



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, page 2-1](#)
- [Running a Host Input API Script, page 2-2](#)
- [Host Input API Functions, page 2-5](#)
- [Example Host Input API Scripts, page 2-38](#)

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:

- [Calling the Host Input Module, page 2-1](#)
- [Setting the Source Type, page 2-1](#)
- [Obtaining a Source ID, page 2-2](#)
- [Required Fields, page 2-2](#)

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 2-39](#) 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 2-39](#) 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 2-39](#) 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, page 2-3](#)
- [Setting a Third-Party Vulnerability Map, page 2-3](#)
- [Setting a Third-Party Product Map, page 2-3](#)

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

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

The User Third-Party Mappings page appears.

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.

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

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

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.

6. Type the vendor string used by the third-party product in the **Vendor String** field.
7. Type the product string used by the third-party product in the **Product String** field.
8. Type the version string used by the third-party product in the **Version String** field.
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.

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

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

The Application Detectors page appears.

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

The Third-Party Product Mappings page appears.

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

The Edit Third-Party Product Mappings page appears.

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, page 2-5](#)
- [Server Functions, page 2-10](#)
- [Client Application Functions, page 2-16](#)
- [Protocol Functions, page 2-19](#)
- [Package Fix Functions, page 2-22](#)
- [Host Attribute Functions, page 2-24](#)
- [Vulnerabilities Functions, page 2-29](#)
- [Third-Party Mapping Functions, page 2-33](#)
- [AddScanResult Function, page 2-34](#)

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, page 2-5](#)
- [DeleteHost, page 2-6](#)
- [SetOS, page 2-7](#)
- [UnsetOS, page 2-9](#)

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 2-39](#) for an example of this function used in a script.

Use this syntax:

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

Table 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 2-1 .
\$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 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 2-1 .
\$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 .

Table 2 DeleteHost Fields (continued)

Field	Description	Required	Allowed Values
\$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 2-33](#).

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 2-40](#) 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 3 SetOS Fields

Field	Description	Required	Allowed Values
\$source_type_id	Indicates the type of the host input source.	Yes	<p>"Application" or "Scanner"</p> <p>Note you should set the <code>\$source_type_id</code> variable to contain the appropriate value before invoking the <code>SetOS</code> function, and then reference <code>\$source_type_id</code> in your function call. For more information, see Setting the Source Type, page 2-1.</p>
\$source_id	Indicates the source ID for the source adding the host input.	Yes	<p>"source_id"</p> <p>Note you should set the <code>\$source_id</code> variable to contain the source ID before invoking the <code>SetOS</code> function, and then reference <code>\$source_id</code> in your function call. For more information, see Obtaining a Source ID, page 2-2.</p>
\$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)	<p>A list of attribute value hash pairs of the format:</p> <pre>{attribute => "Department", value => "Development"},</pre> <p>Note that <code>\$attrib_list</code> must be an array or reference an array.</p>
\$os	Contains a hash with keys describing the details of an operating system definition.	If you set a current third-party map before calling <code>SetOS</code> , only the rendering keys are required.	The <code>\$os</code> 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 2-33](#).

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.
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 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 <code>\$source_type_id</code> variable to contain the appropriate value before invoking the <code>UnsetOS</code> function, and then reference <code>\$source_type_id</code> in your function call. For more information, see Setting the Source Type, page 2-1 .
\$source_id	Indicates the source ID for the source adding the host input.	Yes	"source_id" Note you should set the <code>\$source_id</code> variable to contain the source ID before invoking the <code>UnsetOS</code> function, and then reference <code>\$source_id</code> 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: <pre>{attribute => "Department", value => "Development"},</pre> Note that <code>\$attrib_list</code> 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, page 2-10](#)
- [SetService, page 2-11](#)
- [UnsetService, page 2-13](#)
- [Service Keys, page 2-14](#)

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 2-42](#) 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 7 AddService Fields

Field	Description	Required	Allowed Values
\$source_type_id	Indicates the type of the host input source.	Yes	<p>"Application" or "Scanner"</p> <p>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 2-1.</p>
\$source_id	Indicates the source ID for the source adding the host input.	Yes	<p>"source_id"</p> <p>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.</p>
\$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)	<p>A list of attribute value hash pairs of the format:</p> <pre>{attribute => "Department", value => "Development"},</pre> <p>Note that \$attrib_list must be an array or reference an array.</p>
\$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 2-14 .

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 2-33](#)), 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 8 SetService Fields

Field	Description	Required	Allowed Values
\$source_type_id	Indicates the type of the host input source.	Yes	<p>"Application" or "Scanner"</p> <p>Note you should set the <code>\$source_type_id</code> variable to contain the appropriate value before invoking the <code>SetService</code> function, and then reference <code>\$source_type_id</code> in your function call. For more information, see Setting the Source Type, page 2-1.</p>
\$source_id	Indicates the source ID for the source adding the host input.	Yes	<p>"source_id"</p> <p>Note you should set the <code>\$source_id</code> variable to contain the source ID before invoking the <code>SetService</code> function, and then reference <code>\$source_id</code> in your function call. For more information, see Obtaining a Source ID, page 2-2.</p>
\$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)	<p>A list of attribute value hash pairs of the format:</p> <pre>{attribute => "Department", value => "Development"},</pre> <p>Note that <code>\$attrib_list</code> must be an array or reference an array.</p>
\$service	Contains a hash with keys describing the details of a server definition.	If you set a current third-party map before calling <code>SetService</code> , only the <code>rendering</code> and <code>service_name</code> or <code>service_id</code> keys are required.	The <code>\$service</code> variable is a server definition hash that supports several keys. For more information, see Service Keys, page 2-14 .

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 9 UnsetService Fields

Field	Description	Required	Allowed Values
<code>\$source_type_id</code>	Indicates the type of the host input source.	Yes	"Application" or "Scanner" Note you should set the <code>\$source_type_id</code> variable to contain the appropriate value before invoking the <code>UnsetService</code> function, and then reference <code>\$source_type_id</code> in your function call. For more information, see Setting the Source Type, page 2-1 .
<code>\$source_id</code>	Indicates the source ID for the source adding the host input.	Yes	"source_id" Note you should set the <code>\$source_id</code> variable to contain the source ID before invoking the <code>UnsetService</code> function, and then reference <code>\$source_id</code> in your function call. For more information, see Obtaining a Source ID, page 2-2 .
<code>\$addr_string</code>	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.
<code>\$port</code>	Indicates the port for the server to be unset.	Yes	Integers in the range of 1-65535.
<code>\$proto</code>	Indicates the protocol for the server to be unset.	Yes	Either the strings <code>tcp</code> or <code>udp</code> or the appropriate protocol IDs 6 (<code>tcp</code>) or 17 (<code>udp</code>).

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 10 DeleteService Fields

Field	Description	Required	Allowed Values
\$source_type_id	Indicates the type of the host input source.	Yes	<p>"Application" or "Scanner"</p> <p>Note you should set the <code>\$source_type_id</code> variable to contain the appropriate value before invoking the <code>DeleteService</code> function, and then reference <code>\$source_type_id</code> in your function call. For more information, see Setting the Source Type, page 2-1.</p>
\$source_id	Indicates the source ID for the source adding the host input.	Yes	<p>"source_id"</p> <p>Note you should set the <code>\$source_id</code> variable to contain the source ID before invoking the <code>DeleteService</code> function, and then reference <code>\$source_id</code> in your function call. For more information, see Obtaining a Source ID, page 2-2.</p>
\$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)	<p>A list of attribute value hash pairs of the format:</p> <pre>{attribute => "Department", value => "Development"},</pre> <p>Note that <code>\$attrib_list</code> must be an array or reference an array.</p>
\$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 <code>tcp</code> or <code>udp</code> or the appropriate protocol IDs 6 (<code>tcp</code>) or 17 (<code>udp</code>).

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
<code>port</code>	<code>uint</code>	Use this key in combination with the <code>proto</code> key and the address or attribute specifications to specify the server on the hosts where you want to modify the vulnerability listings.
<code>proto</code>	<code>string</code>	Use this key in combination with the <code>port</code> 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 <code>tcp</code> or <code>udp</code> or the appropriate protocol IDs 6 (<code>tcp</code>) or 17 (<code>udp</code>).

Table 2-12 *Keys for Rendering*

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

Table 2-13 *Keys for Vulnerability Mapping (continued)*

Key	Data Type	Definition
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 <code>fix_ids</code> 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

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

The Object Management page appears.

2. Click **Application Filters.**

The Application Filters section appears.

3. Click **Add Application Filter.**

The Application Filter pop-up window appears.

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 2-16](#)
- [DeleteClientApp, page 2-17](#)
- [DeleteClientAppPayload, page 2-18](#)

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 2-42](#) 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 14 AddClientApp Fields

Field	Description	Required	Allowed Values
\$source_type_id	Indicates the type of the host input source.	Yes	<p>"Application" or "Scanner"</p> <p>Note you should set the <code>\$source_type_id</code> variable to contain the appropriate value before invoking the <code>AddClientApp</code> function, and then reference <code>\$source_type_id</code> in your function call. For more information, see Setting the Source Type, page 2-1.</p>
\$source_id	Indicates the source ID for the source adding the host input.	Yes	<p>"source_id"</p> <p>Note you should set the <code>\$source_id</code> variable to contain the source ID before invoking the <code>AddClientApp</code> function, and then reference <code>\$source_id</code> in your function call. For more information, see Obtaining a Source ID, page 2-2.</p>
\$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)	<p>A list of attribute value hash pairs of the format:</p> <pre>{attribute => "Department", value => "Development"},</pre> <p>Note that <code>\$attrib_list</code> must be an array or reference an array.</p>
\$id	Indicates the client application name.	Yes	<p>A string consisting of alphanumeric characters or spaces, enclosed in double quotes.</p> <p>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.</p>
\$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 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 2-1 .
\$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 16 DeleteClientAppPayload Fields

Field	Description	Required	Allowed Values
\$source_type_id	Indicates the type of the host input source.	Yes	<p>"Application" or "Scanner"</p> <p>Note you should set the <code>\$source_type_id</code> variable to contain the appropriate value before invoking the <code>DeleteClientAppPayload</code> function, and then reference <code>\$source_type_id</code> in your function call. For more information, see Setting the Source Type, page 2-1.</p>
\$source_id	Indicates the source ID for the source adding the host input.	Yes	<p>"source_id"</p> <p>Note you should set the <code>\$source_id</code> variable to contain the source ID before invoking the <code>DeleteClientAppPayload</code> function, and then reference <code>\$source_id</code> in your function call. For more information, see Obtaining a Source ID, page 2-2.</p>
\$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)	<p>A list of attribute value hash pairs of the format:</p> <pre>{attribute => "Department", value => "Development"},</pre> <p>Note that <code>\$attrib_list</code> must be an array or reference an array.</p>
\$id	Indicates the client application name.	Yes	<p>A string consisting of alphanumeric characters or spaces, enclosed in double quotes.</p> <p>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.</p>
\$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	<p>A string consisting of alphanumeric characters or spaces, enclosed in double quotes.</p> <p>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.</p>

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 2-20
- [AddProtocol](#), page 2-21

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 17 DeleteProtocol Fields

Field	Description	Required	Allowed Values
<code>\$source_type_id</code>	Indicates the type of the host input source.	Yes	"Application" or "Scanner" Note you should set the <code>\$source_type_id</code> variable to contain the appropriate value before invoking the <code>DeleteProtocol</code> function, and then reference <code>\$source_type_id</code> in your function call. For more information, see Setting the Source Type , page 2-1.
<code>\$source_id</code>	Indicates the source ID for the source adding the host input.	Yes	"source_id" Note you should set the <code>\$source_id</code> variable to contain the source ID before invoking the <code>DeleteProtocol</code> function, and then reference <code>\$source_id</code> in your function call. For more information, see Obtaining a Source ID , page 2-2.
<code>\$addr_string</code>	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.
<code>\$attrib_list</code>	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 <code>\$attrib_list</code> must be an array or reference an array.
<code>\$mac_list</code>	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 <code>\$mac_list</code> must be an array or reference an array.
<code>\$proto</code>	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 <code>/etc/protocols</code> file are acceptable. For network protocols ("net"), see Network Protocol Values , page A-1.
<code>\$type</code>	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 2-41](#) 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 18 AddProtocol Fields

Field	Description	Required	Allowed Values
<code>\$source_type_id</code>	Indicates the type of the host input source.	Yes	<p>"Application" or "Scanner"</p> <p>Note you should set the <code>\$source_type_id</code> variable to contain the appropriate value before invoking the <code>AddProtocol</code> function, and then reference <code>\$source_type_id</code> in your function call. For more information, see Setting the Source Type, page 2-1.</p>
<code>\$source_id</code>	Indicates the source ID for the source adding the host input.	Yes	<p>"source_id"</p> <p>Note you should set the <code>\$source_id</code> variable to contain the source ID before invoking the <code>AddProtocol</code> function, and then reference <code>\$source_id</code> in your function call. For more information, see Obtaining a Source ID, page 2-2.</p>
<code>\$addr_string</code>	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.
<code>\$attrib_list</code>	Indicates the host attribute or attributes specifying the hosts affected by the host input.	Yes (unless IP addresses or MAC addresses are provided)	<p>A list of attribute value hash pairs of the format:</p> <pre>{attribute => "Department", value => "Development"},</pre> <p>Note that <code>\$attrib_list</code> must be an array or reference an array.</p>
<code>\$mac_list</code>	Indicates the list of MAC addresses for the affected hosts.	Yes (unless IP addresses or attribute lists are provided)	<p>A list of MAC address strings, with or without separating colons.</p> <p>Note that <code>\$mac_list</code> must be an array or reference an array.</p>

Table 18 AddProtocol Fields (continued)

Field	Description	Required	Allowed Values
\$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 2-22](#)
- [RemoveFix, page 2-23](#)

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 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 2-1 .
\$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 2-33 .

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.

Note: 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 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 <code>\$source_type_id</code> variable to contain a value before invoking the <code>RemoveFix</code> function, and then reference <code>\$source_type_id</code> in your function call. For more information, see Setting the Source Type, page 2-1 .
\$source_id	Indicates the source ID for the source adding the host input.	Yes	“source_id” Note you should set the <code>\$source_id</code> variable to contain the source ID before invoking the <code>RemoveFix</code> function, and then reference <code>\$source_id</code> 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 <code>\$attrib_list</code> must be an array or reference an array.
\$port	With the <code>\$proto</code> 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 <code>\$port</code> field, indicates the server affected by the fix.	No	Either the strings <code>tcp</code> or <code>udp</code> or the appropriate protocol IDs 6 (<code>tcp</code>) or 17 (<code>udp</code>).
\$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 <code>SetCurrent3rdPartyMap</code> function before invoking the <code>RemoveFix</code> function. For more information, see SetCurrent3rdPartyMap, page 2-33 .

Host Attribute Functions

For more information, see the following sections:

- [AddHostAttribute, page 2-25](#)
- [DeleteHostAttribute, page 2-25](#)
- [SetAttributeValue, page 2-26](#)
- [DeleteAttributeValue, page 2-27](#)

- [SetCriticality, page 2-28](#)

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 2-26](#).

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 21 AddHostAttribute Fields

Field	Description	Required	Allowed Values
<code>\$source_type_id</code>	Indicates the type of the host input source.	Yes	"Application" or "Scanner" Note you should set the <code>\$source_type_id</code> variable to contain a value before invoking the <code>AddHostAttribute</code> function, and then reference <code>\$source_type_id</code> in your function call. For more information, see Setting the Source Type, page 2-1 .
<code>\$source_id</code>	Indicates the source ID for the source adding the host input.	Yes	"source_id" Note you should set the <code>\$source_id</code> variable to contain the source ID before invoking the <code>AddHostAttribute</code> function, and then reference <code>\$source_id</code> in your function call. For more information, see Obtaining a Source ID, page 2-2 .
<code>\$attrib_name</code>	Indicates the host attribute or attributes specifying the hosts affected by the host input.	Yes	"Department"
<code>\$attrib_type</code>	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 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 <code>\$source_type_id</code> variable to contain a value before invoking the <code>AddHostAttribute</code> function, and then reference <code>\$source_type_id</code> in your function call. For more information, see Setting the Source Type, page 2-1 .
\$source_id	Indicates the source ID for the source adding the host input.	Yes	“source_id” Note you should set the <code>\$source_id</code> variable to contain the source ID before invoking the <code>DeleteHostAttribute</code> function, and then reference <code>\$source_id</code> 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 2-42](#) 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 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 <code>\$source_type_id</code> variable to contain a value before invoking the <code>SetAttributeValue</code> function, and then reference <code>\$source_type_id</code> in your function call. For more information, see Setting the Source Type, page 2-1 .
\$source_id	Indicates the source ID for the source adding the host input.	Yes	“source_id” Note you should set the <code>\$source_id</code> variable to contain the source ID before invoking the <code>SetAttributeValue</code> function, and then reference <code>\$source_id</code> in your function call. For more information, see Obtaining a Source ID, page 2-2 .

Table 23 SetAttributeValue Fields (continued)

Field	Description	Required	Allowed Values
\$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 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 <code>DeleteAttributeValue</code> function, and then reference \$source_type in your function call. For more information, see Setting the Source Type, page 2-1 .
\$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 <code>DeleteAttributeValue</code> 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.

Table 24 DeleteAttributeValue 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.
\$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 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 <code>SetCriticality</code> function, and then reference \$source_type_id in your function call. For more information, see Setting the Source Type, page 2-1 .
\$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 <code>SetCriticality</code> 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.

Table 25 SetCriticality 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.
\$criticality	Indicates the criticality level for the host.	Yes	The identification number or string for the criticality level: <ul style="list-style-type: none"> ■ "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 2-29](#)
- [SetValidVulns, page 2-30](#)

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 26 SetInvalidVulns Fields

Field	Description	Required	Allowed Values
\$source_type_id	Indicates the type of the host input source.	Yes	<p>“Application” or “Scanner”</p> <p>Note you should set the <code>\$source_type_id</code> variable to contain a value before invoking the <code>SetInvalidVulns</code> function, and then reference <code>\$source_type_id</code> in your function call. For more information, see Setting the Source Type, page 2-1.</p>
\$source_id	Indicates the source ID for the source adding the host input.	Yes	<p>“source_id”</p> <p>Note you should set the <code>\$source_id</code> variable to contain the source ID before invoking the <code>SetInvalidVulns</code> function, and then reference <code>\$source_id</code> in your function call. For more information, see Obtaining a Source ID, page 2-2.</p>
\$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)	<p>A list of attribute value hash pairs of the format:</p> <pre>{attribute => “Department”, value => “Development”},</pre> <p>Note that <code>\$attrib_list</code> must be an array or reference an array.</p>
\$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 2-31 .
\$vuln_type	Indicates the type of the vulnerability.	Yes	<p>Any of the following:</p> <ul style="list-style-type: none"> ■ rna ■ name of custom third-party vulnerability map set <p>For more information on mapping third-party vulnerabilities, see the <i>FireSIGHT System User Guide</i> or see SetCurrent3rdPartyMap, page 2-33.</p>

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 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 <code>\$source_type_id</code> variable to contain a value before invoking the <code>SetValidVulns</code> function, and then reference <code>\$source_type_id</code> in your function call. For more information, see Setting the Source Type, page 2-1 .
\$source_id	Indicates the source ID for the source adding the host input.	Yes	“source_id” Note you should set the <code>\$source_id</code> variable to contain the source ID before invoking the <code>SetValidVulns</code> function, and then reference <code>\$source_id</code> 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: <pre>{attribute => “Department”, value => “Development”},</pre> Note that <code>\$attrib_list</code> 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 2-31 .
\$vuln_type	Indicates the type of the vulnerability.	Yes	Any of the following: <ul style="list-style-type: none"> ■ rna ■ name of custom third-party vulnerability map set For more information on mapping third-party vulnerabilities, see the <i>FireSIGHT System User Guide</i> or see SetCurrent3rdPartyMap, page 2-33 .

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 28 Keys for Vulnerability Mapping

Key	Data Type	Used by	Definition
cve_ids	string	\$mapping_vuln_list	<p>A comma-separated list of CVE IDs, with each ID enclosed in single quotes.</p> <p>If this field, vuln_id, and bugtraq_ids are empty, this is a generic scan result</p> <p>Use this key to specify the CVE ID for any vulnerabilities on the hosts.</p>
bugtraq_ids	uint	\$mapping_vuln_list	<p>A comma-separated list of BugTraq IDs, with each ID enclosed in single quotes.</p> <p>If this field, vuln_id, and cve_ids are empty, this is a generic scan result.</p> <p>Use this key to specify the BugTraq ID for any vulnerabilities on the hosts.</p>
vuln_id	string	\$vulns and \$mapping_vuln_list	<p>A string, enclosed in single quotes.</p> <p>If this field, bugtraq_ids, and cve_ids are empty, this is a generic scan result.</p> <p>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.</p>

Table 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 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 2-33](#)
- [UnsetCurrent3rdPartyMap, page 2-33](#)

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 2-40](#) 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 2-43](#), [Example: Adding a Generic Scan Result to a Host, page 2-43](#), and [Full Example Script, page 2-44](#).

Use this syntax:

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

Table 31 AddScanResult Fields

Field	Description	Required	Allowed Values
\$scanner_id	Indicates the scanner ID for the scanner that obtained the scan results.	Yes	<p>"scanner_id"</p> <p>where <code>scanner_id</code> is a string indicating the name of the scanner that is the source of the vulnerability data you add.</p> <p>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.</p> <p>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 <i>FireSIGHT System User Guide</i>.</p>
\$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 31 AddScanResult Fields (continued)

Field	Description	Required	Allowed Values
\$mapping_vuln_list	Indicates scan results with vulnerability IDs for the affected hosts.	Yes	<p>A list of vulnerability hash values of the format:</p> <pre>{ '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...', }</pre> <p>Note that \$mapping_vuln_list must be an array or reference an array.</p> <p>Uses a hash of vulnerability keys to set vulnerability information. For more information, see Vulnerability Keys, page 2-31.</p>

Table 31 AddScanResult Fields (continued)

Field	Description	Required	Allowed Values
\$generic_item_list	Indicates scan results without vulnerability IDs for the affected hosts.	Yes	<p>A list of vulnerability hash values of the format:</p> <pre>{ 'port' => 107, 'proto' => 17, 'name' => 'Using NetBIOS to retrieve info from a Windows host', 'desc' => 'The following 2 NetBIOS names have been gathered...'</pre> <p>Note that \$generic_item_list must be an array or reference an array.</p> <p>Uses a hash of vulnerability keys to set vulnerability information. For more information, see Vulnerability Keys, page 2-31.</p>
\$flag	Indicates how the scan result should be processed.	Yes	<p>The number for the action to be performed:</p> <ul style="list-style-type: none"> ■ 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 2-44](#) and [Full Example Script, page 2-44](#).

Use this syntax:

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

Table 32 DeleteScanResult Fields

Field	Description	Required	Allowed Values
\$scanner_id	Indicates the scanner ID for the scanner that obtained the scan results.	Yes	<p>"scanner_id"</p> <p>where <code>scanner_id</code> is a string indicating the name of the scanner that is the source of the vulnerability data you add.</p> <p>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.</p> <p>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 <i>FireSIGHT System User Guide</i>.</p>
\$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	<p>A list of vulnerability hash values of the format:</p> <pre>{ 'vuln_id' => 10150, 'port' => 107, 'proto' => 17, }</pre> <p>Note that <code>\$mapping_vuln_list</code> must be an array or reference an array.</p> <p>Uses a hash of vulnerability keys to set vulnerability information. For more information, see Vulnerability Keys, page 2-31.</p> <p>If <code>vuln_id</code> is used, only the specified <code>vuln_id</code> is removed.</p>

Table 32 DeleteScanResult Fields (continued)

Field	Description	Required	Allowed Values
\$generic_item_list	Indicates scan results without vulnerability IDs for the affected hosts.	Yes	<p>A list of vulnerability hash values of the format:</p> <pre>{ 'port' => 107, 'proto' => 17, 'name' => 'Using NetBIOS to retrieve info from a Windows host', 'desc' => 'The following 2 NetBIOS names have been gathered...', }</pre> <p>Note that <code>\$generic_item_list</code> must be an array or reference an array.</p> <p>Uses a hash of vulnerability keys to set vulnerability information. For more information, see Vulnerability Keys, page 2-31.</p>
\$flag	Indicates how the scan result should be processed.	Yes	<p>The number for the action to be performed:</p> <ul style="list-style-type: none"> ■ 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 <code>\$mapping_vuln_list</code> 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 <code>\$generic_item_list</code> 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 <code>\$mapping_vuln_list</code> and <code>\$generic_item_list</code> 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, page 2-39](#)
- [Example: Setting the Source Type, page 2-39](#)

- [Example: Setting the Source ID, page 2-39](#)
- [Example: Adding a Host to the Network Map, page 2-39](#)
- [Example: Setting the Operating System on the Host, page 2-40](#)
- [Example: Adding a Protocol to the Host, page 2-41](#)
- [Example: Adding a Server to the Host, page 2-42](#)
- [Example: Setting the Host Criticality, page 2-42](#)
- [Example: Adding a Client Application to Multiple Hosts, page 2-42](#)
- [Example: Adding a Scan Result to a Host, page 2-43](#)
- [Full Example Script, page 2-44](#)

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:

Mapping Set Name	Asset Management App
Description	Product Mapping Set for an asset manag...

Product Maps + Add Product Map

Vendor String	Product String	Version String

Fix Maps + Add Fix Map

Fix String

Save Cancel

371628

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);
```

