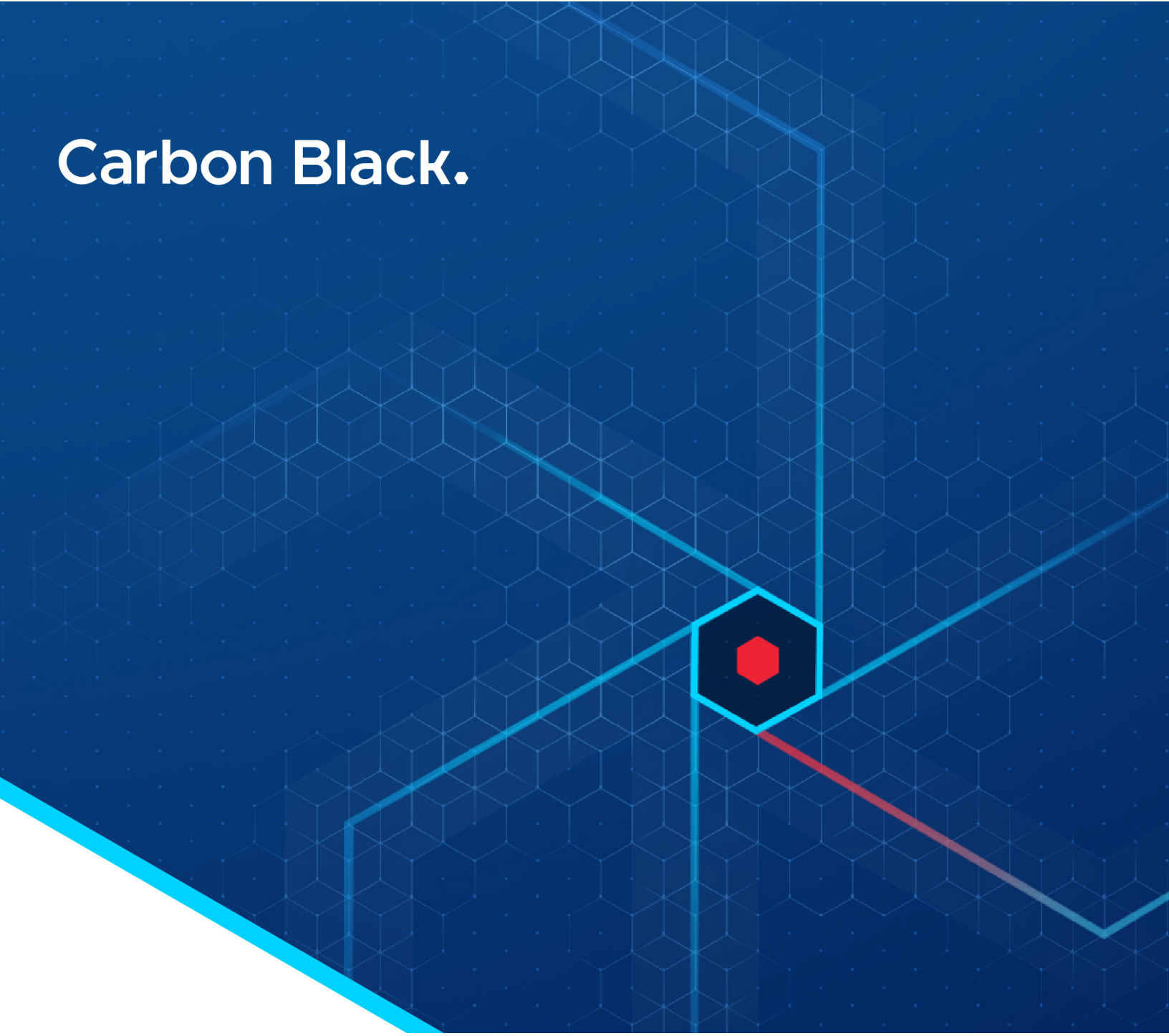


CB Response Integration Guide

Server Version: 6.5

Document Date: August 2019/April 2020

Copyrights and Notices

- Python sqlalchemy - Copyright (c) 2005–2014 Michael Bayer and contributors. SQLAlchemy is a trademark of Michael Bayer.
- Python sqlalchemy-migrate - Copyright (c) 2009 Evan Rosson, Jan Dittberner, Domen Kozar Python tempita - Copyright (c) 2008 Ian Bicking and Contributors
- Python urllib3 - Copyright (c) 2012 Andy McCurdy
- Python werkzeug - Copyright (c) 2013 by the Werkzeug Team, see AUTHORS for more details. QUnitJS - Copyright (c) 2013 jQuery Foundation, http://jquery.org/
- RabbitMQ - Copyright (c) 2007–2013 GoPivotal, Inc. All Rights Reserved. redis - Copyright (c) by Salvatore Sanfilippo and Pieter Noordhuis
- Simple Logging Facade for Java - Copyright (c) 2004–2013 QOS.ch Six - Copyright (c) 2010–2015 Benjamin Peterson
- Six - yum distribution - Copyright (c) 2010–2015 Benjamin Peterson
- Spymemcached / Java Memcached - Copyright (c) 2006–2009 Dustin Sallings and Copyright (c) 2009–2011 Couchbase, Inc.
- Supervisord - Supervisor is Copyright (c) 2006–2015 Agendaless Consulting and Contributors. Underscore - (c) 2009–2012 Jeremy Ashkenas, DocumentCloud Inc.
- Zlib - Copyright (c) 1995–2013 Jean-loup Gailly and Mark Adler

Permission is hereby granted, free of charge, to any person obtaining a copy of the above third-party software and associated documentation files (collectively, the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notices and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE LISTED ABOVE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

CB Response Integration Guide
Product Version: 6.5
Document Revision Date: April 30, 2020 11:39 am

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

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 identify providers (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* (this document) – Provides information for administrators who are responsible for integrating CB Response with various tools, such as CB Protection, EMET, VDI, SSO, and more.
- *CB Response Operating Environment Requirements (OER)* – Describes performance and scalability considerations in deploying a CB Response server. Note that this was called the *Server Sizing Guide* in previous releases.
- *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.
- *CB Response API* – Documentation for the CB Response REST API is located at https://developer.carbonblack.com/reference/enterprise-response. Documentation for the Python module that can be used for easy access to the REST API is hosted at https://cbapi.readthedocs.io.
- *CB Response Connectors* – Documentation describing how to install, configure and maintain various Carbon Black connectors is located at https://developer.carbonblack.com/guide/enterprise-response/#connectors. A connector enables communication between a third-party product and CB Response server.

Other CB Response documentation is referenced in the appropriate sections of this guide.

Contacting Technical Support

Carbon Black Technical Support offers several channels for resolving support questions:

Technical Support Contact Options
Carbon Black User eXchange: https://community.carbonblack.com
Email: support@carbonblack.com
Phone: 877.248.9098
Fax: 617.393.7499

Reporting Problems

When you call or email 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
Hardware configuration	Hardware configuration of the server or computer the product is running on (processor, memory, and RAM)
Document version	For documentation issues, specify the title, version and date of the manual you are using. The date and version of the document appear on the cover page, or for longer manuals, at the end of the Copyrights and Notices section.
Problem	Action causing the problem, error message returned, and any other appropriate output
Problem severity	Critical, serious, minor, or enhancement

Contents

List of Tasks

How to . . .

Chapter 3

Integrating CB Response with CB Protection

This chapter describes how to integrate CB Response with CB Protection. It also describes the features that are 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: Prior to CB Protection v8.0.0, the same product was known as the Bit9 Security platform. You may see the term “Bit9” used in this document where an integration involves a pre-8.0.0 version of 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 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 information from the Carbon Black Predictive Security Cloud, which provides reputation information, threat indicators, and attack classification intelligence. See “Threat Intelligence Feeds” in the *CB Response User Guide* for more information on the information that is available from the cloud.

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** – Updater rules allows seamless installation and upgrades of CB Response Mac and Linux 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 instructs agents to trust (by default) Windows files that are identified as being from the publisher “Carbon Black, Inc.”. This is the publisher for CB Response files. 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 **CB Response** 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 14. 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 and the tab it is on currently shows only for Windows agents.
- **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 “Integration Features in the CB Protection Console” on page 20 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 configuration 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.

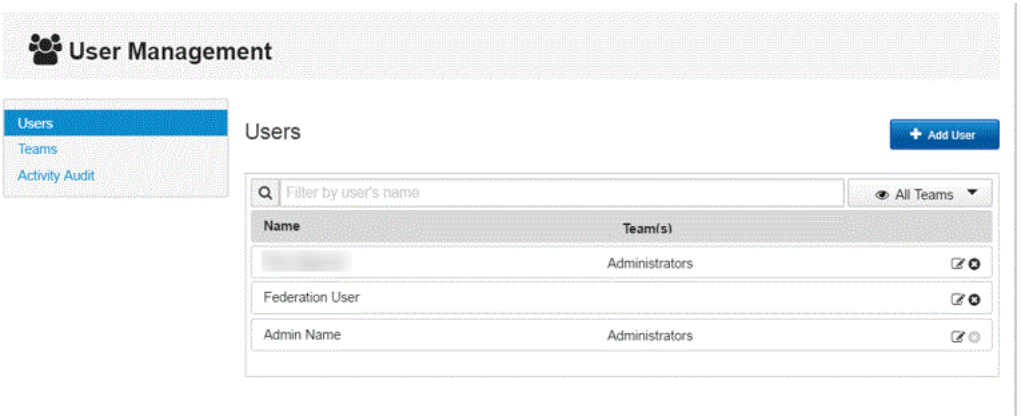
Creating a CB Response User for Integration

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

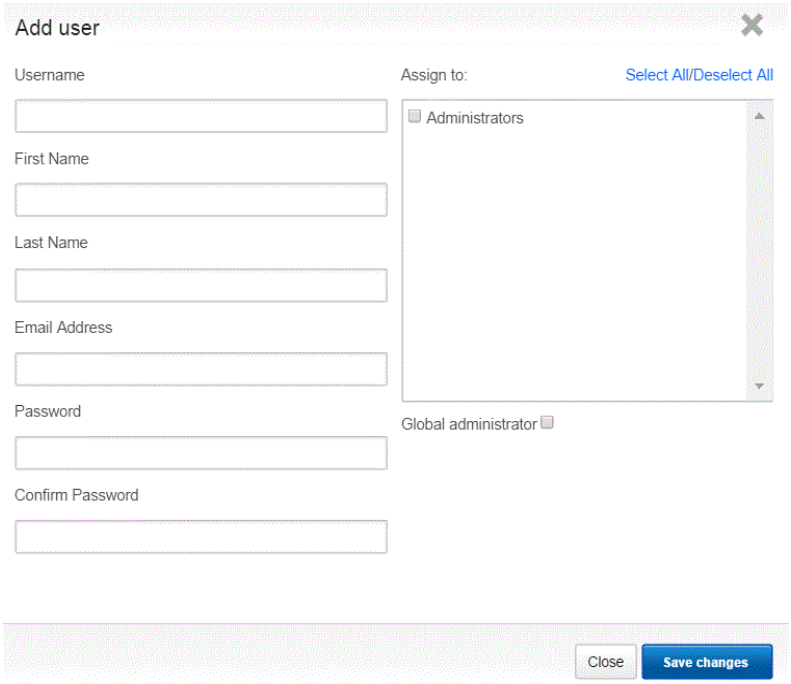
Note: The user account for the integration must be an Administrator with Global Administrator privileges.

To create a CB Protection Integration user and API Token:

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



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



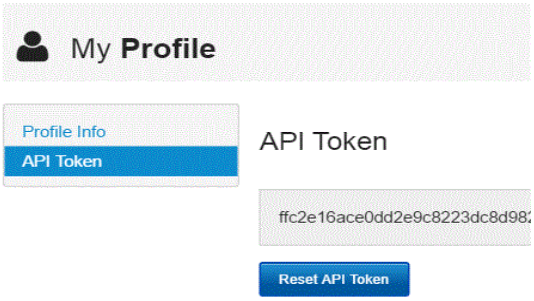
4. On the **Add user** dialog, define the user you will use for the integration:
   a. In the **Username** field, enter a meaningful username such as **CB Protection Integration** to depict the purpose of this account.
   b. In the **First Name** and **Last Name** fields, enter meaningful values. These 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, because it is not used as 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. 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, select **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.
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 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). 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. 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: • **Success, version:** <CB Response product version> • **Invalid API Token** • **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. **Important:** 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.

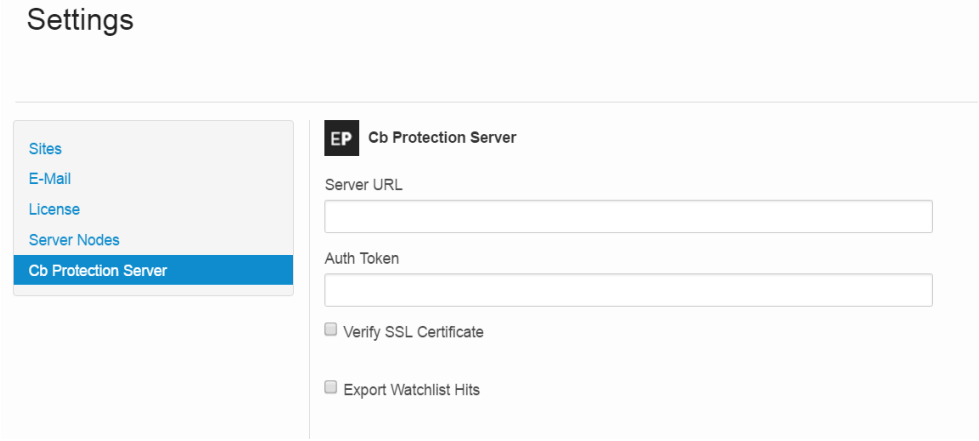
11. When you have entered and successfully tested the CB Protection server settings in the CB Protection console, click **Update** on the **System Configuration/Licensing** page. The configuration should be complete and your servers should be integrated.

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.



Notes

- 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.
- 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:
   - If you are running v7.2.3, choose **Administration > System Configuration**. and click the **Licensing** tab.
   - If you are running v8.0.0, click the Administration (gear) icon and choose **System Configuration** and click the **Licensing** tab.
2. In the **CB Response** 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.

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 they are available.

To view CB Protection-managed computers also running a CB Response sensor:

1. In the CB Protection console menu, select **Assets > Computers**. The **Computers** page appears.
2. On the **Saved Views** menu, select **Carbon Black Deployments** to see computers that are 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.

Click the **View Details** button for any computer in this view to see more CB Response sensor details on the **Computer Details** page. The **CB Response** panel on this page reports the presence, version, 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 *Not installed*. By default, the CB Protection server checks CB Response status every 30 minutes.

The **CB Response** 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 **CB Response Details** link in the **Related Views** menu to go to the CB Response console.

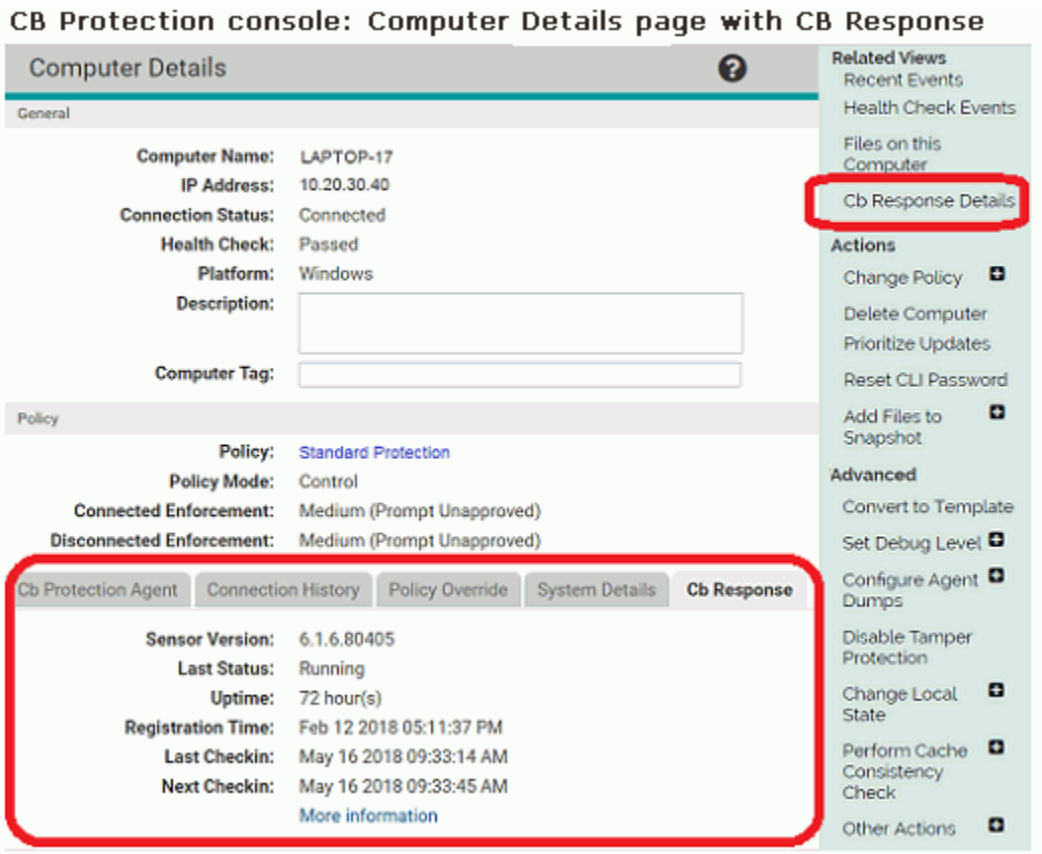


Table 2 describes the information available on the **CB Response** tab for **CB Protection Computer Details** pages. Note that this information currently shows only for Windows agents. OSX and Linux agents will report that CB Response information is "Unknown".

Table 2: CB Response tab fields on the CB Protection Computer Details page

Field	Description
Sensor Version	The version of the CB Response sensor installed on this computer.
Sensor Status **Last Status** **(on Details page)**	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. The possible values for CB Response status in the table are: • **Unknown** • **Installed, initializing** – sensor installed but not fully initialized • **Installed, running** • **Installed, not running** • **Not installed** • **Stopped** On the **Details** page, the **Last Status** field on the **CB Response** tab is similar to Sensor Status in the table. However, it does not appear if sensor status is **Unknown**. Its possible values are: • **Running** • **Service not running** • **Kernel not running** • **Stopped** **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.
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 that is 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. **Note:** You 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 view 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 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.

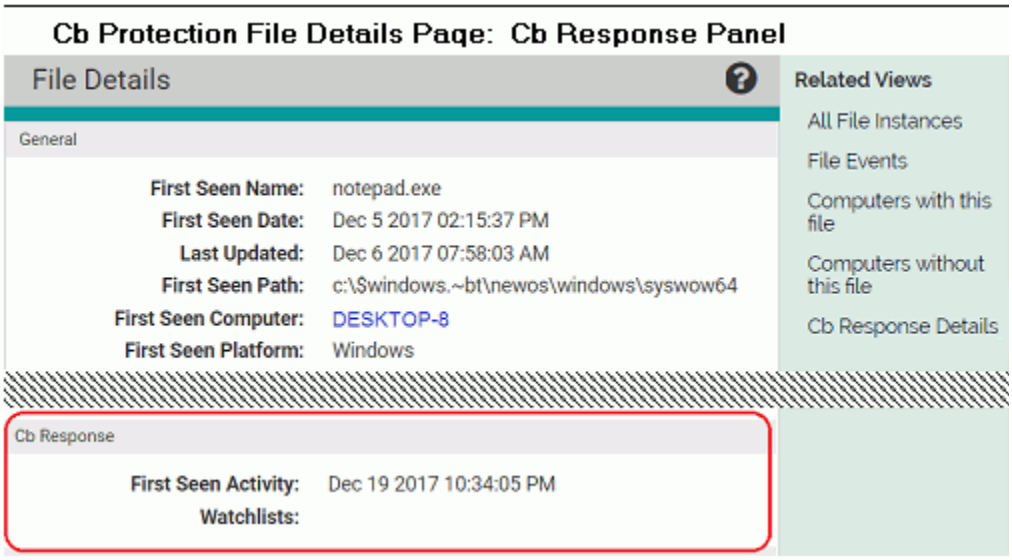


Table 3: CB Response fields on CB Protection File/File Instance Details pages

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.
Unique Paths	The number of unique paths in which this file has been seen. (only appears if data is available)
Network Connections	Whether there have been any network connections associated with this file, and if so, on how many computers. (only appears if data is available)
Registry Modifications	Whether there have been any registry modifications associated with this file, and if so, on how many computers. (only appears if data is available)
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 25.

Event Information

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

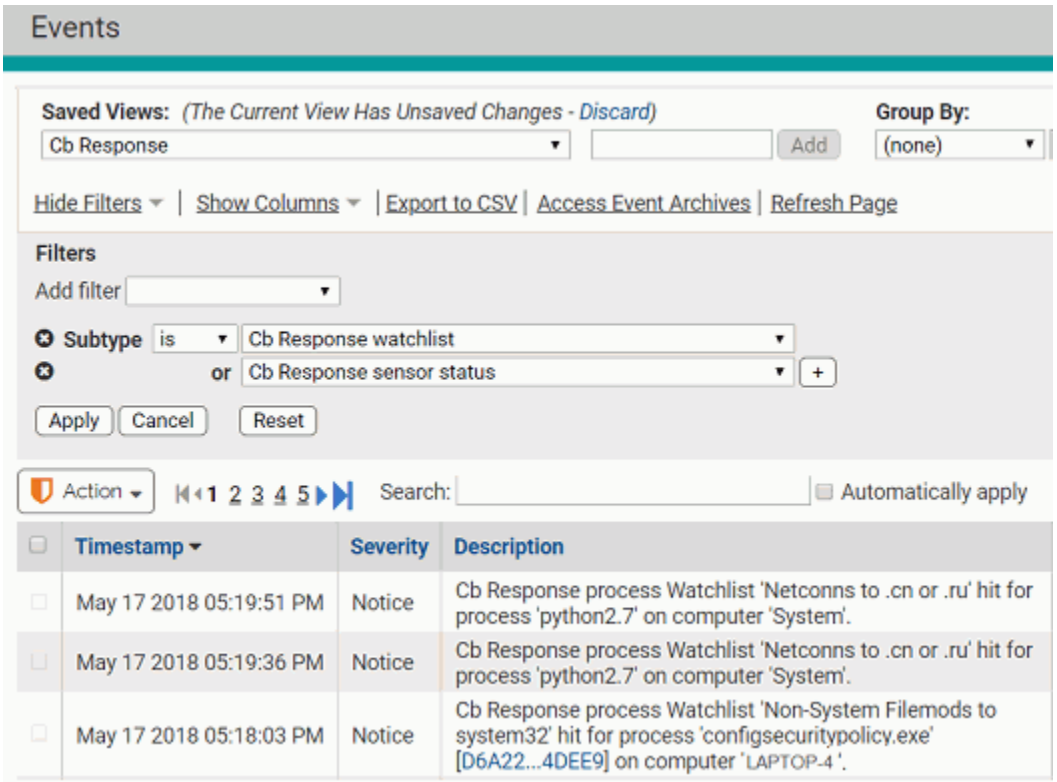
- **CB Response sensor status**
- **CB Response watchlist**

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 **CB Response**.

The following shows the CB Response view with filters displayed:



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 26 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 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.

In other cases, there may be a menu link on a page to “**CB Response 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 23 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 unique “Process Key" 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 CB Protection* guide for information about theses CB Protection features.

Chapter 2

Integrating EMET with CB Response

This chapter describes how to integrate a CB Response server with the Microsoft Enhanced Mitigation Experience Toolkit (EMET).

Important: Beginning with Windows 10 version 1709 (Fall Creators Update), also known as Redstone 3, Microsoft replaced EMET with Windows Defender Exploit Guard. Microsoft also announced that EMET reached end of life on July 31, 2018.

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.

Notes

- 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.
- EMET features and terminology are not detailed in this document. If you need more information about EMET, see the documentation provided by Microsoft.
- Proper functioning of this integration assumes EMET is installed and configured per Microsoft recommendations. EMET 5.x versions should be compatible.
- 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.
- Microsoft has announced that EMET will reach end of life on July 31, 2018 and be replaced by Windows Defender Exploit Guard, which replaced EMET beginning with Windows 10 Fall Creators Update (Redstone 3).

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.

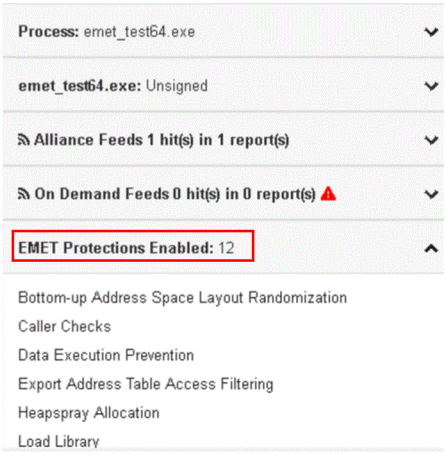
EMET Events in CB Response

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

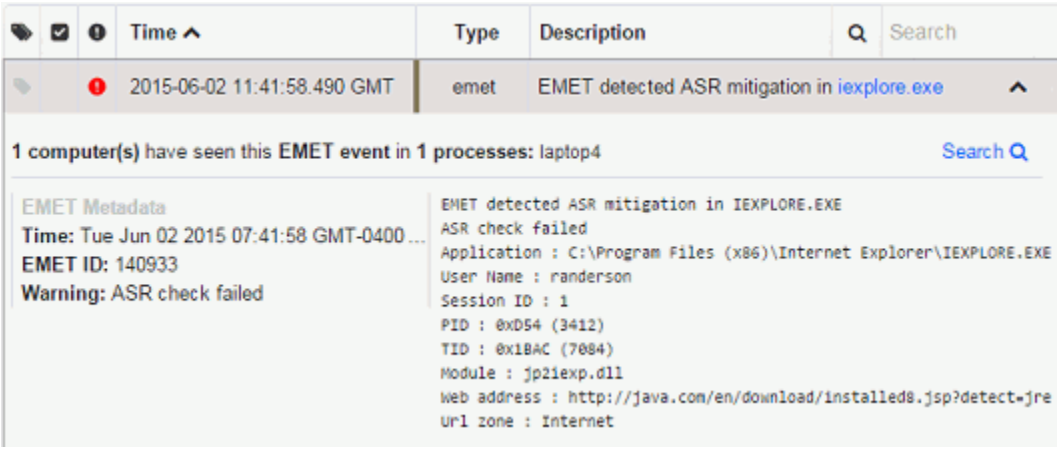
Process Search and Analysis for EMET Events

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 can appear. 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 that is being analyzed*. These will appear even if EMET did not perform any mitigations when an exploit was attempted on the process.



- **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.



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.*

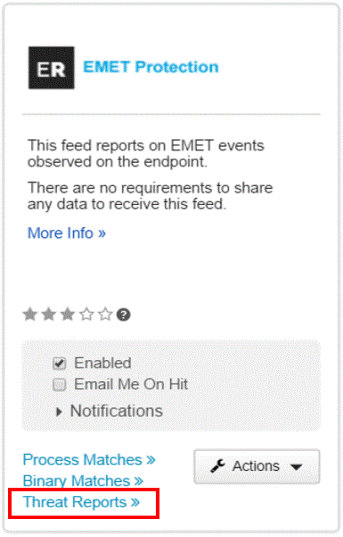
Keep in mind that this search will not only help you determine if EMET is installed on a host but will also help you determine process-specific protections.

EMET Events and Threat Reports

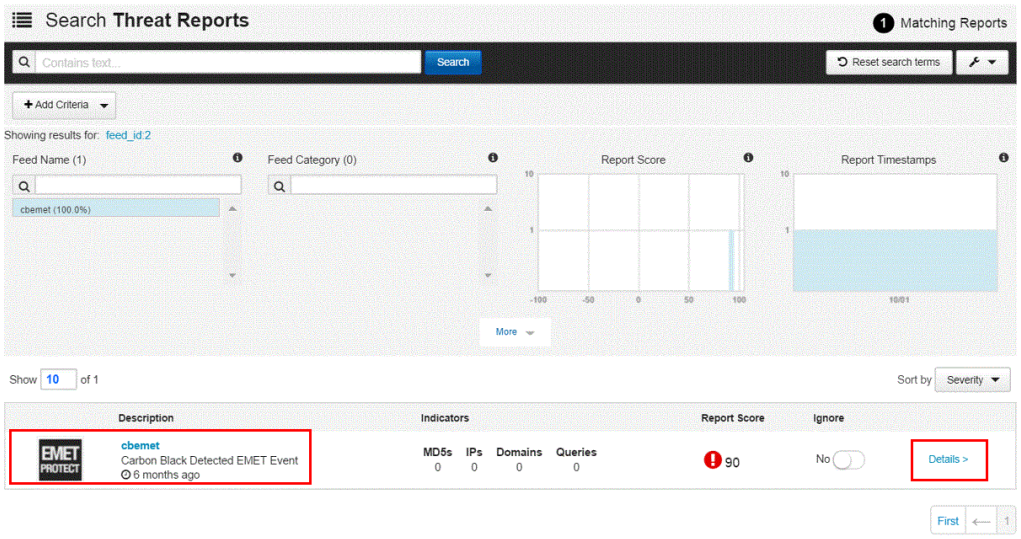
EMET threat reports are delivered as part of the EMET Protection Feed. If this feed is enabled, these reports are searchable and processes with an EMET mitigation event are tagged with “EMET Protection Feed” and get a score of 90. See “Threat Intelligence Feed Scores in *CB Response User Guide*.

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.



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*.

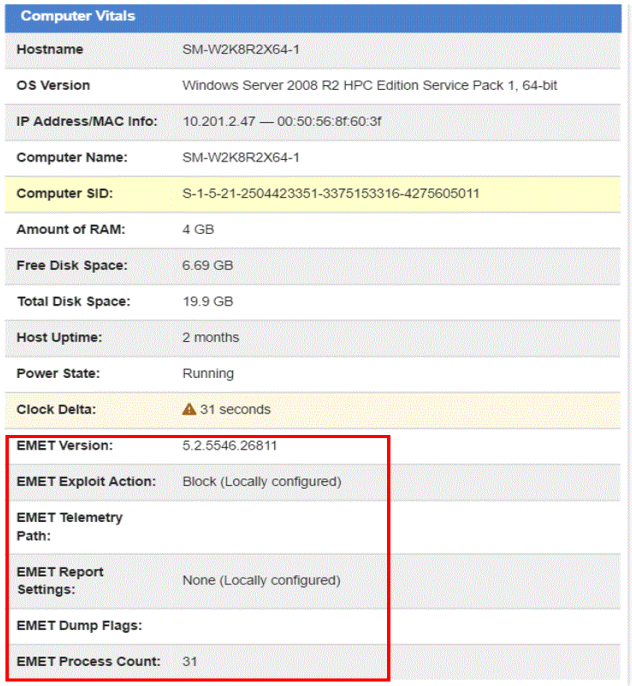
For more information on the **Threat Intelligence Feeds** page, see "Threat Intelligence Feeds" in the *CB Response User Guide*.

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

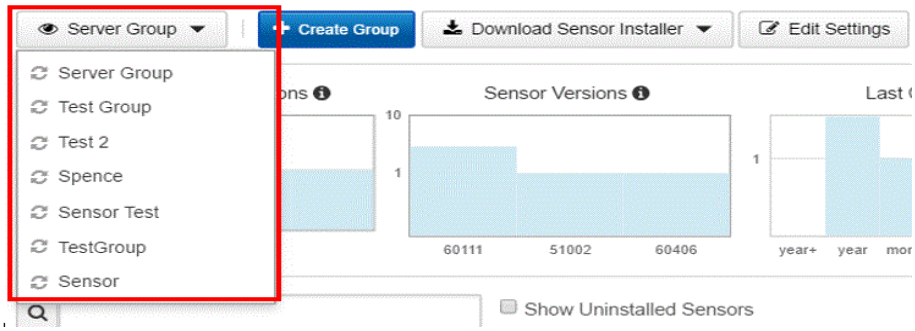
Field	Description
EMET Version	If the Microsoft Enhanced Mitigation Experience Toolkit (EMET) is installed on the endpoint, the toolkit version number.
EMET Exploit Action	The EMET configuration for what to do when an exploit attempt occurs: • Audit -- Do not kill the process, when applicable, but log the exploitation attempt. • Block -- Terminate the program when an exploitation attempt is detected (“Stop” in the EMET interface)
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: • Windows Event Log • Tray Icon • Early Warning In addition, for any active option, there will also be an indication of whether the choice is Locally or GPO configured. **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

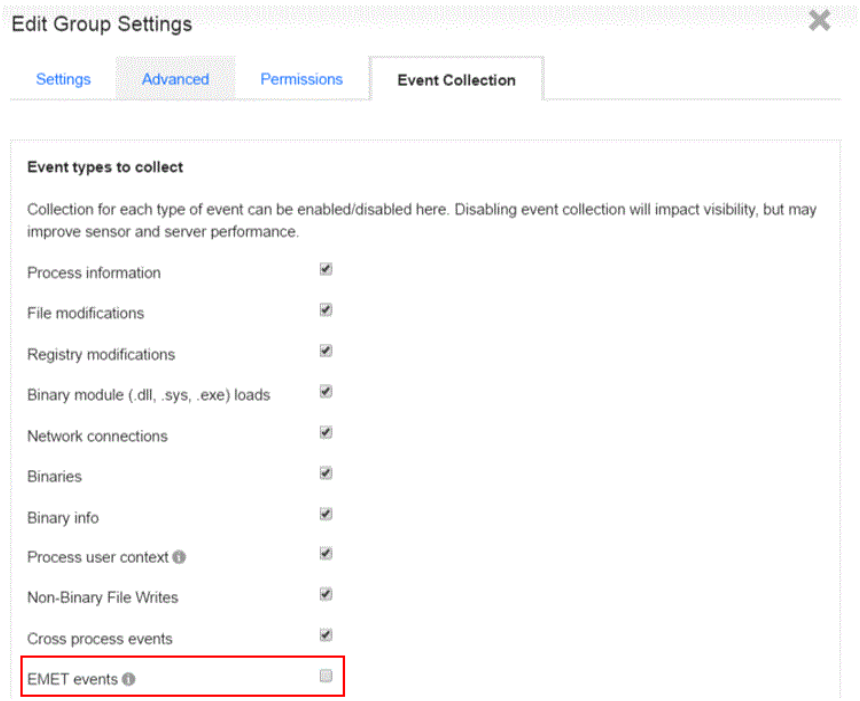
By default, sensors on any host with EMET installed will include EMET events in the logs reported back to their CB Response server. However, you can disable EMET event reporting for all sensors in a Sensor Group.

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.



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 CB Response with the OKTA, Shibboleth, and ADFS IdPs.

Sections

Overview

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

- Users can maintain a single set of credentials for a variety of services.
- Users need not re-authenticate when switching from one system to another since their initial login.
- Context is remembered after the initial login.

CB Response 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 external SAML 2.0-compliant identity providers (IdPs).

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. It comes with a certain level of complexity.

The SAML 2.0 specification is described in 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 that is 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 the following SSO IdPs:

- OKTA
- Shibboleth
- ADFS IdP

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

SAML 2.0 Single Sign-On Setup for CB Response On-Premise

Before establishing a trust relationship between a SAML service provider and an IdP, the two services must have well-established, cryptographically secure identities. This identity information must be exchanged so that the service provider (SP) knows who its IdP 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 On-Premise (acting as the SAML service provider) is `/etc/cb/cert/cb-server.[crt,key]`. This identity is also used for server-sensor authentication and for web user interface HTTPS server authentication.

You can also configure CB Response On-Premise to use a separate set of certificate/key files for SSO by updating configuration fields in the SSO configuration file.

Attribute Mapping

With CB Response (version 5.2 and later), when SSO is configured, users who are authenticated by the configured IdP can be automatically provisioned in CB Response if they are authorized and if the IdP returns 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 vary between IdPs. For CB Response On-Premise, the `attr_map.py` script maps the attributes that are returned by an IdP to the fields that are required by CB Response.

CB Response supports the following seven attributes:

Table 5: Supported Attributes

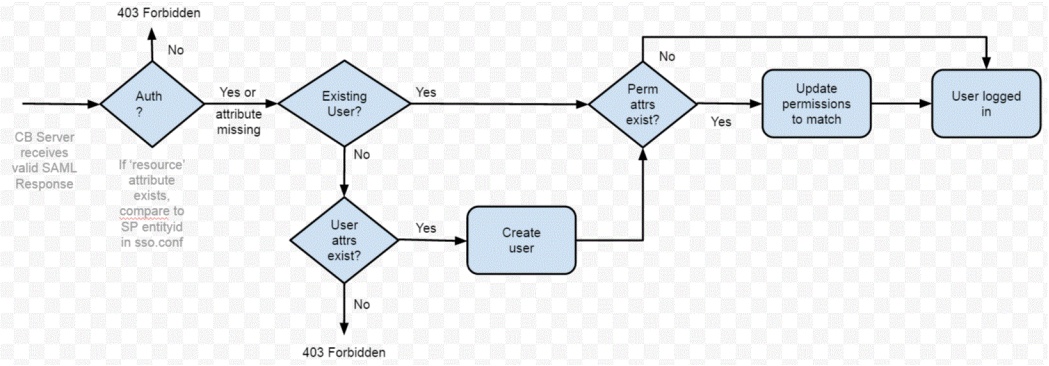
Attribute	Required	Use
username	Required	The username to log in as.
authorized	Optional (but strongly recommended)	Required to authorize access. Boolean. "True" if this user is authorized access to the CB Response server.
first_name	Optional	Required to provision a new user.
last_name	Optional	Required to provision a new user.
email	Optional	Required to provision a new user.
builtin_roles	Optional	Required to set permissions. A list of roles. Valid roles: "GlobalAdmin" If None, this field is ignored. If an empty list ([]), all roles are removed.

Table 5: Supported Attributes

Attribute	Required	Use
teams	Optional	Required to set permissions. A list of teams this in which the user is a member. Names must match an existing configured team. If None, this field is ignored. If an empty list ([]), all team associations are removed.

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

The following diagram describes the attribute mapping process:



Example Attribute Mapping Script

This section provides an example attribute mapping script. It pertains to CB Response On-Premise only; this section does not apply to CB Response Cloud.

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 defined values that are returned by the 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/` together with 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"] = None
    else:
        result["builtin_roles"] = []
        result["teams"] = None

    return result
```

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

- **username** – The user's login ID.
- **givenName** – The user's first name.
- **sn** – The user's last name (surname).
- **mail** – The user'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 Response server. Any user part of **cbserver-owners** is granted global admin and is included in the administrators group.

The following is example debug output of a user being authenticated, authorized, created, added to global admins and the administrators team.

```
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,
```

Integrate OKTA IdP with CB Response On-Premise

This section describes how to integrate the OKTA IdP with CB Response On-Premise.

To integrate the OKTA IdP with CB Response On-Premise:

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`.
   c. 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. Review and edit the `/etc/cb/sso/sso.conf` file as described here.

   Caution: 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.

   Also, 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 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 that was previously created. This single property defines whether CB Response server will be started in standalone vs. federated authentication mode.

   Note: To deactivate SSO integration, 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. Give the XML file to the IdP to complete the trust.

7. Restart CB Response server by issuing this command:

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

Integrate Shibboleth IdP with CB Response On-Premise

This section describes how to integrate the Shibboleth IdP with CB Response On-Premise.

To integrate the Shibboleth IdP with CB Response On-Premise:

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:

   Caution: The syntax of this configuration file must fully conform to the JSON data-interchange format. Failure to do so can 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 for 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. After this file is created, you must give it to the identity provider to complete the trust.
7. Restart the CB Response server by issuing the following command:

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

Integrate ADFS IdP with CB Response On-Premise

This section describes how to integrate the ADFS IdP with CB Response On-Premise. To configure ADFS, see your Microsoft documentation.

To integrate the ADFS IdP with CB Response On-Premise:

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 that are returned from ADFS. Each configurable property is accompanied with inline documentation in the `attr_map.py` file to assist with this process.
3. In the `/etc/cb/sso/sso.conf` file:

   Caution: 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 that 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>"
      ]
    },
```

   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 ADFS IdP. For example:

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

e. 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"
}
```

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: 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. Give the file to the IdP to complete the trust.

7. Restart CB Response server by issuing this command:

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

Troubleshoot SSO Integration with CB Response On-Premise

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.blueprints.api_routes_saml` and `saml2` modules.

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

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

   `[loggers]` section example:

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

   ```
   [logger_cb.flask.blueprints.api_routes_saml]
   level=DEBUG
   handlers=debug_syslog
   qualname=cb.flask.blueprints.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.

   This setting is related to the field specified in the `/usr/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 52.

   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. 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`

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 52):

```
# 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 *CB 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 of Logging

CB Response logs the following types of events to syslog:

- **Notification logs** – for watchlist and feed hits, and binary information events
- **Audit logs** – for banning, isolation, and CBLR sessions

  With audit logging enabled (as described in “Audit Logs” on page 56), audit logs include all user API activity, including HTTP request details.

See the *CB Response Server/Cluster Management Guide* for information about all CB Response server logs.

Notification Logs

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 the “Watchlists” chapter in the *CB Response User Guide* 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 the “Threat Intelligence Feeds” chapter in the *CB Response User Guide* for more information.

  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.
- **Binary information event** – This event occurs when a process execution adds a binary to the CB Response database.

The program name prefix for logged events is `cb-notifications-`. By default, logged 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`).

> **Note**
>
> Currently, the binary information events are not published in the `cb-all-notifications.log` file.

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.

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
```

Audit Logs

CB Response audit logs are located in `/var/log/cb/audit`, and are available through syslog. They include the following files:

- `banning.log` – hash banning activity (add, remove, toggle)
- `isolation.log` – sensor isolation activity (enable and disable)
- `live-response.log` – endpoint CBLR session activity such as `cd`, `exec`, `reg`, and so on; does not include commands issued in CBLR that are not part of an endpoint session, such as `sensor` and `help`.
- `useractivity.log` (audit logging enabled only) – all user API activity, including HTTP request details

  You can enable audit logging with the `EnableExtendedApiAuditLogging=True` parameter in `/etc/cb/cb.conf`. See the *CB Response Configuration Guide (cb.conf)* for details.

Syslog Format

Each watchlist and feed hit consists of a series of key-value pairs. This section provides the following information about the different key-value pairs for watchlist and feed hits:

- Examples of key-value pairs in the different types of hit
- The default process template used to create the key-value pairs for the hit
- Descriptions of the key-value pairs for the different types of hit

Note

To improve readability in the examples, line breaks appear between fields In the template files, however, fields are separated by spaces.

Watchlist Hit on Process

The following describes the process watchlist hit.

Process Watchlist – Example

```
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_sha256=’A31C76B026966D8A01BB929402B17A5C0B35037CF53908770D
7F3D1F28F6A7BD’
process_name='chrome.exe' process_path='c:\program files
(x86)\google\chrome\application\chrome.exe' last_update='2014-11-
13T13:49:39.361Z'
```

Process Watchlist – Default Template

```
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_sha256='{{doc['process_sha256']}}'
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 %}
```

Process Watchlist – 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: • event (process events) • module (binary modules) • 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 hardcoded 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.

Syslog Label	Solr Doc Reference	Description
process_md5	process_md5	MD5 hash value of the executable backing this process.
process_sha256	**process_sha256**	SHA-256 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_* 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.

Watchlist Hit on Binary

The following describes the binary watchlist hit.

Binary Watchlist – Example

```
Mar 13 03:50:19 [1037] <warning>  reason=watchlist.hit type=module
md5=7931ADC31F0180855E05D6666630B3A3 host=WORKSTATION-2'
sha256=F6F9A4834AFE57EBC0B77CF1E83F0C24B298C2FCBBF214CF6CC4DBD24E8
BFB4B
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'
```

Binary Watchlist – Default Template

```
reason=watchlist.hit type=module
md5={{doc["md5"]}}
sha256={{doc["sha256"]}}
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 %}
```

Binary Watchlist – 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.
sha256	sha256	SHA-256 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.

Syslog Label	Solr Doc Reference	Description
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.
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.

Feed Hit on Process Ingress

The following describes the process ingress feed hit.

Process Ingress – Example

```
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'
```

Process Ingress – Default Template

```
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']}}'
```

Process Ingress – 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.

Syslog Label	Solr Doc Reference	Description
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.
timestamp	event_timestamp	Epoch time (seconds since 00:00:00 on 1 January 1970) of the feed hit event.

Feed Hit on Process Storage

The following describes the process storage feed hit.

Process Storage Feed Hit – Example

```
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_sha256='7AFB56DD48565C3C9804F683C80EF47E5333F847F2D3211EC1
1ED13AD36061E1'
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'
```

Process Storage Feed Hit – Default Template

```
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_sha256=’{{doc['process_sha256']}}'
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 %}
```

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.

Syslog Label	Solr Doc Reference	Description
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	_* 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.
timestamp	event_timestamp	Epoch time of the feed hit event.
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_sha256	process_sha256	SHA-256 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.

Feed Hit on Binary Ingress

The following describes the binary ingress feed hit.Binary Ingress Feed Hit – Example

```
Aug 12 14:06:39 [26070] <warning> reason=feed.ingress.hit
type=module md5=B84E2D174DC84916A536572BB8F691A8
sha256=A23B2D174DC84916A539872CD8F775A8B84F1E234DC84916A564738DC4E
A6B76 host='SERV12R2X64-01' sensor_id=1 feed_id=2
feed_name='srstrust' ioc_type='md5'
ioc_value='b84e2d174dc84916a536572bb8f691a8'
timestamp='1407866781.79'
```

Binary Ingress Feed Hit – Default Template

```
reason=feed.ingress.hit type=module
md5={{doc['md5']}}
sha256={{doc['sha256']}}
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']}}'
```

Binary 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 example: • event (process events) • module (binary modules) • 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.
sha256	sha256	SHA-256 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.

Syslog Label	Solr Doc Reference	Description
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	_* 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.
timestamp	event_timestamp	Epoch time of the feed hit event.

Feed Hit on Binary Storage

The following describes the binary storage feed hit.

Binary Storage Feed Hit – Example

```
Aug 12 14:06:39 [26070] <warning> reason=feed.storage.hit
type=module md5=B84E2D174DC84916A536572BB8F691A8
sha256=7AFB56DD48565C3C9804F683C80EF47E5333F847F2D3211EC11ED13AD36
061E1 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'
```

Binary Storage Feed Hit – Default Template

```
reason=feed.storage.hit type=module
md5={{doc['md5']}}
sha256={{doc['sha256']}}
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 %}
```

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.

Syslog Label	Solr Doc Reference	Description
sha256	sha256	SHA-256 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.
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	_* 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.
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.
product_name	product_name	Product name string from the class FileVersionInfo.

Syslog Label	Solr Doc Reference	Description
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.

Feed Hit on Host Ingress

Host Ingress Feed Hit – Example

```
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'
```

Host Ingress Feed Hit – Default Template

```
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']}}'
```

Host Ingress Feed Hit – Key-Value Pairs

Key-value pairs for host ingress feed hits are a subset of those for binary ingress feed hits. See “Binary Ingress Feed Hit – Key-Value Pairs” on page 66 for descriptions.

Feed Hit on Process Query

Process Query Feed Hit – Example

```
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_sha256='a76b4c204d7e28f0e4dcbb6abc910dC3e7f820416ed744874c
ba74849067b71'
process_name='cbquery' process_path='/usr/share/cb/cbquery'
last_update='2015-06-24T18:32:17.345Z'
```

Process Query Feed Hit – Default Template

```
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_sha256='{{doc['process_sha256']}}'
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 %}
```

Process Query Feed Hit – Key-Value Pairs

Key-value pairs for process query feed hits are a subset of those for process storage feed hits. See “Process Storage Feed Hit – Key-Value Pairs” on page 64 for descriptions.

Feed Hit on Binary Query

Binary Query Feed Hit – Example

```
2015-06-24 18:30:14 [13031] <warning>  reason=feed.query.hit
type=module md5=6D4B29FB9307FBE8781E42B7CFDA4CE1
sha256=F6E9D4834CBA57BCD0E77FE1D83C0B24A298B2CDEEF214ED6CC4BAB24C8
DEF4E
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'
```

Binary Query Feed Hit – Default Template

```
reason=feed.query.hit type=module
md5={{doc["md5"]}}
sha256={{doc["sha256"]}}
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 %}
```

Binary Query Feed Hit – Key-Value Pairs

Key-value pairs for binary query feed hits are a subset of those for binary storage feed hits. See “Binary Storage Feed Hit – Key-Value Pairs” on page 68 for descriptions.

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 the “Advanced Search Queries” chapter in the CB Response User Guide for specific fields to use when creating queries, and to the “Watchlists” chapter in the same guide 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 for CB Response syslog integration:

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 Response IP address to be authorized prior to accepting data.

Setting Up Server Data Transmission

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

To set up the CB Response server to send syslog data to remote devices:

1. Access the CB Response server either through the console or with a remote terminal connection using SSH.
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:

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

CB Response stores all logged information in the following directory:

`/var/log/cb/`

When you troubleshoot any server-side activity, start with the logs in this directory.

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 its 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`

Note, however, that Binary Information events are not published in the `cb-all-notifications.log` file.

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 interface options are as follows:

Syslog notification testing utility (cbsyslog) options

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. **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
process_sha256=6635a659bc80def44859f36719ed30618589c4b50abc17de
f38ee7dd913721 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 syslog 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

Key	Description	Example
md5	MD5 hash value of the observed binary module.	44C0CBADFF00F3930B6A01EEAA405C6F
sha256	SHA-256 hash value of the observed binary module.	12B0DCFFEE00FA405C6F3930B6A01EEAA405C6F44C0CBADFF00F3930B6A01EEA
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

Key	Description	Example
md5	MD5 hash value of the observed binary module.	44C0CBADFF00F3930B6A01EEAA405C6F
sha256	SHA-256 hash value of the observed binary module.	1123A659BC80DEF22859F36719ED30618589C4B50ABC17DEF38EE7DDB913721
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

Key	Description	Example
md5	MD5 hash value of the observed binary module.	44C0CBADFF00F3930B6A0 1EEAA405C6F
sha256	SHA-256 hash value of the observed binary module.	1123A659BC80DEF22859F367 19ED30618589C4B50ABC17D EF38EE7DDB913721
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

Key	Description	Example
md5	MD5 hash value of a binary module that triggered a feed hit.	44C0CBADFF00F3930B6A01E EAA405C6F
sha256	SHA-256 hash value of a binary module that triggered a feed hit.	1123A659BC80DEF22859F367 19ED30618589C4B50ABC17D EF38EE7DDB913721
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
cb_version	CB Response server version.	5.0.0.140204.501

Key	Description	Example
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

Key	Description	Example
md5	MD5 hash value of a binary module that triggered a feed hit.	44C0CBADFF00F3930B6A01EEAA405C6F
sha256	SHA-256 hash value of a binary module that triggered a feed hit.	1123A659BC80DEF22859F36719ED30618589C4B50ABC17DEF38EE7DDB913721
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
feed_name	Name of the feed that was matched.	mdl
event_timestamp	Time of the event.	1400695113.17

Key	Description	Example
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: • Signed • Unsigned • Bad Signature • Invalid Signature • Expired • Invalid Chain • Untrusted Root • 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 1EEAA405C6F

Key	Description	Example
sha256	SHA-256 hash value of a process, a parent process, a child process, a loaded module or a written file.	1EEAA405C6F44C0CBADFF00F3930B6A044C0CBADFF00F3930B6A01EEAA405C6F
observed_filename	Full path to the executable backing this process.	c:\program files(x86)\google\chrome\application\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_timestamp	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.	

Key	Description	Example
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.	

feed.ingress.hit.process

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

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_sha256	SHA-256 hash value of the executable backing this process.	2bc63daba64f2d3ad1dcd5bf506708142bc63daba64f2d3ad1dcd5bf50670814
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

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.	506708142bc63daba64f2d3ad1dcd5bf
parent_sha256	SHA-256 hash value of the parent process.	1123a659bc80def22859f36719ed30618589c4b50abc17def38ff7eed913721
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\google update.exe
process_md5	MD5 hash value of the executable backing this process.	506708142bc63daba64f2d3ad1dcd5bf

Key	Description	Example
process_sha256	SHA-256 hash value of the executable backing this process.	1123a659bc80def22859f36719ed30618589c4b50abc17def38ff7eed913721
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
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

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

Key	Description	Example
id	Internal use.	7553512292948143354
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
process_sha256	SHA-256 hash value of the executable backing this process.	1123a659bc80def22859f36719ed30618589c4b50abc17def38ff7eed913721
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

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

Key	Description	Example
digsig_result	If digitally signed, the result. Contains one of the following eight possible values: • Signed • Unsigned • Bad Signature • Invalid Signature • Expired • Invalid Chain • Untrusted Root • 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
md5	MD5 hash value of the process, the parent process, a child process, a loaded module, or a written file.	44C0CBADFF00F3930B6A01EEAA405C6F
sha256	SHA-256 hash value of the process, parent process, a child process, a loaded module, or a written file	1123a659bc80def22859f36719ed30618589c4b50abc17def38ff7eed913721
observed_filename	Full path to the executable backing this process.	c:\program files(x86)\google\chrome\application\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_timestamp	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

Key	Description	Example
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.	
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, enabling 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 `/etc/cb/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 78. 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 re-imaged 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, maintaining the client event history despite having been re-imaged. If the endpoint's System ID (SID) changes, Carbon Black must create a new Sensor ID for that endpoint.

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. Choosing 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. If you have a CB Response cluster, this configuration change is required on the master and on all minions.

   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.

   ```
   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 endpoint.
   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 endpoint.
   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*.