mapping third-party vulnerabilities through the web interface, see the *FireSIGHT System User Guide*. For more information on the AddScanResult function, see AddScanResult Function, page 3-21.

Creating a Third-Party Product Map

When you import operating system or server data to a host, you can map third-party product name details to a Cisco product definition. You can create a third-party product map through the Defense Center web interface.

The third-party product map set allows the system to translate the third-party vendor, product, and version to the corresponding Cisco definition. When you set a third-party product map containing a server definition or an operating system definition, within the same script you can then just define the display strings for a third-party server or operating system when you add or set it using the API.

If you map third-party fixes to Cisco fix definitions using a third-party product map, set the product map, and then add fixes to hosts using the third-party fix name, the system maps the fixes to the appropriate Cisco fix definitions and deactivates vulnerabilities addressed by the fix.

To map a third-party product to a Cisco product definition:

Access: Admin

Step 1 Select **Policies > Application Detectors**, then click **User Third-Party Mappings**.

The User Third-Party Mappings page appears.

Step 2 You have two choices:

- To edit an existing map set, click **Edit** next to the map set.
- To create a new map set, click **Create Product Map Set**.

The Edit Third-Party Product Mappings page appears.

Step 3 Type a name for the mapping set in the **Mapping Set Name** field.

Step 4 Type a description in the **Description** field.

Step 5 You have two choices:

- To map a third-party product, click **Add Product Map.**
- To edit an existing third-party product map, click **Edit** next to the map set.

The Add Product Map page appears.

Step 6 Type the vendor string used by the third-party product in the **Vendor String** field.

Step 7 Type the product string used by the third-party product in the **Product String** field.

Step 8 Type the version string used by the third-party product in the **Version String** field.

Step 9 In the **Product Mappings** section, select the operating system, product, and versions you want to use for vulnerability mapping from the following lists (if applicable):

- **Vendor**
- **Product**
- **Major Version**
- **Minor Version**
- **Revision Version**
- **Build**
- **Patch**
- **Extension**

For example, if you want a host running a product whose name consists of third-party strings to use the vulnerabilities from Red Hat Linux 9, select **Redhat, Inc.** as the vendor, **Redhat Linux** as the product, and **9** as the version.

Step 10 Click **Save**.

After you create the third-party product map, you can import data using the `SetOS`, `SetService`, or `AddService` functions. Note the third-party product name details and Cisco product definition before importing data.

To locate third-party and Cisco product details:

Access: Admin

Step 1 Select **Policies > Application Detectors.**

The Application Detectors page appears.

Step 2 Select **User Third-Party Mappings**.

The Third-Party Product Mappings page appears.

Step 3 Click the edit icon ( ) for your product map set.

The Edit Third-Party Product Mappings page appears.

Step 4 Click the edit icon ( ) for your product map.

The Add Product Map pop-up window appears. Note the **Vendor String**, **Product String**, and **Version String** values.

For more information on mapping third-party products, see the *FireSIGHT System User Guide*.

Writing Host Input Import Files

This chapter provides details on the syntax to import data using the import functions of the host input import tool. When writing your import file, make sure you follow the instructions provided in the following sections:

Understanding Import File Format

To successfully import data using the host input import tool, you must create an import file that conforms to the required format.

Caution The system discards any data in the import file that it cannot interpret, so you may want to test import of your import file before running the import. For more information, see Testing Your Import on the Defense Center, page 3-30.

Host input import files must begin with the SetSource and SetMap commands, to provide an application source name and to set up third-party product name mappings for the imported data. For more information, see Setting the Source ID, page 3-4.

After the SetSource and SetMap commands, you can add additional command lines to the file. Each command line contains a single command with the parameters needed for that command and ends with a hard return. For more information on syntax for individual commands you can include, see the following sections:

- Host Functions, page 3-6
- Server Functions, page 3-9
- Client Application Functions, page 3-13
- Protocol Functions, page 3-15
- Package Fix Functions, page 3-16
- Host Attribute Functions, page 3-18
- Vulnerabilities Functions, page 3-19
- Setting a Third-Party Product Map, page 3-5

To see an example of a complete import file and explanations of each section of the file, see Example Host Input Import File, page 3-24.

Setting the Source Type

At the beginning of your import file, you must identify the source type for the data you plan to import. If you import scan results using this function, be sure to edit the source definition for the input source in your network discovery policy to set the identity source type to Scanner.

To set the source type:

Step 1 Add a line to your import file using the following syntax:

```
SetSourceType, Sourcetype
```

where `SetSourceType` is the name of the function and *`Sourcetype`* is the type of source you want to add or use for the imported data. (Valid values are `2` (scanner) or `3` (application).)

If SetSourceType is not used, the default is type is `3` (application).

Setting the Source ID

At the beginning of your import file, you must set the source ID for the data you plan to import.

To set the source application name:

Step 1 Add a line to your import file using the following syntax:

```
SetSource, SourceID
```

where `SetSource` is the name of the function and *`SourceID`* is the identification string you want to display as the source application for the imported data.

The following is an example of the first lines of an import file:

```
# Example CSV style import file for Host Input API
#
# Set the current SOURCE_ID and Product Map to "Custom Utility"
SetSource, Custom Utility
```

To see these commands in context in an example file, see Entire Example File, page 3-29.

Setting a Third-Party Product Map

If you are planning to import third-party operating system, server, or fix definitions, you must create a user third-party product map for the third-party names. You can use this function to set the current third-party map for the current session. You create third-party mappings using the Defense Center web interface to set up a reusable map between each third-party vendor, product, and version combination and the corresponding Cisco product definition. If you set a third-party map and then add or set host operating system or server data that includes third-party application names included in the map, the system uses the mappings to map the Cisco product definition, and associated vulnerabilities, to each host where the input occurs.

For instance, you could create a map set called `"Custom Utility"`, in which you define the third-party strings as follows:

- **Vendor String - Microsoft**
- **Product String - Win2k**

You could select the following Cisco product mapping in the map set:

- **Vendor - Microsoft, Corp.**
- **Product - Windows 2000**
- **Patch - SP3**

If you set this product map by calling `SetMap, Custom Utility`, it maps `Microsoft Win2k` to the VDB entry for the `Microsoft Windows 2000 SP3` product.

To set the third-party product map set:

Step 1 Add a line to your import file using the following syntax:

```
SetMap, Third-PartyProductMapName
```

where SetMap is the name of the function and *`Third-PartyProductMapName`* is the name of the third-party product map set you want to use for the import.

For example, you could put the following line of code following the SetSource command:

```
SetMap, Custom Utility
```

You can also use this command to change to a different third-party product map within an import file.

Required Fields

Each host input function requires either an IPv4 or IPv6 address, address range, or subnet (for specifying IP hosts by address) or a MAC address or addresses (for specifying MAC-only hosts). The documentation for each function call indicates any additional required fields for that function.

Note that some fields are required only in that you must supply that information to make sure that the host input succeeds and adds meaningful data to the network map. For example, you can add a fix to the system without providing a fix identification number or fix name that matches an existing Cisco fix definition and without mapping the third-party fix to a Cisco fix. However, even if that fix addresses vulnerabilities on the host where you added it, those vulnerabilities cannot be marked invalid if the system cannot map the fix to the vulnerabilities using an Cisco fix definition.

In general, supply as much information as possible for any data you import to ensure that the data can be used for data correlation.

Host Input Import Syntax

After you set the source ID and product map for your import file, as described in Setting the Source ID, page 3-4, you can add lines to your import file to import the specific data you want to add to your network map using various host input functions. Each import function call must end in a hard return and imports one set of import data. For an example of a complete import file, see Example Host Input Import File, page 3-24.

For more information on specific commands you can use, see the following sections:

Host Functions

You can use the host input API to add and remove hosts in the network map and to set operating system definitions for hosts.

For more information on host functions, see the following sections:

AddHost

You can use the `AddHost` function to add a host to the network map. You can add an IP host (a host with an IP address and optionally a MAC address) or a MAC-only host (a host with only a MAC address). Because hosts created using the API are not tracked by the system, these hosts are not subject to the normal host timeout.

Note that you cannot create a MAC-only host for a MAC address if the system detects traffic that indicates that MAC address is mapped as a primary MAC address for an IP host already in the network map.

Use this syntax:

```
AddHost, ip_address, mac_address
```

Table 3-1 AddHost Fields

Field	Description	Required	Values
ip_address	Indicates the IP address for the added host.	Yes (unless a MAC address is provided)	A single IP address
mac_address	Indicates the MAC address for the added host.	Yes (unless an IP address is provided)	A single MAC address

DeleteHost

You can use the `DeleteHost` function to remove a host (or hosts) from the network map. You can remove an IP host (a host with an IP address and optionally a MAC address) by specifying either the IP address or the MAC address for the host. To remove a MAC-only host (a host with only a MAC address), indicate the MAC address as the `mac_list` value.

Use this syntax:

```
DeleteHost, ip_address, mac_address
```

Table 3-2 DeleteHost Fields

Field	Description	Required	Values
ip_address	Indicates the string containing the IP address or addresses for the affected host or hosts.	Yes (unless MAC addresses are provided)	A comma-separated list of IP addresses, CIDR blocks, and ranges of IP addresses.
mac_address	Indicates the list of MAC addresses for the affected host or hosts.	Yes (unless IP addresses are provided)	A list of MAC address strings, with or without separating colons.

SetOS

You can use the `SetOS` function to specify the vendor, product, version, and mobile device information for the operating system for specified hosts. When you import operating system information, you set the display strings for the vendor, product, version, and mobile device information. You can also map the third-party vendor, product, and version strings to a Cisco product definition. See Creating a Third-Party Product Map, page 3-2 for more information.

If you map third-party operating system names to a Cisco definition, the vulnerabilities for that operating system in the Cisco database correspond to the host where the third-party data was imported. If you have already created a third-party product map set using the Defense Center web interface, you can use the `SetMap` function to use the values you specified in that map set for the third-party application strings and corresponding Cisco definitions, as described in Setting a Third-Party Product Map, page 3-5.

The operating system identity displayed in a host profile is set by the highest priority source. Possible sources have the following priority order: user, scanner and application (set in the system policy), FireSIGHT, then NetFlow. Note that a new higher priority operating system identity will not override a current operating system identity if it has less detail than the current identity.

If you define a custom operating system for a host, the Defense Center web interface indicates the source for the change in the Source Type field of the event view or the basic host information of the host profile.

Use this syntax:

```
SetOS, ip_address, vendor_str, product_str, version_str, vendor_id, product_id,
major, minor, revision, build, patch, extension, device_string, mobile, jailbroken
```

Or, to set a new product map before you set the operating system, use this syntax:

```
SetMap:map_name, SetOS, ip_address, vendor_str, product_str, version_str, vendor_id,
product_id, major, minor, revision, build, patch, extension, device_string, mobile,
jailbroken
```

For more information on setting third-party product maps, see Setting a Third-Party Product Map, page 3-5.

Table 3-3 SetOS Fields

Field	Description	Required	Allowed Values
ip_address	Indicates the string containing the IP address or addresses for the affected host or hosts.	Yes	A comma-separated list of IP addresses, CIDR blocks, and ranges of IP addresses
vendor_str	Supplies the operating system vendor display name used by the third-party application.	No	string
product_str	Supplies the operating system product display name used by the third-party application.	No	string
version_str	Supplies the operating system version display name used by the third-party application.	No	string
vendor_id	Supplies the Cisco vendor definition to map to.	No	uint32
product_id	Supplies the Cisco product definition to map to.	No	uint32
major	Supplies the Cisco major version definition to map to.	No	uint32
minor	Supplies the Cisco minor version definition to map to.	No	uint32
revision	Supplies the Cisco revision string to map to.	No	uint32
build	Supplies the Cisco build definition to map to.	No	string
patch	Supplies the Cisco patch definition to map to.	No	string
extension	Supplies the Cisco extension definition to map to.	No	string
device_string	Supplies the detected mobile device hardware information.	No	string
mobile	Indicates whether the operating system is running on a mobile device.	No	uint8
jailbroken	Indicates whether the mobile device operating system is jailbroken.	No	uint8

UnsetOS

You can use the `UnsetOS` function to remove a previously set OS definition from specified hosts. It resets the OS definition to allow the system to track changes to the operating system in the future.

Use this syntax:

```
UnsetOS, ip_address
```

Where *ip_address* is a comma-separated list of IP addresses, CIDR blocks, and ranges of IP addresses representing the host or hosts where you want to reset the operating system identity.

Server Functions

You can update server information for hosts in the network map using the server functions.

For more information, see the following sections:

AddService

You can add a server to an existing host in the network map using the `AddService` function.

The server identity displayed in a host profile is set by the highest priority source. Possible sources have the following priority order: user, scanner and application (set in the system policy), FireSIGHT, then NetFlow. Note that a new higher priority server identity will not be override a current operating server identity if it has less detail than the current identity.

Use this syntax:

```
AddService, ip_address, port, proto, server, vendor_str, version_str, vendor_id,
product_id, major, minor, revision, build, patch, extension
```

Or, to set a new product map before you add the server, use this syntax:

```
SetMap:map_name, AddService, ip_address, port, proto, server, vendor_str,
version_str, vendor_id, product_id, major, minor, revision, build, patch, extension
```

For more information on setting third-party product maps, see Creating a Third-Party Product Map, page 3-2 and Setting a Third-Party Product Map, page 3-5.

Table 3-4 AddService Fields

Field	Description	Required	Values
ip_address	Indicates the string containing the IP address or addresses for the affected host or hosts.	Yes	A comma-separated list of IP addresses, CIDR blocks, and ranges of IP addresses.
port	Use this field in combination with the `ip_address` and `proto` fields to specify the server to be added on the hosts where it should be added.	Yes	Integers in the range of 1-65535.
proto	Use this field in combination with the `ip_address` and `port` fields to specify the server to be added on the hosts where it should be added.	Yes	Either the strings `tcp` or `udp` or the appropriate protocol IDs `6` (tcp) or `17` (udp).
server	The name or ID of the server in the Cisco database.	No	To identify the server, you must include a value for either `service_name` or `service_id`. If neither is provided, the server will be listed as `unknown`. If a server name is provided, The system looks up the server ID. If no ID exists for the server name, the system creates an ID.

Table 3-4 AddService Fields (continued)

Field	Description	Required	Values
vendor_str	Supplies the operating system vendor display name used by the third-party application.	No	string
product_str	Supplies the operating system product display name used by the third-party application.	No	string
version_str	Supplies the operating system version display name used by the third-party application.	No	string
vendor_id	Supplies the Cisco vendor definition.	No	uint32
product_id	Supplies the Cisco product definition.	No	uint32
major	Supplies the Cisco major version definition.	No	uint32
minor	Supplies the Cisco minor version definition.	No	uint32
revision	Supplies the Cisco revision string.	No	uint32
build	Supplies the Cisco build definition to map to.	No	string
patch	Supplies the Cisco patch definition to map to.	No	string
extension	Supplies the Cisco extension definition to map to.	No	string

SetService

You can use the `SetService` function to specify the server protocol, vendor, product, and version for a specified server. You can set display strings for the server using service keys. By mapping a third-party product in the Defense Center web interface (see Creating a Third-Party Product Map, page 3-2) or using the `SetMap` function (see Setting a Third-Party Product Map, page 3-5), you can associate third-party server data with the vulnerability information for specific Cisco product definitions.

If the server protocol does not already exist, this call causes a new server identity to be created for the string. If the specified server does not exist previously, the system creates it.

The server identity displayed in a host profile is set by the highest priority source. Possible sources have the following priority order: user, scanner and application (set in the system policy), FireSIGHT, then NetFlow. Note that a new higher priority server identity will not be override a current server identity if it has less detail than the current identity.

If you define a third-party server definition for a host, the FireSIGHT System web interface indicates the source for the change in the Source Type field of the Servers table view of events or the Servers section of the host profile.

Note If the number of servers stored in the network map for a specific host exceeds 100, new server information is ignored until servers are deleted from the host.

Use this syntax:

```
SetService, ip_address, port, proto, server, vendor_str, version_str, vendor_id,
product_id, major, minor, revision, build, patch, extension
```

Or, to set a new product map before you set the server, use this syntax:

```
SetMap:map_name, SetService, ip_address, port, proto, server, vendor_str,
version_str, vendor_id, product_id, major, minor, revision, build, patch, extension
```

For more information on setting third-party product maps, see Setting a Third-Party Product Map, page 3-5.

Table 3-5 SetService Fields

Field	Description	Required	Values
ip_address	Indicates the string containing the IP address or addresses for the affected host or hosts.	Yes	A comma-separated list of IP addresses, CIDR blocks, and ranges of IP addresses.
port	Use this field in combination with the `ip_address` and `proto` fields to specify the server to be set on the hosts where it should be set.	Yes	Integers in the range of 1-65535.
proto	Use this field in combination with the `ip_address` and `port` fields to specify the server to be set on the hosts where it should be set.	Yes	Either the strings `tcp` or `udp` or the appropriate protocol IDs `6` (tcp) or `17` (udp).
server	The name or ID of the server in the Cisco database.	No	To identify the server, you must include a value for either `service_name` or `service_id`. If neither is provided, the server will be listed as `unknown`. If a server name is provided, the system looks up the server ID. If no ID exists for the server name, the system creates an ID.
vendor_str	Supplies the operating system vendor display name used by the third-party application.	No	string
product_str	Supplies the operating system product display name used by the third-party application.	No	string
version_str	Supplies the operating system version display name used by the third-party application.	No	string
vendor_id	Supplies the Cisco vendor definition.	No	uint32
product_id	Supplies the Cisco product definition.	No	uint32
major	Supplies the Cisco major version definition.	No	uint32
minor	Supplies the Cisco minor version definition.	No	uint32
revision	Supplies the Cisco revision string.	No	uint32
build	Supplies the Cisco build definition to map to.	No	string
patch	Supplies the Cisco patch definition to map to.	No	string
extension	Supplies the Cisco extension definition to map to.	No	string

UnsetService

You can use the `UnsetService` function to remove user-added server definitions from a specified host. `UnsetService` does not remove any server definitions detected through FireSIGHT.

Note If the number of servers stored in the network map for a specific host exceeds 100, new server information is ignored until servers are deleted from the host.

Use this syntax:

```
UnsetService, ip_address, port, proto
```

Table 3-6 UnsetService Fields

Field	Description	Required	Values
ip_address	Indicates the string containing the IP address or addresses for the affected host or hosts.	Yes	A comma-separated list of IP addresses, CIDR blocks, and ranges of IP addresses.
port	Use this field in combination with the `ip_address` and `proto` fields to specify the server to be removed on the hosts where it should be removed.	Yes	Integers in the range of 1-65535.
proto	Use this field in combination with the `ip_address` and `port` fields to specify the server to be removed on the hosts where it should be removed.	Yes	Either the strings `tcp` or `udp` or the appropriate protocol IDs `6` (tcp) or `17` (udp).

DeleteService

You can use the `DeleteService` function to remove a server from a specified host. You must specify the port and protocol of the server you want to delete.

Use this syntax:

```
DeleteService, ip_address, port, proto
```

Table 3-7 DeleteService Fields

Field	Description	Required	Values
ip_address	Indicates the string containing the IP address or addresses for the affected host or hosts.	Yes	A comma-separated list of IP addresses, CIDR blocks, and ranges of IP addresses.
port	Use this field in combination with the `ip_address` and `proto` fields to specify the server to be deleted on the hosts where it should be deleted.	Yes	Integers in the range of 1-65535.
proto	Use this field in combination with the `ip_address` and `port` fields to specify the server to be deleted on the hosts where it should be deleted.	Yes	Either the strings `tcp` or `udp` or the appropriate protocol IDs `6` (tcp) or `17` (udp).

Client Application Functions

You can use the client application functions to modify client application data for hosts in the network map.

For more information, see the following sections:

AddClientApp

You can use the `AddClientApp` function to add client applications to existing hosts in the network map. If the client application name does not already exist in the Cisco database, the system creates a new entry for the client application.

The client application identity displayed in a host profile is set by the highest priority source. Possible sources have the following priority order: user, scanner and application (set in the system policy), FireSIGHT, then NetFlow. Note that a new higher priority client application identity will not be override a current client application identity if it has less detail than the current identity.

Use this syntax:

```
AddClientApp, ip_address, app_name, app_type, version
```

Table 3-8 AddClientApp Fields

Field	Description	Required	Values
ip_address	Indicates the string containing the IP address or addresses for the affected host or hosts.	Yes	A comma-separated list of IP addresses, CIDR blocks, and ranges of IP addresses.
app_name	Indicates the client application name.	Yes	A string consisting of alphanumeric characters or spaces. For existing applications, corresponds to ID values in the database. The system looks up the ID to see if it matches an existing client application ID. If it does not, a new ID is created.
app_type	This field is deprecated.	No	A null value.
version	Indicates the application version.	No	A string consisting of alphanumeric characters or spaces.

DeleteClientApp

You can use the `DeleteClientApp` function to remove a client application from the specified host.

Use this syntax:

```
DeleteClientApp, ip_address, app_name, app_type, version
```

Table 3-9 DeleteClientApp Fields

Field	Description	Required	Values
ip_address	Indicates the string containing the IP address or addresses for the affected host or hosts.	Yes	A comma-separated list of IP addresses, CIDR blocks, and ranges of IP addresses.
app_name	Indicates the client application name.	Yes	A string consisting of alphanumeric characters or spaces. For existing applications, corresponds to ID values in the database. The system looks up the ID to see if it matches an existing client application ID. If it does not, a new ID is created.
app_type	This field is deprecated.	No	A null value.
version	Indicates the application version.	No	A string consisting of alphanumeric characters or spaces.

DeleteClientAppPayload

You can use the `DeleteClientAppPayload` function to remove a web application from the specified host.

Use this syntax:

```
DeleteClientAppPayload, ip_address, app_name, app_type, version, payload_type,
payload_id
```

Table 3-10 DeleteClientAppPayload Fields

Field	Description	Required	Values
ip_address	Indicates the string containing the IP address or addresses for the affected host or hosts.	Yes	A comma-separated list of IP addresses, CIDR blocks, and ranges of IP addresses.
app_name	Indicates the client application name.	Yes	A string consisting of alphanumeric characters or spaces. For existing applications, corresponds to ID values in the database. The system looks up the ID to see if it matches an existing client application ID. If it does not, a new ID is created.
app_type	This field is deprecated.	No	A null value.
version	Indicates the application version.	No	A string consisting of alphanumeric characters or spaces.
payload_type	Indicates the web application category.	Yes	The number `0`. For existing applications, corresponds to ID values in the database. The system looks up the type to see if it matches an existing web application type. If it does not, a new type is created.
payload_id	Indicates the web application name.	Yes	A string consisting of alphanumeric characters or spaces. For existing applications, corresponds to ID values in the database. The system looks up the ID to see if it matches an existing web application ID. If it does not, a new ID is created.

Protocol Functions

You can use the protocol functions to update protocol information for hosts in the network map.

For more information, see the following sections:

- DeleteProtocol, page 3-15
- AddProtocol, page 3-15

DeleteProtocol

You can use the `DeleteProtocol` function to remove a protocol from the specified IP or MAC host.

Use this syntax:

```
DeleteProtocol, ip_address, mac_address, proto, type
```

Table 3-11 DeleteProtocol Fields

Field	Description	Required	Values
ip_address	Indicates the string containing the IP address or addresses for the affected host or hosts.	Yes (unless MAC addresses are provided)	A comma-separated list of IP addresses, CIDR blocks, and ranges of IP addresses.
mac_address	Indicates the list of MAC addresses for the affected host or hosts.	Yes (unless IP addresses are provided)	A list of MAC address strings, with or without separating colons.
proto	Indicates the identification string or name of the protocol to be deleted.	Yes	Valid protocol names consisting of alphanumeric characters or spaces. For transport protocols (`"xport"`), protocols listed in the `/etc/protocols` file are acceptable. For network protocols (“`net`”), see Network Protocol Values, page A-1.
type	Indicates the type of protocol to be deleted.	Yes	`"xport"` or `"net"`

AddProtocol

You can use the `AddProtocol` function to add either a network or transport protocol to an existing host in the network map. You can supply either a protocol ID, a transport protocol name that exists in the `/etc/protocols` file on your Defense Center or a network protocol name from Network Protocol Values, page A-1.

Note You cannot add transport protocols to MAC-only hosts.

Use this syntax:

```
AddProtocol, ip_address, mac_address, proto, type
```

Table 3-12 AddProtocol Fields

Field	Description	Required	Values
ip_address	Indicates the string containing the IP address or addresses for the affected host or hosts.	Yes (unless MAC addresses are provided)	A comma-separated list of IP addresses, CIDR blocks, and ranges of IP addresses.
mac_address	Indicates the list of MAC addresses for the affected host or hosts.	Yes (unless IP addresses are provided)	A list of MAC address strings, with or without separating colons.
proto	Indicates the identification string or name of the protocol to be added.	Yes	Valid protocol names consisting of alphanumeric characters or spaces. For transport protocols (`"xport"`), protocols listed in the `/etc/protocols` file are acceptable. For network protocols (“`net`”), see Network Protocol Values, page A-1.
type	Indicates the type of protocol to be added.	Yes	`"xport"` or `"net"`

Package Fix Functions

You can use the Package Fix functions to apply or remove fixes for hosts in your network map.

For more information, see the following sections:

- AddFix, page 3-16
- RemoveFix, page 3-17

AddFix

You can use the `AddFix` function to map a fix to a specified host or server. You can map a fix using a fix ID from the Cisco vulnerability database (VDB), or using a third-party fix that you map to a fix in the VDB using the Defense Center web interface.

When you apply a fix to a host or server, the vulnerability mappings for the system are adjusted and the fixed vulnerabilities are marked as Invalid in the web interface and are not used for impact assessment. However, note that if the applied fix is not applicable to the OS or server identity the fix has no effect.

Use the following syntax:

```
AddFix, ip_address, port, proto, fix_id
```

Table 3-13 AddFix Fields

Field	Description	Required	Values
ip_address	Indicates the string containing the IP address or addresses for the affected host or hosts.	Yes (unless MAC addresses are provided)	A comma-separated list of IP addresses, CIDR blocks, and ranges of IP addresses.
port	With the `proto` field, identifies the server affected by the fix on the host where the import occurs.	Yes, if the fix applies to a server	Integers in the range of 1-65535.
proto	With the `port` field, identifies the server affected by the fix on the host where the import occurs.	No	Either the strings `tcp` or `udp` or the appropriate protocol IDs `6` (tcp) or `17` (udp).
fix_id	Indicates the identification string for the fix.	Yes	A Cisco fix identification number or a fix name defined in a third-party product map that you use by calling the `SetMap` function before invoking the `AddFix` function. For more information, see Setting a Third-Party Product Map, page 3-5.

RemoveFix

You can use the `RemoveFix` function to remove a fix mapping from the specified host or server. When you remove a fix, vulnerability mappings are updated accordingly.

Use this syntax:

```
RemoveFix, ip_address, port, proto, fix_id
```

Table 3-14 RemoveFix Fields

Field	Description	Required	Values
ip_address	Indicates the string containing the IP address or addresses for the affected host or hosts.	Yes (unless MAC addresses are provided)	A comma-separated list of IP addresses, CIDR blocks, and ranges of IP addresses.
port	With the `proto` field, identifies the server affected by the fix on the host where the import occurs.	Yes, if the fix applies to a server	Integers in the range of 1-65535.

Table 3-14 RemoveFix Fields (continued)

Field	Description	Required	Values
proto	With the `port` field, identifies the server affected by the fix on the host where the import occurs.	No	Either the strings `tcp` or `udp` or the appropriate protocol IDs `6` (tcp) or `17` (udp).
fix	Indicates the identification string for the fix.	Yes	A Cisco fix name or a fix name defined in a third-party product map that you use by calling the `SetMap` function before invoking the `AddFix` function. For more information, see Setting a Third-Party Product Map, page 3-5.

Host Attribute Functions

You can use the host input import tool to set attribute values for hosts in your network map. For more information, see the following sections:

AddHostAttribute

You can use the `AddHostAttribute` function to add text or URL attributes. Note that adding a host attribute does not add a value for the attribute. For more information on setting an attribute value, see SetAttributeValue, page 3-18, below.

Use this syntax:

```
AddHostAttribute, attributename, attributetype
```

where *`attributename`* is the name of the attribute (consisting of alphanumeric characters and spaces.) and *`attributetype`* is the type of attribute (`text` or `URL`).

DeleteHostAttribute

You can use the `DeleteHostAttribute` function to delete attributes.

Use this syntax:

```
DeleteHostAttribute, attributename
```

where *`attributename`* is the name of the attribute. (Valid names consist of alphanumeric characters and spaces.)

SetAttributeValue

You can use the `SetAttributeValue` function to set the value of an existing attribute to the specified value for specified hosts. This function can set the value of user-defined host attributes and the `Criticality` attribute. You can use this function to set the host criticality by using `"criticality"` as the attribute id.

Use this syntax:

```
SetAttributeValue, ip_address, attribute, value
```

Table 3-15 SetAttributeValue Fields

Field	Description	Required	Values
ip_address	Indicates the string containing the IP address or addresses for the affected host or hosts.	Yes	A comma-separated list of IP addresses, CIDR blocks, and ranges of IP addresses.
attribute	Indicates the host attribute name.	Yes	Valid attribute names consisting of alphanumeric characters or spaces.
value	Indicates the host attribute value.	Yes	Valid attribute values for the named attribute, consisting of alphanumeric characters or spaces. If a value is passed in for a list attribute, the value must be an existing named value for the list attribute.

DeleteAttributeValue

You can use the `DeleteAttributeValue` function to remove an attribute value for a host.

Use this syntax:

```
DeleteAttributeValue, ip_address, attribute, value
```

Table 3-16 DeleteAttributeValue Fields

Field	Description	Required	Values
ip_address	Indicates the string containing the IP address or addresses for the affected host or hosts.	Yes	A comma-separated list of IP addresses, CIDR blocks, and ranges of IP addresses.
id	Indicates the host attribute name.	Yes	Valid attribute names consisting of alphanumeric characters or spaces.
value	Indicates the host attribute value.	Yes	Valid attribute values for the named attribute, consisting of alphanumeric characters or spaces. If a value is passed in for a list attribute, the value must be an existing named value for the list attribute.

Vulnerabilities Functions

You can use the vulnerabilities functions to update the status of vulnerabilities on a host.

For more information, see the following sections:

- SetInvalidVulns, page 3-19
- SetValidVulns, page 3-20

SetInvalidVulns

You can use the `SetInvalidVulns` function to deactivate vulnerabilities on a host or set of hosts. For the function call to be effective, the vulnerability must exist on the host and be set to valid.

Use this syntax:

```
SetInvalidVulns, ip_address, port, proto, type, vuln_id
```

Table 3-17 SetInvalidVulns Fields

Field	Description	Required	Values
ip_address	Indicates the string containing the IP address or addresses for the affected host or hosts.	Yes (unless MAC addresses are provided)	A comma-separated list of IP addresses, CIDR blocks, and ranges of IP addresses.
port	With the `proto` field, identifies the server affected by the vulnerability on the host where the import occurs.	Yes, if the fix applies to a server	Integers in the range of 1-65535.
proto	With the `port` field, identifies the server affected by the vulnerability on the host where the import occurs.	Yes, if the fix applies to a server	Either the strings `tcp` or `udp` or the appropriate protocol IDs `6` (tcp) or `17` (udp).
vuln_id	Indicates the vulnerability ID for the vulnerability.	Yes	Valid Cisco vulnerability IDs, or mapped third-party vulnerability IDs. For third-party vulnerabilities, note that you must map the third-party vulnerability ID and reference the vulnerability map set in the `vuln_type` field. For more information, see Creating a Third-Party Vulnerability Map, page 3-1.

SetValidVulns

You can use the `SetValidVulns` function to activate vulnerabilities on a host or set of hosts. Once you set a vulnerability as Valid for a host, Defense Center assigns a red impact to the event if the SID in the event is mapped to the valid vulnerability. For the function call to be effective, the vulnerability must exist on the host and be set to invalid.

Use this syntax:

```
SetValidVulns, ip_address, port, proto, type, vuln_id
```

Table 3-18 SetValidVulns Fields

Field	Description	Required	Values
ip_address	Indicates the string containing the IP address or addresses for the affected host or hosts.	Yes	A comma-separated list of IP addresses, CIDR blocks, and ranges of IP addresses.
port	With the `proto` field, identifies the server affected by the vulnerability on the host where the import occurs.	Yes, if the fix applies to a server	Integers in the range of 1-65535.

Table 3-18 SetValidVulns Fields (continued)

Field	Description	Required	Values
proto	With the `port` field, identifies the server affected by the vulnerability on the host where the import occurs.	Yes, if the fix applies to a server	Either the strings `tcp` or `udp` or the appropriate protocol IDs `6` (tcp) or `17` (udp).
vuln_id	Indicates the vulnerability ID for the vulnerability.	Yes	Valid Cisco vulnerability IDs, or mapped third-party vulnerability IDs. For third-party vulnerabilities, note that you must map the third-party vulnerability ID and reference the vulnerability map set in the `vuln_type` field. For more information, see Creating a Third-Party Vulnerability Map, page 3-1.

Scan Result Functions

You can use the host input import tool to add scan results to your Defense Center and to flush the added results to the database. When adding a scan result you can map third-party vulnerabilities in the results to CVE or BugTraq vulnerabilities.

See the following sections for more information:

- AddScanResult Function, page 3-21
- ScanFlush Function, page 3-23
- ScanUpdate Function, page 3-23
- DeleteScanResult Function, page 3-23

AddScanResult Function

You can use the `AddScanResult` function to add scan results from a third-party vulnerability scanner and map each vulnerability to a BugTraq or CVE ID. If you import scan results using this function, be sure to edit the source definition for the input source in your network discovery policy to set the identity source type to Scanner.

Use this syntax:

```
AddScanResult, ipaddr, 'scanner_id', vuln_id, port, protocol, 'name', 'description',
cve_ids, bugtraq_ids
```

Note How results are added depends on whether you use the ScanUpdate or ScanFlush function. For more information, refer to ScanFlush Function, page 3-23 and ScanUpdate Function, page 3-23.

Table 3-19 AddScanResult Fields

Field	Description	Required	Allowed Values
ipaddr	Indicates the IP address of the scanned host or hosts.	Yes	A single IP address.
scanner_id	Indicates the scanner ID for the scanner that obtained the scan results.	Yes	*'scanner_id'* where `scanner_id` is a string indicating the name of the scanner that is the source of the vulnerability data you add. To add scan results from a previously used scanner, indicate the specific scanner name listed in system policies on the Defense Center where you added the results. Adding results from a new scanner ID adds that scanner to the system policy. New scanners are added as the lowest priority by default. If you want to change the priority of the scanner, you can do so in the system policy. For more information, see the *FireSIGHT System User Guide*.
vuln_id	Indicates the vulnerability ID for the vulnerability.	Yes	Valid Cisco vulnerability IDs, or mapped third-party vulnerability IDs. If this field, port, protocol, bugtraq_ids, and cve_ids are empty, this is a generic scan result.
port	With the `proto` field, identifies the server affected by the vulnerability on the host where the import occurs.	Yes, if the vulnerability applies to a server	Integers in the range of 1-65535.
proto	With the `port` field, identifies the server affected by the vulnerability on the host where the import occurs.	Yes, if the vulnerability applies to a server	Either the strings `tcp` or `udp` or the appropriate protocol IDs `6` (tcp) or `17` (udp).
name	The name of the vulnerability being imported.	No	A string enclosed in single quotes; for example: `'Using NetBIOS to retrieve info from a Windows host'`
description	The description of the vulnerability being imported.	No	A string enclosed in single quotes; for example: `'The following 2 NetBIOS names have been gathered...'`
cve_ids	Space-separated list of CVE vulnerability IDs	No	Valid CVE vulnerability IDs; for example, `'cve_ids: CVE2003-0988'`. If this field, port, protocol, vuln_id, and bugtraq_ids are empty, this is a generic scan result.
bugtraq_ids	Space-separated list of BugTraq vulnerability IDs	No	Valid BugTraq vulnerability IDs; for example, `'bugtraq_ids: 9506'`. If this field, port, protocol, vuln_id, and cve_ids are empty, this is a generic scan result.

ScanFlush Function

After you add scan results to a Defense Center using `AddScanResult`, you must use either the `ScanUpdate` or `ScanFlush` function to cause the `AddScanResult` commands to run on the Defense Center so the scan results upload to the database.

The `ScanFlush` function does not require any arguments, and can be used at whatever point in the import file that you want to upload data to the database.

If you use the `ScanFlush` function, any existing scan results are removed from the host and only the new results are added.

ScanUpdate Function

After you add scan results to a Defense Center using `AddScanResult`, you must use either the `ScanUpdate` or `ScanFlush` function to cause the `AddScanResult` commands to run on the Defense Center so the scan results upload to the database.

The `ScanUpdate` function does not require any arguments, and can be used at whatever point in the import file that you want to upload data to the database.

If you use the `ScanUpdate` function, the existing scan results are not removed from the host. The new scan results are merged with the existing scan results.

If you use the `ScanUpdate` function with the `DeleteScanResult` function, the specific results are deleted.

Note that a `ScanUpdate` automatically occurs when an import finishes even if it is not explicitly included in the import file, because the client connection closes.

DeleteScanResult Function

You can use the `DeleteScanResult` function with the `ScanUpdate` function to remove specific scan results from a specific host.

If you supply values for the optional parameters, this restricts results to those matching the parameters. If you do not supply values for the optional parameters, all results on the specified IP address are deleted.

Use this syntax:

```
DeleteScanResult, ipaddr, 'scanner_id', vuln_id, port, protocol
```

Table 3-20 DeleteScanResult Fields

Field	Description	Required	Allowed Values
ipaddr	Indicates the IP address of the scanned host or hosts.	Yes	A single IP address.
scanner_id	Indicates the scanner ID for the scanner that obtained the scan results.	No	`'scanner_id'` where `scanner_id` is a string indicating the name of the scanner that is the source of the vulnerability data you add. To add scan results from a previously used scanner, indicate the specific scanner name listed in system policies on the Defense Center where you added the results. Adding results from a new scanner ID adds that scanner to the system policy. New scanners are added as the lowest priority by default. If you want to change the priority of the scanner, you can do so in the system policy. For more information, see the *FireSIGHT System User Guide*.
vuln_id	Indicates the vulnerability ID for the vulnerability.	No	A valid third-party vulnerability ID.
port	With the `proto` field, identifies the server affected by the vulnerability on the host where the import occurs.	No	Integers in the range of 1-65535.
proto	With the `port` field, identifies the server affected by the vulnerability on the host where the import occurs.	No	Either the strings `tcp` or `udp` or the appropriate protocol IDs `6` (tcp) or `17` (udp).

Example Host Input Import File

The following sections illustrate how you might construct an import file to import data using the host input import tool.

The following sections, in sequential order, show each portion of the file:

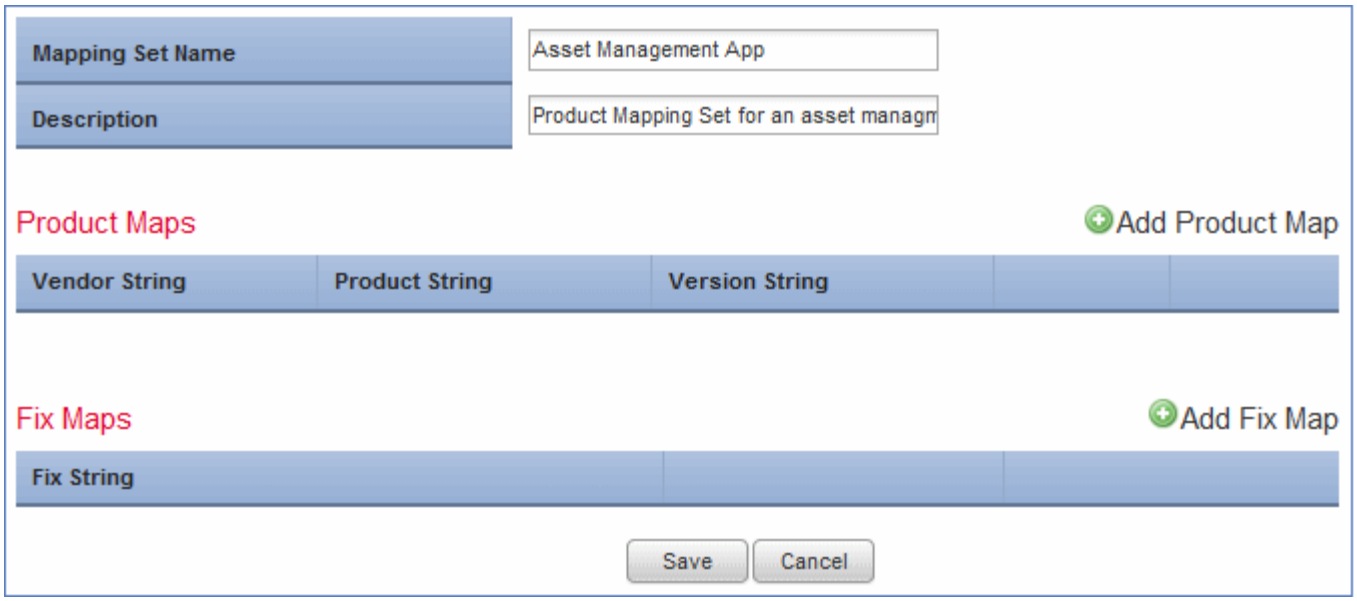
Example: Setting the Source ID and Product Map

The example script starts with calls to set the name of the source application and the product map to be used in the import:

```
# Set the current SOURCE_ID and Product Map to "Asset Management App"
#
SetSource, Asset Management App
SetMap, Asset Management App
```

This source ID value is used to provide an application name for the system to use in host input events resulting from this import. If you viewed a host input event or a host profile for a host modified using this import, the Source Type value would be `Application: Asset Management App`.

Note that the product map called “Asset Management App” referenced by the SetMap command was created using the Defense Center web interface:



Because the third-party product map is the Asset Management App map set, the system maps any third-party operating or server names used in the commands contained in the import file to Cisco definitions using product maps or fix maps defined in that map set, as illustrated in Example: Setting the Operating System, page 3-26.

Example: Adding a Host

After the file sets the source application name and third-party product map, commands to import data follow. The first import function called is the `AddHost` function:

```
# Add an IP host with no Primary MAC
#
AddHost,1.2.3.4
```

Note that the IP address for the added host is `1.2.3.4` and no primary MAC address is set for the host.

Example: Adding a Protocol to the Host

The next command in the import file adds the `ospf` protocol to the `1.2.3.4` host:

```
# Add the ospf protocol to the host
#
AddProtocol, 1.2.3.4,,ospf,xport
```

Note that the protocol type for the protocol is `xport`.

Example: Adding a Server to the Host

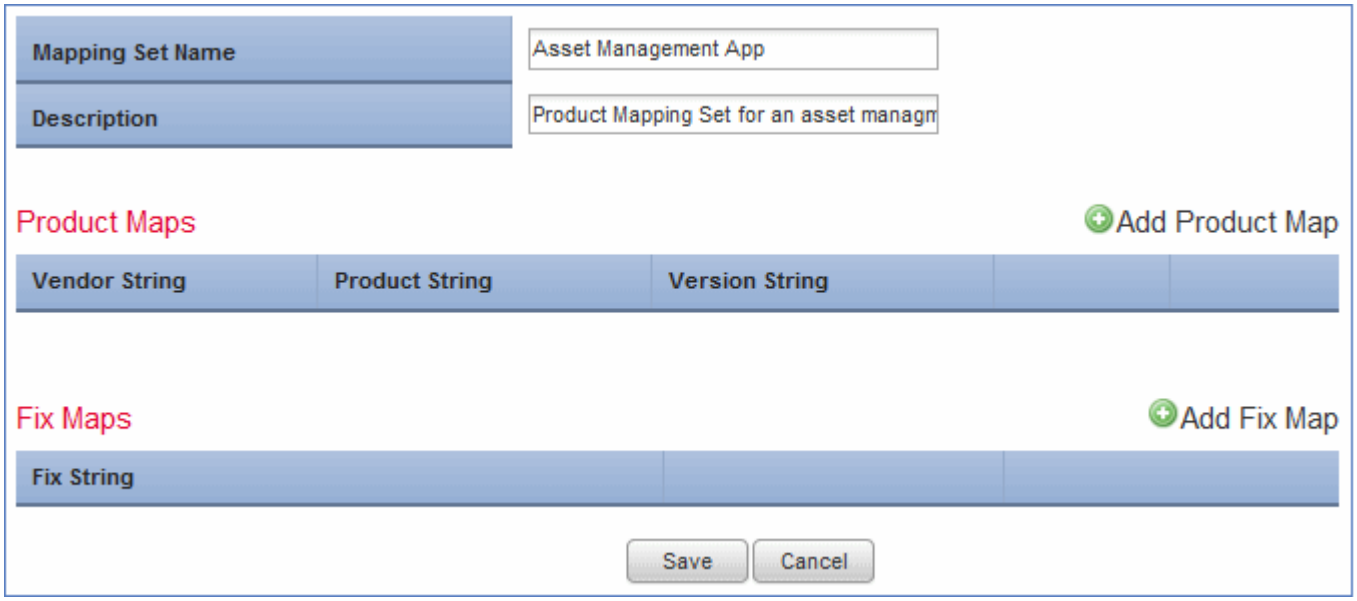
The next command in the import file uses the `AddService` function to add the `OpenSSH` server to the `1.2.3.4` host:

```
# Add a server for the host
#
AddService,1.2.3.4, 22, tcp, ssh, OpenSSH, 4.1
```

Note that the command sets the port to `22`, the protocol to `tcp`, the server type to `ssh`, the vendor display string to `OpenSSH`, and the version display string to `4.1`.

Example: Setting the Operating System

The import file next sets the operating system value for the host using the `SetOS` command. The `Asset Management App` map set contains a product map mapping the third-party product name `Microsoft Win2K` to the Cisco product definition for Microsoft Windows 2000 SP3:



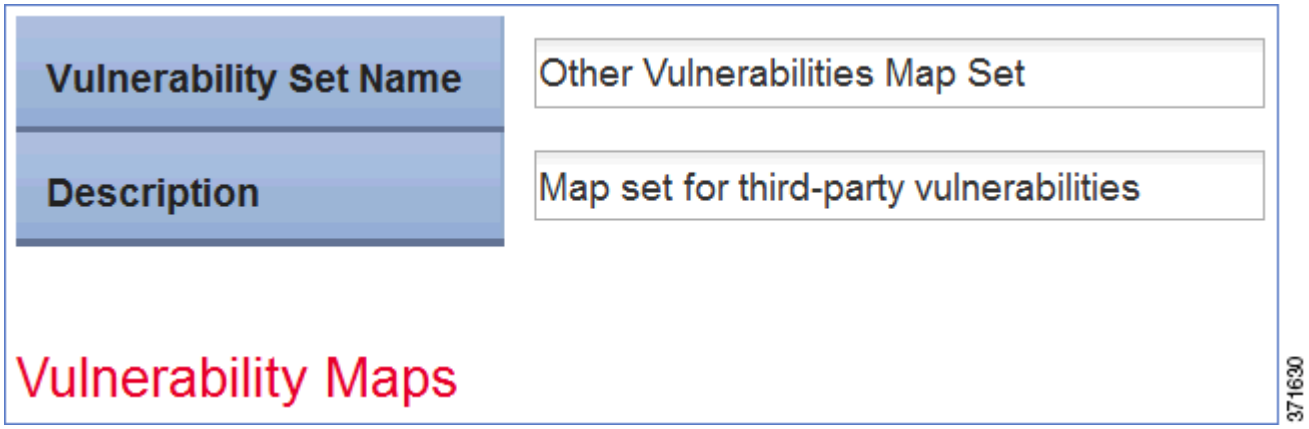
The command in the import file is as follows:

```
# Set the OS. Because the Map is set to "Asset Management App" these values resolve to
the Windows 2000 SP3 definition
#
SetOS, 1.2.3.4, Microsoft, Win2k
```

Note that the `SetOS` command line includes values for the `vendor_str` and `product_str` fields to set the operating system display name to `Microsoft Win2K`. Because those match the **Vendor String** and **Product String** settings defined in the `Asset Management App` product map set, the system maps that third-party operating system name to the Cisco Microsoft Windows 2000 SP3 product definition.

Example: Adding a Third-Party Vulnerability

The import file next imports a third-party vulnerability to the `1.2.3.4` host. This example depends on a third-party vulnerability map set created using the Defense Center web interface:



The command in the import file sets the `Vuln003` vulnerability to valid:

```
# Add a third-party vulnerability (from third-party vulnerability map "Other
Vulnerabilities Map Set") to the host
#
SetValidVuln, 1.2.3.4,,, Other Vulnerabilities Map Set, Vuln0003
```

Example: Setting the Host Criticality

The next command in the import file uses the `SetAttributeValue` command to set the criticality for the `1.2.3.4` host to High.

```
# Set the criticality of the host to "High"
#
SetAttributeValue, 1.2.3.4,criticality,high
```

Note that the attribute name is set to c`riticality` and the attribute value is set to `"high"`.

Example: Add Scan Results

The next set of commands in the import file uses the AddHost command to add a host and then the AddScanResult command to add data for that host from a third-party scanner.

```
# Add IP host for scan results to follow
#
AddHost,1.2.3.5
#
# Add the scan result from a Qualys scanner to the network map
#
AddScanResult,1.2.3.5,"Qualys",82003,,,"ICMP Timestamp Request","ICMP (Internet
Control and Error Message Protocol) is a protocol encapsulated in IP packets. Its
principal purpose is to provide a protocol layer able to inform gateways of the
inter-connectivity and accessibility of other gateways or hosts. ping is a well-known
program for determining if a host is up or down. It uses ICMP echo packets. ICMP
timestamp packets are used to synchronize clocks between hosts.","cve_ids:
CVE-1999-0524","bugtraq_ids:"
```

Example: Running Commands on the Defense Center

The `ScanFlush` command indicates to the Defense Center that it can run the queued commands above the ScanFlush line.

```
ScanFlush
```

Example: Adding a Client Application to the Host

The import file then uses the `AddClientApp` command to add a client application named `BMC Remedy` to the `1.2.3.4` host.

```
# Add a Client App
#
AddClientApp, 1.2.3.4, "BMC Remedy", "Asset Manager", "0.0"
```

Note that the client application ID is set to `BMC Remedy`, the client application type is set to `Asset Manager`, and the version is set to `0.0`.

Example: Adding a MAC-Only Host

Finally, the import file uses the `AddHost` command to add a MAC-only host:

```
# Add a MAC-only host
#
AddHost,,01:02:03:04:05:06
```

Note that the `ip_address` field is left blank and the MAC address is provided instead.

In addition, note that although there is no `ScanFlush` command at the end of the file, the remaining data from the script is sent to the network map when the import file finishes because the session disconnects.

Entire Example File

The full import file explained in the sections above looks like this:

```
# Example import file for Host Input Import Tool
#
# Set the current SOURCE_ID and Product Map to "Asset Management App"
#
SetSource, Asset Management App
SetMap, Asset Management App
#
# Add an IP host with no Primary MAC
#
AddHost,1.2.3.4
#
# Add the ospf protocol to the host
#
AddProtocol, 1.2.3.4,,ospf,xport
#
# Add a server for the host
#
AddService,1.2.3.4, 22, tcp, ssh, OpenSSH, 4.1
#
# Set the OS. Because the Map is set to "Asset Management App" these values resolve to
the Windows 2000 SP3 definition
#
SetOS, 1.2.3.4, Microsoft, Win2k
#
# Add a third-party vulnerability (from third-party map "Other Vulnerabilities Set")
to the host
#
SetValidVuln, 1.2.3.4,,, Other Vulnerabilities Set, Vuln0003
#
# Set the criticality of the host to "High"
#
SetAttributeValue, 1.2.3.4,criticality,high
#
# Add IP host for scan results to follow
#
AddHost,1.2.3.5
#
# Add the scan result from a Qualys scanner to the network map
#
AddScanResult,1.2.3.5,"Qualys",82003,,,"ICMP Timestamp Request","ICMP (Internet
Control and Error Message Protocol) is a protocol encapsulated in IP packets. Its
principal purpose is to provide a protocol layer able to inform gateways of the
inter-connectivity and accessibility of other gateways or hosts. ping is a well-known
```

```
program for determining if a host is up or down. It uses ICMP echo packets. ICMP
timestamp packets are used to synchronize clocks between hosts.","cve_ids:
CVE-1999-0524","bugtraq_ids:"
#
#Send the commands above to the host input service for processing
#
ScanFlush
#
# Add a Client App
#
AddClientApp, 1.2.3.4, "BMC Remedy", "Asset Manager", "0.0"
#
# Add a MAC only host
#
AddHost,,01:02:03:04:05:06
```

Testing Your Import on the Defense Center

You can simulate an import with your import file to make sure it behaves as expected. Because many functions allow you to import duplicate data into the network map, you want to avoid running the same import multiple times. Running a test import avoids that problem. Additionally, the system discards any data in the import file that it cannot interpret, so you want to make sure that the import file will import completely. The test reports the results to the screen (or you can redirect them to a file) so you can then correct any problems with the file before you run the actual import.

Note that if you set up the host input reference client on a remote host with access to the Defense Center, you can use the `ssl_host_input_api_test.pl` script to process an import file from the client. For more information on setting up the reference client, see Setting Up the Host Input Reference Client, page 4-3.

To test an import file:

Step 1 Copy the import file you created to the `/usr/local/sf/bin/` directory on the Defense Center where you want to run the import.

Caution You must log in using an account with read/write access to this directory to import a file.

Step 2 Log into your Defense Center by `ssh`, using an account with `admin` privileges.

Step 3 At the command line, type `/usr/local/sf/bin/nmimport.pl -t` *`filename`*.

Tip To redirect the results of the test import to a log file, add `> logfilename` to the end of the command.

The import test runs, printing messages indicating the results of the import simulation to the screen or to the file you specify.

Running a Host Input Import

You can run the host input import tool from the command line to process the import file you created.

Caution The system discards any data in the import file that it cannot interpret. Additionally, if you run the same import multiple times, you may find duplicate data in your network map for some items. To prevent these issues, you may want to test import of your import file before running the actual import. For more information, see Testing Your Import on the Defense Center, page 3-30.

Note that if you set up the host input reference client on a remote host with access to the Defense Center, you can use the `ssl_host_input_api_test.pl` script to process an import file from the client. For more information on setting up the reference client, see Running the Host Input Reference Client, page 4-5.

To run an import:

Step 1 Copy the import file you created to the `/usr/local/sf/bin/` directory on the Defense Center where you want to run the import.

Caution You must log in using an account with read/write access to this directory in to import a file.

Step 2 Log into your Defense Center using an account with `admin` privileges.

Step 3 At the command line, type `/usr/local/sf/bin/nmimport.pl` *`filename`*.

Tip To redirect the results of the test import to a log file, add `>` *`logfilename`* to the end of the command.

The system adds the imported data to the network map and either displays the result messages on the screen or redirects them to the file you specify.

Configuring Host Input Clients

In addition to accepting host input commands from users on the Defense Center, the Defense Center's host input service also accepts batch import files from authenticated host input clients on external hosts. You can use a host input client to process import files created for the host input import tool and then send the data to the Defense Center to add the information to your network map.

You can use the provided host input API reference client to process and send CSV data or to test your host input client connection to the Defense Center.

Perform the following tasks to manage Defense Center and input client interaction:

Step 1 Establish an authenticated connection to the Defense Center.

See Registering the Host Input Client with the Defense Center, page 4-1 for information about generating authentication credentials to establish an authenticated connection to the Defense Center.

Step 2 Set up a host input client:

- To use the Cisco-provided host input reference client, set up the reference client on the computer where you plan to run it. For more information, see Using the Host Input Reference Client, page 4-2.

  For information on creating import files (also referred to as command files) that you will use your reference client to process, see Writing Host Input Import Files, page 3-3.

- To use your own custom client, make sure it can locate and process the certificate to connect to the Defense Center. For information, see Registering the Host Input Client with the Defense Center, page 4-1.

Registering the Host Input Client with the Defense Center

License: FireSIGHT

Before you can use a host input client, you must register the computer on which the client runs with the Defense Center. The Defense Center then generates an authentication certificate, which you download to your client computer.

To add a host input client:

Access: Admin

Step 1 Select **Local > Registration > Host Input Client.**

The Host Input Client page appears.

Step 2 Click **Create Client**.

The Create Client page appears.

Step 3 In the **Hostname** field, enter the host name or IP address of the host running the host input client.

Note If you use a host name, the host input server **must** be able to resolve the host to an IP address. If you have not configured DNS resolution, you should configure it first or use an IP address.

Step 4 If you want to encrypt the certificate file, enter a password in the **Password** field.

Step 5 Click **Save**.

The host input service allows the client computer to access port 8307 on the Defense Center and creates an authentication certificate to use during client-server authentication. The Host Input Client page re-appears, with the new client listed under **Host Input Clients**.

Step 6 Click the download icon ( ) next to the certificate file.

Step 7 Save the certificate file to the directory used by your client computer for SSL authentication.

The client can now connect to the Defense Center.

Tip To revoke access for a client, click the delete icon ( ) next to the host you want to remove. Note that you do not need to restart the host input service on the Defense Center; access is revoked immediately.

Connecting the Client to the Defense Center

The host input service on the Defense Center reads a version from the client when the client connects. If the client sends a version newer than the version of the server, the service rejects the connection.

In addition, during the initial exchange, the host input service communicates the maximum allowed data size per transaction to the client. If the client attempts to send a data block bigger than the maximum size, the server closes the connection.

Using the Host Input Reference Client

The reference client provided with the host input SDK is a set of sample client scripts and Perl modules that illustrate how you can use the host input API. You can run them to familiarize yourself with host input import, or you can use them to debug problems with installations of your custom-built client. You can also use one of the scripts to process a host input command file from the client.

For more information on setting up reference clients, see the following sections:

- Setting Up the Host Input Reference Client, page 4-3
- Running the Host Input Reference Client, page 4-5

Setting Up the Host Input Reference Client

To use the host input reference client, you must first install the sample scripts and configure your client to fit the script requirements.

For more information, see the following sections:

Understanding the Host Input Reference Client

You can download the `HostInputClientSDK.zip` package, which contains the host input reference client, from the Cisco support site. The Table 4-1Host Input Reference Client Files, page 4-3 lists the files included in the `HostInputClientSDK.zip` package.

Table 4-1 Host Input Reference Client Files

Filename	Description
SFHIClient.pm	This Perl module contains the functions called by the Perl clients.
SFPkcs12.pm	This Perl module parses the client certificate and allows the client to connect to the Defense Center.
ssl_host_input_ api_test.pl	You can use this Perl script to import CSV data by specifying the appropriate input plugin and a command file.
InputPlugins/csv.pm	You can call this Perl module to run a command file that imports CSV data.

Configuring Communications for the Host Input Reference Client

The reference client uses the Secure Sockets Layer (SSL) protocol for data communication. You must install OpenSSL on the computer you plan to use as a client and configure it appropriately for your environment.

To set up SSL on your client:

Step 1 Download OpenSSL from `http://openssl.org/source/`.

Step 2 Unpack the source to `/usr/local/src`.

Step 3 Configure the source by running the Configure script.

Step 4 Make and install the compiled source.

Loading General Prerequisites for the Host Input Reference Client

Before you can run the host input reference client, you must install the `IO::Socket::SSL` Perl module on the client computer. You can install the module manually or use `cpan` to do so.

Note If the `Net::SSLeay` module is not installed on the client computer, install that module as well. `Net::SSLeay` is required for communication with OpenSSL.

You also need to install and configure OpenSSL to support an SSL connection to the Defense Center. For more information, see Configuring Communications for the Host Input Reference Client, page 4-3.

In addition, if you plan to use the Qualys plugin with the host input client, you must install the `XML::Smart` Perl module and its prerequisites. If you plan to use IPv6 to communicate between the client and the Defense Center, you must also install the `IO::Socket::INET6` Perl module.

Downloading and Unpacking the Host Input Reference Client

You can download the `HostInputClientSDK.zip` file that contains the host input reference client from the Support site.

Unpack the zip file to a computer running the Linux operating system, where you plan to run the client.

Creating a Certificate for the Host Input Reference Client

License: FireSIGHT

Before you can use the host input reference client, you need to create a certificate on the Defense Center for the computer where you want to run the client. You then download that file to the client computer.

To create a certificate for the reference client:

Access: Admin

Step 1 Select **Local** > **Registration** > **Host Input Client**.

The Host Input Client page appears.

Step 2 Click **Create Client**.

The Create Client page appears.

Step 3 In the **Hostname** field, enter the host name or IP address of the host running the host input reference client.

If you use a host name, the Defense Center **must** be able to resolve the host to an IP address. If you have not configured DNS resolution on the Defense Center or if a reverse lookup is not available, you should configure DNS first or use an IP address. Refer to the *FireSIGHT System User Guide* or the online help for more information about configuring DNS settings.

Step 4 Click **Save**.

The Defense Center now allows the host to access the Defense Center and creates an authentication certificate to use during client-server authentication. The Host Input Client page appears again, with the new client listed under **Hostname**.

Step 5 Click the download icon ( ) next to the client hostname to download the certificate file.

Step 6 Save the certificate file to the directory where you put the reference client.

The client can now connect to the Defense Center. You do not need to restart the host input service.

Tip To revoke access for a client, click **Delete** next to the host you want to remove. Note that you do not need to restart the host input service on the Defense Center; access is revoked immediately.

Running the Host Input Reference Client

The Host Input Perl reference client scripts are designed for use on an operating system with the Linux kernel but should work on any POSIX-based operating system, as long as the client machine meets the prerequisites defined in Setting Up the Host Input Reference Client, page 4-3.

You can use the reference client to import CSV data from a remote client to the network map on a Defense Center.

Use the following syntax to run the `ssl_host_input_api_test.pl` script:

```
./ssl_host_input_api_test.pl csv CSVCommandFile Defense CenterIPAddress
```

For example, to import using a CSV file named `csv_file.txt` to a Defense Center with an IP address of 10.10.0.4:

```
./ssl_host_input_api_test.pl csv csv_file.txt 10.10.0.4
```

Network Protocol Values

Using the `AddProtocol` and `DeleteProtocol` functions, you can add protocols to or remove protocols from hosts. The following table details the available network protocol values.

Table A-1 Network Protocol Values

Value	Description
`IP`	Internet Protocol version 4
`ARP`	Address Resolution Protocol
`BPDU(STP)`	Bridge Protocol Data Unit (Spanning Tree Protocol)
`RARP`	Reverse Address Resolution Protocol
`OldIPX`	Internetwork Packet Exchange, older version
`IP Version 6`	Internet Protocol version 6
`Loopback`	Loopback
`SNAP`	Subnetwork Access Protocol
`Novell NetWare`	Novell NetWare
`NetBIOS`	Network Basic Input/Output System
`NetBIOS (Response)`	Network Basic Input/Output System response
`IPX`	Internetwork Packet Exchange
`Intel ANS`	Intel Advanced Network Services
`DEC MOP Dump/Load Assistance`	Digital Equipment Corporation Maintenance Operations Protocol dump/load assistance
`DEC MOP Remote Console`	Digital Equipment Corporation Maintenance Operations Protocol remote console
`PPPoE Discovery`	Point-to-Point over Ethernet discovery stage
`PPPoE Session`	Point-to-Point over Ethernet session stage

Symbols

Numerics

A

C

D

E

F

H

I

K

M

O

P

R

S

T

U

V