CARBON
BLACK

# Cb Response Integration Guide

Release: 6.1

Document Date: May 2017

# Copyrights and Notices

**Carbon Black, Inc.**
1100 Winter Street, Waltham, MA 02451 USA
Tel: 617.393.7400
Fax: 617.393.7499
Email: support@carbonblack.com
Web: http://www.carbonblack.com

*Cb Response Integration Guide*
Document Revision Date: May 2, 2017
Product Version: 6.1

# Before You Begin

This preface provides a brief orientation to the *Cb Response Integration Guide*.

**Sections**

## What This Document Covers

This documentation provides information for administrators who are responsible for integrating Cb Response with various tools. It discusses:

- Integration with Cb Protection (formerly Bit9).
- Integration with Microsoft Enhanced Mitigation Experience Toolkit (EMET).
- Supported SAML 2.0 specifications and SAML 2.0 Single Sign-On (SSO) setup. This includes integration with the OKTA, Shibboleth, and ADFS IdPs.
- The Duo plugin, which you can configure two-factor authentication and download the Duo Mobile application on a mobile device.
- Syslog output for Cb Response events.
- Cb Response support for Virtual Desktop Infrastructure (VDI) and how to configure your machines to use it.

The following table summarizes the contents of this guide:

| | Chapter | Description |
|---|---|---|
| **1** | Integrating Cb Response with Cb Protection | Describes the procedure for integrating Cb Response with Cb Protection. It also describes the features available when this integration is active, as well as general features that contribute to the coexistence of the Cb Response sensor and Cb Protection agent on the same computer. |
| **2** | Integrating EMET with Cb Response | Describes the procedure for integrating a Cb Response server with the Microsoft Enhanced Mitigation Experience Toolkit (EMET). |
| **3** | Integrating with SSO Identity Providers | Describes supported SAML 2.0 specifications and SAML 2.0 Single Sign-On (SSO) setup. It also explains how to integrate with the OKTA, Shibboleth, and ADFS IdPs. |
| **4** | Integrating with Third-Party Authentication | Describes how to integrate Duo plugin, you can configure two-factor authentication and download the Duo Mobile application on a mobile device. |
| **5** | Syslog Output for Cb Response Events | Describes syslog output for Cb Response events. It provides descriptions and examples of the output, and explains how you can use syslog output for notification of alerts. |
| **6** | Server VDI Support | Describes Cb Response support for Virtual Desktop Infrastructure (VDI) and how to configure your machines to use it. |

## Other Documentation

You may need some or all of the following documentation to accomplish tasks that are not covered in the *Cb Response Integration Guide.* These documents, as well as other technical support solution documents, are available on the Carbon Black User eXchange.

The technical solutions documents are a source of information that is maintained as a knowledge base. Some of these documents are updated with every new released build, while others are updated only for minor or major version changes.

- *Cb Response User Guide* – Describes the Cb Response product and explains how to use all of its features and perform administration tasks.
- *Cb Response Integration Guide* – Provides information for administrators who are responsible for integrating Cb Response with various tools, such as Cb Protection, EMET, VDI, SSO, and more.
- *Cb Connector Add-On Guide*– Describes how to integrate Carbon Black products with a variety of third-party connectors.
- *Cb Response Server Sizing Guide (OER)* – Describes performance and scalability considerations in deploying a Cb Response server.
- *Cb Response Server Configuration (cb.conf) Guide* – Provides all of the `cb.conf` configuration file functions, descriptions, and parameters.
- *Cb Response Unified View Server Guide* – Describes how to install and use the Cb Response Unified View server, which is a standalone server that ties multiple Cb Response servers together and provides a single interface. The Unified View interface allows users to perform queries across multiple Cb Response servers with a unified result set. The Unified View server also allows users to manipulate the full functionality of a single Cb Response server.
- *Cb Response Release Notes* – Provides information about new and modified features, issues resolved and general improvements in this release, and known issues and limitations. It also includes required or suggested preparatory steps before installing the server.

Other Carbon Black documentation is referenced in the appropriate sections of this guide.

## Carbon Black Technical Support

For your convenience, you can contact Carbon Black Technical Support in a variety of ways.

| Technical Support Contact Options |
|---|
| Web: http://www.carbonblack.com |
| Email: support@carbonblack.com |
| Phone: Tel: 617.393.7400 |
| Fax: 617.393.7499 |

When you call Carbon Black Technical Support, provide the following information to the support representative:

| Required Information | Description |
|---|---|
| **Contact** | Your name, company name, telephone number, and email address. |
| **Product version** | Product name and version number. |
| **Document version** | For documentation issues, specify the date and version of the manual you are using. |
| **Problem** | Action causing the problem, error message returned, and event log output (as appropriate). |
| **Problem severity** | Critical, serious, minor, or enhancement. |

# Contents

Chapter 2

# Integrating Cb Response with Cb Protection

This chapter describes the procedure for integrating Cb Response with Cb Protection. It also describes the features available when this integration is active, as well as general features that contribute to the coexistence of the Cb Response sensor and Cb Protection agent on the same computer.

| **Note** | *The terms "Cb Protection" and "Bit9" are used interchangeably in this document. The product name is in the middle of being changed from Bit9 to Cb Protection.* |
|---|---|

**Sections**

## Overview

Cb Response provides powerful features for endpoint threat detection and response. Cb Protection provides its own powerful features for endpoint threat protection. Both solutions have been engineered to allow you to take advantage of their complementary features by installing both the Cb Response sensor and the Cb Protection agent on your endpoints and adding Cb Response features inside Cb Protection to improve your security posture.

| | |
|---|---|
| ***Note*** | *In addition to integrating with each other, both Cb Response and the Cb Protection can take advantage of Cb Threat Intel, which provides reputation information, threat indicators, and attack classification intelligence. See "Threat Intelligence" in the Cb Response User Guide for more information on Cb Threat Intel.* |

### Built-in Compatibility Features

The Cb Protection agent recognizes the Cb Response sensor and reports its presence to the Cb Protection server. The Cb Protection server has many optimizations to allow efficient operation of both the Cb Protection agent and Cb Response sensor on the same endpoint. These include:

- **Performance Optimizations** – Internal performance optimizations nearly eliminate any performance impact on either product from having the other product's agent or sensor installed.
- **Trust for Sensor Updates** – An Updater rule allows seamless installation and upgrades of Cb Response sensors that would otherwise have been blocked by a Cb Protection agent in Medium and High enforcement modes. See "Approving by Updater" in the Cb Protection console help for more information about updater rules.
- **Publisher Trust for Cb Response** – A Publisher rule in Cb Protection trusts (by default) files that are identified as being from the publisher "Carbon Black, Inc.". See "Approving or Banning by Publisher" in the Cb Protection console help for more information about publisher rules.
- **Server Integration Interface** – The **Licensing** tab on the Cb Protection System Configuration page includes a **Carbon Black** tab that can be used to activate integration with a specific Cb Response server. Step-by-step instructions are detailed in "Activating Cb Response-to-Cb Protection Integration" on page 15. The features associated with this integration are listed in the following section.

## Features when Servers are Integrated

Additional Cb Response-Cb Protection integration features become available when you explicitly configure the two servers to work with each other. The majority of these features involve making information from and about Cb Response available in the Cb Protection console:

- **Cb Response Sensor Details in Cb Protection Console** – Pages displaying details about a computer running the Cb Protection agent show whether the Cb Response sensor is installed, and if so, the version and status of the sensor. Note that this information currently shows only for Windows agents. OSX and Linux agents will report that Cb Response information is "Unknown".
- **Cb Response File Statistics in Cb Protection Console** – Pages displaying details about a file found on a computer running the Cb Protection agent show Cb Response statistics about the file, such as how many watchlists it is on, the number of computers and processes in which the file has been seen, and the number of network connections.
- **Links to the Cb Response server in Cb Protection Console** – Menu and inline links from the Cb Protection console **Events** table, **Computer Details**, and **File Details** pages connect to the Cb Response data for the object being viewed.
- **Cb Protection Agent Status in Cb Response Console** – In the Cb Response console, the **Host Information** page for each computer running a sensor reports whether a Cb Protection agent is installed.
- **Process Data Correlation** – A globally unique process identifier called a *Process Key* makes it possible to know when events on a Cb Response server and a Cb Protection server are referencing the same process. It uniquely identifies an individual instance of this running process. This identifier is available in Syslog output as well as data exported for third-party analysis tools such as Splunk.

See "Server Integration Features in the Cb Protection Console" on page 21 for more details on integration features.

## Activating Cb Response-to-Cb Protection Integration

The configuration settings for a Cb Response-to-Cb Protection integration appear in both the Cb Response and Cb Protection consoles.

You perform the integration confguration on the **Licensing** tab of the **System Configuration** page within the Cb Protection console. After the integration is set up, you can then view the configuration in the Cb Response console.

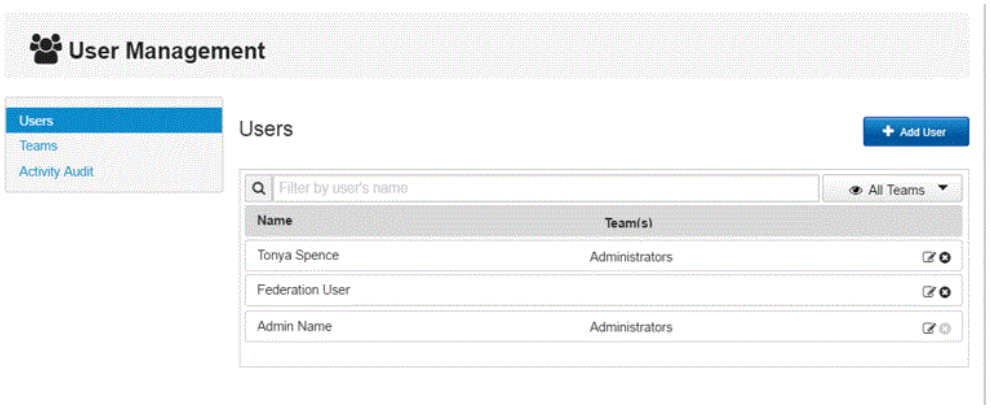
### Creating a Cb Response User for Integration

If you do not already have a user account to use for Cb Protection integration, follow this procedure.

| | |
|---|---|
| ***Note*** | *The user account for the integration must be an Administrator with Global Administrator privileges.* |

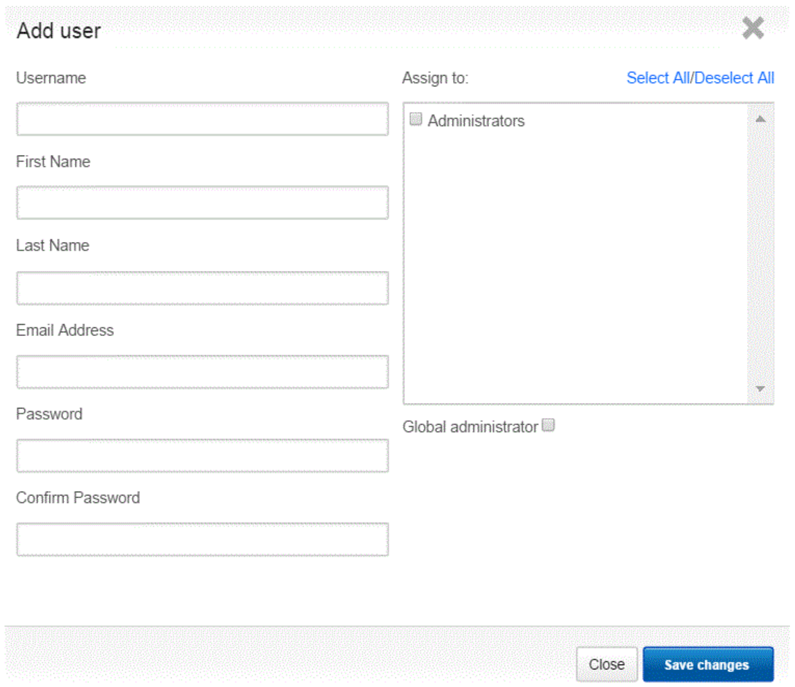
**To create a Cb Protection Integration user and API Token:**

1. Log into Cb Response.
2. In the left navigation menu, select **Users** to display the **User Management** page:



| Name | Team(s) |
|---|---|
| Tonya Spence | Administrators |
| Federation User | |
| Admin Name | Administrators |

3. Click **Add User** in the top-right corner to display the **Add User** page:



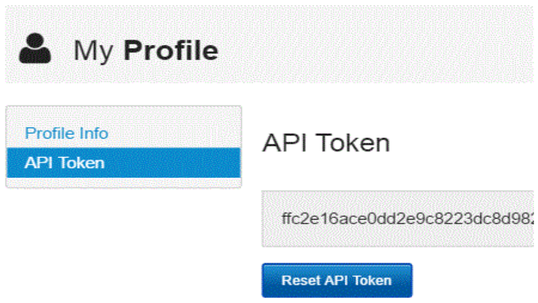
4. On the **Add user** dialog, define the user you will use for the integration:
   - a. In the **Username** field, enter a meaningful username such as **Cb Protection Integration** to depict the purpose of this account.
   - b. In the **First Name** and **Last Name** fields, enter meaningful values. These values will appear in the Cb Response console for this user. For example, you might enter **Cb Protection** as the first name and **Integration** as the last name.
   - c. Provide an email account for the user. An email account is required for Cb Response console users. However, this email account can be fictitious, since it will not be used as it would be for a normal account.
   - d. Enter and confirm a password for the account and keep it available for future use.
   - e. In the **Assign to** panel, select the **Administrators** check box to assign this user to the Administrators team.
   - f. Select the **Global administrator** check box below the **Assign to** panel.
   - g. When you have entered all the required information, click **Save changes**.

## Configuring and Activating the Integration

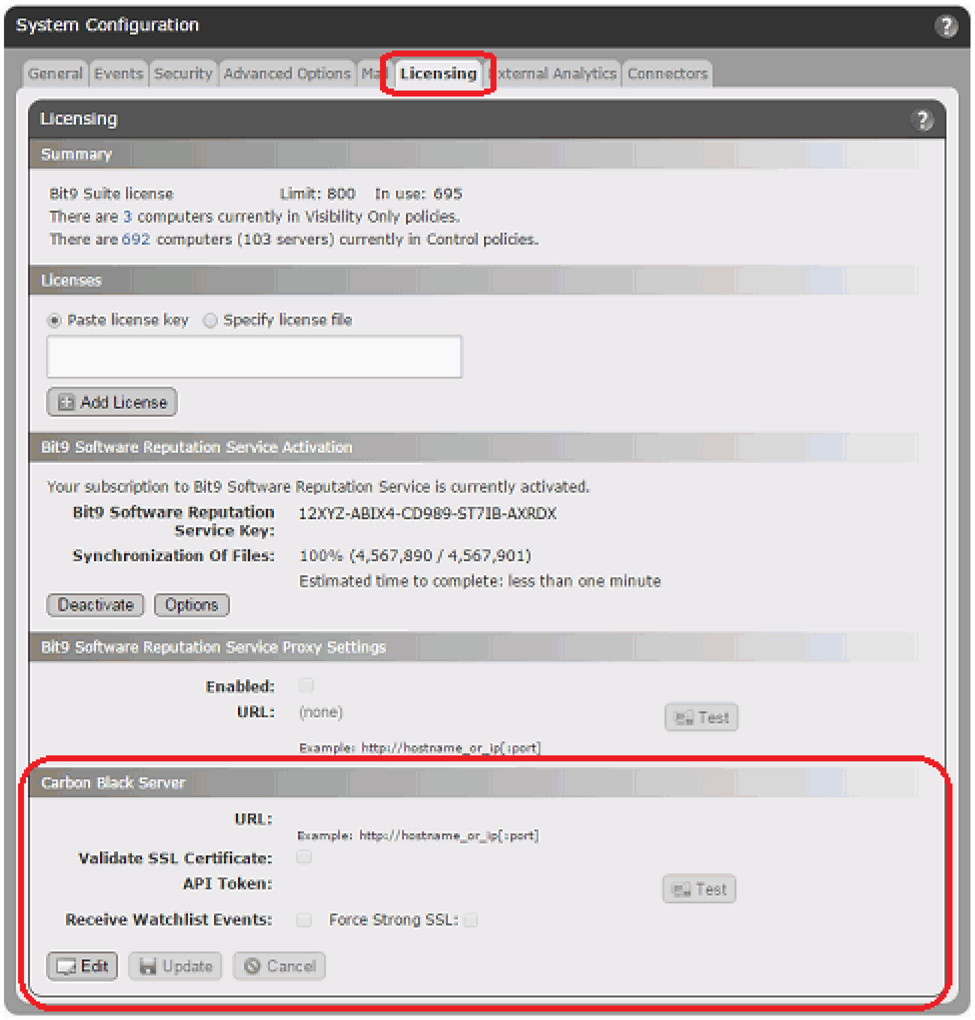
Review the information in Table 1 to prepare for the Cb Response-to-Cb Protection integration. The following procedure involves using both consoles. You can copy information from the Cb Response console and paste it into the Cb Protection console.

**To activate Cb Response-Cb Protection Server integration:**

1. Log into Cb Response
2. In the top-right corner of the console, select your ***username*** > **My Profile**.
3. On the **My Profile** page, choose **API Token** on the left menu.



4. In the **API Token** field, copy the presented API token.
5. Open another browser and log into the Cb Protection console using an account with Administrator privileges.
6. In the Cb Protection console menu:
   - If you are running v7.2.3, choose **Administration** > **System Configuration**.
   - If you are running v8.0.0, click the Administration (gear) icon and choose **System Configuration**.
7. On the **System Configuration** page, click the **Licensing** tab.
8. The Cb Response server panel is at the bottom of the page when the **Licensing** tab is displayed.



Licensing

Summary

Bit9 Suite license Limit: 800 In use: 695
There are 3 computers currently in Visibility Only policies.
There are 692 computers (103 servers) currently in Control policies.

Licenses

Paste license key [✓] Specify license file

Add License

Bit9 Software Reputation Service Activation

Your subscription to Bit9 Software Reputation Service is currently activated.

**Bit9 Software Reputation Service Key:** 12XYZ-ABIX4-CD989-ST7IB-AXRDX
**Synchronization Of Files:** 100% (4,567,890 / 4,567,901)
Estimated time to complete: less than one minute

Deactivate Options

Bit9 Software Reputation Service Proxy Settings

**Enabled:**
**URL:** (none)
Example: http://hostname_or_ip[:port]

Test

Carbon Black Server

**URL:**
Example: http://hostname_or_ip[:port]
**Validate SSL Certificate:**
**API Token:**
**Receive Watchlist Events:** Force Strong SSL:

Test

Edit Update Cancel

9. Enter the Cb Response configuration settings as shown in Table 1.

**Table 1:** Settings on the Cb Protection Server for Cb Response Integration

| Field/Button | Description |
|---|---|
| **URL** | The URL of the Cb Response server you want to link to the Cb Protection server. Port is only necessary if you do not use standard ports on the Cb Response server (80 for HTTP and 443 for HTTPS).<br>You can copy the base URL (without any page-specific additions) from the Cb Response browser and paste it into the relevant section of the Cb Protection configuration page.<br>For example, https://cbresponse.mycompany.local. |
| **Validate SSL Certificate** | Selecting this check box causes a validity check on the Cb Response server certificate. This should be selected only if the Cb Response server certificate is issued by a trusted certificate authority. Without manual configuration, Cb Response uses a self-signed certificate and so this generally should not be checked. |
| **API Token** | Enter the Cb Response server API Token here by copying it from the Cb Response console. Click the **Test** button to confirm that the server is accessible and the key works. The test returns one of the following values:<br>• **Success, version:** <Cb Response product version><br>• **Invalid API Token**<br>• **Server not accessible** |
| **Receive Watchlist Events** | Select this box to activate delivery of Cb Response watchlist events from the configured server to the Cb Protection server. |
| **Force Strong SSL** | Select this box to cause the Cb Response server to check the Cb Protection server certificate before sending events. This should not be selected, if your Cb Protection server uses a self-signed Cb Protection certificate on IIS. |

10. Click the **Test** button to determine whether the servers are able to communicate. Possible causes of failure and their troubleshooting steps are:

    - **Invalid API Token** – Make sure that the API token for the Cb Protection user has been copied correctly from the Cb Response console and pasted into the configuration page on the Cb Protection console. Also make sure that this user is an administrator and has global administrator privileges.
    - **Server not accessible** – Confirm that the correct URL and port number (if needed) has been entered in the configuration page on the Cb Protection console, and that the **Validate SSL certificate** check box was not selected when you are using a self-signed certificate. Also, make sure that access to the Cb Response server is not blocked by the network firewall.
    - **Force Strong SSL** – Selecting this check box causes the Cb Response server to check the Cb Protection server certificate before sending watchlist events. This should be checked only if the Cb Protection console certificate is issued by a trusted authority (for example, not self-signed).

If you are unable to create a successful connection, contact Carbon Black Technical Support.

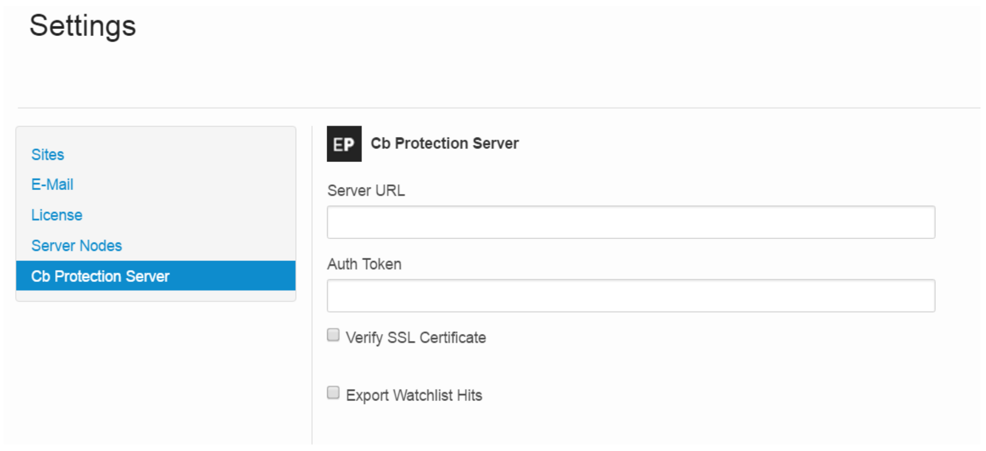
11. When you have entered and successfully tested the Cb Protection server settings in the Cb Protection console, click **Update** on the **System Configuration/Licensing** page. The configuration should be complete and your servers should be integrated.

## Viewing Integration Settings in Cb Response

In the Cb Response console, you can view the current Cb Protection integration settings using the **Cb Protection Server** option.

**To view Cb Protection integration settings in the Cb Response console:**

1. Log into Cb Response
2. In the top-right corner of the console, select your ***username*** > **Settings**.
3. In the left panel of the **Settings** dialog, select **Cb Protection Server** to view the current integration settings.



| | |
|---|---|
| **Note** | *You can change the watchlist and SSL settings in the Cb Response console. However, you cannot change the URL or API Token parameters here. If you need to modify these, use the Cb Protection console.*<br>*The value in the **Server URL** field on this page must be resolvable by the Cb Response server for proper communication to occur with Cb Protection.* |

**To view the Cb Protection integration status in the Cb Response console:**

1. Log into Cb Response
2. In the left navigation menu, select **Server Dashboard**.

   The status of the Cb Protection server connection is shown in the **Server Communication Status** panel in the top-right corner.

3. Three possible statuses can occur. You may need to perform more steps depending on the status:
   a. **Cb Protection Server is connected** – This status indicates that the integration has been configured and the connection is currently functioning properly.
   b. **Cb Protection Server not configured** – If the Cb Protection server connection has not been configured, a **Settings** button appears in the status line for the connection. Do not use this button. Keep in mind that you cannot configure the API Token or URL on this page; they must be entered in the Cb Protection console.
   c. **Unable to connect to Cb Protection Server** – This status may indicate network or firewall problems, or bad URL or port configuration. It can also occur if Force Strong SSL was chosen on the Cb Protection console's **System Configuration** page for Cb Response when a self-signed certificate is being used on the Cb Protection console.

### Regenerating the Authorization ID for Server Communication

The Cb Protection server creates a hidden token that the Cb Response server uses to send back watchlist hits. This token is not visible in the console of either product, but can be retrieved from the database. You can also enter it manually on the Cb Response side for diagnostic purposes. If you believe this token was compromised, or if your configuration stops working (for example, because the Cb Response server lost the token due to a reinstall or a manual token change), you can regenerate a new key.

**To regenerate the authorization key for server communications:**

1. In the Cb Protection console, choose **Administration > System Configuration** and click the **Licensing** tab.
2. In the **Carbon Black Server** panel, uncheck the **Receive Watchlist Events** box and click the **Update** button.
3. Check the **Receive Watchlist Events** box and click the **Update** button. A new key is generated.
4. Verify on both servers that there is a successful connection between the Cb Protection server and the Cb Response server.

## Server Integration Features in the Cb Protection Console

### Sensor Information

If you integrate the Cb Protection server with the Cb Response server, Cb Protection console pages that show computer information include Cb Response sensor details when available.

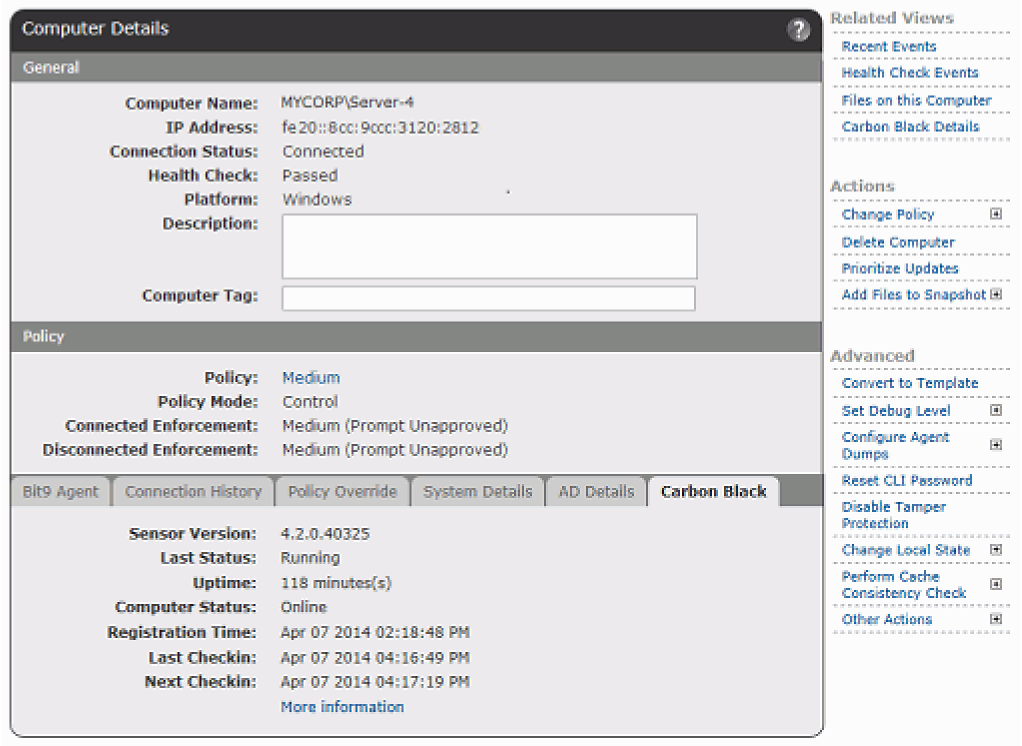
**To view a table of Cb Protection-managed computers also running a Cb Response sensor:**

1. In the Cb Protection console menu, choose **Assets** > **Computers**. The **Computers** page appears.
2. On the **Saved Views** menu, choose **Carbon Black Deployments** to see computers grouped by whether they have had a Cb Response sensor installed on them. This table also shows the Cb Response sensor version and its current status.

You also can click the **View Details** button for any computer in this view to see more Cb Response sensor details on the **Computer Details** page for any computer. The **Carbon Black** panel on this page reports the presence, version and status, and other details of any Cb Response sensor found on a computer running the Cb Protection agent. If a Cb Response server is not configured or the computer is not running a Cb Response sensor, this tab shows only a status of *Not installed*. By default, the Cb Protection server checks Cb Response status every 30 minutes.

The **Carbon Black** panel of the **Computer Details** page also provides a **More information** link to the **Sensors** page of the Cb Response console. You also can use the **Carbon Black Details** link in the **Related Views** menu to go to the Cb Response console.

Bit9 Platform Console: Computer Details Page with Carbon Black Tab



| | |
|---|---|
| Computer Name: | MYCORP\Server-4 |
| IP Address: | fe20::8cc:9ccc:3120:2812 |
| Connection Status: | Connected |
| Health Check: | Passed |
| Platform: | Windows |
| Description: | |
| Computer Tag: | |
| Policy: | Medium |
| Policy Mode: | Control |
| Connected Enforcement: | Medium (Prompt Unapproved) |
| Disconnected Enforcement: | Medium (Prompt Unapproved) |
| Sensor Version: | 4.2.0.40325 |
| Last Status: | Running |
| Uptime: | 118 minutes(s) |
| Computer Status: | Online |
| Registration Time: | Apr 07 2014 02:18:48 PM |
| Last Checkin: | Apr 07 2014 04:16:49 PM |
| Next Checkin: | Apr 07 2014 04:17:19 PM |

Table 2 describes the information available on the **Carbon Black** tab for **Bit9 Computer Details** pages. Note that this information currently shows only for Windows agents. OSX and Linux agents will report that Cb Response information is "Unknown".

**Table 2:** Fields on the Carbon Black tab of the Bit9 Computer Details page

| Field | Description |
|---|---|
| **Sensor Version**<br>**(Carbon Black version in table)** | The version of the Cb Response sensor installed on this computer. |
| **Carbon Black Status (in table)**<br>**Last Status**<br>**(on Details page)** | This field shows the last Cb Response sensor status for this computer, as reported by the Cb Protection agent to the Cb Protection server. The Cb Protection server checks Cb Response status every 30 minutes, and so status changes may be out of sync for up to that amount of time.<br>The possible values for Cb Response status in the table are:<br>• **Unknown**<br>• **Installed, initializing** – sensor is installed but not fully initialized<br>• **Installed, running**<br>• **Installed, not running**<br>• **Not installed**<br>• **Stopped**<br>On the **Details** page, the **Last Status** field on the **Carbon Black** tab is similar to Cb Response status in the table. However, it does not appear if sensor status is **Unknown**. Its possible values are:<br>• **Running**<br>• **Service not running**<br>• **Kernel not running**<br>• **Stopped**<br>**Notes:** In addition to up to a 30-minute gap between sensor installation and Cb Protection polling of Cb Response status, status will continue to report as **Not installed** until the Cb Response sensor connects to the Cb Response server and receives a sensor id. Also, if the Cb Protection agent is offline or uninstalled from a computer, the last Cb Response sensor status reported by the agent is displayed in the Cb Protection console, even if sensor status changes. |
| **Uptime** | Number of minutes and hours that the Cb Response sensor has been running since it was last started. |
| **Computer Status** | The status of this computer as reported by the Cb Response server. |
| **Registration Time** | The date and time the Cb Response sensor on this computer registered with its server. |
| **Last Checkin** | The date and time the Cb Response sensor on this computer last checked in with its server. |
| **Next Checkin** | The date and time of the next scheduled server checkin for the Cb Response sensor on this computer. |
| **More Information** | Connects to the login page of the Cb Response server configured on the **System Configuration** page **Licensing** tab. Logging in takes you to the **Sensors** page in Cb Response, so you can view additional details about this computer.<br>**Note: Y**ou must have valid login credentials for the Cb Response server to successfully open the Cb Response console. |

## File and Process Information

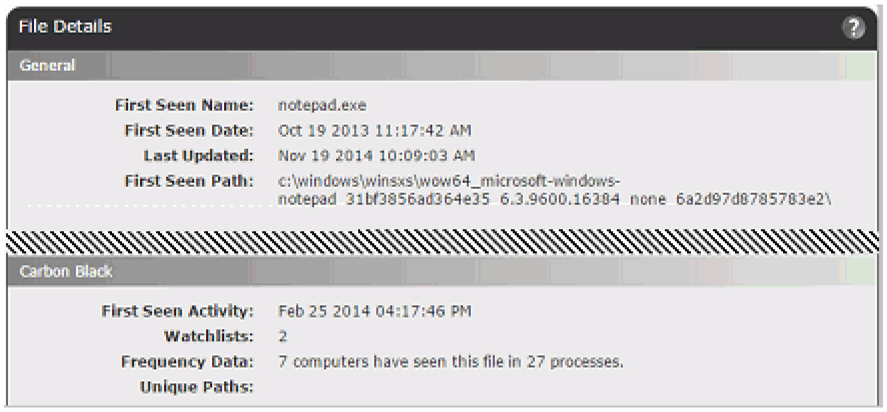
In the Cb Protection console, you can Cb Response statistics on files, such as:

- How many watchlists it is on
- The number of computers and processes in which the file has been found

| | |
|---|---|
| ***Note*** | *For 5.1.1 and previous sensors, Cb Response process information is available to the Cb Protection from Windows sensors only. It is not available in Cb Protection from the Cb Response OS X and Linux sensors.* |

**To view Cb Response details for a file found on a Cb Protection-managed computer:**

1. In the Cb Protection console, choose **Assets > Files** to display the **Files** page.
2. Click the **File Catalog** tab to view a table of unique files discovered on computers managed by this Cb Protection server. You can also choose **Files on Computers** to view a table of all file instances.
3. When you locate the file for which you want to view more details, click the **View Details** icon (file and pencil) on the left of that file row. The **File Details** page appears.


File Details

General

First Seen Name: notepad.exe
First Seen Date: Oct 19 2013 11:17:42 AM
Last Updated: Nov 19 2014 10:09:03 AM
First Seen Path: c:\windows\winsxs\wow64_microsoft-windows-notepad_31bf3856ad364e35_6.3.9600.16384_none_6a2d97d8785783e2\

Carbon Black

First Seen Activity: Feb 25 2014 04:17:46 PM
Watchlists: 2
Frequency Data: 7 computers have seen this file in 27 processes.
Unique Paths:

**Cb Protection File Details Page: Carbon Black Panel**

**Table 3:** Carbon Black fields on Cb Protection File/File Instance Details pages

| Field | Description |
|---|---|
| **First Seen Activity** | The date and time when activity involving this file was first reported to the Cb Response server. |
| **Watchlists** | The number of Cb Response watchlists on which this file appears. This appears if watchlist export is configured on the Cb Protection **System Configuration** page for Cb Response. |
| **Frequency Data** | The frequency of the file is the number of endpoints that have this file associated with a process. The number of processes is the count of all processes that have been associated with this file. |

**Table 3:** Carbon Black fields on Cb Protection File/File Instance Details pages

| Field | Description |
|---|---|
| **Unique Paths** | The number of unique paths in which this file has been seen. |
| **VirusTotal Score** | If available, the VirusTotal score for this file and the date and time of the analysis. |
| **Network Connections** | Whether there have been any network connections associated with this file, and if so, on how many computers. |
| **Registry Modifications** | Whether there have been any registry modifications associated with this file, and if so, on how many computers. |
| **File Icon** | The icon for this file (if any). |
| **More information** | Link to the Cb Response console showing more information about this file (from the **File Details** page) or a particular instance of this file on a particular computer (from the **File Instance Details** page). See "Links to the Cb Response Console" on page 26. |

## Event Information

The Cb Protection console **Events** page can display two different Cb Response-related event subtypes:

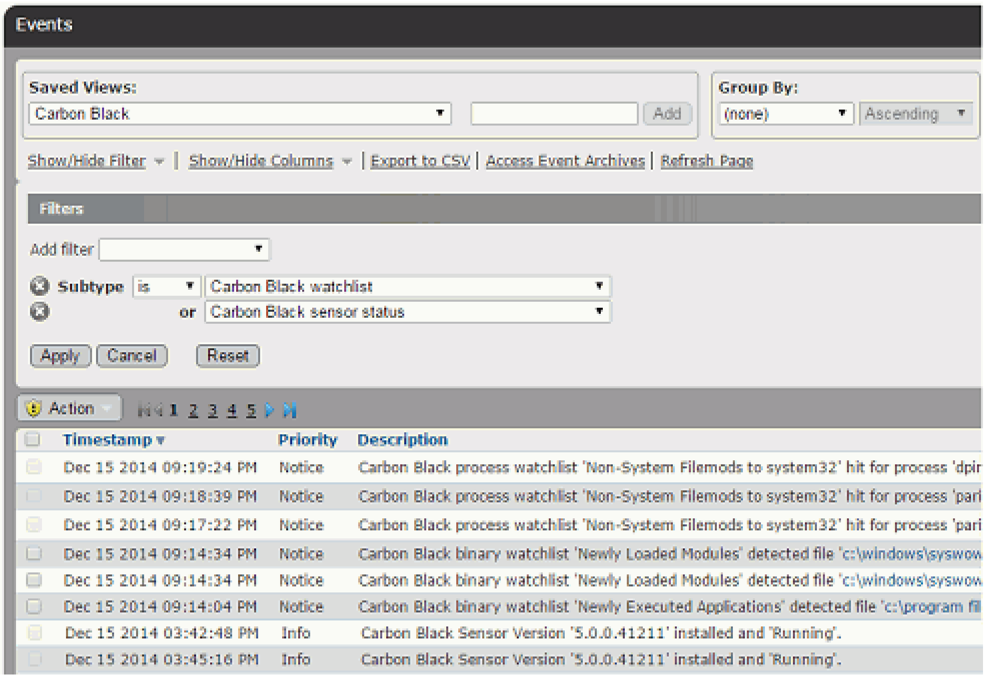
- **Carbon Black sensor status**
- **Carbon Black watchlist**

Cb Response events may be seen in unfiltered views of the events table, but there is also a **Saved View** for Cb Response events.

**To view Cb Response-related events in the Cb Protection Console:**

1. In the Cb Protection console menu, choose **Reports > Events** to display the **Events** page.
2. On the **Saved Views** menu, choose **Carbon Black**.

The following shows this view with filters displayed:


Events
Saved Views:
Carbon Black
Add
Group By:
(none)
Ascending
Show/Hide Filter | Show/Hide Columns | Export to CSV | Access Event Archives | Refresh Page
Filters
Add filter
Subtype is Carbon Black watchlist
or Carbon Black sensor status
Apply Cancel Reset
Action 1 2 3 4 5

| Timestamp | Priority | Description |
|---|---|---|
| Dec 15 2014 09:19:24 PM | Notice | Carbon Black process watchlist 'Non-System Filemods to system32' hit for process 'dpir |
| Dec 15 2014 09:18:39 PM | Notice | Carbon Black process watchlist 'Non-System Filemods to system32' hit for process 'pari |
| Dec 15 2014 09:17:22 PM | Notice | Carbon Black process watchlist 'Non-System Filemods to system32' hit for process 'pari |
| Dec 15 2014 09:14:34 PM | Notice | Carbon Black binary watchlist 'Newly Loaded Modules' detected file 'c:\windows\syswow |
| Dec 15 2014 09:14:34 PM | Notice | Carbon Black binary watchlist 'Newly Loaded Modules' detected file 'c:\windows\syswow |
| Dec 15 2014 09:14:04 PM | Notice | Carbon Black binary watchlist 'Newly Executed Applications' detected file 'c:\program fi |
| Dec 15 2014 03:42:48 PM | Info | Carbon Black Sensor Version '5.0.0.41211' installed and 'Running'. |
| Dec 15 2014 03:45:16 PM | Info | Carbon Black Sensor Version '5.0.0.41211' installed and 'Running'. |

Both process and binary watchlist event are exported to Cb Protection from Cb Response (when export is activated).

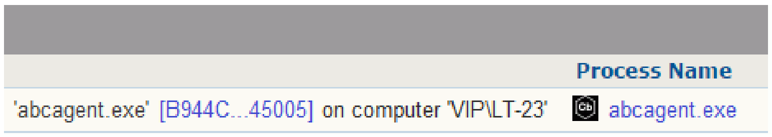
For process watchlist events, you can add a column to display the unique Process Key ID that correlates process information between Cb Protection and Cb Response. See "Correlation of Exported Data" on page 27 for more information.

When Cb Response watchlist hits appear in the Cb Protection **Events** table, the watchlist name appears in the **Rule Name** and **Description** fields of the table.

## Links to the Cb Response Console

Where Cb Response information is displayed in the Cb Protection console, there is often a link back to the relevant location in the Cb Response console. The link is identified with the Cb Response logo and uses the server URL provided in the Cb Response server section of the Cb Protection **System Configuration** page.

The following example from the **Events** page in the Cb Protection console shows how such a link appears.



| | Process Name |
|---|---|
| 'abcagent.exe' [B944C...45005] on computer 'VIP\LT-23' | abcagent.exe |

In other cases, there may be a menu link on a page to "**Carbon Black Details**".

When a user clicks one of these links, the Cb Response login page appears, where the user must provide Cb Response login account credentials.

If this browser remains open, the login credentials stay active for 90 minutes, and subsequent use of links during this period will go directly to the relevant page.

## Correlation of Exported Data

"File and Process Information" on page 24 describes how file and process data correlation is used inside the Cb Protection console. The Cb Response and the Cb Protection servers both make event data available for external use. Cb Response and the Cb Protection users might consider correlating or analyzing data from both sources.

To facilitate this, events that include process data have a "Process Key", which is a unique identifier for each process. The process key is available as follows:

- Syslog output from the Cb Response server
- Syslog output from the Cb Protection server
- Cb Response API queries
- Cb Protection Live Inventory SDK/Public API queries
- Data exported specifically for Splunk from the Cb Protection server
- External event exports and event archives from the Cb Protection server
- Cb Response email alerts

See the Cb Protection console online help or the *Using the Cb Protection Security Platform Guide* for information about theses Cb Protection features.

Chapter 2

# Integrating EMET with Cb Response

This chapter describes the procedure for integrating a Cb Response server with the Microsoft Enhanced Mitigation Experience Toolkit (EMET).

**Sections**

## Overview

One of the endpoint protection applications you might have on some or all of your endpoints is the Microsoft Enhanced Mitigation Experience Toolkit (EMET). EMET is designed to detect and protect against common attack techniques and actions.

Integrating EMET into the Cb Response environment gives you a single place to go investigate attacks detected and stopped by EMET while taking advantage of the additional visibility provided by Cb Response on your endpoints.

When EMET events become part of the Cb Response database, you can search for them, use them to trigger alerts, and perform process analysis to understand the relationships between an EMET event and other events on one or more endpoints in your organization, including the timeline of those relationships. In addition, EMET events can become part of the syslog output from the Cb Response server.

| | |
|---|---|
| ***Note*** | *This documentation uses the term "EMET event" to indicate the case where EMET detects an exploit attempt. It uses the term "EMET configuration" to indicate the protections enabled by EMET for that process.*<br>*EMET features and terminology are not detailed in this document. If you need more information about EMET, see the documentation provided by Microsoft.*<br>*Proper functioning of this integration assumes EMET is installed and configured per Microsoft recommendations. EMET 5.x versions should be compatible.*<br>*Reporting of EMET events to Cb Response requires that the Windows Event Log be selected as one of the Reporting options in the EMET interface on the host. This is explained in this document.* |

By default, EMET-enabled sensors report EMET events and configuration to the Cb Response server. The integration does not require interaction with another server. For any reporting sensor, this information appears in several places in the Cb Response console:

- On the **Search Processes** page, you can search for processes for which an EMET mitigation occurred and/or processes whose sensor has EMET protection enabled. For more information on this page, see "Process Search and Analysis" in *Cb Response User Guide.*
- On the **Process Analysis** page, EMET events are displayed and labeled in the table of events, and the EMET settings *specific to the current process* on the reporting sensor are included on the page. For more information on this page, see "Process Search and Analysis" in *Cb Response User Guide.*
- On the **Sensor Details** page, the general EMET configuration (if any) is shown. For more information on this page, see "Managing Sensors" in *Cb Response User Guide.*

To further enhance the EMET integration, you can enable the EMET Protection Feed on the **Threat Intelligence Feeds** page (see "Threat Intelligence Feeds" in *Cb Response User Guide*). This does not actually enable/disable delivery of events from sensors, but it does enable you to:

- Create alerts based on EMET events and manage them on the **Triage Alerts** page.
- Specify delivery of an email alert when an EMET event occurs.
- Include EMET events in the syslog output from your Cb Response server.

## EMET Events in Cb Response

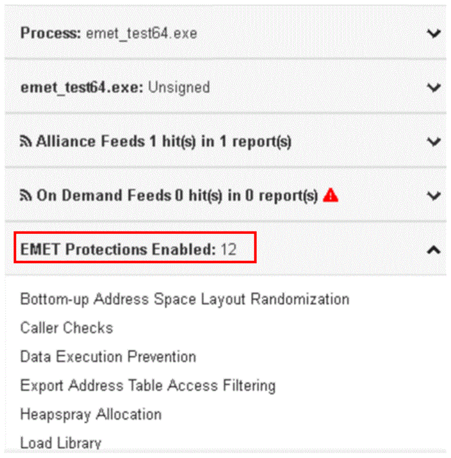
This section describes the places in the Cb Response console where you can view and analyze EMET events.

### Process Search and Analysis for EMET Events

If you open the **Process Analysis** page for a process on a host that has EMET protections enabled for the process, two types of EMET-related information may appear:

| **Note** | *In both cases, the sensor group for the host must have EMET reporting enabled.* |
|---|---|

- **Process-specific Protections on the Host** – The **Process Analysis** page includes a list of the **EMET Protections Enabled** on the host *for the process being analyzed*. These will appear even if EMET did not perform any mitigations when an exploit was attempted on the process.



**Process:** emet_test64.exe

**emet_test64.exe:** Unsigned

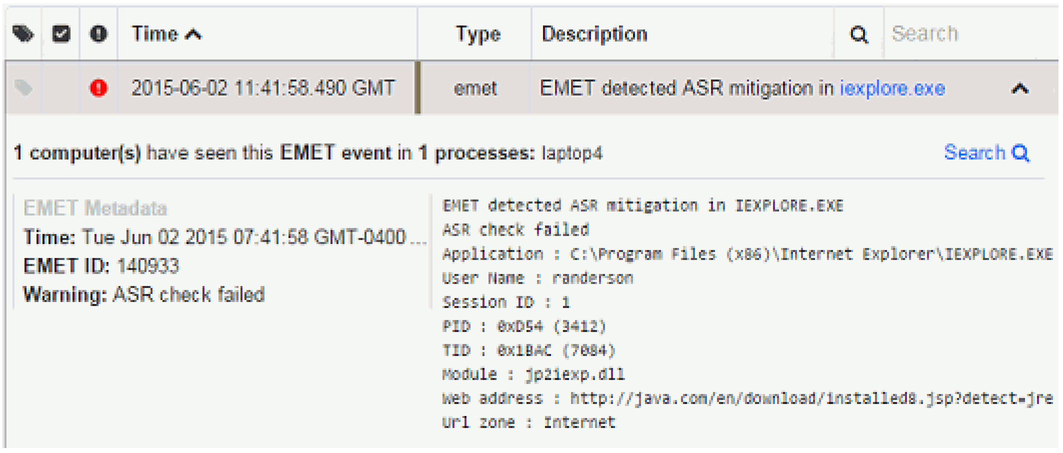
**Alliance Feeds 1 hit(s) in 1 report(s)**

**On Demand Feeds 0 hit(s) in 0 report(s)**

**EMET Protections Enabled:** 12

Bottom-up Address Space Layout Randomization
Caller Checks
Data Execution Prevention
Export Address Table Access Filtering
Heapspray Allocation
Load Library

- **EMET Mitigation Events** – If EMET took mitigation actions related to an exploit of the process, EMET events for these actions are listed in the event table and can be expanded for additional detail.



| | | | Time | Type | Description |
|---|---|---|---|---|---|
| | | | 2015-06-02 11:41:58.490 GMT | emet | EMET detected ASR mitigation in iexplore.exe |

**1 computer(s)** have seen this **EMET event** in **1 processes:** laptop4

EMET Metadata
**Time:** Tue Jun 02 2015 07:41:58 GMT-0400 ...
**EMET ID:** 140933
**Warning:** ASR check failed

```
EMET detected ASR mitigation in IEXPLORE.EXE
ASR check failed
Application : C:\Program Files (x86)\Internet Explorer\IEXPLORE.EXE
User Name : randerson
Session ID : 1
PID : 0xD54 (3412)
TID : 0x1BAC (7084)
Module : jp2iexp.dll
Web address : http://java.com/en/download/installed8.jsp?detect=jre
Url zone : Internet
```

### EMET Configuration Searches

In addition to searching for processes that have an EMET mitigation event associated with them, you can search for processes that are on a host that either has or does not have an *EMET configuration* related to that process. This allows you to determine whether a process might have been reported on a system that did not have endpoint protection enabled. The EMET configuration search option is available on the **Add Criteria** menu on the **Search Processes** page.

For more information on the **Search Processes** page, see "Process Search and Analysis" in the *Cb Response User Guide.*

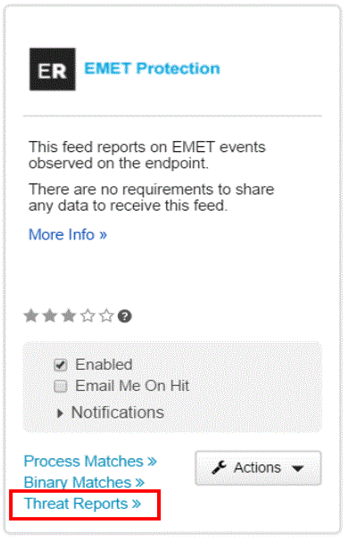
Keep in mind that this search will not only help you determine if EMET is installed on a host but will also help you determine process-specific protections.

## EMET Events and Threat Reports

EMET threat reports are delivered as part of the EMET Protection Feed. If this feed is enabled, these reports are searchable and processes with an EMET mitigation event are tagged with "EMET Protection Feed" and get a score of 90. See "Threat Intelligence Feed Scores in *Cb Response User Guide*.

**To view EMET threat reports:**

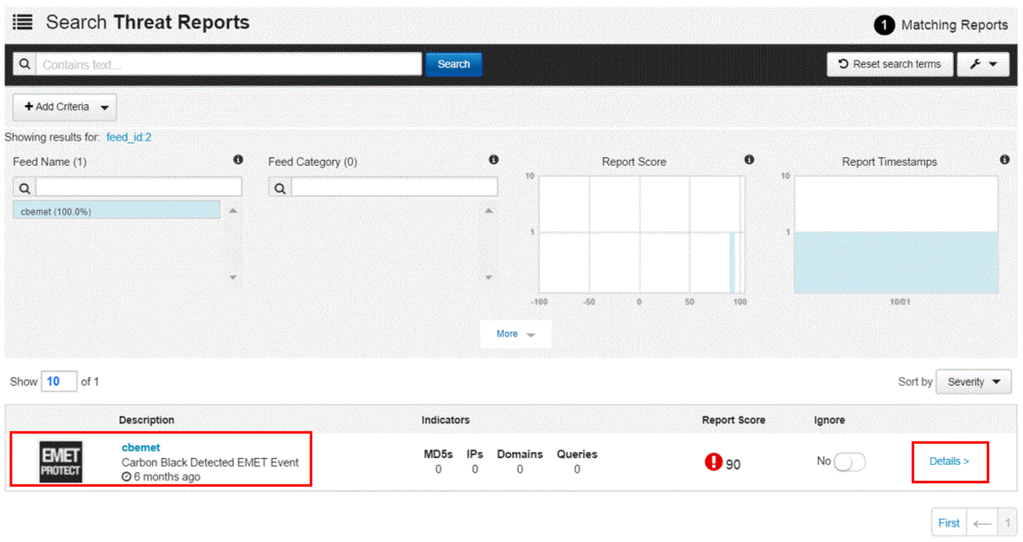
1. Log into Cb Response
2. In the left navigation menu, select **Threat Intelligence**.
3. On the **Threat Intelligence Feeds** page, scroll down to the **EMET Protection** panel and click the **Threat Reports** link.



This feed reports on EMET events observed on the endpoint.

There are no requirements to share any data to receive this feed.

4. The **Search Threat Reports** page appears with the results of the EMET reports search (labeled as **cbemet** in the **Description** column).



You can also click the **Details** link for in the far right column for any report in the table of reports to get additional information about the report.

See "Searching for Threat Reports" in *Cb Response User Guide* for more information on what you can do with Cb Response threat reports.

## Enabling and Disabling the EMET Protection Feed

The **Threat Intelligence Feeds** page (accessed via **Detect > Threat Intelligence** on the Cb Response console menu) includes an EMET Protection feed. This feed is disabled by default. You can enable it to include EMET event reports with the other reports received on your Cb Response server. When the EMET Protection is enabled, you can enable any of the following:

- Cb Response console alerts based on EMET events
- Delivery of an email alert when an EMET event occurs
- Inclusion of EMET events in the syslog output from your Cb Response server

For instructions on enabling a feed and configuring its alert and syslog features, see "Enabling, Disabling, and Configuring a Feed" in the *Cb Response User Guide*.

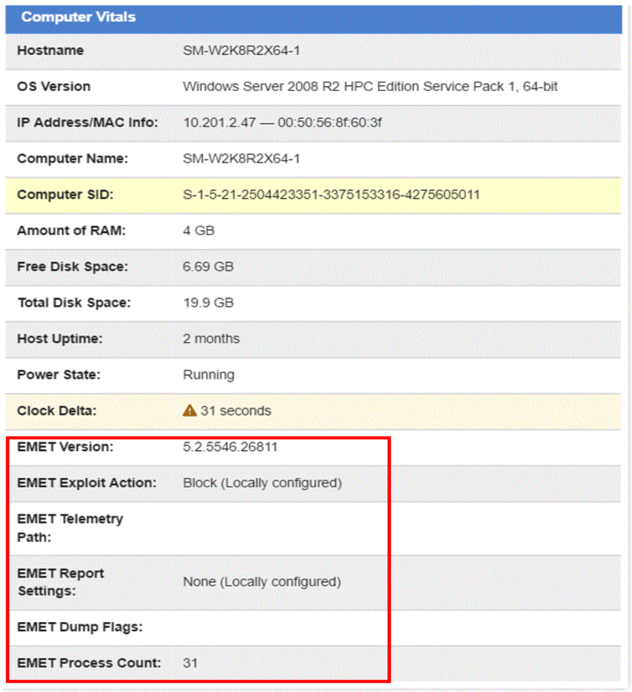
For more information on the **Threat Intelligence Feeds** page, see "Threat Intelligence Feeds" in the *Cb Response User Guide*.

| | |
|---|---|
| ***Note*** | *The Cb Response server receives EMET events regardless of whether the EMET Protection feed is enabled. EMET event collection can be disabled per-sensor group from the Group Settings page on the Cb Response server UI.* |

## EMET Status on an Endpoint

If EMET is installed on a host running a Cb Response sensor, information about that host's EMET configuration is provided to the Cb Response server. That information is displayed in the **Computer Vitals** panel on the **Sensor Details** page for each sensor.

For more information on the **Sensor Details** page, see "Managing Sensors" in the *Cb Response User Guide*.



| Computer Vitals | |
|---|---|
| Hostname | SM-W2K8R2X64-1 |
| OS Version | Windows Server 2008 R2 HPC Edition Service Pack 1, 64-bit |
| IP Address/MAC Info: | 10.201.2.47 — 00:50:56:8f:60:3f |
| Computer Name: | SM-W2K8R2X64-1 |
| Computer SID: | S-1-5-21-2504423351-3375153316-4275605011 |
| Amount of RAM: | 4 GB |
| Free Disk Space: | 6.69 GB |
| Total Disk Space: | 19.9 GB |
| Host Uptime: | 2 months |
| Power State: | Running |
| Clock Delta: | ⚠ 31 seconds |
| EMET Version: | 5.2.5546.26811 |
| EMET Exploit Action: | Block (Locally configured) |
| EMET Telemetry Path: | |
| EMET Report Settings: | None (Locally configured) |
| EMET Dump Flags: | |
| EMET Process Count: | 31 |

Hosts without EMET installed do not include the fields for EMET status on their **Sensor Details** page. If one of the types of EMET data shown in the **Computer Vitals** panel is not available, that field will be blank.

Table 4 shows the EMET-related fields included on the **Computer Vitals** panel.

**Table 4:** EMET Information on the Sensor Details page

| Field | Description |
|---|---|
| **EMET Version** | If the Microsoft Enhanced Mitigation Experience Toolkit (EMET) is installed on the endpoint, the version number of that toolkit. |
| **EMET Exploit Action** | The EMET configuration for what to do when an exploit attempt occurs:<br>• Audit -- Do not kill the process, when applicable, but log the exploitation attempt.<br>• Block -- Terminate the program when an exploitation attempt is detected ("Stop" in the EMET interface) |

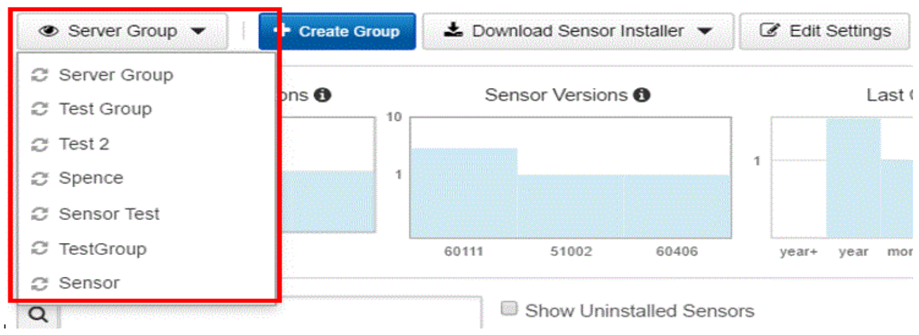
| Field | Description |
|---|---|
| **EMET Telemetry Path** | If Local Telemetry mode is enabled on EMET for this endpoint, this field shows the path where Early Warning information is sent on the local machine. See the Microsoft EMET documentation for the Registry path for this setting. |
| **EMET Report Settings** | Identifies which *Reporting* checkboxes are checked in the EMET interface, which controls where mitigation events are reported on the local host. Options are one or more of the following:<br>• Windows Event Log<br>• Tray Icon<br>• Early Warning<br>In addition, for any active option, there will also be an indication of whether the choice is Locally or GPO configured.<br>**Note:** If Windows Event Log is not active, Cb Response will not received EMET events. |
| **EMET Dump Flags** | If present, identifies the type of MiniDump file created if the LocalTelemetryPath key is set. |
| **EMET Process Count** | The number of active processes that have EMET mitigations configured on the host. |

## Disabling Sensor EMET Event Reporting

By default, sensors on any host with EMET installed will include EMET events in the logs reported back to their Cb Response server. However, you can disable EMET event reporting for all sensors in a Sensor Group.

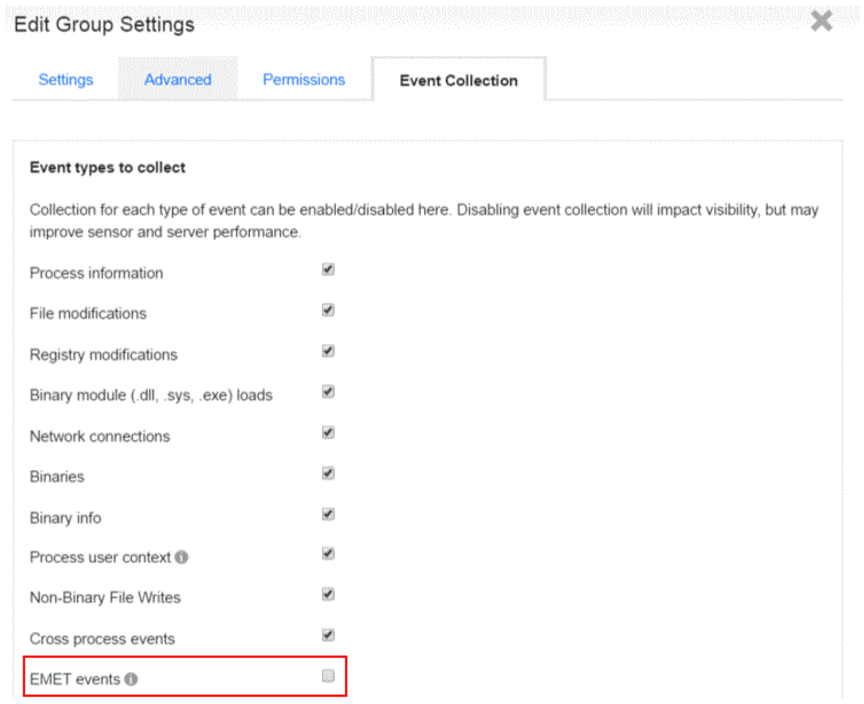
**To disable EMET event reporting from a Sensor group:**

1. Log into Cb Response.
2. In the left navigation menu, select **Sensors**.
3. On the **Sensors** page, choose the sensor group (top-left corner of the page) for which you want to disable EMET event reporting.



4. When you have selected the sensor group, click the **Edit Settings** button to display the **Edit Group Settings** page.

5. Click on the **Event Collection** tab and when it displays, *uncheck* the **EMET events** box.
6. Click **Save Changes**.
7. EMET event collection will no longer be included in the logs sent to the Cb Response server.



Event types to collect

Collection for each type of event can be enabled/disabled here. Disabling event collection will impact visibility, but may improve sensor and server performance.

| | |
|---|---|
| Process information | ✓ |
| File modifications | ✓ |
| Registry modifications | ✓ |
| Binary module (.dll, .sys, .exe) loads | ✓ |
| Network connections | ✓ |
| Binaries | ✓ |
| Binary info | ✓ |
| Process user context | ✓ |
| Non-Binary File Writes | ✓ |
| Cross process events | ✓ |
| EMET events | ☐ |

To re-enable EMET event collection for a sensor group, follow the same procedure but *check* the **EMET events** box before saving the changes.

Chapter 3

# Integrating with SSO Identity Providers

This chapter describes supported SAML 2.0 specifications and SAML 2.0 Single Sign-On (SSO) setup. It also explains how to integrate with the OKTA, Shibboleth, and ADFS IdPs.

**Sections**

## Overview

Single Sign-On (SSO) provides the ability for multiple systems (possibly more than one vendor) to share a user authentication provider so that:

- Users do not have to maintain separate sets of credentials for various services that they are using.
- Users do not have to re-authenticate when switching from one system to another since their initial login.
- Context is remembered after the initial login.

Cb Response server supports the Security Assertion Markup Language (SAML) 2.0 for SSO integration. The following sections provide a summary of supported capabilities and the procedures for configuring and troubleshooting SSO integration with an external SAML 2.0-compliant identity providers.

## Supported SAML 2.0 Specifications

SAML 2.0 is a relatively flexible and generic specification, which allows it be used in many different scenarios and use cases, but also comes with a certain level of complexity.

The SAML 2.0 specification is broken down into four documents:

- SAML 2.0 Core – Describes basic SAML assertions and protocols
- SAML 2.0 Bindings – Describes various types of HTTP calls supported by the protocol
- SAML 2.0 Profiles – Describes a set of profiles (use-cases), each one defining a set of calls made through one of the bindings to exchange SAML messages
- SAML 2.0 Metadata – Describes the format of the metadata XML files that must be exchanged between identity and service providers in order to establish mutual trust

Cb Response supports a subset of functionality described in these specifications:

- Supported SAML 2.0 Bindings:
  - HTTP Redirect Binding – Section 3.4
  - HTTP POST Binding – Section 3.5
- Supported SAML 2.0 Profiles:
  - Web Browser SSO Profile – Section 4.1

### Supported SSO Identity Providers

Cb Response currently supports these SSO Identity Providers (IdPs):

- OKTA IdP – See “OKTA IdP Integration” on page 43.
- Shibboleth IdP – See “Shibboleth IdP Integration” on page 45.
- ADFS IdP – See “ADFS IdP Integration” on page 47.

Any other IdP that implements the SAML 2.0 specification and supports the bindings and profiles listed above will very likely work, but Carbon Black has not validated and cannot support their configuration or operation.

## SAML 2.0 Single Sign-On Setup

Before establishing a trust relationship between a SAML service provider and an identity provider, the two services must have well-established, cryptographically secure identities. This identity information must then be exchanged, so that the service provider knows who its identity provider is and vice versa.

The exchange is performed by having each service generate a metadata XML file that is then provided to the other service. By default, the identity certificate/private key files used by Cb Response (acting as the SAML service provider) is `/etc/cb/cert/cb-server.[crt,key]`. This identity is also used for server-sensor authentication as well as for web user interface HTTPS server authentication.

You can also configure Cb Response to use a separate set of certificate/key files for SSO by updating configuration fields in the SSO configuration file created below:

To set up SAML 2.0 single sign-on, follow the steps in these sections according to the IdP with which you are integrating:

- See "OKTA IdP Integration" on page 43.
- See "Shibboleth IdP Integration" on page 45.
- See "ADFS IdP Integration" on page 47.

### Attribute Mapping

With Cb Response (version 5.2 and later), when SSO is configured, users authenthicated by the configured IdP can be automatically provisioned in Cb Response if they are authorized and the IdP resturns the user's first name, last name, and email address as user attributes.

The fields returned by the IdP are not part of the SAML 2.0 specification and will vary between IdPs. The `attr_map.py` script maps the attributes returned by your IdP to the fields required by Cb Response. Cb Response supports seven attributes:

**Table 5:** Supported Attributes

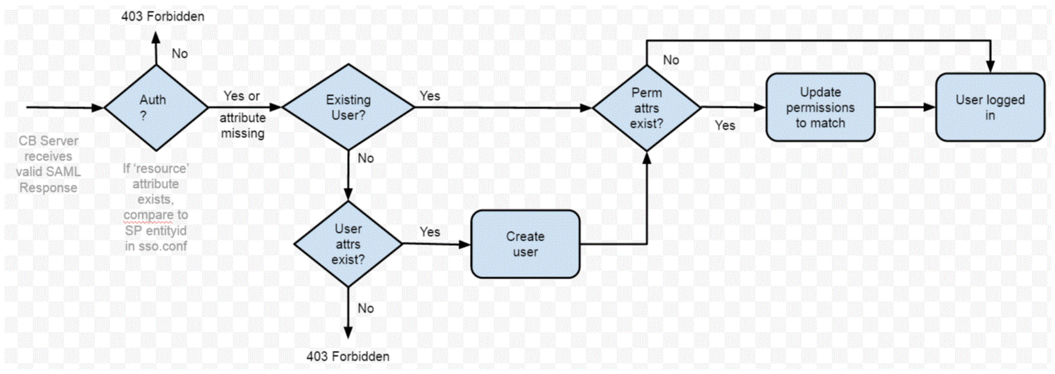
| Attribute | Required | Use |
|---|---|---|
| **uid** | Required | The username to log in as. |
| **authorized** | Optional (but strongly recommended) | Required to authorize access.<br>Boolean.<br>"True" if this user is authorized access to this Cb Response server. |
| **first_name** | Optional | Required to provision a new user.<br>The user's first name. |
| **last_name** | Optional | Required to provision a new user.<br>The user's last name. |
| **email** | Optional | Required to provision a new user.<br>The user's email address. |

**Table 5:** Supported Attributes

| Attribute | Required | Use |
|---|---|---|
| **builtin_rules** | Optional | Required to set permissions.<br>A list of roles.<br>Valid roles: "GlobalAdmin"<br>If None, this field is ignored.<br>If an empty list ([]), all roles are removed. |
| **teams** | Optional | Required to set permissions.<br>A list of teams this the user is a member of.<br>Names must match an existing configured team.<br>If None, this field is ignored.<br>If an empty list ([]), all team associations are removed. |

Your IdP administrator is responsible for providing you a list of attributes returned by your IdP. username, first_name, last_name and email should map directly to the fields returned by your IdP. authorized, builtin_roles and teams are for you to configure, based on your business policies.

This diagram succinctly describes the attribute mapping process:



## Example Attribute Mapping Script

This section provides an example attribute mapping script.

The attribute mapping is not contained in a config file, but in a user-defined Python script. This gives the most flexibility to match Cb Response -required values with the arbitrarily defined values returned by the client's IdP.

```
def callback(saml_response, db_session, logger, sso_config):
    """
        Takes a SAML Response object and returns a dictionary
        of fields.   This is a default implementation, it is
        expected to be overridden by a user in the SSO config.

        This instance will return empty values for all fields so
        behavior maintains backwards compatibility with
        existing SSO configurations.
    """
       logger.debug("Default SAML attribute map, user
authorized, not parsing attributes in SAML Response.")
       result = {}
       result["authorized"] = True
       result["username"] = None
       result["first_name"] = None
       result["last_name"] = None
       result["email"] = None
       result["builtin_roles"] = None
       result["teams"] = None

       return result
```

The default callback returns authorized = True and None for all attributes. This keeps behavior consistent with the current SSO implementation. An example script for a fully featured install is included in `/etc/cb/sso/` alongside the example config file:

```
def callback(saml_response, db_session, logger, sso_config):
  result = {}
  attrs = saml_response.attrs

  result["authorized"] = True if "cbserver" in attrs
  ["groups"] else False

  result["username"] = attrs["uid"][0] if "uid" in attrs
  else None
  result["first_name"] = attrs["givenName"][0] if
  "givenName" in attrs else None
  result["last_name"] = attrs["sn"][0] if "sn" in
  attrs else None
  result["email"] = attrs["mail"][0] if "mail" in
  attrs else None

    if "cbserver-owners" in attrs["groups"]:
        result["builtin_roles"] = ["global_admin",]
        result["teams"] = ["Administrators", ]
    else:
        result["builtin_roles"] = []
        result["teams"] = []

    return result
```

In the example above, the IdP returns the following fields:

- **username** – The user’s login ID.
- **givenName** – The user's first name (given name).
- **sn** – The user's last name (surname).
- **mail** – The users’s email address.
- **groups** – A list of relevant group memberships.

The example uses the `resource` parameter to determine group membership.

Two group names are defined by this IdP:

- **cbserver**
- **cbserver-owners**

A user must be a member of **cbserver** to have access to the Cb Enteprise Response server. Any user part of **cbserver-owners** is granted global admin and made part of the administrators team.

The following is example debug output of a user being authenticated, authorized, created, added to global admins and the administrators team. (For more information on enabling this, see “Troubleshooting” on page 56.)

```
15:08:06.799 api_routes_saml.py(214): <DEBUG> Attributes
returned in SAML response:
15:08:06.800 api_routes_saml.py(216): <DEBUG> mail:
['jguy@carbonblack.com']
15:08:06.801 api_routes_saml.py(216): <DEBUG> givenName:
['Bill']
15:08:06.801 api_routes_saml.py(216): <DEBUG> groups:
['cbserver-owners', 'cbserver']
15:08:06.801 api_routes_saml.py(216): <DEBUG> uid: ['bill']
15:08:06.801 api_routes_saml.py(216): <DEBUG> sn: ['Smith']
15:08:06.801 api_routes_saml.py(218): <DEBUG> Custom SAML
attribute map returned:
15:08:06.802 api_routes_saml.py(220): <DEBUG> username: bill
15:08:06.802 api_routes_saml.py(220): <DEBUG> first_name: Bill
15:08:06.802 api_routes_saml.py(220): <DEBUG> last_name: Smith
15:08:06.802 api_routes_saml.py(220): <DEBUG> builtin_roles:
['global_admin']
15:08:06.802 api_routes_saml.py(220): <DEBUG> teams:
['Administrators']
15:08:06.803 api_routes_saml.py(220): <DEBUG> authorized: True
15:08:06.803 api_routes_saml.py(220): <DEBUG> email:
jguy@carbonblack.com
15:08:06.806 api_routes_saml.py(242): <WARNING> bill
authenticated and authorized, but not found in user database.
Creating user.
15:08:06.812 api_routes_saml.py(261): <DEBUG> Updating bill to
Global Admin role.
15:08:06.814 api_routes_saml.py(269): <DEBUG> Updating team
membership for bill to [{'id': 1,
```

## OKTA IdP Integration

This section explains how to integrate the OKTA IdP with Cb Response.

**To integrate the OKTA IdP with Cb Response:**

1. Acquire metadata XML from the OKTA IdP and place it in the `/etc/cb/sso` directory on the Cb Response server host. (You are not required to use this directory, but it is a good default location.)
2. On the Cb Response server, navigate to `/etc/cb/sso` and:
   a. Copy `/etc/cb/sso/sso.conf.example.okta` to `/etc/cb/sso/sso.conf`.
   b. Copy `attr_map.py.example.okta` to `attr_map.py`.

| | |
|---|---|
| ***Note*** | *Make appropriate changes to the* `attr_map.py` *file based on the attributes returned from Okta. Each configurable property is accompanied with additional inline documentation in the* `attr_map.py` *file to assist with this process.* |

3. In the `/etc/cb/sso/sso.conf` file:

| | |
|---|---|
| ***Warning*** | *The syntax of the* `sso.conf` *configuration file must fully conform to the JSON data-interchange format. Failure to do so may create an invalid configuration file, which will prevent the cb-coreservices services from launching properly. When changes are made to this file and cb-enterprise is restarted, check /*`var/log/cb/coreservices/debug.log` *to ensure there are no errors.*<br>*If the administrator configures the Single Logout (SLO) service in both the IDP Service and Endpoint sections and a user logs out of the Cb Response server, the user is also logged out of the Okta application. (This will, in effect, log the user out of all Okta applications.)* |

   a. Specify the file path to the location of the metadata XML from the OKTA IdP. For example:

```
""metadata": {
      "local": [
        "<file path to location of IdP XML>"
      ]
    },
```

   b. Make sure the `attribute_mapper` field has the path to the Python Mapper file:
   `"attribute_mapper": "/etc/cb/sso/attr_map.py",`
   c. Change the `accepted_time_diff` field if needed:
   `"accepted_time_diff": 600,`

**d.** Update the `service` / `sp` / `idp` section with the appropriate appid from the OKTA IdP. For example:

```
"service": {
 "sp": {
     "nameid_format": "urn:oid:1.3.6.1.4.1.1466.115.121.1.15-
NameID",
     "idp": {
         "http://www.okta.com/<appid>": {
```

**e.** Update the `single_sign_on_service` and `single_logout_service` sections with the appropriate name and appid from the OKTA IdP For example:

```
# URLs in this section MUST be updated to match the URLs defined
by the
# IdP you are integrating with
"single_sign_on_service": {
 "urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect": "https:/
/fakeidp.okta.com/app/<name>/<appid>/sso/saml"
},

"single_logout_service": {
 "urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect": "https:/
/fakeidp.okta.com/app/<name>/<appid>"
}
```

**f.** In the `endpoints` section, update the `assertion_consumer_service` and `single_logout_service` fields with the appropriate IP address of FQDN of the Cb Response. For example:

```
"endpoints": {
   "assertion_consumer_service": {
      "https://<IP Address or FQDN of the CB Server>/api/saml/
assertion":"urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
   },
   "single_logout_service": {
      "https://<IP Address or FQDN of the CB Server>/api/saml/
logout": "urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
   }
},
```

**g.** Update the `entityid` field with the appropriate IP address or FQDN of the Cb Response server. For example:

```
"entityid": "https://<IP Address or FQDN of the CB Server>/",
```

**h.** Search the `sso.conf` file for “TODO” and ensure that all “TODO” tasks are also completed.

4. Open the `/etc/cb/cb.conf` file and edit the `SSOConfig` property so that it contains the full path to the SSO configuration file created in the previous steps. This single property is what defines whether or not Cb Response server will be started in standalone vs. federated authentication mode.

| | |
|---|---|
| ***Note*** | *If you need to deactivate SSO integration at any time, comment out the* `SSOConfig` *property.* |

5. Generate Cb Response server's SSO service provider metadata XML file by issuing this command:

```
/usr/share/cb/cbssl sso --make-metadata > /<your file path>
```

6. Once this file is created, you must give it to the identity provider to complete the trust.
7. Restart Cb Response server by issuing this command:

```
sudo service cb-enterprise restart
```

## Shibboleth IdP Integration

This section explains how to integrate the Shibboleth IdP with Cb Response.

**To integrate the Shibboleth IdP with Cb Response:**

1. Acquire metadata XML from the Shibboleth IdP and place it in the `/etc/cb/sso` directory on the Cb Response server host. (You are not required to use this directory, but it is a good default location.)
2. On the Cb Response server, navigate to `/etc/cb/sso` and:
   a. Copy `/etc/cb/sso/sso.conf.example.shib` to `/etc/cb/sso/sso.conf`.
   b. Copy `attr_map.py.example.shib` to `attr_map.py`.

| | |
|---|---|
| ***Note*** | *Make appropriate changes to the* `attr_map.py` *file based on the attributes returned from Shibboleth. Each configurable property is accompanied with additional inline documentation in the* `attr_map.py` *file to assist with this process.* |

**3.** In the `/etc/cb/sso/sso.conf` file:

| | |
|---|---|
| ***Warning*** | *The syntax of this configuration file must fully conform to the JSON data-interchange format. Failure to do so may create an invalid configuration file, which will prevent the cb-coreservices services from launching properly. When changes are made to this file and cb-enterprise is restarted, check* `/var/log/cb/coreservices/debug.log` *to ensure there are no errors.* |

**a.** Specify the file path to the location of the metadata XML from the Shibboleth IdP. For example:

```
""metadata": {
      "local": [
        "<file path to location of IdP XML>"
      ]
    },
```

**a.** Make sure the `attribute_mapper` field has the path to the Python Mapper file:

`"attribute_mapper": "/etc/cb/sso/attr_map.py",`

**b.** Change the `accepted_time_diff` field if needed:

`"accepted_time_diff": 600,`

**c.** Update the `service` / `sp` / `idp` section with the Shibboleth IdP. For example:

```
"service": {
 "sp": {
   "idp": {
     # EntityId of the IDP
     "https://fakeipd.example.com": {
```

**d.** Update the `single_sign_on_service` and `single_logout_service` sections with the appropriate name and appid from the Shibboleth IdP For example:

```
# URLs in this section MUST be updated to match the URLs defined
by the
# IdP you are integrating with
"single_sign_on_service": {
   "urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect":
"https://fakeipd.example.com/saml2/idp/SSOService"
},
"single_logout_service": {
   "urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect":
"https://fakeipd.example.com/saml2/idp/SingleLogoutService"
}
```

e. In the `endpoints` section, update the `assertion_consumer_service` and `single_logout_service` fields with the appropriate IP address of FQDN of the Cb Response. For example:

```
"endpoints": {
   "assertion_consumer_service": {
      "https://<IP Address or FQDN of the CB Server>/api/saml/
assertion":"urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
   },
   "single_logout_service": {
      "https://<IP Address or FQDN of the CB Server>/api/saml/
logout": "urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
   }
},
```

f. Update the `entityid` field with the appropriate IP address or FQDN of the Cb Response server. For example:

```
"entityid": "https://<IP Address or FQDN of the CB Server>/",
```

g. Search the `sso.conf` file for "TODO" and ensure that all "TODO" tasks are also completed.

4. Open the `/etc/cb/cb.conf` file and edit the `SSOConfig` property so that it contains the full path to the SSO configuration file created in the previous steps. This single property is what defines whether or not Cb Response server will be started in standalone vs. federated authentication mode.

| **Note** | *If you need to deactivate SSO integration at any time, comment out the* `SSOConfig` *property.* |
|---|---|

5. Generate Cb Response server's SSO service provider metadata XML file by issuing this command:

```
/usr/share/cb/cbssl sso --make-metadata > /<your file path>
```

6. Once this file is created, you must give it to the identity provider to complete the trust.

7. Restart Cb Response server by issuing this command:

```
sudo service cb-enterprise restart
```

## ADFS IdP Integration

This section explains how to integrate the ADFS IdP with Cb Response.

**To integrate the ADFS IdP with Cb Response:**

1. Acquire metadata XML from the ADFS IdP and place it in the `/etc/cb/sso` directory on the Cb Response server host. (You are not required to use this directory, but it is a good default location.)

2. On the Cb Response server, navigate to `/etc/cb/sso` and:
   a. Copy `/etc/cb/sso/sso.conf.example.adfs` to `/etc/cb/sso/sso.conf`.
   b. Copy `attr_map.py.example.adfs` to `attr_map.py`.

| **Note** | *Make appropriate changes to the `attr_map.py` file based on the attributes returned from ADFS. Each configurable property is accompanied with additional inline documentation in the `attr_map.py` file to assist with this process.* |
|---|---|

3. In the `/etc/cb/sso/sso.conf` file:

| **Warning** | *The syntax of this configuration file must fully conform to the JSON data-interchange format. Failure to do so may create an invalid configuration file, which will prevent the cb-coreservices services from launching properly. When changes are made to this file and cb-enterprise is restarted, check /`var/log/cb/coreservices/debug.log` to ensure there are no errors.* |
|---|---|

   a. Specify the file path to the location of the metadata XML from the ADFS IdP. For example:

```
""metadata": {
      "local": [
        "<file path to location of IdP XML>"
      ]
    },
```

   a. Make sure the `attribute_mapper` field has the path to the Python Mapper file:
   `"attribute_mapper": "/etc/cb/sso/attr_map.py",`
   b. Change the `accepted_time_diff` field if needed:
   `"accepted_time_diff": 600,`
   c. Update the `service` / `sp` / `idp` section with the appropriate appid from the ADFS IdP. For example:

```
"service": {
 "sp": {
   "idp": {
     # EntityId of the IDP
     "https://fakeipd.example.com": {
```

**d.** Update the `single_sign_on_service` and `single_logout_service` sections with the appropriate name and appid from the ADFS IdP For example:

```
# URLs in this section MUST be updated to match the URLs defined
by the
# IdP you are integrating with
"single_sign_on_service": {
   "urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect":
"https://fakeipd.adfs.com/adfs/ls/"
},

"single_logout_service": {
   "urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect":
"https://fakeipd.adfs.com/adfs/ls/?wa=wsignout1.0"
}
```

**e.** In the `endpoints` section, update the `assertion_consumer_service` and `single_logout_service` fields with the appropriate IP address of FQDN of the Cb Response. For example:

```
"endpoints": {
   "assertion_consumer_service": {
      "https://<IP Address or FQDN of the CB Server>/api/saml/
assertion":"urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
   },
   "single_logout_service": {
      "https://<IP Address or FQDN of the CB Server>/api/saml/
logout": "urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
   }
},
```

**f.** Update the `entityid` field with the appropriate IP address or FQDN of the Cb Response server. For example:

```
"entityid": "https://<IP Address or FQDN of the Cb Server>/",
```

**g.** Search the `sso.conf` file for “TODO” and ensure that all “TODO” tasks are also completed.

**4.** Open the `/etc/cb/cb.conf` file and edit the `SSOConfig` property so that it contains the full path to the SSO configuration file created in the previous steps. This single property is what defines whether or not Cb Response server will be started in standalone vs. federated authentication mode.

| | |
|---|---|
| **Note** | *If you need to deactivate SSO integration at any time, comment out the* `SSOConfig` *property.* |

**5.** Generate Cb Response server’s SSO service provider metadata XML file by issuing this command:

```
/usr/share/cb/cbssl sso --make-metadata > /<your file path>
```

6. Once this file is created, you must give it to the identity provider to complete the trust.
7. Restart Cb Response server by issuing this command:

```
sudo service cb-enterprise restart
```

## Configure ADFS 2.0 for SSO Integration

This procedure explains how to configure ADFS 2.0 for SSO integration with Cb Response:

**To configure ADFS 2.0 for SSO Integration:**

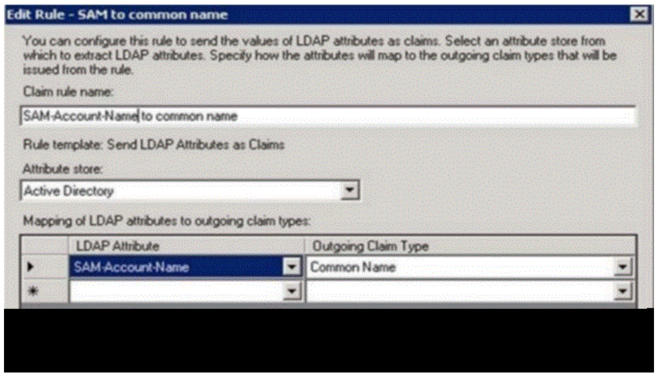
1. Access the ADFS Management Tool.
2. Create a new relying party trust:
   - Import data about the relying party from the `cb-metadata.xml` file that you generated in the previous section.
   - Ensure that the Display Name matches the "sp":"name" value from `/etc/cb/sso/sso.conf`.
3. Edit the claim rules to create rules for Cb Response:
   - Ensure that the value in **NameID** matches the Cb Response login name. (The Cb Response server depends on this.)
   - The Cb Response server uses the transient **NameID** policy, so be sure that claim rules comply with this policy.

**Example:**

**a.** Right-click on the relying party trust created above and select **Edit Claim Rules**.

**b.** Add a new rule to use **Active Directory SAM-Account-Name** as the **Common Name**.

**c.** Click **OK**.



Edit Rule - SAM to common name

You can configure this rule to send the values of LDAP attributes as claims. Select an attribute store from which to extract LDAP attributes. Specify how the attributes will map to the outgoing claim types that will be issued from the rule.

Claim rule name: SAM-Account-Name to common name

Rule template: Send LDAP Attributes as Claims

Attribute store: Active Directory

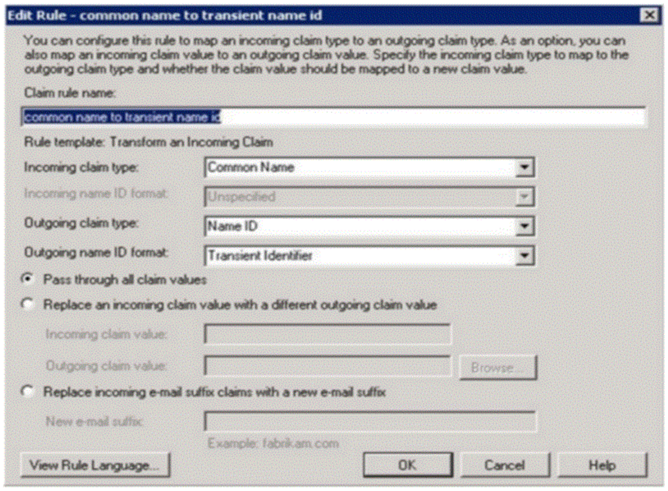
Mapping of LDAP attributes to outgoing claim types:

| | LDAP Attribute | Outgoing Claim Type |
|---|---|---|
| ▸ | SAM-Account-Name | Common Name |
| * | | |

**d.** Right-click on the relying party trust again and select **Edit Claim Rules**.

**e.** Add a new rule to translate **Common Name** to the **NameID** transient format.

**f.** Click **OK**.



Edit Rule - common name to transient name id

You can configure this rule to map an incoming claim type to an outgoing claim type. As an option, you can also map an incoming claim value to an outgoing claim value. Specify the incoming claim type to map to the outgoing claim type and whether the claim value should be mapped to a new claim value.

Claim rule name: common name to transient name id

Rule template: Transform an Incoming Claim

Incoming claim type: Common Name

Incoming name ID format: Unspecified

Outgoing claim type: Name ID

Outgoing name ID format: Transient Identifier

(●) Pass through all claim values

( ) Replace an incoming claim value with a different outgoing claim value

Incoming claim value:

Outgoing claim value:

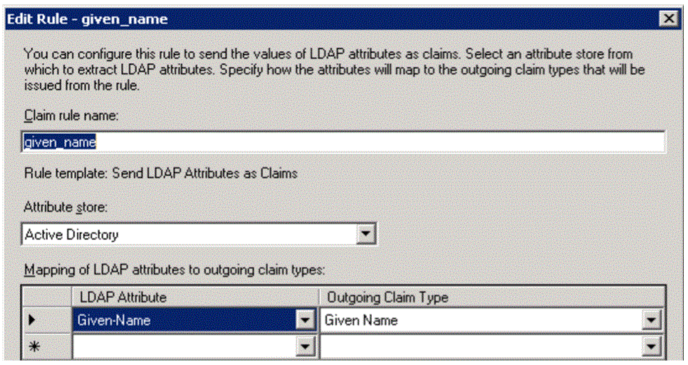
( ) Replace incoming e-mail suffix claims with a new e-mail suffix

New e-mail suffix:

Example: fabrikam.com

4. Above the **SAM Account Name** and **NameID** claim rules, add these additional claim rules:

   a. Add a **given_name** rule as follows:



Edit Rule - given_name

You can configure this rule to send the values of LDAP attributes as claims. Select an attribute store from which to extract LDAP attributes. Specify how the attributes will map to the outgoing claim types that will be issued from the rule.

Claim rule name: given_name

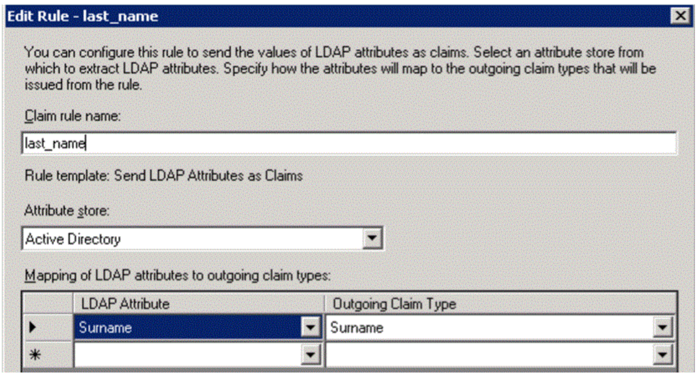
Rule template: Send LDAP Attributes as Claims

Attribute store: Active Directory

Mapping of LDAP attributes to outgoing claim types:

| | LDAP Attribute | Outgoing Claim Type |
|---|---|---|
| ▶ | Given-Name | Given Name |
| * | | |

   b. Add a **last_name** rule as follows:



Edit Rule - last_name

You can configure this rule to send the values of LDAP attributes as claims. Select an attribute store from which to extract LDAP attributes. Specify how the attributes will map to the outgoing claim types that will be issued from the rule.

Claim rule name: last_name

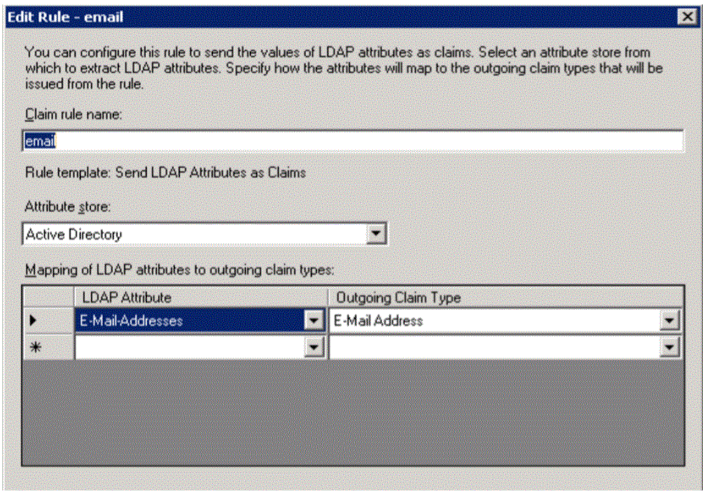
Rule template: Send LDAP Attributes as Claims

Attribute store: Active Directory

Mapping of LDAP attributes to outgoing claim types:

| | LDAP Attribute | Outgoing Claim Type |
|---|---|---|
| ▶ | Surname | Surname |
| * | | |

   c. Add an **email** rule as follows:



Edit Rule - email

You can configure this rule to send the values of LDAP attributes as claims. Select an attribute store from which to extract LDAP attributes. Specify how the attributes will map to the outgoing claim types that will be issued from the rule.

Claim rule name: email

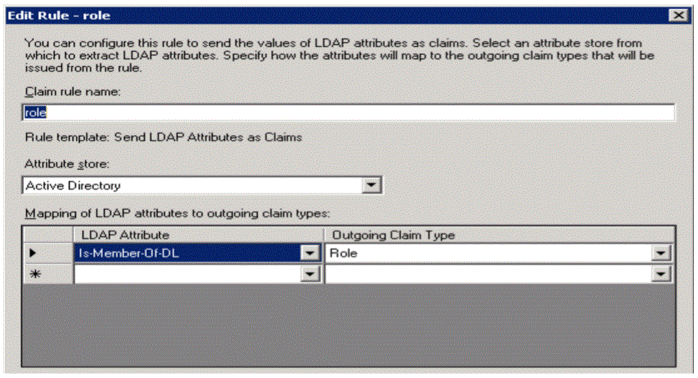
Rule template: Send LDAP Attributes as Claims

Attribute store: Active Directory

Mapping of LDAP attributes to outgoing claim types:

| | LDAP Attribute | Outgoing Claim Type |
|---|---|---|
| ▶ | E-Mail-Addresses | E-Mail Address |
| * | | |

d. Add a **role** rule as follows. This rule can contain any mapping you need to the groups in the AD. Make sure to reflect your changes in the `attr_map.py` file.


Edit Rule - role

You can configure this rule to send the values of LDAP attributes as claims. Select an attribute store from which to extract LDAP attributes. Specify how the attributes will map to the outgoing claim types that will be issued from the rule.

Claim rule name: role

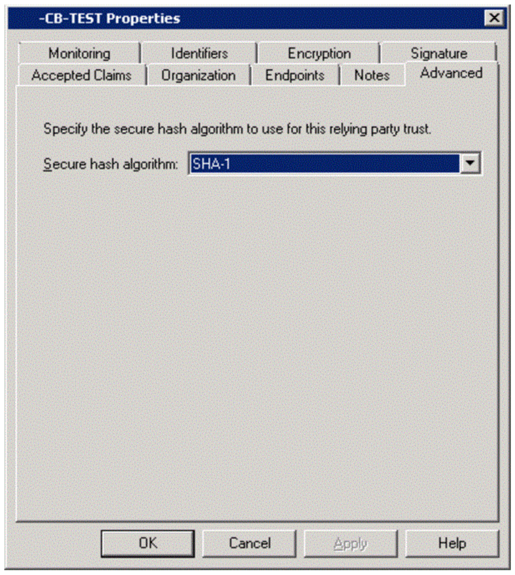
Rule template: Send LDAP Attributes as Claims

Attribute store: Active Directory

Mapping of LDAP attributes to outgoing claim types:

| LDAP Attribute | Outgoing Claim Type |
|---|---|
| Is-Member-Of-DL | Role |
| | |

5. Change the default **Secure hash algorithm** to **SHA-1** as follows:


-CB-TEST Properties

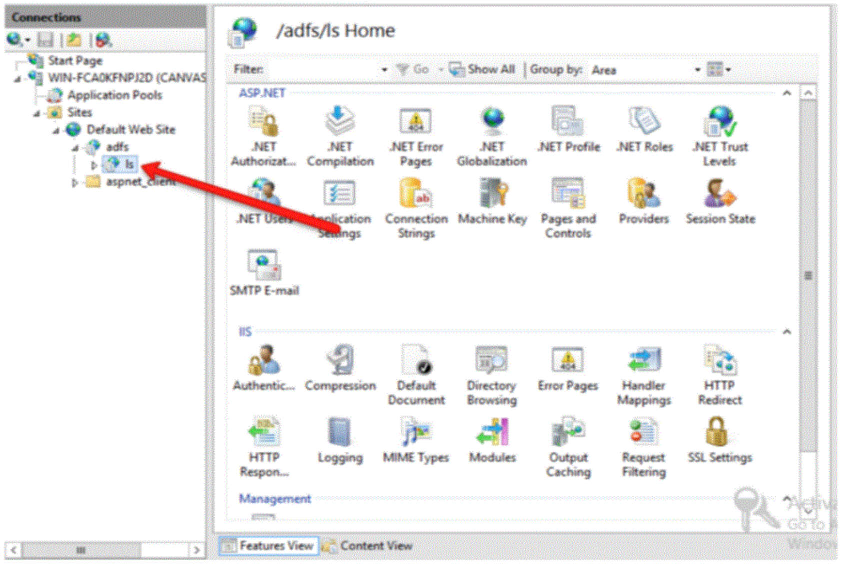
Specify the secure hash algorithm to use for this relying party trust.

Secure hash algorithm: SHA-1

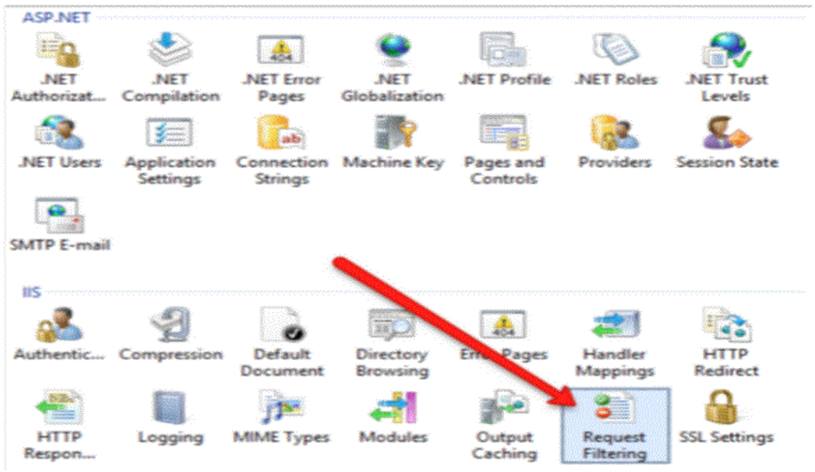
Next, you must add a special filter in IIS to allow logout requests to go through.

**To add an IIS filter for ADFS logout:**

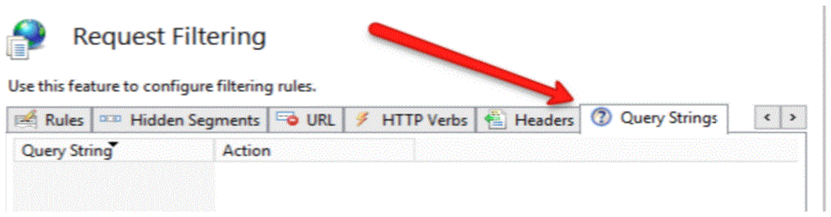
1. Open your IIS Manager.
2. In the left pane, locate the `adfs/ls` directory:



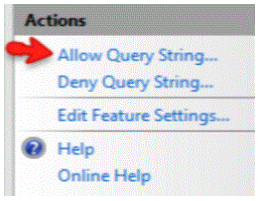
3. In the right pane, select **Request Filtering**:



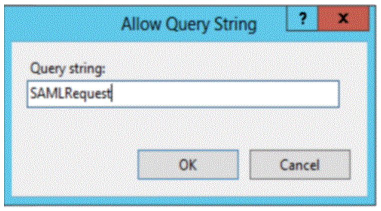
4. Select the **Query Strings** tab:



5. Select the **Allow Query String** action:

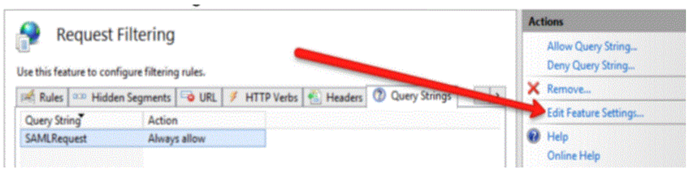


6. Add in a new **Query string** called **SAMLRequest**.


Allow Query String

Query string:
SAMLRequest

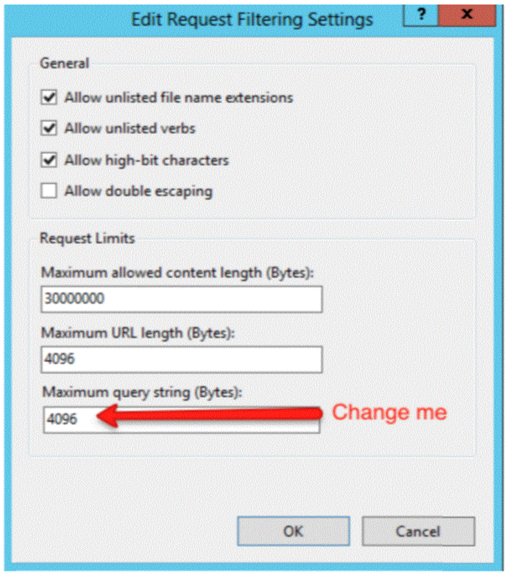
7. Select the **Edit Feature Settings** action:


Request Filtering

Use this feature to configure filtering rules.

| Query String | Action |
|---|---|
| SAMLRequest | Always allow |

8. Change **Maximum query string (Bytes)** to **4096**:


Edit Request Filtering Settings

General

- [✓] Allow unlisted file name extensions
- [✓] Allow unlisted verbs
- [✓] Allow high-bit characters
- [ ] Allow double escaping

Request Limits

Maximum allowed content length (Bytes): 30000000

Maximum URL length (Bytes): 4096

Maximum query string (Bytes): 4096

9. At this point, you may need to restart the adfs/ls IIS web pages to implement these changes.

## Troubleshooting

If SSO does not function as expected, review the log file located at:

`/var/log/cb/coreservices/debug.log`

You can also inspect the actual SAML requests being sent and the responses being received by increasing the logging level of the `cb.flask.api_routes_saml` and `saml2` modules.

**To increase the logging level of the `cb.flask.api_routes_saml` and `saml2` modules:**

1. Open the `/etc/cb/coreservices-logger.conf` file.
2. Append `cb.flask.api_routes_saml` and `saml2` to the list of keys under the `[loggers]` section.

   `[loggers]` section example:

   ```
   keys=root, gunicorn.access, cb.flask.api_routes_saml, saml2
   ```
3. Paste the following below the `[loggers]` section in the `coreservices-logger.conf` file:

   ```
   [logger_cb.flask.api_routes_saml]
   level=DEBUG
   handlers=debug_syslog
   qualname=cb.flask.api_routes_saml
   propagate=1
   ```
4. Also, paste the following below the `[loggers]` section in the `coreservices-logger.conf` file:

   ```
   [logger_saml2]
   level=DEBUG
   handlers=debug_syslog
   propagate=0
   qualname=saml2
   ```

Chapter 4

# Integrating with Third-Party Authentication

**Sections**

## Overview

Using the Duo plugin, you can configure two-factor authentication and download the Duo Mobile application on a mobile device.

Duo two-factor authentication:

- Adds a second authentication factor to your server
- Facilitates authentication management and security monitoring

You can add several device types for two-factor authentication, including mobile phone devices, tablets, and landlines. For information on how to add these devices using Duo Mobile, see  Duo's product documentation at https://duo.com.

## Set Up Duo Administrator Unix Application Account

The Cb Response server administrator must create a Unix application account in order to integrate with Cb Response.

**To set up a Duo Administrator Unix application account:**

1. Go to https://duo.com.
2. Click **Sign Up** in the top-right corner of this page and follow the prompts to create an account and activate the Duo Mobile application on your preferred device.

| | |
|---|---|
| **Note** | *If you already have a Duo Mobile account, you can use this. You do not have to create a new account for the Cb Response-Duo plugin.* |

3. When your account is active and you have activated the Duo Mobile application on your device, log in with your account at https://duo.com.
4. Select **Applications** in the left panel.
5. Click **Protect an Application**.
6. Scroll down to locate the **Unix Application** selection and click the **Protect this Application** blue link.
7. Scroll down to the **Settings > General** panel.
8. In the **Name** field, enter a name for the Unix application, such as "Cb Response Server".

9. For **Username normalization**, select the choice that best represents your security posture.

| | |
|---|---|
| ***Note*** | *This setting is related to the field specified in the* `/user/share/cb/plugins/duo/secrets.ini` *settings file for mapping Cb Response users to Duo users. See "Map Cb Response Users to Duo Users" on page 60.* *If the Duo Mobile application is setup for **Simple** normalization of the username, and Cb Response-Duo integration is configured for email, then only the name in the Cb Response user email field (the value before the "@" symbol) is used to match the Duo user.* *If password is setup for **None** and Cb Response-Duo integration is configured for email or username, then the entire string must match a Duo user account.* |

10. Click **Save Changes**.

## Configure Duo Plugin

To configure the Duo plugin and enable communication of the Duo plugin through an Internet proxy, you will edit the following settings file:

`/usr/share/cb/plugins/duo/secrets.ini`

**To configure the Duo plugin:**

Enter values for the following Duo plug settings:

| | |
|---|---|
| ***Note*** | *These values are specific to the administrator's Duo Mobile application configuration, which is available on the Duo Admin Portal. For more information on using the Duo Admin Portal, see https://duo.com/docs/administration.* |

- `ikey`
- `skey`
- `host`

**To enable communication of the Duo plugin through an Internet proxy:**

Enter values for the following Internet proxy settings:

- `hostname`
- `port`
- `username`
- `password`

| | |
|---|---|
| ***Note*** | *The Duo plugin supports the basic_auth username/password proxy authentication.* |

## Map Cb Response Users to Duo Users

Refer to the following section in the `secrets.ini` settings file (see "secrets.ini Settings File" on page 60):

```
# how to map Cb users to Duo users? Choices are "username" or
"email". Default to "email"
duo_mapping=email
```

This specifies the data in the Cb Response user account that will be sent as the username to the Duo authorization server.

## secrets.ini Settings File

The following is the `secrets.ini` settings file for your reference:

```
# Secret keys for your Duo security integration. Get your ikey,
skey, and host from your Duo
# Security administrative console.

[duo]
ikey=
skey=
host=

[config]
# number of seconds a "session" should last until 2fa is
required again. Default to 60
session_lifetime=500

# how to map Cb users to Duo users? Choices are "username" or
"email". Default to "email"
duo_mapping=email

# create a new 2fa session for each source IP address? True or
false. Default to "false"
use_ipaddr_session_key=false

# Uncomment the next section if you need a web proxy to reach
the outside world

#[proxy]
#hostname=
#port=
#
## if the proxy requires authentication, put the username &
password here
#username=
#password=
```

## Enable Two-Factor Authentication

**To enable two-factor authentication on your Cb Response server:**

1. Search for the following section in the the `cb.conf` configuration file:

   ```
   # Two factor authentication plugin path
   #TwoFactorAuthCallbackModulePath=/usr/share/cb/plugins/duo/
   duo_2fa_auth_callback.py
   ```

   For more information, see the *Carbon Black Response Server Configuration (cb.conf) Guide* located on the Carbon Black User eXchange.

2. Uncomment the `TwoFactorAuthCallbackModulePath` setting in the `cb.conf` configuration file as follows:

   ```
   # Two factor authentication plugin path
   TwoFactorAuthCallbackModulePath=/usr/share/cb/plugins/duo/
   duo_2fa_auth_callback.py
   ```

Chapter 5

# Syslog Output for Cb Response Events

This chapter describes syslog output for Cb Response events. It provides descriptions and examples of the output, and explains how you can use syslog output for notification of alerts.

**Sections**

## Overview

Cb Response logs the following events to syslog:

- **Watchlist hit** – This event occurs when activity or binaries found on one of your endpoints matches a query in a watchlist. See Chapter 13, “Watchlists,” for more information.
- **Feed hit** – This event occurs when activity or binaries found on one of your endpoints matches an IOC reported by a threat intelligence feed. See Chapter 11, “Threat Intelligence Feeds,” for more information.
- **Binary Information event** – This event occurs when a process execution adds a binary to the Cb Response database.

The program name prefix for these events is `cb-notifications-`. By default, these events are written to log files at `/var/log/cb/notifications` on the Cb Response server (based on the syslog configuration at `/etc/rsyslog.d/cb-coreservices`).

In the `/var/log/cb/notifications` directory, there is one file for all hits, one file for each watchlist, and one file for each feed. Watchlist files include the watchlist ID in the program name and in the log file name, while feed files include the feed ID in the program name and in the log file name. Binary information events are logged in a separate file.

| **Note** | *Currently, the Binary Information events are not published in the* `cb-all-notifications.log` *file.* |
|---|---|

For example, the `/var/log/cb/notifications` directory listing below contains log files for the following events:

- All watchlist and feed hits
- Hits to Watchlist ID 10
- Hits to Feed ID 8
- All binary information events

```
[root@localhost coreservices]# ll /var/log/cb/notifications/*.log
-rw-------. Jun 9 15:30 /var/log/cb/notifications/cb-all-
notifications.log

-rw-------. Jun 9 15:30 /var/log/cb/notifications/cb-
notifications-watchlist-10.log

-rw-------. Jun 9 18:02 /var/log/cb/notifications/cb-
notifications-feed-8.log

-rw-------. Jun 9 18:04 /var/log/cb/notifications/cb-
notifications-binaryinfo.log
```

## Syslog Format

Each watchlist and feed hit consists of a series of key-value pairs. The key-value pairs for binary and process hits are different. The following sections provide examples and descriptions of the different types of key-value pairs for watchlist hits and feed hits.

## Watchlist Hits

This section describes process and binary hits on watchlists.

### Process Hit on a Watchlist

The following examples show key-value pairs in a process watchlist hit and the default process template that is used to create the key-value pairs for the hit.

#### Key-value Pairs

```
Mar 13 10:00:09 [1037] <warning>  reason=watchlist.hit type=event
process_guid=00000c42-0000-172c-01d0-5d6cca2adbb2
segment_id=1488563344023 host='WORKSTATION-8LT' sensor_id=4322
watchlist_id=3 watchlist_name='Netconns to .cn or .ru'
timestamp='1426255209.17' start_time='2014-11-13T09:05:02.34Z'
group='Default Group'
process_md5='b9d6d7e6e5c4fcd8dd7f88ec9d563085'
process_name='chrome.exe' process_path='c:\program files
(x86)\google\chrome\application\chrome.exe' last_update='2014-11-
13T13:49:39.361Z'
```

#### Default Process Template

| | |
|---|---|
| **Note** | *Line breaks are added in the following example for readability. Template fields are separated by spaces.* |

```
reason=watchlist.hit type=event
process_guid={{doc['id']}}
segment_id={{doc["segment_id"]}}
host='{{doc['hostname']}}'
comms_ip='{{doc['comms_ip']}}'
interface_ip='{{doc['interface_ip']}}'
sensor_id={{doc['sensor_id']}}
watchlist_id={{doc['watchlist_id']}}
watchlist_name='{{doc['watchlist_name']}}'
timestamp='{{doc['event_timestamp']}}'
start_time='{{doc['start']}}'
group='{{doc['group']}}'
process_md5='{{doc['process_md5']}}'
process_name='{{doc['process_name']}}'
process_path='{{doc['path']}}'
last_update='{{doc['last_update']}}'
{% for k in doc %}{% if k.startswith("alliance_") %}
{{k}}='{{doc[k]}}'{% endif %}{% endfor %}
```

Table 6 describes the process watchlist hit key-value pairs.

**Table 6:** Process Watchlist Hit Key-value Pairs

| Syslog Label | Solr Doc Reference | Description |
|---|---|---|
| **reason** | No doc reference | Text that describes the entry. The **reason** label is hard-coded in the syslog template and identifies the type of the event as "watchlist.hit," "feed.hit," or "binaryinfo." |
| **type** | No doc reference | Text that identifies the type of data that is returned with the event. For example:<br>• event (process events)<br>• module (binary modules)<br>• host (host level information only) |
| **process_guid** | id | Process doc identifier. |
| **segment_id** | segment_id | Process doc segment identifier. |
| **host** | hostname | Hostname of the computer on which the process executed. |
| **comms_ip** | comms_ip | IP address from which Cb Response received the event (which could be a NAT or proxy address, if one is configured for the computer on which the process executed; otherwise will be the same as interface_ip). |
| **interface_ip** | Interface_ip | IP address of the computer on which the process executed. |
| **sensor_id** | sensor_id | Sensor ID of the computer on which the process executed. |
| **watchlist_id** | watchlist_id | The ID of the watchlist that matched the hit criteria (-1 is the internal syslog test watchlist ID. When you use the cbsyslog command line tool to test the output of your custom template, -1 is hard-coded to be the watchlist_id attribute, and is displayed in place of the actual watchlist_id when you run the test.) |
| **watchlist_name** | watchlist_name | Name of the watchlist that matched the hit criteria. |
| **timestamp** | event_timestamp | Epoch time of the watchlist hit event. |
| **start_time** | start | Start time of this process, in the computer's local time. |
| **group** | group | Sensor group to which this sensor was assigned at the time of process execution. |

**Table 6:** Process Watchlist Hit Key-value Pairs

| Syslog Label | Solr Doc Reference | Description |
|---|---|---|
| **process_md5** | process_md5 | MD5 hash value of the executable backing this process. |
| **process_name** | process_name | Filename of the executable backing this process. |
| **process_path** | path | Full path to the executable backing this process. |
| **last_update** | last_update | Last activity in this process, in the computer's local time. |
| **for/if loops** | alliance_*<br>ioc_attr | • `alliance_*` identifies and prints all attributes whose names start with "alliance_" in all documents that contain feed hits, including documents reported by watchlist hits. These attributes represent feed hits.<br>• `ioc_attr` identifies and prints additional attributes on IOC values that were matched.<br>**NOTE:** for/if loops are not required. Their purpose is to report attributes that do not have predefined sets. You can create customized templates that do not contain them if you do not need to report on alliance`_*` or `ioc_attr` attributes. |

## Binary Hit on a Watchlist

The following examples show key-value pairs in a binary watchlist hit and the default process template that is used to create the key-value pairs.

### Key-value Pairs

```
Mar 13 03:50:19 [1037] <warning>  reason=watchlist.hit type=module
md5=7931ADC31F0180855E05D6666630B3A3 host=WORKSTATION-2'
sensor_id=74 watchlist_id=8 watchlist_name='Newly Loaded Modules'
timestamp='1426233008.59' first_seen='2014-11-13T07:47:17.862Z'
group=['Default Group'] desc='AntiMalware Definition Update'
company_name='Microsoft Corporation' product_name='Microsoft
Malware Protection' product_version='1.193.2512.0'
file_version='1.193.2512.0' signed='Signed'
```

### Default Binary Template

| Note | *Line breaks are added in the following example for readability. Template fields are separated by spaces.* |
|---|---|

```
reason=watchlist.hit type=module
md5={{doc["md5"]}}
host='{{doc.get('hostname')}}'
sensor_id={{doc.get('sensor_id')}}
watchlist_id={{doc['watchlist_id']}}
watchlist_name='{{doc['watchlist_name']}}'
timestamp='{{doc['event_timestamp']}}'
first_seen='{{doc["server_added_timestamp"]}}'
group={{doc["group"]}}
desc='{{doc["file_desc"]}}' company_name='{{doc["company_name"]}}'
product_name='{{doc["product_name"]}}'
product_version='{{doc["product_version"]}}'
file_version='{{doc["file_version"]}}'
signed='{{doc["signed"]}}'
{% for k in doc %}{% ifk.startswith("alliance_") %}
{{k}}='{{doc[k]}}'{% endif %}{% endfor %}
```

Table 7 describes the binary watchlist hit key-value pairs.

**Table 7:** Binary Watchlist Hit Key-value Pairs

| Syslog Label | Solr Doc Reference | Description |
|---|---|---|
| **reason** | No doc reference | Text that describes the entry. The **reason** label is hard-coded in the syslog template and identifies the type of the event as "watchlist.hit," "feed.hit," or "binaryinfo." |
| **type** | No doc reference | Text that identifies the type of data that is returned with the event. For binary modules, the value is **module**. |
| **md5** | md5 | MD5 hash value of a process, a parent process, a child process, a loaded module or a written file. |
| **host** | hostname | Hostname of the computer on which the binary was observed. |
| **sensor_id** | sensor_id | Sensor ID of the computer on which the binary was observed. |
| **watchlist_id** | watchlist_id | The ID of the watchlist that matched the hit criteria (-1 is the internal syslog test). |
| **watchlist_name** | watchlist_name | Name of the watchlist that matched the hit criteria. |
| **timestamp** | event_timestamp | Epoch time of the watchlist hit event. |
| **first_seen** | server_added_ timestamp | Time that this binary was first seen by the Cb Response server. |
| **group** | group | First sensor group in which this binary was observed. |
| **desc** | file_desc | File description string from the class FileVersionInfo. |
| **company_name** | company_name | Company name string from the class FileVersionInfo. |
| **product_name** | product_name | Product name string from the class FileVersionInfo. |
| **product_version** | product_version | Product version string from the class FileVersionInfo. |
| **file_version** | file_version | File version string from the class FileVersionInfo. |
| **signed** | signed | Digital signature status of the binary. |

**Table 7:** Binary Watchlist Hit Key-value Pairs

| Syslog Label | Solr Doc Reference | Description |
|---|---|---|
| **for/if loops** | _*<br>ioc_attr | • `alliance_*` identifies and prints all attributes whose names start with "alliance_" in all documents that contain feed hits, including documents reported by watchlist hits. These attributes represent feed hits.<br>• `ioc_attr` identifies and prints additional attributes on IOC values that were matched.<br>**NOTE:** for/if loops are not required. Their purpose is to report attributes that do not have predefined sets. You can create customized templates that do not contain them if you do not need to report on alliance_* or `ioc_attr` attributes. |

## Feed Hits

Each feed hit is a series of key-value pairs. By default, a feed hit is logged only at the ingress as feed events arrive at the Cb Response server. These are `feed.ingress.hit` events.

Optionally (when enabled via the `EnableSolrFeedNotifications` configuration option in `/etc/cb/cb.conf`), the feed hit is also logged when committed to persistent storage. In the latter case, the notification can contain additional key-value pairs in the binary or process hit. These are `feed.storage.hit` events.

## Process Ingress Hit on Feed

The following examples show key-value pairs in a process ingress feed hit and the default process template that is used to create the key-value pairs.

### Key-value Pairs

```
Aug 12 14:24:19 [26070] <warning> reason=feed.ingress.hit
type=event process_guid=00000001-0000-de04-01cf-b65a8ecb26cf
host='SERV12R2X64-01' sensor_id=1 feed_id=10 feed_name='tor'
ioc_type='ipv4' ioc_value='38.229.70.52' direction='Outbound'
protocol='TCP' port='22' timestamp='1407867859.64'
```

### Default Template

| **Note** | *Line breaks are added in the following example for readability. Template fields are separated by spaces.* |
|---|---|

```
reason=feed.ingress.hit type=event
process_guid={{doc['process_id']}}
host='{{doc['hostname']}}'
sensor_id={{doc['sensor_id']}}
feed_id={{doc['feed_id']}}
feed_name='{{doc['feed_name']}}'
```

```
ioc_type='{{doc['ioc_type']}}'
ioc_value='{{doc['ioc_value']}}'
{% for k in doc['ioc_attr'] %}
{{k}}='{{doc['ioc_attr'][k]}}'{% endfor %}
timestamp='{{doc['event_timestamp']}}'
```

Table 8 describes the process ingress feed hit key-value pairs.

**Table 8:** Process Ingress Feed Hit Key-value Pairs

| Syslog Label | Solr Doc Reference | Description |
|---|---|---|
| **reason** | no doc reference | Text that describes the entry. The **reason** label is hard-coded in the syslog template and identifies the type of the event as "watchlist.hit," "feed.hit," or "binaryinfo." |
| **type** | no doc reference | Text that identifies the type of data that is returned with the event. For process events, the value is 'event'. |
| **process_guid** | process_id | Process doc identifier. |
| **host** | hostname | Hostname of the computer on which the feed hit was detected. |
| **sensor_id** | sensor_id | Sensor ID of endpoint that observed the feed hit. |
| **feed_id** | feed_id | ID of the feed that matched the hit criteria. |
| **feed_name** | feed_name | Name of the feed that matched the hit criteria. |
| **ioc_type** | ioc_type | Type of the IOC that caused the hit. |
| **ioc_value** | ioc_value | Value of the IOC that matched the hit criteria. |
| **for/if loops** | _* <br> ioc_attr | • `alliance_*` identifies and prints all attributes whose names start with "alliance_" in all documents that contain feed hits, including documents reported by watchlist hits. These attributes represent feed hits. <br> • `ioc_attr` identifies and prints additional attributes on IOC values that were matched. <br> **NOTE:** for/if loops are not required. Their purpose is to report attributes that do not have predefined sets. You can create customized templates that do not contain them if you do not need to report on alliance`_*` or `ioc_attr` attributes. |
| **timestamp** | event_timestamp | Epoch time of the feed hit event. |

## Process Storage Hit on Feed

The following examples show key-value pairs in a process storage feed and the default process template that is used to create the key-value pairs.

### Key-value Pairs

```
Aug 12 14:26:10 [26070] <warning> reason=feed.storage.hit
type=event process_guid=00000001-0000-de04-01cf-b65a8ecb26cf
segment_id=1488563344023 host='SERV12R2X64-01' sensor_id=1
feed_id=10 feed_name='tor' ioc_type='ipv4'
ioc_value='38.229.70.52' direction='Outbound' protocol='TCP'
port='22' timestamp='1407867970.49' start_time='2014-08-
12T18:23:47.602Z' group='Default Group'
process_md5='a3ccfd0aa0b17fd23aa9fd0d84b86c05'
process_name='putty.exe'
process_path='c:\users\ssmith\desktop\putty.exe'
last_update='2014-08-12T18:23:55.415Z' alliance_link_tor='http://
www.torproject.org' alliance_score_tor='0'
alliance_updated_tor='2014-05-06T17:15:23.000Z'
alliance_data_tor='TOR-Node-38.229.70.52'
```

### Default Process Storage Feed Template

| **Note** | *Line breaks are added in the following example for readability. Template fields are separated by spaces.* |
|---|---|

```
reason=feed.storage.hit type=event
process_guid={{doc['process_id']}
segment_id={{doc['segment_id']}}
host='{{doc['hostname']}}'
comms_ip='{{doc['comms_ip']}}'
interface_ip='{{doc['interface_ip']}}'
sensor_id={{doc['sensor_id']}}
feed_id={{doc['feed_id']}}
feed_name='{{doc['feed_name']}}'
ioc_type='{{doc['ioc_type']}}'
ioc_value='{{doc['ioc_value']}}'
{% for k in doc['ioc_attr'] %}
{{k}}='{{doc['ioc_attr'][k]}}'{% endfor %}
timestamp='{{doc['event_timestamp']}}'
start_time='{{doc['start']}}'
group='{{doc['group']}}'
process_md5='{{doc['process_md5']}}'
process_name='{{doc['process_name']}}'
process_path='{{doc['path']}}'
last_update='{{doc['last_update']}}'
{% for k in doc %}{% if k.startswith("alliance_") %}
{{k}}='{{doc[k]}}'{% endif %}{% endfor %}
```

Table 9 describes the default process storage feed hit key-value pairs.

**Table 9:** Process Storage Feed Hit Key-value Pairs

| Syslog Label | Solr Doc Reference | Description |
|---|---|---|
| **reason** | no doc reference | Text that describes the entry. The **reason** label is hard-coded in the syslog template and identifies the type of the event as "watchlist.hit," "feed.hit," or "binaryinfo." |
| **type** | no doc reference | Text that identifies the type of data that is returned with the event. For process events, the value is ‘event’. |
| **process_guid** | process_id | Process doc identifier. |
| **segment_id** | segment_id | Process doc segment identifier. |
| **host** | hostname | Hostname of the computer on which the feed hit was detected. |
| **comms_ip** | comms_ip | IP address from which Cb Response received the event (which could be a NAT or proxy address, if one is configured for the computer on which the process executed; otherwise will be the same as interface_ip). |
| **interface_ip** | Interface_ip | IP address of the computer on which the process executed. |
| **sensor_id** | sensor_id | Sensor ID of endpoint that observed the feed hit. |
| **feed_id** | feed_id | ID of the feed that was matched. |
| **feed_name** | feed_name | Name of the feed that was matched. |
| **report_title** | report_title | Name of the item in the feed that was matched. |
| **ioc_type** | ioc_type | Type of the IOC that caused the hit. |
| **ioc_value** | ioc_value | Value of the IOC that matched. |
| **for/if loops** | _*<br>ioc_attr | • `alliance_*` identifies and prints all attributes whose names start with "alliance_" in all documents that contain feed hits, including documents reported by watchlist hits. These attributes represent feed hits.<br>• `ioc_attr` identifies and prints additional attributes on IOC values that were matched.<br>**NOTE:** for/if loops are not required. Their purpose is to report attributes that do not have predefined sets. You can create customized templates that do not contain them if you do not need to report on alliance_* or `ioc_attr` attributes. |
| **timestamp** | event_timestamp | Epoch time of the feed hit event. |

| Syslog Label | Solr Doc Reference | Description |
|---|---|---|
| **start_time** | start | Start time of this process, in the computer's local time. |
| **group** | group | Sensor group to which the sensor was assigned at the time of process execution. |
| **process_md5** | process_md5 | MD5 hash value of the executable backing this process. |
| **process_name** | process_name | Filename of the executable backing this process. |
| **process_path** | path | Full path to the executable backing this process. |
| **last_update** | last_update | Last activity in this process, in the computer's local time. |

## Binary Ingress Hit on Feed

The following examples show key-value pairs in a binary ingress feed hit and the default process template that is used to create the key-value pairs.

### Key-value Pairs

```
Aug 12 14:06:39 [26070] <warning> reason=feed.ingress.hit
type=module md5=B84E2D174DC84916A536572BB8F691A8
host='SERV12R2X64-01' sensor_id=1 feed_id=2 feed_name='srstrust'
ioc_type='md5' ioc_value='b84e2d174dc84916a536572bb8f691a8'
timestamp='1407866781.79'
```

### Default Binary Ingress Feed Hit Template

| | |
|---|---|
| ***Note*** | *Line breaks are added in the following example for readability. Template fields are separated by spaces.* |

```
reason=feed.ingress.hit type=module
md5={{doc['md5']}}
host='{{doc['hostname']}}'
sensor_id={{doc['sensor_id']}}
feed_id={{doc['feed_id']}}
feed_name='{{doc['feed_name']}}'
ioc_type='{{doc['ioc_type']}}'
ioc_value='{{doc['ioc_value']}} '
{% for k in doc['ioc_attr'] %} {{k}}='{{doc['ioc_attr'][k]}}'{%
endfor %}
timestamp='{{doc['event_timestamp']}}'
```

Table 10 describes the binary ingress feed hit key-value pairs.

**Table 10:** Binary Ingress Feeds Key-value Pairs

| Syslog Label | Solr Doc Reference | Description |
|---|---|---|
| **reason** | no doc reference | Text that describes the entry. The **reason** label is hard-coded in the syslog template and identifies the type of the event as "watchlist.hit," "feed.hit," or "binaryinfo." |
| **type** | no doc reference | Text that identifies the type of data that is returned with the event. For example:<br>• event (process events)<br>• module (binary modules)<br>• host (host level information only) |
| **md5** | md5 | MD5 hash value of a process, a parent process, a child process, a loaded module or a written file. |
| **host** | hostname | Hostname of the computer on which the feed hit was detected. |
| **sensor_id** | sensor_id | Sensor ID of the endpoint that detected the feed hit. |
| **feed_id** | feed_id | ID of the feed that was matched. |
| **feed_name** | feed_name | Name of the feed that was matched. |
| **ioc_type** | ioc_type | Type of IOC that caused the hit. |
| **ioc_value** | ioc_value | Value of the IOC that matched the hit criteria. |
| **for/if loops** | _*<br>ioc_attr | • `alliance_*` identifies and prints all attributes whose names start with "alliance_" in all documents that contain feed hits, including documents reported by watchlist hits. These attributes represent feed hits.<br>• `ioc_attr` identifies and prints additional attributes on IOC values that were matched.<br>**NOTE:** for/if loops are not required. Their purpose is to report attributes that do not have predefined sets. You can create customized templates that do not contain them if you do not need to report on alliance_* or `ioc_attr` attributes. |
| **timestamp** | event_timestamp | Epoch time of the feed hit event. |

## Binary Storage Hit on Feed

The following examples show key-value pairs in a binary storage feed hit and the default process template that is used to create the key-value pairs.

### Key-value Pairs

```
Aug 12 14:06:39 [26070] <warning> reason=feed.storage.hit
type=module md5=B84E2D174DC84916A536572BB8F691A8
host='SERV12R2X64-01' sensor_id=1 feed_id=2 feed_name='srstrust'
ioc_type='md5' ioc_value='b84e2d174dc84916a536572bb8f691a8'
timestamp='1407866797.20' first_seen='2014-08-12T18:06:22.190Z'
group=['Default Group'] desc='Windows Security Center ISV API'
company_name='Microsoft Corporation' product_name='MicrosoftÂ®
WindowsÂ® Operating System' product_version='6.1.7600.16385'
file_version='6.1.7600.16385 (win7_rtm.090713-1255)'
signed='Signed' alliance_updated_srstrust='2014-05-
16T04:39:55.000Z' alliance_score_srstrust='-100'
alliance_data_srstrust='['b84e2d174dc84916a536572bb8f691a8']'
alliance_link_srstrust='https://services.carbonblack.com/Services/
extinfo.aspx?ak=b8b4e631d4884ad1c56f50e4a5ee9279&sg=0313e1735f6cec
221b1d686bd4de23ee&md5=b84e2d174dc84916a536572bb8f691a8'
```

### Default Binary Storage Feed Hit Template

| **Note** | *Line breaks are added in the following example for readability. Template fields are separated by spaces.* |
|---|---|

```
reason=feed.storage.hit type=module
md5={{doc['md5']}}
host='{{doc['hostname']}}'
sensor_id={{doc['sensor_id']}}
feed_id={{doc['feed_id']}}
feed_name='{{doc['feed_name']}}'
ioc_type='{{doc['ioc_type']}}'
ioc_value='{{doc['ioc_value']}} '
{% for k in doc['ioc_attr'] %} {{k}}='{{doc['ioc_attr'][k]}}'{%
endfor %}
timestamp='{{doc['event_timestamp']}}'
first_seen='{{doc["server_added_timestamp"]}}'
group={{doc["group"]}}
desc='{{doc["file_desc"]}}'
company_name='{{doc["company_name"]}}'
product_name='{{doc["product_name"]}}'
product_version='{{doc["product_version"]}}'
file_version='{{doc["file_version"]}}'
signed='{{doc["digsig_result"]}}'
{% for k in doc %}{% if k.startswith("alliance_") %}
{{k}}='{{doc[k]}}'{% endif %}{% endfor %}
```

Table 11 describes the default binary storage feed hit key-value pairs.

**Table 11:** Binary Storage Feed Hit Key-value Pairs

| Syslog Label | Solr Doc Reference | Description |
|---|---|---|
| **reason** | no doc reference | Text that describes the entry. The **reason** label is hard-coded in the syslog template and identifies the type of the event as "watchlist.hit," "feed.hit," or "binaryinfo." |
| **type** | no doc reference | Text that identifies the type of data that is returned with the event. For binary events, the value is ‘module’. |
| **md5** | md5 | MD5 hash value of a process, a parent process, a child process, a loaded module, or a written file. |
| **host** | hostname | Hostname of the computer on which the feed hit was detected. |
| **sensor_id** | sensor_id | Sensor ID of the endpoint that observed the feed hit. |
| **feed_id** | feed_id | ID of the feed that was matched. |
| **feed_name** | feed_name | Name of the feed that was matched. |
| **ioc_type** | ioc_type | Type of the IOC that caused the hit. |
| **ioc_value** | ioc_value | Value of the IOC that matched. |
| **for/if loops** | _*<br>ioc_attr | • `alliance_*` identifies and prints all attributes whose names start with "alliance_" in all documents that contain feed hits, including documents reported by watchlist hits. These attributes represent feed hits.<br>• `ioc_attr` identifies and prints additional attributes on IOC values that were matched.<br>**NOTE:** for/if loops are not required. Their purpose is to report attributes that do not have predefined sets. You can create customized templates that do not contain them if you do not need to report on alliance`_*` or `ioc_attr` attributes. |
| **timestamp** | event_timestamp | Epoch time of the feed hit event. |
| **first_seen** | server_added_ timestamp | The time that this binary was first seen by the server. |
| **group** | group | First sensor group in which this binary was observed. |
| **desc** | file_desc | File description string from the class FileVersionInfo. |
| **company_name** | company_name | Company name string from the class FileVersionInfo. |

| Syslog Label | Solr Doc Reference | Description |
|---|---|---|
| **product_name** | product_name | Product name string from the class FileVersionInfo. |
| **product_version** | product_version | Product version string from the class FileVersionInfo. |
| **file_version** | file_version | File version string from the class FileVersionInfo. |
| **signed** | signed | Digital signature status of the binary. |
| **for/if loops** | _* ioc_attr | • `alliance_*` identifies and prints all attributes whose names start with "alliance_" in all documents that contain feed hits, including documents reported by watchlist hits. These attributes represent feed hits. • `ioc_attr` identifies and prints additional attributes on IOC values that were matched. **NOTE:** for/if loops are not required. Their purpose is to report attributes that do not have predefined sets. You can create customized templates that do not contain them if you do not need to report on alliance_`*` or `ioc_attr` attributes. |

## Host Ingress Hit on Feed

### Key-value Pairs

```
2015-06-24 17:18:58 [3032] <warning>  reason=feed.ingress.hit
type=host host='WIN2008R2DC01' sensor_id=2 feed_id=2
feed_name='cbtamper' ioc_type='class'
ioc_value='com.carbonblack.cbfs.ingress_search.detectors.SensorTam
per$Terminate' hit_field_tamper_type='AlertCbServiceStopped'
timestamp='1435180738.63'
```

### Default Template

| **Note** | *Line breaks are added in the following example for readability. Template fields are separated by spaces.* |
|---|---|

```
reason=feed.ingress.hit type=host host='{{doc['hostname']}}'
sensor_id={{doc['sensor_id']}} feed_id={{doc['feed_id']}}
```

```
feed_name='{{doc['feed_name']}}'
ioc_type='{{doc['ioc_type']}}'
ioc_value='{{doc['ioc_value']}}'
{% for k in doc['ioc_attr'] %} {{k}}='{{doc['ioc_attr'][k]}}'{%
endfor %}
timestamp='{{doc['event_timestamp']}}'
```

The key-value pairs in host ingress feed hits are a subset of those in Table 10, "Binary Ingress Feeds Key-value Pairs", on page 74, and their descriptions can be found there.

**Table 12:** Host Ingress Feed Hit Key-value Pairs

| Syslog Label | Solr Doc Reference | Description |
|---|---|---|
| **reason** | no doc reference | Text that describes the entry. The **reason** label is hard-coded in the syslog template and identifies the type of the event as "watchlist.hit," "feed.hit," or "binaryinfo." |
| **type** | no doc reference | Text that identifies the type of data that is returned with the event. For host events, the value is 'host'. |
| **host** | hostname | Hostname of the computer on which the feed hit was detected. |
| **sensor_id** | sensor_id | Sensor ID of the endpoint that detected the feed hit. |
| **feed_id** | feed_id | ID of the feed that was matched. |
| **feed_name** | feed_name | Name of the feed that was matched. |
| **ioc_type** | ioc_type | Type of IOC that caused the hit. |
| **ioc_value** | ioc_value | Value of the IOC that matched the hit criteria. |
| **timestamp** | event_timestamp | Epoch time of the feed hit event. |

## Process Query Hit on Feed

### Key-value Pairs

```
2015-06-24 14:40:06 [10982] <warning>  reason=feed.query.hit
type=event process_guid=0000000d-0000-564b-01d0-aeac18ce56e9
segment_id=1488563344023 host='stress03' comms_ip=''
interface_ip='' sensor_id=13 feed_id=4
feed_name='bit9endpointvisibility' timestamp='1435171205.89'
start_time='2015-06-24T18:32:16.752Z' group='Default Group'
process_md5='ab611b1f6c952654665a4cda027581f4'
process_name='cbquery' process_path='/usr/share/cb/cbquery'
last_update='2015-06-24T18:32:17.345Z'
```

### Default Template

| **Note** | *Line breaks are added in the following example for readability. Template fields are separated by spaces.* |
|---|---|

```
reason=feed.query.hit type=event
process_guid={{doc['process_id']}}
segment_id={{doc["segment_id"]}}
host='{{doc['hostname']}}'
comms_ip='{{doc['comms_ip']}}'
interface_ip='{{doc['interface_ip']}}'
sensor_id={{doc['sensor_id']}}
feed_id={{doc['feed_id']}}
feed_name='{{doc['feed_name']}}'
{% for k in doc['ioc_attr'] %} {{k}}='{{doc['ioc_attr'][k]}}'{%
endfor %}
timestamp='{{doc['event_timestamp']}}'
start_time='{{doc['start']}}'
group='{{doc['group']}}'
process_md5='{{doc['process_md5']}}'
process_name='{{doc['process_name']}}'
process_path='{{doc['path']}}'
last_update='{{doc['last_update']}}'
{% for k in doc %}{% if k.startswith("alliance_") %}
{{k}}='{{doc[k]}}'{% endif %}{% endfor %}
```

The key-value pairs in process query feed hits are a subset of those in Table 9, “Process Storage Feed Hit Key-value Pairs”, on page 72, and their descriptions can be found there.

## Binary Query Hit on Feed

### Key-value Pairs

```
2015-06-24 18:30:14 [13031] <warning>  reason=feed.query.hit
type=module md5=6D4B29FB9307FBE8781E42B7CFDA4CE1
host='WIN2008R2DC01' sensor_id=2 feed_id=18
feed_name='cbtestquery'  timestamp='1435185013.38' first_seen=''
group='Default Group' desc='XML Resources' company_name='Microsoft
Corporation' product_name='Microsoft XML Core Services'
product_version='8.110.7600.16385' file_version='8.110.7600.16385'
signed='Signed'
```

### Default Template

| | |
|---|---|
| ***Note*** | *Line breaks are added in the following example for readability. Template fields are separated by spaces.* |

```
reason=feed.query.hit type=module
md5={{doc["md5"]}}
host='{{doc.get('hostname')}}'
sensor_id={{doc.get('sensor_id')}}
feed_id={{doc['feed_id']}}
feed_name='{{doc['feed_name']}}'
{% for k in doc['ioc_attr'] %} {{k}}='{{doc['ioc_attr'][k]}}'{%
endfor %}
timestamp='{{doc['event_timestamp']}}'
first_seen='{{doc["server_added_timlestamp"]}}'
group='{{doc["group"]}}'
desc='{{doc["file_desc"]}}'
company_name='{{doc["company_name"]}}'
product_name='{{doc["product_name"]}}'
product_version='{{doc["product_version"]}}'
file_version='{{doc["file_version"]}}'
signed='{{doc["signed"]}}'
{% for k in doc %}{% if k.startswith("alliance_") %}
{{k}}='{{doc[k]}}'{% endif %}{% endfor %}
```

The key-value pairs in process query feed hits are a subset of those in Table 11, “Binary Storage Feed Hit Key-value Pairs”, on page 76, and their descriptions can be found there.

## Syslog Integration

You can use syslog features for notification and data intelligence sharing. Directing Cb Response alerts to syslog files enables a variety of integration options for numerous platforms. Specific fields might vary depending upon the watchlist parameters chosen during creation. Refer to Chapter 10, "Advanced Search Queries" for specific fields to use when creating queries, and to Chapter 13, "Watchlists" for more information about watchlists.

### Setting up Remote Devices

Remote devices must be configured with a new receiver to accept the rsyslog feed from Cb Response. Whether the remote device is an instance of SPLUNK, ArcSight, or another manager-of-managers platform such as Tivoli, the basic setup requirements are the same.

| | |
|---|---|
| ***Note*** | *The procedure for setting up remote devices differs depending upon the device itself. Only the basics are described below. Adapt the procedure to your particular platform.* |

**To set up the remote device to integrate with:**

1. Add a new UDP receiver to the remote device.
2. Enable the new receiver to communicate using a new and unique UDP port number for the communication with Cb Response. Verify that the receiver is working and listening on the appropriate port.

| | |
|---|---|
| ***Note*** | *The system might require the Cb ResponseCb Response IP address to be authorized prior to accepting data.* |

### Setting Up Server Data Transmission

On the Cb Response Cb Responseserver, the rsyslog feature is used to transmit each watchlist hit to a remote device or to multiple remote devices.

**To set up the Cb ResponseCb Response server to send data to remote devices:**

1. Access the Cb Responseserver either through the console or with a remote terminal connection using SSH. For information about logging into Cb Response, see "Logging In" on page 34.
2. Edit the rsyslog file to enable syslog information to be redirected:
   `/etc/rsyslog.d/cb-coreservices.conf`

This example shows example output from an unaltered `cb-coreservices.conf` file:

| **Note** | *The contents of the actual /etc/rsyslog.d/cb-coreservices.conf file may be different.* |
|---|---|

```
# By default the value of this directive is 'on' so that any special
character (ASCII < 32) is escaped. However,
# that causes multiline messages to be rather unreadable. While the
practice of printing multiple lines in a log
# should be discouraged, it is useful when error exception stack tracers
are being reported.  This option might
# also cause problems if other log file reader software is being used as
it may not be able to read additional
# lines as those lines wouldn't have any timestamp/souce information.
#
# If this option is causing problems, it can be disabled which would make
interpretting stack traces a bit more
# difficult. However, the following command can be used when reading log
files to make stack traces readable again:
#     cat /path/to/log/file | sed 's/#012/\n\t/g'
#
$EscapeControlCharactersOnReceive off

$template AccessLogFormat,"%msg%\n"
$template CbLogFormatWithPID,"%timegenerated:1:10:date-rfc3339%
%timegenerated:8:15:% [%procid%] <%syslogseverity-text%> %msg%\n"
$template CbSyslogStandardFormatWithPID,"%timegenerated% [%procid%]
<%syslogseverity-text%> %msg%\n"

$template DynaFile,"/var/log/cb/notifications/%PROGRAMNAME%.log"

if $programname startswith 'process' then -?DynaFile

if $programname == 'cb-coreservices' and $syslogfacility-text == 'local0'
then /var/log/cb/coreservices/debug.log;CbLogFormatWithPID
& ~
if $programname == 'cb-coreservices' and $syslogfacility-text == 'local7'
then /var/log/cb/coreservices/access.log;AccessLogFormat
& ~
if $programname == 'cb-sensorservices' and $syslogfacility-text ==
'local0' then /var/log/cb/sensorservices/debug.log;CbLogFormatWithPID
& ~
if $programname == 'cb-sensorservices' and $syslogfacility-text ==
'local7' then /var/log/cb/sensorservices/access.log;AccessLogFormat
& ~
if $programname == 'cb-allianceclient' and $syslogfacility-text ==
'local0' then /var/log/cb/allianceclient/
allianceclient.log;CbLogFormatWithPID
& ~
if $programname == 'cb-job-runner' then /var/log/cb/job-runner/job-
runner.log;CbLogFormatWithPID
& ~
if $programname == 'cb-notifications' then /var/log/cb/notifications/cb-
all-notifications.log;CbSyslogStandardFormatWithPID
& ~
if $programname startswith 'cb-notifications-' then -
?DynaFile;CbSyslogStandardFormatWithPID
```

```
& ~
if $programname == 'cb-services' then /var/log/cb/services/
init.log;CbLogFormatWithPID
& ~
if $programname == 'cb-enterprised' then /var/log/cb/enterprise/
enterprise.log;CbLogFormatWithPID
& ~
if $programname == 'cb-liveresponse' and $syslogfacility-text == 'local0'
then /var/log/cb/liveresponse/debug.log;CbLogFormatWithPID
& ~
if $programname == 'cb-liveresponse' and $syslogfacility-text == 'local7'
then /var/log/cb/liveresponse/access.log;AccessLogFormat
```

## Sending All Data to a Remote Device

You can direct all watchlist output a specific remote device by adding the remote device IP address to the `cb-all-notifications` parameter in the `/etc/rsyslog.d/cb-coreservices.conf` file.

**To set up the Cb Response server to send data to a remote device:**

1. Log into the Cb Response console.
2. Edit the `cb-coreservices.conf` file as shown in the following example:
   `vi /etc/rsyslog.d/cb-coreservices.conf`
3. Add the following line (**`highlighted`**) to the configuration file under the `cb-allnotifications` line:

   ```
   if $programname == 'cb-notifications' then /var/log/cb/
   notifications/cb-allnotifications.log;CbLogFormatWithPID
   & @<remote device IP address>:<UDP port>;CbLogFormatWithPID
   & ~
   ```
4. Restart the rsyslog daemon so that the changes take effect:
   `service rsyslog restart`
5. Verify that the data is now present on the remote device.

## Sending Watchlist Data to a Remote Device

To direct specific watchlist output to a remote device, you must configure Cb Response to filter each watchlist independently.

**To configure Cb Response to send watchlist data to a remote device:**

1. Login to the Cb Response console.
2. Edit the `cb-coreservices.conf` file:
   `vi /etc/rsyslog.d/cb-coreservices.conf`

3. Add the following line to the configuration file:

| | |
|---|---|
| **Note** | *The entire section below must be added to the cb-coreservices.conf file. The example here specifies watchlist number 105.* |

| | |
|---|---|
| **Note** | *Ensure that the correct watchlist is specified. Verify the watchlist ID from the Cb Response console before you add these lines to the cb-coreservices.conf file to ensure that the correct watchlist is sent to the remote device.* |

```
if $programname == 'cb-notifications-watchlist-105' then
/var/log/cb/notifications/cb-notifications-watchlist-
105.log;CbLogFormatWithPID
& @<remote device IP address>:<UDP port>;CbLogFormatWithPID
& ~
```

4. Restart the rsyslog daemon so that the changes take effect:
```
service rsyslog restart
```
5. Verify that the data is now present on the remote device.

## Enabling Communication Persistence (Spooling)

If communication with the remote device is interrupted, you can enable spooling for notifications on the Cb Response server.

**To enable spooling of notifications on the Cb Response server:**

1. Log into the Cb Response console.
2. Locate and open the `/etc/rsyslog.d/cb-coreservices.conf` file.
3. Add the following lines after the section in which you are capturing logs (this line starts with `if $programname`) and before each action item for that section:
```
//
# An on-disk queue is created for this action.If the remote host
is
# down, messages are spooled to disk and sent when it is up
again.
$WorkDirectory /var/lib/rsyslog # where to place spool files
$ActionQueueFileName fwdRule1 # unique name prefix for
spoolfiles
$ActionQueueMaxDiskSpace 1g # 1gb space limit (use as much as
possible)
$ActionQueueSaveOnShutdown on # save messages to disk on
shutdown
$ActionQueueType LinkedList # run asynchronously
$ActionResumeRetryCount -1 # infinite retries if host is down
//
```

The following is an example:

```
if $programname startswith 'cb-notifications-' then -
?DynaFile;CbSyslogStandardFormatWithPID
$WorkDirectory /var/lib/rsyslog  # location of spoolfiles on
the disk
$ActionQueueFileName cbtest   # unique name prefix for spool
files
$ActionQueueMaxDiskSpace 1g   # 1gb space limit (use as much as
possible)
$ActionQueueSaveOnShutdown on # save messages to disk on
shutdown
$ActionQueueType LinkedList   # run asynchronously
$ActionResumeRetryCount -1    # infinite retries if host is down
& @@192.168.10.252:514;CbSyslogStandardFormatWithPID
& ~
```

## Cb Response Syslog Architecture

### Logging Location

Cb Response stores all logged information in the following directory:

`/var/log/cb/`

| **Note** | *When troubleshooting any server-side activity start with the logs in this directory first.* |
|---|---|

### Watchlist Log Location

Cb Response maintains two separate syslog files for watchlists created in the Cb Response console. The first syslog file is a single file with all watchlist hits consolidated in one place. The second syslog file saves each watchlist hit to it’s own file. All the watchlist syslog files are stored in the following location on the Cb Response server:

`/var/log/cb/notifications`

Each watchlist is assigned a specific number, which can be viewed from the Cb Response server per this example:

`https://<server name>/#/watchlist/`**`105`**

In this example the watchlist number is 105.

Cb Response creates a numbered syslog that matches the watchlist number. In the example above, the watchlist 105 syslog creates the output file:

`cb-notifications-watchlist-105.log-20131031`

The syslog file name format follows a standard convention for all watchlists as shown below:

`cb-notifications-watchlist-`**`<watchlist#>`**`.log-`**`YYYYMMDD`**

The single summary syslog with all watchlist hits in one consolidated file uses the following naming convention:

`cb-all-notifications.log-`**`YYYYMMDD`**

## Syslog Templates

You can use Cb Response syslog templates to build custom-formatted syslog notifications for Cb Response watchlist hits, feed hits, and binary information events.

Syslog output is formatted using Jinja2 templates. There is a command line utility at:

```
/usr/share/cb/cbsyslog
```

that supports:

```
# /usr/share/cb/cbsyslog --help
Usage: cbsyslog.py [options]
```

This utility provides an interface for testing Cb Response's notifications syslog output. The options in this interface are described in Table 13.

**Table 13:** Options for Testing Notifications Syslog Output

| Option | Description |
|---|---|
| -h, --help | Displays a help message and then closes the message. |
| -v, --verbose | Provides detailed output. |
| -l, --list-events | Outputs the list of events, which can be sent to syslog, and then exits. |
| -e EVENT_NAME, --event=EVENT_NAME | Identifies specific event types. Use the `--listevents` option for a list of event names that can be passed here.<br>**NOTE:** Some event output of the cbsyslog -e contains sample data, while other output contains the results from actual database queries. See the output results to determine if the data is sample data; sample data contains a string such as " "*** Note: This event type uses example content for testing ***". |
| -g, --get | Saves the system default templates to the current directory. |
| -t TEMPLATE, --template=TEMPLATE | Formats the syslog message using the specified template instead of the system default. |
| -f, --fire | Formats and sends an event through the syslog message system. For example, you can use this option to manually execute the same process that occurs when the Cb Response server sends an event to syslog when there is a hit. |
| -q QUERY, --query=QUERY | Processes the first Solr doc that matches the query string. You can use this query to identify which document to test with. |

**To build custom-formatted syslog notifications:**

1. Use the `--get` switch to write the system default templates to the local directory:

   ```
   # /usr/share/cb/cbsyslog --get
   # ll
   -rw-rw-r--. 1 root root 246 May 22 00:16
   binaryinfo.group.observed.template
   -rw-rw-r--. 1 root root 285 May 22 00:16
   binaryinfo.host.observed.template
   -rw-rw-r--. 1 root root 221 May 22 00:16
   binaryinfo.observed.template
   -rw-rw-r--. 1 root root 194 May 22 00:16
   feed.ingress.hit.binary.template
   -rw-rw-r--. 1 root root 210 May 22 00:16
   feed.ingress.hit.process.template
   -rw-rw-r--. 1 root root 194 May 22 00:16
   feed.storage.hit.binary.template
   -rw-rw-r--. 1 root root 243 May 22 00:16
   feed.storage.hit.process.template
   -rw-rw-r--. 1 root root 575 May 22 00:16
   watchlist.hit.binary.template
   -rw-rw-r--. 1 root root 460 May 22 00:16
   watchlist.hit.process.template
   ```

   The templates are given a context with a single python dictionary called `doc` that contains the set of all possible key-value pairs.

2. To view the set of all possible keys, use the Jinja For loop to iterate over the indexed fields in the Solr document with this template:

   a. Create a `'forloop.txt'` template with the following contents:

      ```
      {% for k in doc %}{{k}}={{doc[k]}} {% endfor %}
      ```

   b. Use the `--template` switch to output all of the available keys for a specific event type:

      ```
      # /usr/share/cb/cbsyslog --template ./forloop.txt --event
      watchlist.hit.process
      process_md5=506708142bc63daba64f2d3ad1dcd5bf sensor_id=15
      modload_count=45
      filemod_count=0 servername=cbent-qa-nodesvr02 watchlist_id=-
      1
      watchlist_name=SyslogTest id=1068044553602656801
      group=SetSensor
      hostname=CB-WIN81X64-01 last_update=2014-02-28T02:29:00.09Z
      start=2014-02-28T02:29:00.043Z netconn_count=0
      username=SYSTEM
      process_name=googleupdate.exe path=c:\program files
      (x86)\google\update\googleupdate.exe
      regmod_count=1 segment_id=1488563344023
      host_type=workstation cb_version=4.1.1.140225.1913
      childproc_count=0 unique_id=00000c42-0000-172c-01d0-
      5d6cca2adbb2-015A954A1297
      ```

3. To get a list of available event types, use the `-list-events` option:

```
[root@localhost mytemplates]# /usr/share/cb/cbsyslog --list-
events
binaryinfo.group.observed
binaryinfo.host.observed
binaryinfo.observed
feed.ingress.hit.binary
feed.ingress.hit.host
feed.ingress.hit.process
feed.storage.hit.binary
feed.storage.hit.process
watchlist.hit.binary
watchlist.hit.process
feed.query.hit.binary
feed.query.hit.process
```

## Overriding System Default Templates

**To override the system default templates:**

1. To use a new template, add one of the following entries to `/etc/cb/cb.conf`:

```
BinaryInfoSyslogTemplateGroupObserved=/etc/cb/
my_bininfo_group_observed_template.txt
BinaryInfoSyslogTemplateHostObserved=/etc/cb/
my_bininfo_host_observed_template.txt
BinaryInfoSyslogTemplateObserved=/etc/cb/
my_bininfo_observed_template.txt
FeedIngressSyslogTemplateBinary=/etc/cb/
my_feed_ingress_binary_template.txtx
FeedIngressSyslogTemplateProcess=/etc/cb/
my_feed_ingress_process_template.txt
FeedStorageSyslogTemplateBinary=/etc/cb/
my_feed_storage_binary_template.txt
FeedStorageSyslogTemplateProcess=/etc/cb/
my_feed_storage_process_template.txt
WatchlistSyslogTemplateBinary=/etc/cb/
my_wathlist_process_template.txt
WatchlistSyslogTemplateProcess=/etc/cb/
my_watchlist_binary_template.txt
FeedQuerySyslogTemplateBinary=/etc/cb/
my_feed_query_binary_template.txt
FeedQuerySyslogTemplateProcess=/etc/cb/
my_feed_query_process_template.txt
```

2. The watchlist search process will automatically pick up the new template when the next watchlist hit occurs.

## Available Keys by Event Type

### binaryinfo.observed

**Table 14:** binaryinfo.observed Key Description and Example

| Key | Description | Example |
|---|---|---|
| md5 | MD5 hash value of the observed binary module. | 44C0CBADFF00F3930B6A01EE AA405C6F |
| scores | List of threat intelligence feed scores with which the binary is tagged. | [50, 100, 75] |
| watchlists | List of strings, each one identifying a watchlist that was matched with a binary. | [“x”, “a”] |
| event_timestamp | Event timestamp. | 1400695113.17 |

### binaryinfo.group.observed

**Table 15:** binaryinfo.group.observed Key Description and Example

| Key | Description | Example |
|---|---|---|
| md5 | MD5 hash value of the observed binary module. | 44C0CBADFF00F3930B6A01EE AA405C6F |
| scores | List of threat intelligence feed scores with which the binary is tagged. | [50, 100, 75] |
| watchlists | List of strings, each one identifying a watchlist that was matched with a binary. | [“x”, “a”] |
| event_timestamp | Event timestamp. | 1400695113.17 |
| group | Name of the sensor group in which a binary was observed. | Default Group |

## binaryinfo.host.observed

**Table 16:** binaryinfo.observed Key Description and Example

| Key | Description | Example |
|---|---|---|
| md5 | MD5 hash value of the observed binary module. | 44C0CBADFF00F3930B6A0 1EEAA405C6F |
| scores | List of threat intelligence feed scores with which the binary is tagged. | [50, 100, 75] |
| watchlists | List of strings, each one identifying a watchlist that was matched with a binary. | [“x”, “a”] |
| event_timestamp | Event timestamp. | 1400695113.17 |
| hostname | Name of the host endpoint on which a binary was observed. | PANTHER |
| sensor_id | Sensor identifier of the endpoint on which a binary was observed. | 1 |

## feed.ingress.hit.binary

**Table 17:** feed.ingress.hit.binary Key Description and Example

| Key | Description | Example |
|---|---|---|
| md5 | MD5 hash value of a binary module that triggered a feed hit. | 44C0CBADFF00F3930B6A01EEAA4 05C6F |
| report_id | ID of the report that was matched. | report_01 |
| ioc_type | Type of the IOC that was matched. | dns |
| ioc_value | IOC value that was matched. | www.google.com |
| ioc_attr | Additional attributes on the IOC value that were matched. | {port:80, protocol:tcp} |
| hostname | Hostname of the computer on which the feed hit was detected. | PANTHER |
| sensor_id | Sensor ID of the endpoint. | 1 |

| Key | Description | Example |
|---|---|---|
| cb_version | Cb Response server version. | 5.0.0.140204.501 |
| server_name | Name of the Cb Response server | cbserver |
| feed_id | ID of the feed that was matched. | 15 |
| feed_name | Name of the feed that was matched. | mdl |
| event_timestamp | Time of the event. | 1400695113.17 |

## feed.storage.hit.binary

**Table 18:** feed.storage.hit.binary Key Description and Example

| Key | Description | Example |
|---|---|---|
| md5 | MD5 hash value of a binary module that triggered a feed hit. | 44C0CBADFF00F3930B6A01EEA A405C6F |
| report_id | ID of the report that was matched. | report_01 |
| ioc_type | Type of the IOC that was matched. | dns |
| ioc_value | IOC value that was matched. | www.google.com |
| ioc_attr | Additional attributes on the IOC value that were matched. | {port:80, protocol:tcp} |
| hostname | Name of the host on which the feed hit was detected. | PANTHER |
| sensor_id | Sensor ID of the endpoint. | 1 |
| cb_version | Cb Response server version. | 5.0.0.140204.501 |
| server_name | Name of the Cb Response server. | cbserver |
| feed_id | ID of the feed that was matched. | 15 |

| Key | Description | Example |
|---|---|---|
| feed_name | Name of the feed that was matched. | mdl |
| event_timestamp | Time of the event. | 1400695113.17 |
| copied_mod_len | Number of bytes collected. | 73544 |
| endpoint | Hostname and sensor ID of the endpoint on which the binary was first observed. | [PANTHER\|2] |
| group | First sensor group in which this binary was observed. | [Default Group] |
| digsig_issuer | If digitally signed, the issuer. | VeriSign Class 3 Code Signing 2010 CA |
| digsig_publisher | If digitally signed, the publisher. | Google Inc |
| digsig_result | If digitally signed, the result. Contains one of the following eight possible values:<br>• Signed<br>• Unsigned<br>• Bad Signature<br>• Invalid Signature<br>• Expired<br>• Invalid Chain<br>• Untrusted Root<br>• Explicit Distrust | Signed |
| digsig_result_code | Internal use. | 0 |
| digsig_sign_time | If digitally signed, the time of signing. | 2015-02-02T04:42:00Z |
| digsig_subject | If digitally signed, the subject. | Google Inc |
| is_executable_image | True if the binary is an EXE (versus DLL or SYS). | True |
| is_64bit | True if the architecture is x64. | True |
| md5 | MD5 hash value of a process, a parent process, a child process, a loaded module or a written file. | 44C0CBADFF00F3930B6A0<br>1EEAA405C6F |

| Key | Description | Example |
|---|---|---|
| observed_filename | Full path to the executable backing this process. | c:\program files(x86)\google\chrome\applicatio n\wow_helper.exe |
| orig_mod_len | Size, in bytes, of a binary at the time of collection. | 73544 |
| os_type | Operating system type of the host. | Windows |
| server_added_timesta mp | The time this binary was first seen by the server. | 2014-02-04T07:50:56.9 17Z |
| server_name | Name of the Cb Response server | cbserver |
| watchlist_<id> | For each watchlist that matched a binary, the timestamp of the match. | '2014-02-04T07:55:03. 007Z' |
| file_version | File version string from the class FileVersionInfo. | |
| product_name | Product name string from the class FileVersionInfo. | |
| company_name | Company name string from the class FileVersionInfo. | |
| internal_name | Internal name string from the class FileVersionInfo. | |
| original_filename | Original name string from the class FileVersionInfo. | |
| file_desc | File description string from the class FileVersionInfo. | |
| product_desc | Product description string from the class FileVersionInfo. | |
| comments | Comment string from the class FileVersionInfo. | |
| legal_copyright | Legal copyright string from the class FileVersionInfo. | |
| legal_trademark | Legal trademark string from the class FileVersionInfo. | |

| Key | Description | Example |
|---|---|---|
| private_build | Private build string from the class FileVersionInfo. | |
| special_build | Special build string from the class FileVersionInfo. | |
| product_version | Product name string from the class FileVersionInfo. | |

## feed.ingress.hit.process

**Table 19:** feed.ingress.hit.process Key Description and Example

| Key | Description | Example |
|---|---|---|
| process_id | Process doc identifier. | 00000064-0000-07f0-01d2-8e03fc88f25e |
| report_id | ID of the report that was matched. | report_01 |
| ioc_type | Type of the IOC that was matched. | dns |
| ioc_value | IOC value that was matched. | www.google.com |
| ioc_attr | Additional attributes on the IOC value that were matched. | {port:80, protocol:tcp, direction:'Outbound'} |
| hostname | Hostname of the computer on which the feed hit was detected. | PANTHER |
| sensor_id | Sensor ID of the endpoint. | 1 |
| cb_version | Cb Response server version. | 5.0.0.140204.501 |
| server_name | Name of the Cb Response server. | cbserver |
| feed_id | ID of the feed that was matched. | 15 |
| feed_name | Name of the feed that was matched. | mdl |
| event_timestamp | Time of the event. | 1400695113.17 |

## feed.query.hit.process

**Table 20:** feed.query.hit.process Key Descriptions and Examples

| Key | Description | Example |
|---|---|---|
| process_id | Process doc identifier. | 00000064-0000-07f0-01d2-8e03fc88f25e |
| segment_id | Process Solr doc segment identifier. | 1 |
| hostname | Hostname of the computer on which the feed hit was detected. | PANTHER |
| comms_ip | IP address from which Cb Response received the event (which could be a NAT or proxy address, if one is configured for the computer on which the process executed; otherwise will be the same as interface_ip). | |
| interface_ip | IP address of the computer on which the process executed. | |
| sensor_id | Sensor ID of the endpoint. | 1 |
| feed_id | ID of the feed that was matched. | 15 |
| feed_name | Name of the feed that was matched. | mdl |
| event_timestamp | Time of the event. | 1400695113.17 |
| start | | 2015-06-24T18:32:16.752Z |
| process_md5 | MD5 hash value of the executable backing this process. | 506708142bc63daba64f2d3ad1dcd5bf |
| process_name | Filename of the executable backing this process. | googleupdate.exe |
| path | Full path to the executable backing this process. | c:\program files(x86)\google\update\googleupdate.exe |
| last_update | Last activity in this process, in the computer's local time. | 2014-02-04T16:23:22.5 47Z |

## feed.storage.hit.process

**Table 21:** feed.storage.hit.process Key Description and Example

| Key | Description | Example |
|---|---|---|
| process_id | Process Solr doc identifier. | 00000064-0000-07f0-01d2-8e03fc88f25e |
| segment_id | Process Solr doc segment identifier. | 1488563344023 |
| report_id | ID of the report that was matched. | report_01 |
| ioc_type | Type of the IOC that was matched. | dns |
| ioc_value | IOC value that was matched. | www.google.com |
| ioc_attr | Additional attributes on the IOC value that were matched. | {port:80, protocol:tcp, direction:'Outbound'} |
| hostname | Hostname of the computer on which the feed hit was detected. | PANTHER |
| comms_ip | IP address from which Cb Response received the event (which could be a NAT or proxy address, if one is configured for the computer on which the process executed; otherwise will be the same as interface_ip). | 10.101.301.4 |
| interface_ip | IP address of the computer on which the process executed. | 10.101.301.4 |
| sensor_id | Sensor ID of the endpoint. | 1 |
| cb_version | Cb Response server version. | 5.0.0.140204.501 |
| server_name | Name of the Cb Response server. | cbserver |
| feed_id | ID of the feed that was matched. | 15 |
| feed_name | Name of the feed that was matched. | mdl |
| event_timestamp | Time of the event. | 1400695113.17 |

| Key | Description | Example |
|---|---|---|
| childproc_count | Total count of child processes that were created by this process. | 0 |
| cmdline | Process command line. | “c:\net.exe” /user |
| filemod_count | Total count of files that were modified by this process. | 0 |
| group | Sensor group to which this sensor was assigned at the time of process execution. | Default Group |
| host_type | Type of the computer: workstation, server, or domain controller. | server |
| last_update | Last activity in this process, in the computer’s local time. | 2014-02-04T16:23:22.5 47Z |
| modload_count | Total count of modules that were loaded by this process. | 45 |
| netconn_count | Total count of network connections made by this process. | 0 |
| os_type | Operating system type of the host. | Windows |
| parent_name | Name of the parent process. | svchost.exe |
| parent_md5 | MD5 hash value of the parent process. | 506708142bc63daba64f2d3ad1dcd 5bf |
| parent_pid | Parent process PID. | 2532 |
| parent_unique_id | Parent process unique ID. | 00000c42-0000-172c-01d0-5d6cca2adbb2-000000000001 |
| path | Full path to the executable backing this process. | c:\program files(x86)\google\update\googleupdate.exe |
| process_md5 | MD5 hash value of the executable backing this process. | 506708142bc63daba64f2d3ad1dcd 5bf |
| process_name | Filename of the executable backing this process. | googleupdate.exe |
| process_pid | Process PID. | 44988 |

| Key | Description | Example |
|---|---|---|
| regmod_count | Total count of registry modifications made by this process. | 0 |
| start | Start time of this process, in the computer's local time. | 2014-02-04T16:23:22.5 16Z |
| unique_id | Process unique ID. | 00000c42-0000-172c-01d0-5d6cca2adbb2-015A954A1297 |
| username | User context in which the process was executed. | SYSTEM |
| watchlist_id | Watchlist that matched (-1 is the internal syslog test). | -1 |
| watchlist_name | Name of the watchlist that matched. | SyslogTest |

## watchlist.hit.process

**Table 22:** watchlist.hit.process Key Description and Example

| Key | Description | Example |
|---|---|---|
| cb_version | Cb Response server version. | 5.0.0.140204.501 |
| childproc_count | Total count of child processes that were created by this process. | 0 |
| cmdline | Process command line. | "c:\net.exe" /user |
| filemod_count | Total count of files that were modified by this process. | 0 |
| group | Sensor group to which this sensor was assigned at the time of process execution. | Default Group |
| host_type | Type of the computer: workstation, server, or domain controller. | server |
| hostname | Hostname of the computer on which the process executed. | PANTHER |
| id | Internal use. | 7553512292948143354 |

| Key | Description | Example |
|---|---|---|
| last_update | Last activity in this process, in the computer's local time. | 2014-02-04T16:23:22.5 47Z |
| modload_count | Total count of modules that were loaded by this process. | 45 |
| netconn_count | Total count of network connections made by this process. | 0 |
| os_type | Operating system type of the host. | Windows |
| parent_unique_id | Parent process unique ID. | 00000c42-0000-172c-01d0-5d6cca2adbb2 |
| path | Full path to the executable backing this process. | c:\program files (x86)\google\update\googleupdate.exe |
| process_md5 | MD5 hash value of the executable backing this process. | 506708142bc63daba64f2d3ad1dcd5bf |
| parent_pid | Parent process PID. | 2532 |
| process_name | Filename of the executable backing this process. | googleupdate.exe |
| process_pid | Process PID. | 44988 |
| regmod_count | Total count of registry modifications made by this process. | 0 |
| segment_id | Internal use. | 1488563344023 |
| comms_ip | IP address from which Cb Response received the event (which could be a NAT or proxy address, if one is configured for the computer on which the process executed; otherwise will be the same as interface_ip). | 123.101.301.4 |
| interface_ip | IP address of the computer on which the process executed. | 10.432.123.9 |

| Key | Description | Example |
|---|---|---|
| sensor_id | The internal Cb Response sensor Global Unique Identifier (GUID) of the computer on which this process was executed. | 6 |
| server_name | Name of the Cb Response server. | cbserver |
| start | Start time of this process, in the computer's local time. | 2014-02-04T16:23:22.5 16Z |
| unique_id | Process unique ID. | 00000c42-0000-172c-01d0-5d6cca2adbb2-015A954A1297 |
| username | User context in which the process was executed. | SYSTEM |
| watchlist_id | Watchlist that matched (-1 is the internal syslog test). | -1 |
| watchlist_name | Name of the watchlist that matched. | SyslogTest |

## watchlist.hit.binary

**Table 23:** feed.storage.hit.binary Key Description and Example

| Key | Description | Example |
|---|---|---|
| cb_version | Cb Response server version. | 5.0.0.140204.501 |
| copied_mod_len | Number of bytes collected. | 73544 |
| endpoint | Hostname and sensor ID of the endpoint on which the binary was first observed. | [PANTHER\|2] |
| group | First sensor group in which this binary was observed. | [Default Group] |
| digsig_issuer | If digitally signed, the issuer. | VeriSign Class 3 Code Signing 2010 CA |
| digsig_publisher | If digitally signed, the publisher. | Google Inc |
| digsig_result | If digitally signed, the result. Contains one of the following eight possible values:<br>• Signed<br>• Unsigned<br>• Bad Signature<br>• Invalid Signature<br>• Expired<br>• Invalid Chain<br>• Untrusted Root<br>• Explicit Distrust | Signed |
| digsig_result_code | Internal use. | 0 |
| digsig_sign_time | If digitally signed, the time of signing. | 2015-02-02T04:42:00Z |
| digsig_subject | If digitally signed, the subject. | Google Inc |
| is_executable_image | True if the binary is an EXE (versus DLL or SYS). | True |
| is_64bit | True if architecture is x64. | True |

| Key | Description | Example |
|---|---|---|
| md5 | MD5 hash value of the process, the parent process, a child process, a loaded module, or a written file. | 44C0CBADFF00F3930B6A01EEA A405C6F |
| observed_filename | Full path to the executable backing this process. | c:\program files(x86)\google\chrome\applicatio n\wow_helper.exe |
| orig_mod_len | Size, in bytes, of binary at time of collection. | 73544 |
| os_type | Operating system type of the host. | Windows |
| server_added_timesta mp | The time that this binary was first seen by the server. | 2014-02-04T07:50:56.9 17Z |
| server_name | Name of Cb Response server. | cbserver |
| signed | Internal use. | Signed |
| timestamp | Time that the binary was seen. | 2014-02-04T07:50:56.9 17Z |
| watchlist_name | Name of the watchlist that matched this binary. | SyslogTest |
| watchlists | All watchlists that matched this binary. | [{‘wid’: ‘5’, ‘value’:‘2014-02-04T07:55:03. 007Z’}] |
| watchlist_<id> | For each watchlist that matched this binary, the timestamp of the match. | ‘2014-02-04T07:55:03. 007Z’ |
| file_version | File version string from the class FileVersionInfo. | |
| product_name | Product name string from the class FileVersionInfo. | |
| company_name | Company name string from the class FileVersionInfo. | |
| internal_name | Internal name string from the class FileVersionInfo. | |
| original_filename | Original name string from the class FileVersionInfo. | |
| file_desc | File description string from the class FileVersionInfo. | |

| Key | Description | Example |
|---|---|---|
| product_desc | Product description string from the class FileVersionInfo. | |
| comments | Comment string from the class FileVersionInfo. | |
| legal_copyright | Legal copyright string from the class FileVersionInfo. | |
| legal_trademark | Legal trademark string from the class FileVersionInfo. | |
| private_build | Private build string from the class FileVersionInfo. | |
| special_build | Special build string from the class FileVersionInfo. | |
| product_version | Product name string from the class FileVersionInfo. | |

## Syslog Common Event Format

The Common Event Format is an ArcSight standard that aligns the output format of various technology vendors into a common form.

Cb Response Watchlist Syslog output supports fully-templated formats, which enables easy modification of the template to match the CEF-defined format.

### Applying the Default CEF Templates

CEF syslog templates are located at `/usr/share/cb/syslog_templates`. To use them, add the following lines to `cb.conf`:

```
WatchlistSyslogTemplateProcess=/usr/share/cb/syslog_templates/
process_cef.txt
WatchlistSyslogTemplateBinary=/usr/share/cb/syslog_templates/
binary_cef.txt
```

The watchlist searcher process will automatically pick up the new template when the next watchlist hit occurs.

- The following is an example Process Watchlist hit in CEF format:

  ```
  CEF:0|Carbon Black|Carbon
  Black|4.1.0.131118.1540|reason=process_watchlist_-1|
  SyslogTest|10|dproc=wmiprvse.exe
  fname=c:\\windows\\system32\\wbem\\wmiprvse.exe
  start=2014-01-14T20:36:19.526Z dhost=J-8205A0C27A0C4
  msg=group:Default Group
  process_md5:0ffae66e6d5b1c87cbd22d1f3b6079fd last_update:2014-
  01-14T20:36:19.526Z
  guid:-5850106436655859636 segment_id:1488563344023
  ```

- The following is an example Binary Watchlist hit in CEF format:

  ```
  CEF:0|Carbon Black|Carbon
  Black|4.1.0.131118.1540|reason=binary_watchlist_-1|
  SyslogTest|10|start=2014-01-13T14:49:55.189Z
  msg=md5:6D778E0F95447E6546553EEEA709D03C
  desc:Windows Command Processor company_name:Microsoft
  Corporation
  product_name:MicrosoftÂ:registered: WindowsÂ:registered:
  Operating System
  product_version:5.1.2600.5512 file_version:5.1.2600.5512
  (xpsp.080413-2111)
  signed:Signed
  ```

## Extension Dictionary

The CEF specification is influenced by network device vendors and, to a lesser extent, host-based antivirus products. Products like Cb Response, with rich endpoint visibility, did not exist when the specification was developed and, as a result, the built-in key names supported by the extension dictionary do not map well to the data in Cb Response.

In the default template, the catch-all msg parameter is used for the fields that do not map well to the specified list of default keys. This limits required configuration and avoids the limitations of custom extensions.

If you would like to use custom extension keys, you can configure your SIEM device to support the custom keys and modify the Cb Response default CEF template as desired. Details are available in the CEF specification and in "Syslog Templates" on page 86. Contact your support representative with any questions.

Chapter 6

# Server VDI Support

This chapter describes Cb Response support for Virtual Desktop Infrastructure (VDI) and how to configure your machines to use it.

**Sections**

## Overview

Cb Response provides support for environments in which client machines are frequently reimaged or reverted back to a master image. In these environments, agent-based software can experience complex agent management issues such as duplicate systems or sensor id collisions. These issues are typical in environments where the sensors are installed on a master image or on Virtual Desktop Infrastructure (VDI) images, particularly when using non-persistent images.

Cb Response provides a means to resolve these VDI issues. If VDI behavior is configured and enabled for some or all sensors, the sensor communicates first with the Cb Response server (single or clustered), attempting to register itself. The server then tries to correlate that sensor's and client machine's characteristics (i.e., hostname and DNS name) to an existing sensor. If the server can correlate the new client to a sensor it has seen already, it assigns that sensor its previous Sensor ID. If there is no correlation, the server performs a new registration for that client. This allows the client machine to report to the server with the same Sensor ID, and so maintain the client event history, despite having been reimaged.

The process for setting up VDI support can be divided into two stages:

1. Configuring the Cb Response server to support the VDI behavior.
2. Determining the appropriate VDI support implementation option (global or sensor group).

## Configuring the Server for VDI Support

Before you configure VDI support, the Cb Response server must be at version 5.0 or later.

The process of configuring the server includes two tasks:

- Enabling VDI support in the `cb.conf` file.
- Deploying a VDI support plug-in that correlates a sensor to a Sensor ID.

### Enabling VDI Support

**To enable VDI support:**

1. Add the following configuration options to the `/etc/cb/cb.conf` file:

| | |
|---|---|
| ***Note*** | *If you are copying the following lines of code to paste directly into the file and a line break occurs or special characters appear, delete the line break and any special characters. Ensure that each line is added to the file as one continuous line.*<br>*This configuration is only required on the master server of a Cb Response cluster.* |

```
NewRegistrationCallbackModulePath=/usr/share/cb/plugins/
default_new_sensor_registration_callback.py

NewRegistrationCallbackClassName=DefaultNewRegistrationCallback
```

2. With root-level access, restart cb-enterprise:

   For a single Cb Response server:

   ```
   service cb-enterprise restart
   ```

   For clustered Cb Response servers:

   ```
   /usr/share/cb/cbcluster stop
   /usr/share/cb/cbcluster start
   ```

### Deploying a VDI Support Plug-in

When implementing VDI, you can either use the existing default plug-in or create your own. The default plug-in uses a sensor-provided client hostname and DNS name to correlate to an existing Sensor ID. You can create additional plug-ins that correlate the sensor to an existing Sensor ID based on other characteristics of the system, for example, a MAC address or an IP address.

The default plug-in provided with the Cb Response server is located here:

```
 /usr/share/cb/plugins/
default_new_sensor_registration_callback.py
```

## Specifying the Scope of VDI Support

VDI support can be implemented using one of two approaches:

- Global VDI support
- Sensor group VDI support

An integral part of implementing VDI support is the installation and configuration of Cb Response sensors. Each sensor collects data on running processes and binaries.

When installing a Cb Response sensor on a master image, it is recommended that you utilize Global VDI Support. While not required for sensor-group-based VDI support, the combination of the two solutions provides additional assurance that the master image will not cause any sensor conflicts.

A sensor collects data upon installation and its collection process can be optimized by clearing out two types of Cb Response directories: those storing binary or event log data. Clearing out these directories before the sensor becomes operational ensures that the sensor does not propagate a backlog of data from processes that ran while installing Cb Response to any or all of the images. Such a propagation can have adverse effects while deploying the image.

After stopping Cb Response sensor services on the client, clear the directories and files for the following types of data:

- Windows binary data
  - Directory: `%WINDIR%\CarbonBlack\store`
  - Sub-directories: `MD5_*`
- Windows event data
  - Directory: `%WINDIR%\CarbonBlack\EventLogs`
  - Files: `eventlog_*.log.zip` and `active-event.log`

- OSX binary data
  - Directory: `/var/lib/cb/store`
  - Files: `MD5_*`
- OSX event data
  - Directory: `/var/lib/cb`
  - Files: `event.log*`

Now that the directories have been cleared, you can configure either global or sensor group VDI support.

## Global VDI Support

With the global VDI option configured on the Cb Response sensor, the server will attempt to correlate all sensors registering with it to a previously registered sensor, regardless of the assigned sensor group. For this to occur, the client's sensor must be configured to initially register/check-in with a Sensor ID of 0. This setting is globally enabled on the server by default.

Before you can install a sensor on a master image, you must download a sensor package from the appropriate sensor group. When you install the sensor, ensure that the client is in "private mode" so that the sensor will not make any connections yet, such as checking in or registering with the Cb Response server.

Once installed, you can select the procedure for one of the following operating systems:

- Windows
- OSX

Note that VDI Behavior features are not supported for Linux at this time.

### Setting up Global VDI Support on Windows

**To setup global VDI support on Windows:**

1. Stop the Cb Response services on the server (only the master server if clustered).
   a. Open a command prompt with administrator privileges.
   b. Execute the following commands:
   ```
   sc stop carbonblack
   sc stop carbonblackk
   ```
2. Set the Sensor ID by setting the registry value `SensorId` in the registry key `/HKEY_LOCAL_MACHINE/SOFTWARE/CarbonBlack/config` to `0`.

| **Note** | *If the `SensorId` value does not already exist, create it as a `QWORD` value.* |
|---|---|

3. Save the image and deploy.

### Setting up Global VDI Support on OSX

**To setup global VDI support on OSX:**

1. Stop the Cb Response services on the server (only the master server if clustered).
   a. Open a terminal.
   b. Execute the following commands:
   ```
   sudo launchctl unload /Library/LaunchDaemons/
   com.carbonblack.daemon.plistSet
   ```
2. Set the Sensor ID by editing `/var/lib/cb/sensor.id` and replacing `current id` with *0*.
3. Save the image and deploy.

## Sensor Group VDI Support

With the sensor group VDI option, the server attempts to correlate only sensors that are in a VDI-enabled group. For this to occur, the desired sensor group VDI behavior setting must be enabled.

**To set up group-based VDI support:**

1. Login to the Cb Response console.
2. Configure a group for VDI support by navigating to **Sensors** in the left navigation menu.
3. From the **Sensors** menu in the top-left corner of the **Sensors** dialog, select the sensor group to configure for VDI support.
4. Click the **Edit Settings** tab.

   The **Edit Settings** page appears.
5. On the **Advanced** tab, select the **VDI Behavior Enabled** checkbox.
6. Click the **Save Changes** button to enable the configuration.

For more information about creating and editing sensor groups, see “Sensor Groups” in the *Cb Response User Guide*.