vmware

Carbon Black
EDR

Process :cmd.exe
PID :1672
OS Type: :windows
Path: c:\windows\system32\cmd.exe
Username: BIT9SEAD\admin
MD5: ad7b9c14083b52bc532fba5948342b98
Start Time: 2017-07-17T19:20:34.682Z
Interace IP: 192.168.230.5
Server Comms IP: 192.168.230.5
eventguide...
VT 1
notepad.exe
VT 1
user.exe
cmd.exe
cmd.exe
powersh
powersh

VMware Carbon Black EDR Server Configuration Guide

Server Version: 7.4

Document Date: January 2021

Copyrights and Notices

VMware Carbon Black EDR Server Configuration Guide
Document Revision Date: December 1, 2020
Product Version: 7.4

VMware Carbon Black
1100 Winter Street, Waltham, MA 02451 USA
Tel: 617.393.7400
Fax: 617.393.7499
VMware Carbon Black Web Site: http://www.carbonblack.com
VMware Carbon Black User eXchange (user community): https://community.carbonblack.com
Support E-mail: support@carbonblack.com

Chapter 1

Introduction

This document describes the contents of the `cb.conf` file, the primary configuration file for VMware Carbon Black EDR. By changing the values of parameters in `cb.conf`, you can change the behavior and performance of Carbon Black EDR.

Before editing `cb.conf`, you should be sufficiently familiar with the features and operation of Carbon Black EDR to make judgments about whether and when to change its configuration. For details about using Carbon Black EDR, see the *VMware Carbon Black EDR User Guide*.

Sections

Overview

The primary configuration file for the Carbon Black EDR server is:

`/etc/cb/cb.conf`

The first time you install the Carbon Black EDR server, running `cbinit` creates the `cb.conf` file from a template that includes the standard parameters and default settings.

In a many environments, there might be no need to modify `cb.conf` directly — many configuration options are either set during installation or through the Carbon Black EDR console. Configuration options described in this document, however, can be useful for troubleshooting issues with the server, customizing the configuration for local integration, or making other customizations and enabling features such as Live Response.

Some advanced settings in the console can be additionally limited for use via a corresponding `cb.conf` setting. For these settings, if there is no value in `cb.conf` , the console interface controls behavior. If there is a value in `cb.conf`, that value is fixed and cannot be altered through the console interface. This lets you control whether console users can modify these settings. Among the settings are:

- CbLREnabled -- controls whether Live Response is available
- ShowGDPR Banner – controls displays of a banner to indicate an EU instance
- CbServerCertWarnBeforeExpirationDays -- determines whether, and how far in advance, a warning appears when a server communication certificate is expiring
- CbServerSSLCertStrictCheck – determines whether a strict certificate check is required for server-sensor communication certificates, a setting that if set incorrectly for your environment could prevent sensor check-in
- ForceBlockLeadingWildcardsInSearchTo and ForceBlockCoreJoinsInSearchTo -- determines whether to allow certain process searches that have high performance impact

In some cases you will change the value of a setting that is already in the `cb.conf` file. In others, you must add both the setting and its value — they are not in the `cb.conf` file that is created during server initialization.

Note:

While not always required, the best practice is to restart the server (or in a clustered environment, all members of the cluster) when you change the `cb.conf` file.

Changes on Upgrade

Each new version of the Carbon Black EDR server might add or remove some parameters in the `cb.conf` file, or change the defaults for existing parameters. When you install a new version of the server, however, the upgrade program does not overwrite the existing `cb.conf` file. Instead, it preserves any custom changes you have made to `cb.conf` and creates a new file called `cb.conf.upgrade`.

Examine the `cb.conf.upgrade` file as well as this document to determine what parameters are new or changed in the current release. You can then either copy the relevant new (or changed) sections from `cb.conf.upgrade` into `cb.conf`, or replace `cb.conf` with an edited version of the upgrade file.

Note

The `cb.conf.upgrade` file is for information purposes only and has no functional effect on your server configuration (unless it is renamed to `cb.conf`).

Format Guidelines

Configuration file settings affect Carbon Black EDR services as well as the Bash shell on your server. If you edit the file, carefully observe the following formatting rules to avoid parsing errors:

- All comment lines begin with a hash mark (#).
- Place comments on their own line.
- Do not add comments to the end of a line that contains a property setting.
- Define all properties as name=value pairs.
- Omit whitespace (spaces or tabs) around the equals sign (=) as follows:
  - Correct format:
    ```
    name=value
    ```
  - Incorrect format:
    ```
    name =value
    name= value
    name = value
    ```

Other Documentation

Visit the VMware Carbon Black User Exchange website at https://community.carbonblack.com to locate documentation for tasks that are not covered in this guide, and other documents maintained as a knowledge base for technical support solutions. Documents include:

- *VMware Carbon Black EDR Operating Environment Requirements Guide (OER)* – Describes performance and scalability considerations in deploying a Carbon Black EDR server. This guide was called the *Server Sizing Guide* in previous releases.
- *VMware Carbon Black EDR Server/Cluster Management Guide* – Describes how to install, manage, backup/restore, etc. a Carbon Black EDR server/cluster.
- *VMware Carbon Black EDR User Guide* – Describes the Carbon Black EDR product and explains how to use all of its features and perform administration tasks.
- *VMware Carbon Black EDR Unified View User Guide* – Describes how to install and manage Carbon Black EDR Unified View.
- *VMware Carbon Black EDR Integration Guide* – Provides information for administrators who are responsible for integrating Carbon Black EDR with various tools and applications, such as VMware Carbon Black App Control, EMET, VDI, SSO, and more.
- *VMware Carbon Black EDR API* – Documentation for the Carbon Black EDR 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.

- *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 a Carbon Black EDR server.

Community Resources

The VMware Carbon Black User Exchange website at https://community.carbonblack.com provides access to information shared by Carbon Black customers, employees and partners. It includes information and community participation for users of all VMware Carbon Black products.

When you log in to this resource, you can:

- Ask questions and provide answers to other users' questions.
- Enter a "vote" to bump up the status of product ideas.
- Download the latest user documentation.
- Participate in the VMware Carbon Black developer community by posting ideas and solutions or discussing those posted by others.
- View the training resources available for VMware Carbon Black products.

You must have a login account to access the VMware Carbon Black User Exchange. Contact your Technical Support representative to get an account.

VMware Carbon Black Technical Support

VMware Carbon Black Technical Support provides the following channels for resolving support questions:

- **Web (User Exchange):** http://community.carbonblack.com
- **Email:** support@carbonblack.com
- **Phone:** Tel: 617.393.7400
- **Fax:** 617.393.7499

When contacting VMware Carbon Black Technical Support, provide the following information:

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

Chapter 2

Data Storage

This section describes the data storage settings in the `cb.conf` file.

Settings in this Chapter

DatastoreRootDir

Default: `/var/cb/data`

Sets the path to the root directory where runtime data for the Carbon Black EDR server is stored. This data includes Solr, PostgreSQL, and flat-file storage of module files. Each of these storage types has additional parameters, which are described herein.

Note

Consult VMware Carbon Black Technical Support to move your data root directory from one volume to another.

AllianceClientStorefilePurgeMax

Default: `100`

Specifies the maximum number of storage files (binaries uploaded from sensors to the server) that the VMware Carbon Black EDR Alliance client purges from a local hard drive, if it determines that the files should be deleted. See AllianceClientNoStorefileDelete for information about the criteria that is used to delete these files.

AllianceClientNoStorefileDelete

Default: `0`

Specifies if the Carbon Black EDR Alliance client should keep binary files locally after they have been uploaded to the central Carbon Black EDR Alliance server. If set to `0`, the Carbon Black EDR Alliance client deletes binary storefiles after uploading them to preserve local hard drive space. If set to `1`, binary modules are not deleted after they have been uploaded.

Caution

The *`Purge`* script still erases binary files to recover disk space unless *`KeepAllModuleFiles`* has been set to `1`.

You can always download binary files from the Carbon Black EDR Alliance server, even if the files have been deleted from your Carbon Black EDR server. Download attempts through the console will search the Alliance server if no local copy of a file is available. The Carbon Black EDR API also provides access to Alliance server downloads.

EnableSolrBinaryInfoNotifications

Default: `False`

When set to `True`, this parameter enables notifications for binaries that are entirely new to Carbon Black EDR or new to specific sensor hosts or sensor groups. One notification event occurs for each of the following cases:

- The binary is completely new to the sensors reporting to this server.
- The binary is new to the host that reported it.
- The binary is new to the sensor group to which the reporting host belongs. This means that one newly discovered binary can trigger up to three notifications. Notifications are sent to syslog as log messages.

EnableSolrFeedNotifications

Default: `True`

When set to `True`, this parameter enables notifications when documents that register as feed hits are committed to the Solr index. Users can configure different types of notifications per feed to be alerted of these events.

EnableExtendedApiAuditLogging

Default: `False`

When this parameter is set to `True` (enabled), Carbon Black EDR logs all REST API requests from either the console or other sources, such as scripts, and stores the data in `/var/log/cb/audit/useractivity.log`, as shown in the following example:

```
2017-12-22 11:30:54:  username='dave' userid='1'
ip='::ffff:192.168.56.1' status='200' method='GET' path='/api/v2/
sensor'
2017-12-22 11:30:54:  username='dave' userid='1'
ip='::ffff:192.168.56.1' status='200' method='GET' path='/api/v1/
alert'
2017-12-22 11:30:55:  username='dave' userid='1'
ip='::ffff:192.168.56.1' status='200' method='GET' path='/api/v1/
detect/report/currentmonitoringstatus'
2017-12-22 11:30:55:  username='dave' userid='1'
ip='::ffff:192.168.56.1' status='200' method='GET' path='/api/v3/
group'
2017-12-22 11:30:57:  username='dave' userid='1'
ip='::ffff:192.168.56.1' status='200' method='GET' path='/api/v1/
feed'
2017-12-22 11:30:57:  username='dave' userid='1'
ip='::ffff:192.168.56.1' status='200' method='GET' path='/api/v1/
process'
```

The audit information also appears in the User Management section of the console on the Activity Audit tab, and in a CSV file downloaded from that page, as shown in the following example:

```
username, timestamp, ip_address, request_method, request_path,
result, description
"dave","2017-12-22 21:33:29.031897-
05:00","::ffff:192.168.56.1","GET","/api/v2/group","200","OK"
"dave","2017-12-22 21:33:28.988702-
05:00","::ffff:192.168.56.1","GET","/api/users","200","OK"
"dave","2017-12-22 21:33:28.979345-
05:00","::ffff:192.168.56.1","GET","/api/teams","200","OK"
```

When this parameter is set to `False`, no REST API requests are logged or stored in `useractivity.log`, and the Activity Audit tab shows only login and logout events.

Change Note: New in version 6.2.1.

CbSolrConnectionTimeout

Default: `0`

Sets the connection timeout from the Solr backend engine to the datastore in milliseconds. If the internal defaults for the Solr client are in use, this value is `0`.

CbSolrSocketTimeout

Default: `0`

Sets the socket read timeout from the Solr backend engine to the datastore in milliseconds. If the internal defaults for the Solr client are in use, this value is `0`.

SolrOptimizationSkipOnWriterVolume

Default: False

If set to True, the system never optimizes the partition on the volume that is currently being written to. This helps to keep reads and writes below EBS volume limits for EC2 instances in an Amazon Web Services environment.

Change Note: New in version 6.1.

SolrOptimizeHour

Default: 0 (midnight)

Controls when Solr optimization is done (in local time).

The Cron syntax is supported for this parameter. For example, using the value */12 for the hour optimizes Solr every 12 hours.

Change Note: New in version 6.1.

SolrTimePartitioningMinutes

Default: 4320

Controls the time partitioning for Solr collections, in minutes per partition. If set to zero (0), Solr Time Partitioning will be disabled. The default is equivalent to 3 days.

Important: Set either SolrTimePartitioningDailyStartTime or SolrTimePartitioningMinutes, but not both. If both are configured, SolrTimePartitioningDailyStartTime takes precedence.

Change Note: New in version 6.1.

SolrTimePartitioningDailyStartTime

Default: (no default)

Controls the time partitioning for Solr collections by specifying a daily start time. This is represented by the number of minutes after 00:00 (UTC) that the partitioning should occur. For example, a value of 60 would represent 01:00 UTC, and a new partition would be created at that time each day.

Important: Set either SolrTimePartitioningDailyStartTime or SolrTimePartitioningMinutes, but not both. If both are configured, SolrTimePartitioningDailyStartTime takes precedence.

Change Note: New in version 6.1.

SolrTimePartitioningActivePartitions

Default: 30

Controls the number of actively searched Solr partitions that remain in the query index.

Change Note: New in version 6.1.

SolrTimePartitioningOptimizeMaxSegments

Default: 10

Controls the maximum number of segments used to optimize warm partitions.

Note: Setting this parameter lower than the default of 10 could cause optimization to take a very long time per core.

Change Note: New in version 6.1.

SolrTimePartitioningFreeSpaceThresholdPerc

Default: 5

If there are multiple event storage volumes, event storage switches to a different volume when this threshold for the percent of remaining space is reached. For example, when volume 1 is 95% full, and volume 2 is less full, event storage switches to a new partition on volume 2.

Change Note: New in version 6.1.

SolrTimePartitioningMaxSizeMB

Default: 0

Sets a maximum size (MB) for partitions, after which a new partition is started. This is an optional way to partition based on partition size instead of time (that is, SolrTimePartitioningMinutes or SolrTimePartitioningDailyStartTime).

If set to 0, only time-based partitioning is used.

Change Note: New in version 6.1.

KeepAllModuleFiles

Default: `0`

Enables or disables the deletion of binary files uploaded from sensors.

The default value of `0` indicates that the uploaded files are deleted at the following times:

- After they are uploaded to the Carbon Black EDR Alliance server.
- When data is purged to free up storage volume.

Changing this value to `1` sets the server to never delete module files.

MaxEventStoreSizeInDocs

Change Note: Removed from version 6.1. There is no longer an absolute event core size in docs.

MaxEventStoreDays

Default: `30`

Controls how old warm (mounted) partitions can get before they are unmounted or deleted. When threshold is reached, oldest partition is either unmounted (converted to cold) or deleted from disk, based on value of AlwaysDeleteColdPartitions.

This setting can affect the availability of process data for sensors that have not checked in recently. See "CoreServicesEventlogBytesCap" on page 26 for details.

Change Note: Beginning with version 6.1, the behavior of the parameter is affected by the AlwaysDeleteColdPartitions parameter.

Note: Partition age is not determined by the timestamp of the partition itself, which is assigned at the partition *start time*. Instead, partition end time is used for this purpose, and this is determined by the timestamp of the *next partition* in line.

MaxEventStoreSizeInMB

Default: `0`

By default, process data is purged automatically when disk space is required. If this value is set, process data is unmounted or deleted, starting from the earliest date, until the size of the process store is less than this value. This determines how big the total event store can get in MB before warm partitions are purged. When the threshold is reached, the oldest partition are either be unmounted (converted to cold) or deleted from disk, based on value of AlwaysDeleteColdPartitions.

Change Note: Beginning with version 6.1, the behavior of the parameter is affected by the AlwaysDeleteColdPartitions parameter.

PruneIndividualDocumentsByTime

Default: `False`

Controls whether individual documents will be pruned if their age exceeds the MaxEventStoreDays setting.

- If `False`, the system relies on unloading *partitions* to prune old data, when the entire partition ages out.
- If `True`, the system will delete *older documents* from the oldest warm partition when MaxEventStoreDays is reached.

This flag allows more expensive per-document purging in cases when SolrTimePartitioningMinutes is set to be higher than a day. This means that the partition will not be purged until youngest document in partition reaches MaxEventStoreDays threshold, which could leave an amount of data up to the value of SolrTimePartitioningMinutes over this boundary.

Note: Because deletion of individual documents is a more “expensive” operation than partition-based purging, Carbon Black recommends setting this to `True` only if you use large partitions (>3 days) and want to save on disk space.

Change Note: New in version 6.1.

MaxEventStoreSizeInPercent

Default: `90`

Determines what percent of total disk space can be taken up by the event store before cleanup is triggered. The total disk space that is available to the event store is calculated as the sum of the current event store size and free disk space. When the threshold is reached, the oldest partition will be either unmounted (converted to cold) or deleted from disk, based on value of AlwaysDeleteColdPartitions.

Change Note: The default was raised to `90` (percent) in version 6.1. In addition, beginning with version 6.1, the behavior of the parameter is affected by the AlwaysDeleteColdPartitions parameter.

MinAvailableSizeInMB

Default: `0`

Sets a lower limit on the available disk space that must be maintained on the mount point where the event store resides. This parameter takes precedence over all other storage-size parameters, except for MaxEventStoreSizeInDocs. It is an optional parameter.

This determines how much free space to leave on the events data disk before purging warm partitions. If this condition is met, partitions are deleted from disk rather than converted into cold partitions

AlwaysDeleteColdPartitions

Default: `True`

Controls the handling of cold (unmounted) partitions, including partitions unmounted due to conditions defined by other parameters in `cb.conf`.

- If `True`, unmounted partitions will be automatically deleted from the disk.
- If `False`, unmounted partitions will not be deleted unless available disk size is less than the MinAvailableSizeInMB setting. In this case, unmounted partitions will be deleted rather than unmounted, regardless of related settings, such as SolrTimePartitioningActivePartitions or MaxEventStoreDays..

Change Note: New in version 6.1.

EnableRawSensorDataBroadcast

Default: `False`

Determines how the data store publishes to the message bus. If you set this to `True` and set DatastoreBroadcastEventTypes to empty, then you get data in the same format in which the sensors send it by subscribing to the `api.rawsensordata` channel instead of the `api.events` channel.

That format will be <4 byte integer length (in little endian)><n bytes of protobuf data as specified in the header> ... repeated.

Note

Users should check the content type header for each RabbitMQ message to determine if the content is zipped or not. If the content type is `application/zip`, the message should be unzipped before processing.

DatastoreBroadcastEventTypes

Default: (None)

If this property is not empty, it enables publishing of incoming events from sensors onto RabbitMQ PUBSUB enterprise bus. See Chapter 8, “RabbitMQ” (cb-rabbitmq service) settings in this file.

The value of this property consists of `1` or more of the following comma-separated event types that should be published:

- `procstart` (or `process`)
- `procend`
- `childproc`
- `moduleload`
- `module` (a new module/binary has been encountered)
- `filemod`
- `regmod`
- `netconn`
- `crossproc`

To subscribe for all of the above event types, specify the value as "`*`". Each event type will be published to its own topic: `ingress.event.<event type>`

ProcessDocumentSplitThreshold

Change Note: Removed in 6.1.

WatchlistEndTimeOffset

Change Note: Removed in 6.1. Replaced by WatchlistEndTimeOffsetSeconds

WatchlistStartTimeOffset

Change Note: Removed in 6.1. Replaced by WatchlistStartTimeOffsetSeconds.

WatchlistEndTimeOffsetSeconds

Default: `0`

Changes the search window end-time offset (in seconds) for watchlist search jobs.

Important: Watchlist parameters are optimized based on the commit interval of the Solr backend. Contact Carbon Black Support before you change these values.

Change Note: New in version 6.1. Replaces WatchlistEndTimeOffset.

WatchlistStartTimeOffsetSeconds

Default: `90`

Specifies the search window start-time offset (in seconds) for watchlist search jobs.

Important: Watchlist parameters are optimized based on the commit interval of the Solr backend. Contact Carbon Black Support before you change these values.

Change Note: New in version 6.1. Replaces WatchlistStartTimeOffset.

MaxSyslogSenderMessageSize

Default: `1024`

Configures the maximum syslog message size (in bytes) for `cb-enterprise` syslog notifications. This configuration does not automatically adjust the maximum message size setting in rsyslog configuration.

MaxCbLoggingMessageSize

Default: `2048`

Configures the maximum syslog message size (in bytes) for `cb-enterprise` log output under `/var/log/cb`. This configuration does not automatically adjust the maximum message size setting in rsyslog configuration.

MaxSearchResultRows

Default: `1000`

Configures the maximum number of search result rows to display in the console per page.

DatastoreLogTotalSizeCap

Default: `4GB`

Maximum allowable disk space of all datastore debug logs. If this number is exceeded, old logs are purged.

Change Note: New in version 6.1.

DatastoreLogMaxFileSize

Default: `500MB`

Maximum allowable file size of a single debug log file. If this number is exceeded, a log rotation occurs.

Change Note: New in version 6.1.

DatastoreLogMaxHistory

Default: `14`

Maximum number of days to keep datastore debug logs. Logs older than this number of days are deleted.

Change Note: New in version 6.1.

SolrLogTotalSizeCap

Default: `4GB`

Maximum allowable disk space of all Solr debug logs. If this number is exceeded, old logs are purged.

Change Note: New in version 6.1.

SolrLogMaxFileSize

Default: `500MB`

Maximum allowable file size of a single debug log file. If this number is exceeded, a log rotation occurs.

Change Note: New in version 6.1.

SolrLogMaxHistory

Default: `14`

Maximum number of days to keep Solr debug logs . Logs older than this number of days are deleted.

Change Note: New in version 6.1.

Chapter 3

Sensor

This section describes sensor-related settings in the `cb.conf` file.

Settings in this Chapter

CoreServicesEventlogBytesCap

Default: `157286400`

Default value for maximum bytes (157286400, or 150MB) that can be upload by a group of sensors that will check-in within the next 465 seconds.

DatastoreReservationTimeoutMs

Default: 60000 (60 seconds)

Sets the length of time the throttle queue reservation will be valid before expiring.

Change Note: New in version 6.1. Default changed in 6.2.1.

DatastoreSubmitQMemoryPercent

Default: `10`

Sets the memory used by the front-end throttle queue, as percentage of JVM memory.

Change Note: New in version 6.1.

DatastoreSubmitTimeoutMs

Default: 0

Sets the time (in milliseconds) to wait to reserve data in the front-end throttle queue before timing out.

Change Note: New in version 6.1.

EventExclusionsEnabled

Default: `False`

If set to True, enables a console feature that allows you to exclude collection of certain process events from macOS/OS X hosts based on the path of the parent process. The settings are applied on a per Sensor Group basis and defined on an Exclusions tab on the Edit Group Settings page (only visible when this setting is true).

This is currently supported only on macOS/OS X for Carbon Black EDR macOS/OS X sensor versions 6.0.4 and above (in the 6.x series) and 5.2.7 and above (in the 5.x series).

Note: To enable this feature, you must also restart cb-enterprise services.

Change Note: New in version 6.1.

EventlogLegacyThrottleMaxRatio

Default: `1.0`

See EventlogLegacyThrottleMinRatio.

Change Note: New in version 6.1.

EventlogLegacyThrottleMinRatio

Default: `0.1`

Part of the algorithm to calculate throttle for legacy sensors relies on these three settings:

- EventlogLegacyThrottleMinRatio
- EventlogLegacyThrottleMaxRatio
- EventlogLegacyThrottleMultiplier

Final throttle represents the probability [0..1] that any given sensor will send event logs within the next check-in period and is calculated based on the current value of "retry-after" for a given cluster node.

Final calculation is multiplied with EventlogLegacyThrottleMultiplier and also bounded with EventlogLegacyThrottleMinRatio and EventlogLegacyThrottleMaxRatio. To linearly reduce or increase volume of legacy sensor event logs, reduce or increase EventlogLegacyThrottleMaxRatio from default value of `1.0`.

Change Note: New in version 6.1.

EventlogLegacyThrottleMultiplier

Default: `1.0`

See EventlogLegacyThrottleMinRatio (above).

Change Note: New in version 6.1.

EventLogThresholdAgeSeconds

Default: `5 * 50`

Sets the minimum age of sensor event log to cache before sending to the server. This is a numeric setting that can accept any valid Python math expression (as shown in the default). Designates a time period in seconds. Therefore, "5 * 60" means 5 minutes.

Pushed to 6.x sensor in check-in response. Used in legacy throttle calculation during check-in to compare age of legacy sensor backlog.

Change Note: New in version 6.1.

EventLogThresholdSizeBytes

Default: `10 * 1024 * 1024`

Sets the minimum size of sensor event log to cache before sending to the server. This is a numeric setting that can accept any valid Python math expression (see default), Designates a size in bytes. Thus, "10 * 1024 * 1024" means 10 MB.

Pushed to 6.x sensor in check-in response. Used in legacy throttle calculation during check-in to compare size of legacy sensor backlog.

Change Note: New in version 6.1.

RetryAfterMultiplier

Default: `1.0`

If the value is not `1.0`, modifies the "retry-after" calculation that occurs when the front-end throttle queue is full. A value of `0` makes the result of the calculation 0. A value of `1.0` keeps the calculation as is. Any value higher than `1.0` makes the retry-after value larger by that factor. Floating point values are allowed.

Change Note: New in version 6.1.

SensorCheckinDelayRate

Default: `100`

Sets the maximum number of check-ins per second, per minion.

Change Note: New in version 6.0.

SensorCheckinDelaySecOverride

Default: `0` (off)

Overrides the calculated check-in delay.

Change Note: New in version 6.0.

SensorCheckinDelayVariancePct

Default: `0.1`

Smoothing factor for determining the next check-in for individual sensors. For example, if calculated check-in offset (calculated at runtime based on the number of active sensors, divided by SensorCheckinDelayRate) is `60`, and SensorCheckinDelayVariancePct is `0.1`, then actual next sensor check-in time is 60 +- 6. This helps to distribute sensor check-ins evenly.

Change Note: New in version 6.0.

SensorServicesIP

Default: `[::]`

Sets the listening address for sensor services. Change to `0.0.0.0` for hosts without IPv6 stack.

SensorServicesLoggerConf

Default: `/etc/cb/sensorservices-logger.conf`

Complete path to the logging configuration file for sensor services.

SensorServicesPort

Default: `6500`

Two sensor service instances per minion listening on the configured port, and the configured port + 1. By default, this would be ports 6500 and 6501. This service handles distributed sensor check-ins.

Change Note: New in version 6.1.

SensorThrottleType

Default: `titan`

Throttling strategy. The default `titan` means post-6.0. The other valid value is `legacy`, which means pre-6.0 throttling strategies.

Note: A `titan` throttling type will support pre-6.0 sensors, but a `legacy` throttling type will not support 6.0 sensors.

ServiceUnavailableRetryDelay

Default: `60`

Specifies the number of seconds to wait before retrying when the console returns a 503 (service unavailable) message from the datastore when the site throttle or maximum low priority request thresholds have been exceeded.

Change Note: New in version 6.1.

Chapter 4

Communication

This section describes the communication settings in the `cb.conf` file.

These settings adjust the communications between the Carbon Black EDR server and other components in the Carbon Black EDR environment (such as sensors and the Carbon Black EDR Alliance server).

Settings in this Chapter

AllowNullSensorHostRegister

Default: `1`

If set to `1`, the server will request and require the sensor computer's Security Identifier (SID). If this value is empty, the server rejects the registration. If the server rejects the sensor registration, the sensor re-attempts registration in a few minutes, which includes another attempt to get the sensor computer SID.

Important: You should not need to change the default value of this configuration property. Contact VMware Carbon Black Support before attempting to change it.

Note: This setting is only relevant to Windows sensors.

CoreServicesWorkerCount

Default: `4`

Number of worker processes `cb-coreservices` should create to handle incoming client requests.

CoreServicesWorkerConnections

Default: `10`

The maximum number of simultaneous requests that can be handled by a single worker process. If the incoming request rate is greater than can be handled, requests are queued up to be handled when one of the existing ones completes.

Important: Worker processes maintain a pool of database connections and that pool has a limit of 10 connections. Carbon Black strongly recommends that this value not exceed the default value of *`10`*.

SSOConfig

Default: (no default)

Use this option to enable Carbon Black EDR server integration with an external Single Sign-On (SSO) provider by providing a path to a SSO configuration file. This is not enabled by default, but if it is enabled, the default path (which is commented out) is `/etc/cb/sso/sso.conf`.

AllianceNoClientCert

Default: `0`

The Carbon Black EDR Alliance server uses SSL client certificates to authenticate communication with Carbon Black EDR servers. Many SSL inspection devices do not support client certificates and immediately end the connection when they receive a client certificate.

Set this parameter to `1` to prevent transmission of the SSL client certificate.

Note: Contact VMware Carbon Black Technical Support for alternate authentication arrangements.

AllianceVerifyServerCert

Default: `1`

Indicates that the Carbon Black EDR Alliance server's SSL certificate must be validated with the Carbon Black EDR Certificate Authority. If the server's SSL certificate was not signed by the Carbon Black EDR Certicate Authority, the connection will fail. If your network uses an SSL inspection device, this parameter must be disabled.

AllianceClientProxyUrl

Default: (no default)

Specifies the proxy to be used for internet access. This is disabled by default. If enabled, the default value is http://127.0.0.1:3128.

AllianceClientProxyAuth

Default: `basic`

Specify the type of authentication the proxy uses. Supported types are either `basic` or `ntlm` (NT Lan Manager).

EnforceClientCerts

Default: `True`

Carbon Black EDR sensors validate servers by using SSL server certificates. The Carbon Black EDR server also validates sensors by using SSL client certificates. This setting specifies whether the Carbon Black EDR server allows sensors that do not provide an SSL certificate to communicate with it.

This value should generally be `True`, but can be disabled for troubleshooting, addressing mismatched certificates, or upgrading pre-v3.1.0 sensors that did not support SSL client certificates.

DatastoreReservationCountThrottlingEnabled

Default: `False`

Determines whether site throttling is enabled to control data flow from compatible sensors.

When set to `False`, throttling settings (listed below) have no effect and the throttling feature is bypassed.

When set to `True`, the sensor will notify the server of how much data it has to send and requests a "reservation" to upload it. The server controls the number of sensors that can submit data concurrently based on two things: the amount of data they submit on average and the limiting rate of submission per sensor, which is based on site configuration. Regardless of this and its related settings, the minimum number of sensors allowed to check in at any time is 1.

Minimum sensor versions for compatibility with this feature are: OSX 6.2.0, LNX 5.2.10, WIN 6.1.2 HF2. It is unsupported on previous sensor versions.

This setting must be set to True for the following other settings to take effect: DatastoreReservationCountThrottlingBufferPercent, DatastoreMaxTimeToSubmitS, DatastoreUseSensorUploadDynamicAveraging, DatastoreSensorUploadInitialAverageKB, DatastoreSensorUploadAverageTimeRangeS, DatastoreSensorUploadAverageMinCount.

Important: A cb-enterprise services restart is required after making this `cb.conf` change.

Change Note: New in version 6.2.2.

DatastoreReservationCountThrottlingBufferPercent

Default: `0`

Reduces amount of capacity usage by the specified percentage. For example, if you set the throttle on the Settings > Sites page of the console to 500KB/s, if this value is set to 10, Carbon Black EDR attempts to use 450KB/s. Effective only if DatastoreReservationCountThrottlingEnabled is true.

Change Note: New in version 6.2.2.

DatastoreMaxTimeToSubmitS

Default: 60

The maximum amount of time in seconds that a sensor has to submit data before a timeout. Max Count = Max Rate / (Average / DatastoreMaxTimeToSubmitS). Effective only if DatastoreReservationCountThrottlingEnabled is true.

Change Note: New in version 6.2.2.

DatastoreUseSensorUploadDynamicAveraging

Default: `True`

Enables/Disables dynamic averaging of sensor upload amounts. If false, sensors always use the DatastoreSensorUploadInitialAverageKB value. Effective only if DatastoreReservationCountThrottlingEnabled is true.

Change Note: New in version 6.2.2.

DatastoreSensorUploadInitialAverageKB

Default: `1536`

The initial amount to assume the average upload amount will be (1.5MB by default). Otherwise before the first sensor requested to submit, the average would be 0. This value is more important when DatastoreUseSensorUploadDynamicAveraging is turned off. Effective only if DatastoreReservationCountThrottlingEnabled is true.

Change Note: New in version 6.2.2.

DatastoreSensorUploadAverageTimeRangeS

Default: `300`

The time range (in seconds) used for the averages of sensor submissions. For example, the default value of 300 means that if a submit occurs prior to the last 300 seconds, it's value will no longer be counted. This is used to make a rolling average. Effective only if DatastoreReservationCountThrottlingEnabled is true.

Change Note: New in version 6.2.2.

DatastoreSensorUploadAverageMinCount

Default: `20`

The minimum number of submissions to keep, even if they've become stale (outside the time range specified by DatastoreSensorUploadAverageTimeRangeS). This keeps the average more stable on systems where there is not a lot of activity. Effective only if DatastoreReservationCountThrottlingEnabled is true.

Change Note: New in version 6.2.2.

Chapter 5

Network

This section describes the network settings in the `cb.conf` file. Review, update, or modify these settings to adjust the Carbon Black EDR server listener IP addresses and ports.

Settings in this Chapter

CoreServicesIP

Default: `[::]`

The coreservices daemon binds to this interface. This parameter allows you to specify an option that makes sense in your environment, such as:

- `[::]` – listen on ALL IPv4 and IPv6 interfaces
- `0.0.0.0` – listen on ALL IPv4 ONLY interfaces
- `127.0.0.1` – listen on local IPv4 loopback interface
- `[::1]` – listen on local IPv6 loopback interface
- `127.0.0.1|[::1]` – listen on IPv4 AND IPv6 loopback interfaces

CoreServicesPort

Default: `5000`

The coreservices daemon binds to the port specified by this parameter.

DatastorePort

Default: `9000`

The data store service `cbfs-http` binds to this port.

DatastoreIP

Default: `0.0.0.0`

The data store service `cbfs-http` binds to this IP address. With the default value of `0.0.0.0`, the service will bind to all network interfaces.

MinionApiPort

Default: `443`

This port is used for API calls from the master to minion servers.

NginxSensorHttpPort

Default: `443`

Nginx maintains its own configuration files. However, this property must be kept in sync with the configuration of the listen directive in `/etc/cb/nginx/conf.d/cb.conf`, so that other components (such as firewall management) know which ports are used for HTTP communications.

NginxWebApiHttpHostname

Default: If unset, the server uses the value that is returned by python's `` `socket.getfqdn()` ``. You can view this default value by running the following command:

```
/usr/share/cb/virtualenv/bin/python -c 'import socket;
print(socket.getfqdn());'
```

This property is referenced by the server when the server performs origin header checks. Therefore, this property should be set to the hostname that users type in to access the server. See also WebsocketOriginCheck, NginxWebApiHttpScheme, and NginxWebApiHttpPort.

NginxWebApiHttpPort

Default: `443`

See "NginxSensorHttpPort" for more details about this property. This value is also referenced by the server when the server performs origin header checks. Therefore, this property should be set to the port that users type in to access the server. See also WebsocketOriginCheck, NginxWebApiHttpHostname, and NginxWebApiHttpScheme.

NginxWebApiHttpScheme

Default: https://

This value is referenced by the server when the server performs origin header checks. Therefore, this property should be set to the URL scheme (http:// or https://) that users type in to access the server. See also WebsocketOriginCheck, NginxWebApiHttpHostname, and NginxWebApiHttpPort.

ReverseProxyIP

Default: `::ffff:192.168.1.10`

If this IP address is set, Nginx does not check client certificates from a reverse proxy. For sensors reporting through the reverse proxy, the proxy must be configured with the client certificate and private key from the Carbon Black EDR server for the sensors.

In addition, these headers should be set:

- The `X-Client-Cert-Id` header must be set by the reverse proxy to the ID of the client certificate used by the sensor.
- The `X-Real-IP header` must be set to the correct address on the reverse proxy.

Details for the configuration and requirements for a reverse proxy are available from VMware Carbon Black Technical Support.

Note

The IPv4 address of a reverse proxy is in IPv6-wrapped format.

RedisHost

Default: `localhost`

Sets the Redis general cache host.

RedisPort

Default: `6379`

Sets the Redis general cache listener port (TCP).

RedisStatsHost

Default: `localhost`

Sets the Redis statistics cache host.

RedisStatsPort

Default: `6379`

Sets the Redis statistics cache listener port (TCP).

SolrIP

Default: `127.0.0.1`

Sets the network binding IP address for the cb-solr service.

SolrPort

Default: `8080`

Sets the binding between the cb-solr service and specified port. This identifies the HTTP port that is used for processes on localhost and other nodes in a cluster configuration.

WebsocketOriginCheck

Default: `False`

If set to `True`, websocket-backed features such as alert notifications and streaming Live Query results will be blocked whenever the server cannot guarantee that user websockets originated from your server. This occurs when the websocket handshake origin header does not match the server's expected origin.

The server’s expected origin value is constructed referencing NginxWebApiHttpScheme, NginxWebApiHttpHostname, and NginxWebApiHttpPort.

Chapter 6

SSL (TLS) Certificate

This section describes the SSL certificate usage in the `cb.conf` file.

Carbon Black EDR uses certificates in the following ways:

- Sensors use SSL server certificates to validate that they are communicating with the correct Carbon Black EDR server.
- The Carbon Black EDR server uses SSL client certificates to validate that it is communicating with authentic sensors.
- The Carbon Black EDR server uses an SSL server certificate to validate that it is communicating with the correct Carbon Black EDR Alliance server.
- The Carbon Black EDR Alliance server uses SSL client certificates to validate that it is communicating with authentic Carbon Black EDR servers.

Settings in this Chapter

SSLCertFile

Default: /`etc/cb/certs/cb-server.crt`

Sets the location of the SSL certificate file that is used for HTTPS communications between sensors and the Carbon Black EDR server. These certificates are generated during cbinit and are unique for each server.

Important: If these paths are modified, a corresponding change must also be made in the *`etc/nginx/conf.d/cb.conf`* file.

SSLKeyFile

Default: `/etc/cb/certs/cb-server.key`

Sets the location of the SSL private key file that is used for HTTPS communications between sensors and the Carbon Black EDR server. These certificates are generated during `cbinit` and are unique for each server.

Important: If these paths are modified, a corresponding change must also be made in the *`etc/nginx/conf.d/cb.conf`* file.

CbServerCertWarnBeforeExpirationDays

Default: <`none`>

Sets the number of days before a server certificate expires that a warning appears in the console. You must restart the server or cluster for a change to this setting to take effect.

- If no value is provided (the default) this can be changed via a menu on the Settings page in the console.
- If a value is provided here, it cannot be changed through the console. To match the options in the console interface, the value chosen should be one of the following values: 0, 15, 30, 60 or 90.
  - If the value provided is 0 (zero), no expiration warning is displayed.
  - If you use a value other than the ones available through the console, that value will take effect but the console will show “--” instead of a number.

Change Note: New in version 6.4.0.

CbServerSSLCertStrictCheck

Default: <`none`>

Sets the validation method for server certificates used for server-sensor communication. You must restart the server or cluster for a change to this setting to take effect.

- If no value is provided (the default), validation method this can be changed via radio buttons on the Settings page in the console.
- If the value provided here is `False`, the legacy certificate pinning method is used, and certificate expiration does not interrupt server-sensor communication.
- If the value provided here is `True`, strict certificate validation is used. If the certificate has expired or fails any other validation requirements, server-sensor communication is disabled.

Caution: For sensors that support custom certificates, do not enable strict validation if you are using the legacy certificate created during server installation. Using strict validation for legacy certificates (or any other certificate that cannot pass validation) will disable communication between the sensor and server, and may require uninstalling and reinstalling sensors. This is not an issue with sensors that do not support custom certificates

Change Note: New in version 6.4.0.

AllianceCert

Default: `/etc/cb/certs/carbonblack-alliance-client.crt`

Sets a SSL certificate file that is used for client-side authentication when an HTTPS connection with a Carbon Black EDR Alliance server is established.

These files are loaded onto the machine when the Carbon Black EDR Release RPM is installed. The files are used whenever the Carbon Black EDR server must communicate with central Carbon Black EDR Alliance server(s). This includes yum repositories for installing and upgrading the Carbon Black EDR server software as well as the Carbon Black EDR Alliance client service.

Important: These certificates are specific to each customer organization and should be treated with care. Do not share them with other organizations or people outside your company.

AllianceCertKey

Default: `/etc/cb/certs/carbonblack-alliance-client.key`

Sets SSL private key files that are used for client-side authentication when an HTTPS connection with a Carbon Black EDR Alliance server is established.

These files are loaded onto the machine when the Carbon Black EDR Release RPM is installed. The files are used whenever your Carbon Black EDR server must communicate with central servers at Carbon Black EDR. This includes yum repositories for installing and upgrading the Carbon Black EDR server software as well as the Carbon Black EDR Alliance client service.

Chapter 7

General

This section describes general settings in the `cb.conf` file for Carbon Black EDR server.

Settings in this Chapter

CbUser

Default: `cb`

Defines the user with which the Carbon Black EDR services are run. The user is created during RPM installation. To use another user, create the user, and then restart the server (cb-enterprise).

CbGroup

Default: `cb`

Defines the Linux group with which the Carbon Black EDR services are run. The sensor group is created during RPM installation. To use another service group, create the group in Linux, update this value, and then restart the server (cb-enterprise).

CbFileDescriptorLimit

Default: `80000`

By default, CentOS allows only 1024 file descriptors per process. This number is too low for Carbon Black EDR. Carbon Black EDR updates the process file descriptor limit in the `cb-enterprise init` script to the default value with `ulimit -n` .

Change Note: Previous versions of this document showed an incorrect default (8000).

CbLicenseFile

Default: `/etc/cb/server.lic`

The path to the Carbon Black EDR server license file.

Note: Consult VMware Carbon Black Technical Support before attempting to change this parameter.

CbServerTokenFile

Default: `/etc/cb/server.token`

A random hexadecimal string used to uniquely identify this Carbon Black EDR server installation.

Note: Consult Carbon Black Support before attempting to change this parameter.

ClusterMembership

Default: (no default)

Indicates whether or not this server node is part of a cluster. Valid values are: `Standalone`, `Master`, and `Slave`.

ClusterNodeId

Default: `0`

A server node unique identifier. In a standalone installation, a one-to-one relationship exists between this field and server token. However, if this node is part of a cluster, the server token represents the entire cluster, while this identifier uniquely identifies each node in the cluster.

CbJavaHome

Defaults: `/usr/lib/jvm/jre-1.8.0/` (EL6) or `/usr/lib/jvm/jre-11/` (EL7/EL8)

Carbon Black EDR requires JRE version 1.8.0 or higher for EL6, or 1.11.0 or higher for EL7 or EL8. If the JRE is installed at a different location on your server, change this value to reflect the correct location.

EventForwarderEnabled

Default: `false`

With the release of Carbon Black EDR 7.1.0 server, admins can customize the Event Forwarder from directly within the Carbon Black EDR console.

By default, this feature is enabled for cloud instances, and disabled for on-prem deployments. You can enable the feature for on-prem by adding `EventForwarderEnabled=true` in `cb.conf` and restarting services.

ManageFirewall

Default: `True`

Indicates whether or not the Carbon Black EDR server setup tools will manage the configuration of firewall on behalf of the user. Set this value to `False` if you want to administer the firewall configuration yourself.

A `True` setting is only effective if neither manage_firewall nor manage_iptables is passed in from cbinit.conf during an unattended install.

ManageIptables

Change Note: Removed from version 6.2.3. For customers upgrading from a pre-v6.2.3 release, this setting is automatically removed and replaced with ManageFirewall.

CbShutdownKillJobsTimeout

Default: `30`

The time in seconds to wait before killing cb cron jobs when cb-enterprise is shut down.

CoreServicesEnableProfiling

Default: `Off`

Specifies whether or not to enable profiler on start. Valid values for this property are `Off`, `CpuTicks`, and `WallClock`.

CoreServicesEnableApiProfiling

Default: `False`

Specifies whether detailed API profiling is enabled. If enabled, your browser's development console will contain timing info on each API call.

CoreServicesSmallScaleSensorCount

Default: `25`

If the number of sensors that are currently active is less than this value, the sensor check-in interval is always 30 seconds. If it is greater, Carbon Black EDR calculates a dynamic check-in interval.

CoreServicesMaxCheckinInterval

Default: `1335`

Configures the maximum interval, in seconds, between successive sensor check-ins from a single sensor. Raising this value decreases the load on the server, as there are fewer sensor check-ins and fewer modifications to the event store.

CoreServicesEnableProcessFacets

Default: `True`

Enables or disables all console facets (small graphic data displays) on the **Search Processes** page. This is enabled by default.

CoreServicesEnableBinaryFacets

Default: `True`

Enables or disables all console facets (small graphic data displays) on the **Search Binaries** page. This is enabled by default. See the *VMware Carbon Black EDR User Guide*.

Note: This setting is comma delimited.

CoreServicesDisabledProcessFacets

Default: (no default)

Disables specified console facets (small graphic data displays at the top of the page) on the **Search Processes** page. See the *VMware Carbon Black EDR User Guide*.

Note: This setting is comma delimited.

CoreServicesDisabledBinaryFacets

Default: (no default)

Disable specified console facets (small graphic data displays at the top of the page) on the **Search Binaries** page.

CoreServicesMaxFacetThreads

Default: `None`

Configures the maximum number of threads used for console facets. The default of `None` disables facet threading.

Change Note: Default value changed since version 6.0. It was formerly `1`.

CoreServicesEnableFuzzyProcessFacets

Default: `True`

Enables and disables the use of statistical sampling for calculating the terms in facets. This provides significantly improved runtime performance and reduced memory usage.

If `True`, fuzzy faceting of process facets begins when the following other configuration parameters reach a specified threshold:

- CoreServicesFuzzyProcessFacetsThreshold (Default = 10000)
- CoreServicesFuzzyProcessFacetsSamplingPerc (Default = 0)

See the descriptions of these parameters for additional details.

Change Note: This parameter existed in pre-6.1 versions but the underlying parameters for fine-tuning were not exposed.

CoreServicesFuzzyProcessFacetsThreshold

Default: `10000`

One of two parameters that determine whether fuzzy faceting will start, if enabled by CoreServicesEnableFuzzyProcessFacets.

Specifies the maximum number of documents returned for any given process search before fuzzy faceting begins.

Change Note: New in version 6.1.

CoreServicesFuzzyProcessFacetsSamplingPerc

Default: `0`

One of two parameters that determine whether fuzzy faceting will start, if enabled by CoreServicesEnableFuzzyProcessFacets.

Defines the sampling rate as a percentage of the total number of documents returned for any given process search.

Note: The default value of `0` percentage dynamically adjusts based on total result count, in order to reach a sample size of at least CoreServicesFuzzyProcessFacetsThreshold.

Change Note: New in version 6.1.

CoreServicesEnableFuzzyBinaryFacets

Default: `True`

Enables and disables the use of statistical sampling for calculating the terms in binary facets. This provides significantly improved runtime performance and reduced memory usage. You can disable this setting to make the resulting data more accurate, but it will significantly slow down performance.

If `True`, fuzzy faceting of binary facets will begin when the following other configuration parameters reach a specified threshold:

- CoreServicesFuzzyBinaryFacetsThreshold (Default = 10000)
- CoreServicesFuzzyBinaryFacetsSamplingPerc (Default = 0)

See the listings for these parameters for additional details.

Change Note: This parameter existed in pre-6.1 versions but the underlying parameters for fine-tuning were not exposed.

CoreServicesFuzzyBinaryFacetsThreshold

Default: `10000`

One of two parameters that determine whether fuzzy faceting of binary facets will start, if enabled by CoreServicesEnableFuzzyBinaryFacets.

Specifies the maximum number of documents returned for any given process search before fuzzy faceting begins.

Change Note: New in version 6.1.

CoreServicesFuzzyBinaryFacetsSamplingPerc

Default: 0

One of two parameters that determine whether fuzzy faceting of binary facets will start, if enabled by CoreServicesEnableFuzzyBinaryFacets.

Defines sampling rate as a percentage of total documents returned for any given process search.

Note: The default value of `0` percentage dynamically adjusts based on total result count, in order to reach a sample size of at least CoreServicesFuzzyBinaryFacetsThreshold.

Change Note: New in version 6.1.

CoreServicesEnableFuzzyAlertFacets

Default: `False`

Enables and disables the use of statistical sampling for calculating the terms in alert facets. This provides significantly improved runtime performance and reduced memory usage.

If `True`, fuzzy faceting of alert facets will begin when the following other configuration parameters reach a specified threshold:

- CoreServicesFuzzyAlertFacetsThreshold (Default = 10000)
- CoreServicesFuzzyAlertFacetsSamplingPerc (Default = 0)

See the listings for these parameters for additional details.

Change Note: This parameter existed in pre-6.1 versions but the underlying parameters for fine-tuning were not exposed.

CoreServicesFuzzyAlertFacetsThreshold

Default: `10000`

One of two parameters that determine whether fuzzy faceting of alert facets will start, if enabled by CoreServicesEnableFuzzyAlertFacets.

Specifies the maximum number of documents returned for any given process search before fuzzy faceting begins.

Change Note: New in version 6.1.

CoreServicesFuzzyAlertFacetsSamplingPerc

Default: `0`

One of two parameters that determine whether fuzzy faceting of alert facets will start, if enabled by CoreServicesEnableFuzzyAlertFacets.

Defines sampling rate as a percentage of total documents returned for any given process search.

Note: The default value of `0` percentage dynamically adjusts based on total result count, in order to reach a sample size of at least CoreServicesFuzzyAlertFacetsThreshold.

Change Note: New in version 6.1.

CoreServicesEnableFuzzyFeedFacets

Default: `False`

Enables and disables the use of statistical sampling for calculating the terms in feed facets. This provides significantly improved runtime performance and reduced memory usage.

If `True`, fuzzy faceting of feed facets begins when the following other configuration parameters reach a specified threshold:

- CoreServicesFuzzyFeedFacetsThreshold (Default = 10000)
- CoreServicesFuzzyFeedFacetsSamplingPerc (Default = 0)

Enabling this setting improves runtime performance and reduces memory usage. Disabling this setting increases the accuracy of the resulting data, but slows down performance.

Change Note: This parameter existed in pre-6.1 versions but the underlying parameters for fine-tuning were not exposed.

CoreServicesFuzzyFeedFacetsThreshold

Default: `10000`

One of two parameters that determine whether fuzzy faceting of feed facets will start, if enabled by CoreServicesEnableFuzzyFeedFacets.

Specifies the maximum number of documents returned for any given process search before fuzzy faceting begins.

Change Note: New in version 6.1.

CoreServicesFuzzyFeedFacetsSamplingPerc

Default: `0`

One of two parameters that determine whether fuzzy faceting of feed facets will start, if enabled by CoreServicesEnableFuzzyFeedFacets.

Defines sampling rate as a percentage of total documents returned for any given process search.

Note: The default value of `0` percentage dynamically adjusts based on total result count, in order to reach a sample size of at least CoreServicesFuzzyFeedFacetsThreshold.

Change Note: New in version 6.1.

CoreServicesEventlogBytesCap

Default: `157286400` (157MB)

Sets the upper limit on the aggregate number of bytes that can be uploaded by a group of sensors that will check-in during the next monitoring interval. You can disable this setting to make the resulting data more accurate, but it will significantly slow down performance.

CoreServicesMaxEventlogBytesPerSensor

Default: `10485760` (10MB)

Sets the maximum number of bytes a sensor can push per check-in.

SensorMaxUpgradeRate

Default: `600`

Sets the maximum auto-upgrades per hour. If this property is specified, it places a limit on the number of auto-upgrade requests triggered from a sensor group version setting.

By default, if this option is not specified, there is no cap and any sensor that must be upgraded will be instructed to do so with its next check-in.

CoreServicesProcessSearchOrder

Default: (no default)

Sets the sort order of process search results as seen in the console. The format of this field is: `fieldname direction`, where `direction` is either `asc` (for ascending) or `desc` (for descending).

CoreServicesBinarySearchOrder

Default: (no default)

Sets the sort order of binary search results as seen in the console. The format of this field is: `fieldname direction` where `direction` is either `asc` (for ascending) or `desc` (for descending).

CoreServicesProcessSearchPageSize

Default: `10`

Sets the number of matching process documents that display on each page as seen in the **Search Processes** page in the console.

Change Note: This setting functions as described in release 6.3.0 and later. It had no effect in previous releases.

CoreServicesBinarySearchPageSize

Default: `10`

Sets the number of matching binary documents that display on each page as seen in the **Search Binaries** page in the console.

Change Note: This setting functions as described in release 6.3.0 and later. It had no effect in previous releases.

CoreServicesProcessAutocomplete

Default: `Suggester`

Sets the backend method for the auto-complete function for search queries entered in the **Search Processes** page. Valid values are:

- `Suggester: Faster` – This value does not include counts or infrequent terms.
- `Terms: Slower` – This value includes counts and all terms.

CoreServicesBinaryAutocomplete

Default: `Terms`

Sets the backend method for the auto-complete function for search queries entered in the **Search Binaries** page. Valid values are:

- `Suggester: Faster` -This value does not include counts or infrequent terms.
- `Terms: Slower` – This value includes counts and all terms.

TimestampDeltaThreshold

Default: `5`

Sets the time (in seconds) used as a threshold for identifying sensors with unsynchronized clocks.

CoreServicesPidFile

Default: `/var/run/cb/coreservices.pid`

Contains the current process ID of the coreservices daemon.

SensorInstallerDir

Default: `/usr/share/cb/coreservices/installers`

Directory path for sensor installers on Windows. Installers are loaded from this directory at server startup through coreservices, or with the command `/usr/share/cb/cbcheck sensor-builds -u`, where `-u` is for update.

After installer packages are loaded, they are available for installing or upgrading endpoint sensors. See the *VMware Carbon Black EDR User Guide* for details.

SensorInstallerDirOsx

Default: `/usr/share/cb/coreservices/installers`

Directory path for sensor installers on macOS. Installers are loaded from this directory at server startup through coreservices, or with the command `/usr/share/cb/cbcheck sensor-builds -u`, where `-u` is for update.

After installer packages are loaded, they are available for installing or upgrading endpoint sensors. See the *VMware Carbon Black EDR User Guide* for details.

SensorInstallerDirLinux

Default: `/usr/share/cb/coreservices/installers`

Directory path for sensor installers on Linux. Installers are loaded from this directory at server startup through coreservices, or with the command `/usr/share/cb/cbcheck sensor-builds -u`, where `-u` is for update.

After installer packages are loaded, they are available for installing or upgrading endpoint sensors. See the *VMware Carbon Black EDR User Guide* for details.

EmailNotificationsFromAddress

Default: `no-reply@carbonblack.com`

Configure email from the address for watchlist and feed notifications.

FlaskSecret

Default: (no default)

This required value is a random string of ASCII-printable characters. It is unique for each server and auto-generated during `cbinit`. It is used to encrypt session cookies that are used when a user authenticates with the console.

FailedLogonLockoutCount

Default: `10`

Sets the number of times a user can fail authentication before the account is locked.

AccountUnlockInterval

Default: `30`

Sets the number of minutes after which a locked account unlocks.

UserActivityQuota

Default: `10000`

Carbon Black EDR logs all user authentication in the PostgreSQL database. This setting defines the minimum number of authentication records that are kept.

UserActivityQuotaDelta

Default: `.1`

Defines when to start trimming the number of user authentication records. It is a percentage of "UserActivityQuota".

For example, if UserActivityQuota is set to `10000` and UserActivityQuotaDelta is set to `.1`, when the number of records reaches 11000, it is reduced to 10000. This ensures that you always have the most recent 10000 records.

SolrQueryExecutionQuota

Default: `10000`

Total number of records retained in the SQL table SolrQueryExecution, which records expensive queries.

Controls recording of slow Solr queries by setting a threshold for keeping a history of SOLR query executions. Slower queries are recorded in SQL and then saved in cbdiags when cbdiags are requested.

Change Note: New in version 6.1.

SolrQueryRecorderDurationThresholdMs

Default: `1000`

Controls recording of slow Solr queries by setting a threshold on the execution time (in milliseconds) allowed for recording slow queries. Slower queries are recorded in SQL and then saved in cbdiags when cbdiags are requested.

Change Note: New in version 6.1.

SolrQueryRecorderTopLevelOnly

Default: `True`

When true, record only top-level Solr queries. Queries on individual cores (including minions) will not be recorded.

Change Note: New in version 6.1.

AllianceClientPidFile

Default: `/var/run/cb/allianceclient.pid`

Sets the path to the PID file used for the Carbon Black EDR Alliance client service control.

AllianceSyncIntervalSecs

Default: `60`

Sets the time (in seconds) between periodic connection attempts to the Carbon Black EDR Alliance server.

AllianceURL

Default: `https://api.alliance.carbonblack.com`

Sets the URL of the Carbon Black EDR Alliance server.

DatastoreJvmMax

Default: `10%`

Sets the maximum amount of RAM to be used for the JVM's memory heap. This can be specified either as a number of megabytes (for example, 4096) or as a percentage of the host machine's physical RAM by appending `%` on the end (for example, `30%`).

DatastoreEventCoreClientThreads

Default: `0`

Sets the number of worker threads that process data from the throttle queue and insert it into Solr. The default of zero causes auto-calculation of threads based on CPU cores.

DatastoreAllowUnregisteredSensor

Default: `0`

Controls whether the datastore accepts data from a sensor that has not been registered with a Carbon Black EDR server. The default of `0` disables this capability, and there is generally no reason to enable it.

DatastoreShutdownTimeout

Default: `60`

Sets the number of seconds to wait (when the datastore is being stopped) for all buffers and cached data to be cleanly written to disk. After this time, if the service is still running, it is forcibly stopped.

DatastoreDisableJMXRemote

Default: `0`

Allows external Java management or a debugging process on the local machine to communicate with the datastore. If this setting is not `0`, the datastore process is launched without this setting.

DisableDatastoreCache

Change Note: Removed from version 6.1. See SmallDeploymentMode for equivalent functionality.

SmallDeploymentMode

Default: `False`

If set to `True`, this option disables datastore caching and causes Solr to commit process document updates within 15 seconds. This option trades performance for reduced latency.

Change Note: The default for this parameter was `True` in pre-6.1 versions.

DatastoreDbPoolSize

Default: `4`

Sets the maximum database connections from a single datastore instance.

IngresScannerEventProcessorDir

Default: `/etc/cb/datastore/processors`

Sets the location of ingress scanner event processor libs and configuration.

Important: Consult Carbon Black Support before attempting to change this parameter.

EnableProcessMD5FeedHits

Default: `True`

If `True` (the default), ingress and subsequent storage feed hits triggered by MD5 of the process are enabled.

If `False`, MD5 feed hits are only triggered when metadata for a newly observed binary file is recorded.

EnableProcessSHA256FeedHitEvents

Default: `True`

If `True` (the default), ingress and subsequent storage feed hits triggered by the SHA-256 of the process are enabled.

If `False`, SHA-256 feed hits are only triggered when metadata for a newly observed binary file is recorded.

Change Note: New in version 6.3.0

FeedHitMinScore

Default: `1`

Sets the cap on the minimum feed hit score that will trigger a feed hit event.

Note: This does not control whether or not a document will get tagged but only controls the creation of events that drive email, syslog, and alert notifications.

FeedHitMinScore<XXXXX>

Default: `1`

Sets the cap on the minimum feed hit score that will trigger a feed hit event *for a specific feed*, where 'XXXXX' is the `feed_name` attribute of the feed obtaining the special value. This can be used to override the value set for feed hit events in FeedHitMinScore. If set, this specifies the minimum feed hit score that will trigger a feed hit event for the specified feed.

For example, to require a minimum feed hit score of 3 for events from the CB Reputation threat feed (formerly the Software Reputation Service or 'SRS'), you would use FeedHitMinSrsThreat=3.

FeedNotificationsRateLimiterEnabled

Default: `False`

Enables limiting of feed hit notification rate using a limit specified by FeedNotificationsRateLimit for a period specified by FeedNotificationsRateLimitDuration.

FeedNotificationsRateLimit

Default: `5`

Specifies the maximum number of feed hit notifications that can be sent for a given feed within a period specified by FeedNotificationsRateLimitDuration. Applied only when FeedNotificationsRateLimiterEnabled is set to `True`.

FeedNotificationsRateLimitDuration

Default: `1`

Specifies the duration in hours for which the FeedNotificationsRateLimit value is valid.

Rate limit counters will be reset after FeedNotificationsRateLimitDuration number of hours elapses. Applied only when FeedNotificationsRateLimiterEnabled is set to `True`.

For example, if FeedNotificationsRateLimiterEnabled is `True`, and if FeedNotificationsRateLimit`=5` and FeedNotificationsRateLimitDuration`=1`, the effective rate limit is 5 hits per hour. This means that after sending five notifications for a particular IOC in a given feed, no more notifications will be sent for the rest of the hour.

EventStoreSolrCore

Change Note: Removed from version 6.1.

ModInfoStoreSolrCore

Default: `cbmodules`

Sets the name of the Solr core to be used for module information storage.

ModInfoStoreFlushInterval

Default: `1000`

Sets the time interval, in milliseconds, with which buffered module information events are pushed to the module information Solr core.

PgSqlDataDir

Default: `/var/cb/data/pgsql`

Sets the location of the PostgreSQL data directory.

PgSqlPidFile

Default: `/var/run/cb/cb-pgsql.pid`

Sets the path to the PID file, which is used for `cb-pgsql` service control.

PgSqlLogfilePath

Default: `/var/log/cb/pgsql/startup.log`

Sets the path to the `cb-pgsql` startup log file. This file captures output that is generated prior to the initialization of the logging framework.

PgSqlHost

Default: `*`

Sets the network interfaces on which `cb-pgsql` listens. Specify '`*`' to listen on all available interfaces. More than one interface can be specified with the use of a comma (`,`) separator.

PgSqlPort

Default: `5002`

Sets the port on which `cb-pgsql` listens.

DatabaseURL

Default: `postgresql+psycopg2://cb:<passwd>@localhost:5002/cb`

Sets the SQLAlchemy database URL that is used to connect with PostgreSQL. Substitute your password in the default shown above. For example:
`postgresql+psycopg2://cb:ZX1234Hbn987G4tk@localhost:5002/cb`

ModstorePath

Default: `/var/cb/data/modulestore`

Sets the flat-file storage location for module file storage.

CoreServicesMaxEventResultsPerProcess

Default: `10000`

Sets the maximum number of events to return from the `/process/<guid>/<segment>/event` API.

Change Note: New in version 6.1.

CoreServicesMaxSegmentsPerProcess

Default: `1000`

Sets the maximum number of segments to return from `/process/<guid>/0/preview`, `/process/<guid>/0`, `/process/<guid>/0/report`

Change Note: New in version 6.1.

WatchlistSearchMaxTags

Default: 100

Sets the number of tags to set in a single watchlist search.

Change Note: New in version 6.1. This was previously hard-coded to 100.

SearchRestrictFirstSegment

Default: `False`

Determines whether to use special logic to restrict searches only at segment_id:1 as long as query doesn't contain event fields.

Changing this to `True` will make some queries on old data return 0 results since immutable sensors will not send segment_id:1.

Change Note: New in version 6.1.

SearchUseTerminatedOnCounts

Default: `True`

Adds accuracy for queries that use event count fields (for example, filemod_count, netconn_count) with immutable documents, which are the default in version 6.1.

Change Note: New in version 6.1.

ModulesCacheMemoryPercent

Default: `5`

Sets the percent of memory that should be used in the datastore for the module partition cache structures. This cache holds md5 values that have been observed in process documents so that Carbon Black EDR can periodically and efficiently update module documents with the set of partition id's that they have been observed in.

Change Note: New in version 6.1.

ModulesCacheWritePeriodSecs

Default: `30`

Sets the frequency (in seconds) for writing out partition updates to modules observed by Carbon Black EDR.

Change Note: New in version 6.1.

ModulesRecentCacheTimeoutMultiplier

Default: `4`

Sets a multiplier used in combination with ModulesCacheWritePeriodSecs to determine how long the cache of recently observed md5 values are held in memory. For example, if the defaults are used, the timeout will be 4x30=120 seconds.

A cache of recently observed md5 values is kept in case a partition rolls over, so that Carbon Black EDR can send the recently seen values to the new partition to make sure nothing was missed. This cache is only restricted by time and is a multiple of the write frequency. The two parameters allow the user to choose between memory limits and time limits for controlling the cache.

Note: Large timeout values can have a detrimental impact on memory.

Change Note: New in version 6.1.

ForceComprehensiveSearch

Default: `True`

Determines whether to run “comprehensive” search automatically when needed, without confirming with the user.

Note: The console UI will still prompt for confirmation of the comprehensive search if the search runs against a legacy 5.x core (that is, a Solr database built in 5.2 that is replaced by upgrading to 6.x).

Change Note: New in version 6.1.

ForceBlockLeadingWildcardsInSearchTo

Default: `(none)`

Determines whether process searches with leading wildcards, which can cause performance problems, are always blocked, always allowed, or can be configured through the console:

- If not set, console users with Global Admin privileges can block or allow searches that use leading wildcards by checking or unchecking the “Block Searches with Leading Wildcards” box on the Advanced Settings page.
- If `True`, process searches with leading wildcards are **blocked**, and no user, including a Global Admin, can change this through the console.
- If `False`, process searches with leading wildcards **allowed**, and no user, including a Global Admin, can change this through the console.

Change Note: New in version 6.2.3.

ForceBlockCoreJoinsInSearchTo

Default: `(none)`

Determines whether process searches with joins of large module cores, which can cause performance issues, are always blocked, always allowed, or can be configured through the console:

- If not set, console users with Global Admin privileges can block or allow searches that use large module cores by checking or unchecking the “Block Searches that Include Binary Metadata with Large Binary Stores” box on the Advanced Settings page.
- If `True`, process searches using joins of large module cores are **blocked**, and no user, including a Global Admin, can change this through the console.
- If `False`, process searches using joins of large module cores are **allowed**, and no user, including a Global Admin, can change this through the console.

The definition of “large” for this setting is determined by the value of ModuleCoreDocumentCountWarningThreshold.

Change Note: New in version 6.2.3.

ModuleCoreDocumentCountWarningThreshold

Default: `10000000`

For process searches with binary joins, sets the number of module core documents that is considered “large” enough to potentially cause performance problems. Process searches involving binary metadata with large binary stores can be blocked using the ForceBlockCoreJoinsInSearchTo cb.conf setting and also through the “Block Searches that Include Binary Metadata with Large Binary Stores” setting on the **Settings > Advanced Settings** page.
Change Note: New in version 6.2.3.

DefaultSolrTimeoutS

Default: `60`

Solr timeout (in seconds) for all UI and API queries.

Change Note: New in version 6.1.

RebuildEventSuggestersMins

Default: `30`

Frequency (in minutes) for rebuilding event suggesters.

Change Note: New in version 6.1.

RebuildEventSuggestersTimeoutS

Default: `120`

Timeout (in seconds) for event suggesters rebuilds.

Change Note: New in version 6.1.

RebuildModuleSuggestersMins

Default: `30`

Frequency (in minutes) for rebuilding module suggesters.

Change Note: New in version 6.1.

RebuildModuleSuggestersTimeoutS

Default: `120`

Timeout (in seconds) for module suggesters rebuilds.

Change Note: New in version 6.1.

WatchlistSearchTimeoutS

Default: `120`

Solr timeout (in seconds) for all feed/watchlist queries.

Change Note: New in version 6.1.

CbDiagTmpDir

Default: `/tmp`

Location to write cbdiags data. You can choose a different directory if you prefer not to add these diagnostic files to `/tmp`. For example: `CbDiagTmpDir=/var/cb/data`

Change Note: New in version 6.2.1. This setting is not included in the default `cb.conf` file.

Change Note: New in version 7.1: Temp directory for cbdiag

The feature is for servers that have limited space in `/tmp`. Currently, `--tmpdir=/desired/location` must be passed with each cbdiag execution to make sure that `/tmp` is not used, and another mount that has less space constraint is used instead. Now, a user can permanently configure the tmp location for staging and temp files that are generated during execution of cbdiags.

To set a temp directory for cbdiag in `cb.conf`:

1. Edit `/etc/cb/cb.conf`
2. Add token `CbDiagTmpDir=/desired/location` (example: `/var/cb/data`). This should ideally be done on all nodes.

After it is set, the staging and temp files are generated at the `CbDiagTmpDir` location instead of `/tmp`.

ShowGdprBanner

Default: `(none)`

Controls the state of the EU Data Sharing Banner, which can be displayed at the top of each console page to caution users about sharing data. If `True`, the banner is always displayed. If `False`, the banner is never displayed. If no value is given, display of the banner is controlled by a setting on the Advanced tab of the Settings page in the console.

Change Note: New in version 6.3.0.

Chapter 8

RabbitMQ

This section describes RabbitMQ (cb-rabbitmq service) settings in the `cb.conf` file.

Settings in this Chapter

RabbitMQPort

Default: `5004`

The RabbitMQ AMQP Broker listening port (TCP).

RabbitMQManagementPort

Default: `5005`

The RabbitMQ Management HTTP API listening port (TCP). If this property value is updated, it must also be changed in the `rabbitmq_management/listener` property in `/etc/cb/rabbitmq/rabbitmq.config` file.

RabbitMQDistPort

Default: `25004`

The RabbitMQ Distributed Node Port (TCP). If this property value is updated, it must also be changed in the `kernel/inet_dist_listen_(min/max)` properties in the `/etc/cb/rabbitmq/rabbitmq.config` file

RabbitMQEpmdPort

Default: `4369`

The Erlang Port Mapper Daemon port (TCP). This port is used by the underlying runtime that RabbitMQ is based on. It is needed for distributed node discovery in clustered environments.

RabbitMQUser

Default: `cb`

The user account for broker authentication.

RabbitMQPassword

Default: (no default)

Sets the password to use for authentication with the broker. If the RabbitMQ HTTP Management Console is enabled, use the credentials in this file to gain initial access to the admin interface. From there, you can create a different user account with a set of credentials that are easier to mange.

RabbitMQDataPath

Default: `/var/cb/data/rabbitmq`

The data directory to which persistent queues will be written.

RabbitMQPidFile

Default: `/var/run/cb/rabbitmq/pid`

The RabbitMQ service PID file path.

Chapter 9

Live Response, Banning, Isolation

This section describes `cb.conf` settings that enable and disable or affect the behavior of Live Response (cb-liveresponse service), hash banning, and network isolation features.

Settings in this Chapter

CbLREnabled

Default: (none)

Enable/Disable Live Response functionality.

- If CbLREnabled has no value in cb.conf or is commented out, a console user with administrator privileges can enable or disable Live Response using the Advanced Settings page in the console.
- If CbLREnabled is `True`, Live Response is enabled and cannot be disabled through the console.
- If CbLREnabled is `False`, Live Response is disabled and cannot be enabled through the console.

Change Note: In version 6.3.0, the default for this setting was changed from `False` to no value. This means that newly installed Carbon Black EDR servers default to allowing administrators to enable and disable CB Live Response through the console. If you do not want the console interface for enabling or disabling CB Live Response activated, provide a value for this setting and be sure the setting not commented out.

CbLRCheckinTimeout

Default: `1200`

Timeout (in seconds) to wait for a sensor to initialize a Live Response session. The default is `1200` (20 minutes). When the time expires, the cblr session is discarded.

CbLRSessionTimeout

Default: `300`

The maximum time (in seconds) for a sensor to wait for a Live Response command to complete. The default is `300` (5 minutes).

CbLRSensorWaitTimeout

Default: `120`

The long poll duration (in seconds) of the sensor command query before returning a keepalive message to keep the Live Response session open.

CbLRMaxStoreSizeMB

Default: `0`

Maximum disk space (in megabytes) usable by Live Response functionality. When exceeded, all requests that could possibly require more disk space will be rejected. A value of `0` indicates unlimited disk space.

CbLrDefaultSessionTTLDays

Default: `7`

Default time-to-live (in days) for session data. Unless overwritten via the API, data from a Live Response session is deleted the specified number of days after the session closes.

CbLRMaxActiveSessions

Default: `10`

The maximum number of concurrent, active Live Response sessions allowed. Requests to create more than the specified number of sessions will be rejected.

BanningEnabled

Default: `True`

Enable/Disable banning functionality. If this is disabled, the banning of MD5 hashes via the UI or API will not be allowed by any user.

IsolationExclusionsEnabled

Default: (none)

Enable/Disable exclusion of specified IP addresses or URLs from network isolation when a sensor is isolated. If `True`, the Sensor Group add or edit page includes an interface for address exclusion. If not present or `False`, the interface is not active.

When this setting is changed in the cb.conf file, the server must be restarted for it to take effect.

Note: As of version v6.5.0, network isolation exclusions are supported for Windows sensors versions 6.2.4 and later, and OSX sensor version 6.2.7. Monitor sensor release notes for any additional platform support.

Chapter 10

Event Actions

This section describes the event actions settings in the `cb.conf` file.

Settings in this Chapter

AlertWriterSeverityCalcConfig

Default: (no default)

If this parameter is present, it specifies a path to the configuration file for the alert severity calculation algorithm. If the file does not exist, one will be written out and populated with the Carbon Black EDR server's default settings.

Chapter 11

Service Init Script

This section describes the service init script settings in the `cb.conf` file.

Settings in this Chapter

ProfileSvcInitBash

Default: `0`

Set this option to `1` if troubleshooting slow cb-enterprise services startup or shutdown. When enabled, it uses the bash shell's built-in `set -x` command in conjunction with a custom PS4 environment variable to report each command with its execution time in the `/var/log/cb/services/cb-enterprise.timing.out` file.

Important: Due to certain bash limitations, this command redirects all standard error data to the log file. As a result, if anything else also writes to `stderr`, there might be some unexpected text output while starting/stopping services.

Chapter 12

Statistics Reporting

This section describes the statistics reporting settings in the `cb.conf` file.

Important: Consult VMware Carbon Black Technical Support before attempting to change these parameters.

Carbon Black EDR server supports several methods of sharing runtime statistics information:

- Statistics that are viewable via the `/usr/share/cb/cbstats` command can also be sent to graphite by using the `--graphite` option.
- Some statistics collected by Java services use Coda Hale's Metrics package. These statistics are not accessible from cbstats, but can be enabled to sent directly to graphite in this section.

Settings in this Chapter

GraphiteHost

Default: `localhost`

Sets the host name of the graphite server to which runtime statistics should be sent.

GraphiteStatsUploadPort

Default: `2003`

Sets the graphite server port to which statistics should be sent.

GraphitePrefix

Default: `<Hostname of local host>`

Sets the metrics namespace prefix that should be used when uploading statistics to graphite. By default, if this property is not specified, the prefix will be the hostname of the local machine.

Chapter 13

Service Reporting

This chapter describes the service reporting settings in the `cb.conf` file. These settings are set up in this format:

`<SvcName><ReporterType><Prop>`

where ...

`<SvcName>` is one of the following:

- `CbSolr` – The cb-solr service configuration.
- `CbDatastore` – The cb-datastore service configuration.

and...

`<ReporterType>` is one of the following:

- `GraphiteReporter` – Data is sent over network sockets to a graphite server identified by `GraphiteHost`/`GraphiteStatsUploadPort` above properties.
- `LogReporter` – Data is written to the logback logging framework's `com.carbonblack.cbfs.Metrics` logger object. By default, this output will appear in `debug.log` (if INFO level logging is enabled). Also, the `logback.conf.xml` file can be modified so that metrics are written to a separate file.

and ...

`<Prop>` is one of the following:

- `Enabled` – Indicates whether or not particular reporter is enabled.
- `Interval` – Specifies in seconds how often the data is reported.
- `Filter` – An optional comma separate string of filter regular expressions that can be used to limit which metrics are to be reported. For example `.*jvm.*` will report all metrics that have 'jvm' somewhere in the name whereas `jvm\..+` will report all metrics that begin specifically with "jvm." as the first hierarchy element. If this property is not specified, all metrics are reported.

Use the component definitions above as a guide to the function of the parameters listed here.

Settings in this Chapter

CbSolrStatsGraphiteReporterEnabled

Default: `False`

CbSolrStatsGraphiteReporterInterval

Default: `60`

CbSolrStatsGraphiteReporterFilter

Default: (no default)

CbSolrStatsLogReporterEnabled

Default: `False`

CbSolrStatsLogReporterInterval

Default: `60`

CbSolrStatsLogReporterFilter

Default: (no default)

CbDatastoreStatsGraphiteReporterEnabled

Default: `False`

CbDatastoreStatsGraphiteReporterInterval

Default: `60`

CbDatastoreStatsGraphiteReporterFilter

Default: (no default)

CbDatastoreStatsLogReporterEnabled

Default: `False`

CbDatastoreStatsLogReporterInterval

Default: `60`

CbDatastoreStatsLogReporterFilter

Default: (no default)

Chapter 14

Threat Intel

This section describes the VMware Carbon Black Threat Intel settings in the `cb.conf` file.

Settings in this Chapter

FeatureThirdPartySharing

Default: `False`

This setting relates to third-party file analysis. It allows administrators to choose whether to share full binaries with VMware Carbon Black analysis partners in the **Endpoint Activity Sharing** window. Administrators can also share this data at the sensor group level within the **Sharing** tab of the **Create/Edit Group Settings** windows. If this is set to `False`, the administrator will not see the inspection options.

Important: Consult VMware Carbon Black Technical Support before you change this parameter.

TicUrl

The address of the Threat Intel server.

Chapter 15

Syslog Template

This section describes the syslog template settings in the cb.conf file. The default template files are located in `/usr/share/cb/syslog_templates`.

For each of these options, if a value is not specified, the system default template is used. Use the cbsyslog tool to retrieve the system default template.

Settings in this Chapter

WatchlistSyslogTemplateProcess

Sets the path to the Jinja2 Template that is used to format process watchlist hits before sending the data to syslog. Use `/usr/share/cb/cbsyslog` to modify and test this path.

See the *VMware Carbon Black EDR User Guide*.

WatchlistSyslogTemplateBinary

Sets the path to the Jinja2 Template that is used to format binary watchlist hits before sending the data to syslog. Use `/usr/share/cb/cbsyslog` to modify and test this path.

See the *VMware Carbon Black EDR User Guide*.

BinaryInfoSyslogTemplateObserved

Sets the path to the Jinja2 Template used to format binary information events before sending the data to syslog. These events are created the first time a binary (identified by its MD5 hash), is observed on any sensor that is associated with the Carbon Black EDR server.

See the *VMware Carbon Black EDR User Guide*.

BinaryInfoSyslogTemplateGroupObserved

Sets the path to the Jinja2 Template that is used to format binary information for new sensor group events before sending the data to syslog. These events are created the first time a binary, as identified by its MD5 hash value, is observed by a new sensor group.

See the *VMware Carbon Black EDR User Guide*.

BinaryInfoSyslogTemplateHostObserved

Sets the path to the Jinja2 Template that is used to format binary information for new host events before sending the data to syslog. These events are created the first time a binary, as identified by its MD5 hash value, is observed by a new sensor.

See the *VMware Carbon Black EDR User Guide*.

Chapter 16

Third-Party Authentication

This section describes the third-party authentication settings in the `cb.conf` file.

Settings in this Chapter

TwoFactorAuthCallbackModulePath

This setting enables two-factor authentication on your Carbon Black EDR server using the Duo plugin.

To enable two-factor authentication, you must perform additional configuration steps, which are discussed in “Integrating with Third-Party Authentication” in the *VMware Carbon Black EDR User Guide*.

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
```

CbStatsDaemon

Default: `False`

Run and monitor cb-stats metrics collection daemon. This is a supplemental tool that queries the Carbon Black EDR statistics framework and records data to a file or graphite. See
`/usr/share/cb/setup/cbstats.conf.template`.

Change Note: New in version 6.1.

SolrDiskType

Default: `auto`

Indicates the disk type (spinning/solid state) for solr cores. The values are:

- `auto` – autodetect (default)
- `hdd` – spinning disk
- `ssd` – solid state

Change Note: New in version 6.1.

NetfilterConntrackMax

Default: `262144`

If this is non-zero, cbstartup will update the value of `/proc/sys/net/netfilter/nf_conntrack_max` on the server's system.

Change Note: New in version 6.1.