

# Carbon Black Cloud Sensor Installation Guide

18 July 2024

VMware Carbon Black Cloud

You can find the most up-to-date technical documentation on the VMware by Broadcom website at:

<https://docs.vmware.com/>

**VMware by Broadcom**  
3401 Hillview Ave.  
Palo Alto, CA 94304  
[www.vmware.com](http://www.vmware.com)

Copyright © 2017-2024 Broadcom. All Rights Reserved. The term “Broadcom” refers to Broadcom Inc. and/or its subsidiaries. For more information, go to <https://www.broadcom.com>. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

# Contents

Preface	9
<b>1 Getting Started with Sensor Installation</b>	<b>10</b>
Before you Install Sensors on Endpoints	10
About Sensors and Policy Assignments	11
Local Scan Settings	12
Setting Antivirus Exclusion Rules	12
Method 1: Invite Users to Install Sensors on Endpoints	13
Invite Users to install Sensors	14
Send a new Installation Code	14
Method 2: Install the Sensor on the Endpoint by using the Command Line or Software Distribution Tools	15
Obtain a Company Registration Code	15
Obtain a Deregistration Code	16
Download Sensor Kits	17
Installing Sensors on Endpoints	18
<b>2 Installing Linux Sensors on Endpoints</b>	<b>19</b>
Unpack the Agent	20
Verify the Unpacked Tar-ball Contents	20
Linux Kernel Requirements for Linux Sensor Versions 2.10+	22
Check BPF-based Sensor Requirements	23
Check Requirements for CentOS, RHEL, Oracle RHCK, and Amazon Linux	24
Check Requirements for Oracle UEK	25
Check Requirements for SUSE and OpenSUSE	26
Check Requirements for Debian	28
Check Requirements for Ubuntu	29
About the Linux Sensor cfg.ini File	31
Linux Installer Command Line Parameters	33
Customize the Linux Feature Selection	33
Install a Linux Sensor on a Single Endpoint	34
Install a Linux Sensor on an Endpoint using the RPM/DPKG Installer	35
Install a Linux Sensor on an Endpoint using the RPM Installer on an Amazon Linux 2 Graviton EC2 Instance	36
Install a Linux Sensor on an Endpoint that Automatically Registers the First Time it is Started	36
<b>3 Installing macOS Sensors on Endpoints</b>	<b>38</b>
Approve the Kernel Extension (macOS 10.13 – macOS 11)	39

- Manually Approve the KEXT 39
- Approve the KEXT via MDM 40
- Approving the KEXT via MDM for Big Sur 40
- Identify Devices with Sensors that do not support the Operating System or need KEXT Approval 41
- Approving the System Extension and Network Extension for macOS 11+ 41
  - Approve the System Extension via MDM 41
  - Approve the Network Extension Component of the System Extension via MDM 43
- Full Disk Access Requirement for the macOS Sensor 44
  - Grant the Sensor Full Disk Access by using MDM 44
  - Manually Grant Sensor Full Disk Access 46
- Restart Requirements for macOS 10.15+ 49
- Special Considerations for the macOS Sensor on Big Sur 49
  - Install a KEXT-enabled Sensor on Big Sur 49
  - Switch the macOS Sensor Kernel Type on Big Sur during Update 50
  - Toggle between Kernel Extension and System Extension in Big Sur 50
- Manually Install and Approve the Sensor on macOS 11+ 51
- macOS Sensor Command Line Install 54
  - Extract and Prepare the macOS Install Files 55
  - Perform a macOS Sensor Command Line Installation 56
  - macOS Command Line Parameters 56
  - macOS Command Line Install Examples 58
- Deploying macOS Sensors on Big Sur and Later by using Jamf Pro 58
  - Obtain and Prepare the Sensor 59
  - Create a Package by using Jamf Composer 59
  - Modify the Installation Script 60
  - Upload macOS Sensor DMG and Installation Script to Jamf Pro 61
  - Creating a Configuration Profile 61
    - Configure Configuration Profile General Settings 61
    - Set Privacy Preferences Policy Control in the Configuration Profile 62
    - Enable System Extension Payloads in the Configuration Profile 63
    - Enable Kernel Extension Payloads in the Configuration Profile 63
    - Set the Content Filter in the Configuration Profile 64
  - Create a Software Distribution Policy 65
  - Create and Assign Smart Computer Groups 66
- Validate a Healthy System Extension Sensor through RepCLI 68
- Address the Extension Warning Post-install 69
- Resolve macOS Sensor Upgrade or Install Error due to Existing System Extension 70
- macOS Services, Utilities, and Uninstaller 70
- Installing macOS Sensors on Endpoints by using Workspace ONE UEM 71
  - Prepare to Install macOS Sensors 72
  - Deploying the Carbon Black Cloud sensor for macOS Manually with System Extensions 72

- Creating a Configuration Profile 73
- Manually Install and Approve the Sensor on macOS 11+ 78
- Confirm the Carbon Black Cloud Sensor Installed on the macOS Device 81
- Deploying the Carbon Black Cloud sensor for macOS as Managed Application 82
  - Set Up the Application Installer 83
  - Install the Carbon Black Cloud Sensor for macOS as a Managed Application 84
  - Confirm the Carbon Black Cloud Sensor for macOS Installed as Managed Application 88

## 4 Installing Sensors on Endpoints in a VDI Environment 90

- Creating Multiple Golden or Primary Images 90
- Carbon Black Windows Sensors with Horizon Virtual Desktops 91
  - Carbon Black Windows Sensor Policy Setting Recommendations for Horizon Instant Clones 92
  - Install the Carbon Black Windows Sensor on Horizon Instant Clones 94
  - Horizon Instant Clones and Carbon Black Windows Sensor Installation Known Issues and Mitigation 97
  - Carbon Black Windows Sensor Policy Setting Recommendations for Horizon Full Clones 98
  - Install Carbon Black Windows Sensors on Horizon Full Clones 100
  - Install Carbon Black Windows Sensors in Horizon Full and Instant Clone Mixed Environments 102
  - Horizon Linked-Clones and Carbon Black 3.6+ Windows Sensor Best Practices 104
- Carbon Black Linux Sensors with VMware Horizon Virtual Desktops 104
  - Carbon Black Linux Sensor Policy Setting Recommendations for Horizon Golden Images 105
  - Horizon Golden Image Considerations for Carbon Black Linux Sensors 105
  - Horizon Instant Clone Considerations for Carbon Black Linux Sensors 106
  - Install the Carbon Black Cloud Linux Sensor on a Horizon Golden Image and Create Instant Clones 106
- Carbon Black Windows Sensors with Citrix Virtual Desktops 108
  - Citrix Golden Image Considerations for Carbon Black Sensors 108
  - Carbon Black Policy Setting Recommendations for Citrix Golden Images 109
  - Install the Sensor on a Citrix Golden Image 111
  - Citrix Clone Considerations for Carbon Black Windows Sensors 112
  - Carbon Black Policy Setting Recommendations for Citrix Clones 113
  - Citrix MCS and Carbon Black Windows Sensor 115
  - Citrix PVS and Carbon Black Windows Sensor 116
  - Citrix PVS and Carbon Black Windows 3.7 Sensor 117
- Carbon Black Windows Sensors with vSphere Clients 118
- Carbon Black Linux Sensors with vSphere Clients 121

## 5 Installing Windows Sensors on Endpoints 123

- Verifying Windows Sensor Digital Signatures 124

- Windows Sensor Rollback 127
- Local Scan Settings and the AV Signature Pack 129
  - Disable Automatic Signature Updates and use the Standalone Installer 129
  - To Update the AV Signature Pack by using the RepCLI Command 130
- Windows Sensor Command Line Parameters 130
- Windows Sensor Supported Commands 131
  - Obfuscation of Command Line Inputs 134
- Windows Command Line Install on Endpoints — Examples 135
- Windows Sensor Log Files and Installed Services 136
- Installing Windows Sensors on Endpoints by using Group Policy 136
  - Create a Microsoft Installer Transform (.MST) File 136
  - Automatically Create a Windows Installer .MSI Log 137
  - Install Sensors by using Group Policy 138
- Installing Windows Sensors on Endpoints by using SCCM 138
  - Add the Sensor Application to SCCM 139
  - Deploy the Sensor Application using SCCM 141
  - Verify that the Sensor Application was Deployed via SCCM 142
- Installing Carbon Black Cloud Sensor for Windows by Using Workspace ONE UEM 143
  - Deploy the Carbon Black Cloud Sensor for Windows as Managed Application in Workspace ONE UEM 143
  - Verify that Carbon Black Cloud Sensor for Windows Installed as Managed Application with Workspace ONE UEM 144

## 6 Searching for Sensors 145

## 7 Updating Sensors on Endpoints 148

- About Updating Sensors on Endpoints through the Console 149
  - Update Sensors through the Console 150
- Update Windows Sensors on Endpoints through the Command Line 151
- Update Sensors on Endpoints by using Group Policy 152
- Update Sensors on Endpoints that were Deployed by using SCCM 153
- Update Linux Sensors on Endpoints through the Command Line 153
- View Progress of Sensor Updates 155
- Sensor Status and Details — Asset Groups 158
- Sensor Status and Details — Sensor Groups 163
- Endpoint Filters 167
- Bypass Reasons 171

## 8 Uninstalling Sensors from Endpoints 174

- Uninstall Sensors from the Endpoint by using the Console 174
- Require Codes to uninstall Sensors at an Endpoint 175
- Uninstall a Linux Sensor from an Endpoint 176

- Uninstall a 3.5+ macOS Sensor from an Endpoint 176
- Uninstall a pre-3.5.1 macOS Sensor from an Endpoint 177
- Uninstall a Windows sensor from an Endpoint 177
  - Uninstall Windows Sensors from an Endpoint by using Group Policy 177
  - Enable SCCM to Uninstall a Windows Sensor from an Endpoint 178
- Delete Deregistered Sensors from Endpoints 178

## 9 Managing Sensors for VM Workloads 179

- Installing Sensors on VM Workloads 179
  - Prepare Your Workloads Environment for Sensor Installation 181
  - Install Sensors on VM Workloads 181
- Update Sensors for Workloads from the Console 186
- Update Linux Sensors on Workloads through the Command Line 186
- Uninstall Linux Sensors from Workloads 188
- Uninstall Windows Sensors from Workloads 188
- Delete Deregistered Sensors from Workloads 188

## 10 Installing Sensors on AWS Workloads 190

- Install Sensors on AWS Workloads by Using Sensor Installation Scripts 190
- Install Sensors on AWS Workloads by Using the Console 191

## 11 Setting up the Container Security Environment 194

- Roles and Users for Containers 194
  - Using and Creating Roles for Containers 195
    - Add a Container Role 197
  - Create a User Account for Containers 200
- Adding Clusters and Installing Kubernetes Sensors 200
  - Add a Cluster and Install the Kubernetes Sensor 201
    - Private Container Registry 205
- Check the Kubernetes Sensor Status and Health 205
- Installing a Containerized Sensor 207
  - Set up a Containerized Sensor 208
  - Install a Containerized Sensor 210
    - Install a Containerized Sensor on a Docker Client 213
    - Install a Containerized Sensor on an ECS Cluster 214
  - Validate the Container Image Signature 217
- Setting up CLI Client for Image Scanning 219
  - Download a CLI Client 220
  - Add and Configure a CLI Client 221
- Carbon Black Container Operator Technical Reference 224
  - Manually Deploy the Container Operator 224

- Uninstall the Container Operator 225
- Manually Deploy the Container Agent 225
- Openshift 226
- Reading Metrics by using Prometheus 229
- Custom Resources Definitions 230
- Changing Components Resources 234
- Configuring Container Services to use HTTP Proxy 236
- Changing the Image Source 238
- Operator Role-based Access Control 239
- Container Operator Developer Instructions 240
- Helm Charts 242

## **12** Signature Mirror Instructions 247

## **13** Configuring Carbon Black Cloud Communications 248

- Configure a Firewall 248
  - Disable CURL CRL CHECK 252
- Configure a Proxy 252
  - Connection Mechanism Precedence 254
  - Configure a Proxy for Windows after Sensor Installation 255
  - Configure a Proxy for Linux (all Sensor Versions) 256
  - Configure a Proxy for Linux (Sensor Versions 2.11.1+) 257
  - macOS Proxy Server Information 257
- Cryptographic Protocol Requirements 257



# Preface

This guide provides installation and configuration instructions for Carbon Black Cloud Sensors.

You can install a Carbon Black Cloud sensor on Windows, macOS, and Linux endpoints and on endpoints in VDI environments. The sensor provides data from the endpoints to Carbon Black Cloud analytics. You can also secure VMware workloads by using the Carbon Black Cloud.

---

**Note** For instructions on securing Kubernetes containers, see [Chapter 11 Setting up the Container Security Environment](#) .

---

## Intended Audience

This documentation provides sensor installation, update, and uninstall instructions for administrators, incident responders, and others who will operate the Carbon Black Cloud.

Staff who manage Carbon Black Cloud activities should be familiar with operating systems, web applications, installed software, desktop infrastructure (especially in-house procedures for software roll-outs, patch management, and anti-virus software maintenance), and the effects of unwanted software.

# Getting Started with Sensor Installation

# 1

You can install a Carbon Black Cloud sensor on Windows, macOS, and Linux endpoints, and on endpoints in VDI environments. The sensor provides data from the endpoints to CB analytics.

The following instructions describe how to install sensors on endpoints. To install and manage sensors on workloads, see [Chapter 9 Managing Sensors for VM Workloads](#).

If you are using a Sensor Gateway, see [Installing Carbon Black Cloud Sensors for Sensor Gateway](#).

## Method 1: Invite Users to Install Sensors on Endpoints

- Invited users receive an email that contains an installation code; each invited user installs the sensor directly on an endpoint. This method is not available for Linux sensors.
- This method is useful for installing sensors to a small number of endpoints.

## Method 2: Install the Sensor on the Endpoint by using the Command Line or Software Distribution Tools

- The command line method allows for small-scale deployments and testing.
- A scripted or automated method installs the sensor by using software distribution tools. This method is useful when installing sensors across a large number of endpoints.

Read the following topics next:

- [Before you Install Sensors on Endpoints](#)
- [Method 1: Invite Users to Install Sensors on Endpoints](#)
- [Method 2: Install the Sensor on the Endpoint by using the Command Line or Software Distribution Tools](#)

## Before you Install Sensors on Endpoints

Make sure that endpoints meet the operating environment requirements (OER) for the Carbon Black Cloud products that you have purchased.

See the following Carbon Black Cloud Sensor Operating Environment Requirements:

- [Windows Sensor \(on Windows Desktop\) OER](#)
- [Windows Sensor \(on Windows Server\) OER](#)
- [Linux Sensor OER](#)
- [macOS Sensor OER](#)

---

**Note** Some sensor names contain the product name “CB Defense.” This is correct: the same sensors apply for all Carbon Black Cloud products.

Before you install sensors on endpoints, read the following topics. Set up your AV exclusions, and configure your environment for proxy and firewall settings (see also [Chapter 13 Configuring Carbon Black Cloud Communications](#)).

---

## About Sensors and Policy Assignments

Each sensor is assigned a policy that determines what policy rules apply to the sensor.

---

**Important** Asset Groups is available to Carbon Black Cloud customers on 27 November 2023. Carbon Black recommends that you upgrade from Sensor Groups to Asset Groups as soon as it is operationally feasible for your organization. Sensor Groups will be phased out by 01 December 2024. See [Asset Groups and Sensor Groups](#) in the *User Guide*.

---

By default, each new sensor is initially assigned the Standard policy unless one of the following conditions applies:

- You define an alternate policy during a command line installation.
- You have previously created Asset Groups, the installed sensor matches an Asset Group’s criteria, and the target policy is not the Standard policy. All the sensors in the Asset Group receive an automatic assignment to a policy, which is based on the metadata that is associated with the sensor and the criteria that you define.
- You have previously created Sensor Groups, the installed sensor matches a Sensor Group’s criteria, and the target policy is not the Standard policy. All the sensors in the Sensor Group receive an automatic assignment to a policy, which is based on the metadata that is associated with the sensor and the criteria that you define.

Sensor Groups require the following sensor versions:

- Windows sensors v3.1+
- macOS sensors v3.2 +
- Linux sensors v2.5 +

---

**Note** Policy assignments do not apply to the Carbon Black Cloud Audit and Remediation standalone product.

---

## Local Scan Settings

The local scan feature is only available for Windows sensors 2.0 and later. It is not available for the Audit and Remediation Standalone product, Linux sensors, or macOS sensors.

For more information about Local Scan Settings for Windows, see [Local Scan Settings and the AV Signature Pack](#). To configure local scan settings in the console, see *Configure Local Scan Settings* in the User Guide.

## Setting Antivirus Exclusion Rules

You can create antivirus (AV) exclusion rules, including those specific to various endpoint platforms.

To run as usual, other AV products require custom rules.

If you use other security products, create the following exclusions for the Carbon Black Cloud sensor:

### Linux

<code>/var/opt/carbonblack/</code>
<code>/opt/carbonblack/</code>

### macOS

<code>/Applications/Confer.app/</code>
<code>/Applications/VMware Carbon Black Cloud</code>
<code>/Library/Application Support/com.vmware.carbonblack.cloud/</code>
<code>/Library/Extensions/CbDefenseSensor.kext</code>

### Windows Folders

<code>C:\Program Files\Confer\</code>
<code>C:\ProgramData\CarbonBlack\</code>

### Windows Files

<code>C:\Windows\System32\drivers\cti file.sys</code>	<code>C:\Windows\System32\drivers\ct inet.sys</code>	<code>C:\Windows\System32\drivers\cbe lam.sys</code>
<code>C:\Windows\system32\drivers\cbd isk.sys</code>	<code>C:\windows\system32\CbAMSI.dll</code>	<code>C:\windows\system32\ctiuser.dll</code>
<code>C:\windows\syswow64\CbAMSI.dll</code>	<code>C:\windows\syswow64\ctiuser.dl l</code>	<code>C:\Windows\Syswow64\ctintev.dll</code>

C:\Program Files\Confer\BladeRunner.exe	C:\Program Files\Confer\CbNativeMessaging Host.exe	C:\Program Files\Confer\RepCLI.exe
C:\Program Files\Confer\RepMgr.exe	C:\Program Files\Confer\RepUtils.exe	C:\Program Files\Confer\RepUx.exe
C:\Program Files\Confer\RepWAV.exe	C:\Program Files\Confer\RepWmiUtils.exe	C:\Program Files\Confer\RepWSC.exe
C:\Program Files\Confer\Uninstall.exe	C:\Program Files\Confer\VHostComms.exe	C:\Program Files\Confer\Blades\LiveQuery\o squeryi.exe
C:\Program Files\Confer\scanner\scanhost.e xe	C:\Program Files\Confer\scanner\upd.exe	

## Set Antivirus Exclusion Rules

Use this procedure to create AV exclusion rules, including those specific to various endpoint platforms.

---

**Note** Some security vendors may require a trailing asterisk (\*) to signify all directory contents.

---

### Procedure

- 1 On the left navigation pane, click **Enforce > Policies**.
- 2 Select the policy.
- 3 Click the **Prevention** tab and expand **Permissions**.
- 4 Click **Add application path**.
- 5 Enter the AV's recommended file/folder exclusions from the security vendor.
- 6 Set the operation attempt **Performs any API operation** to **Bypass**.
- 7 To apply the changes, click **Confirm** and then click **Save**.

## Method 1: Invite Users to Install Sensors on Endpoints

This method is useful when you have a small number of sensors to install, or when software distribution tools are not available. This method is not available for Linux sensors.

The installation code will expire after seven (7) days.

---

**Important** The user on the endpoint must have administrator privileges to install the sensor.

---

**Note** With the release of the Windows 3.6 sensor, you can supply either the installation code or the company code to install the sensor.

---

## Invite Users to install Sensors

You can invite users to install sensors on their Windows and macOS endpoints.

### Procedure

- 1 On the navigation bar, click **Inventory** > **Endpoints**.
- 2 Click **Sensor Options** and then click **Send installation request (Windows and macOS)**.
- 3 Add a single user or multiple users. To add multiple users, type a comma-separated list of email addresses.

---

**Note** If you are adding multiple users, the following maximums apply:

- 700K characters
  - 28.5K addresses with an average address length of 21 characters
- 

- 4 Click **Send**.

### Results

Users receive an email invitation that contains the installer download link and a unique single use installation code. The installation code expires after one week. If the installation code expires, follow the procedure [Send a new Installation Code](#)

Users should follow the instructions in the email to install the sensor. In the email, users click on the appropriate OS installer link to download the sensor.

---

**Note** We recommend that you inform users in advance that you're sending the email invitation. In the advance notification, tell the users which version to download (32-bit or 64-bit). The 32-bit variant of the sensor does not run on a 64-bit version of Windows. Instruct the users that they should copy/paste the installation code into a plain text editor, and then copy/paste that entry into the installer. Copy/pasting the installation code directly from the console does not always work properly.

If the user is installing a sensor version prior to 3.0, they must use the legacy 6-digit code instead of the extended 3.0+ installation code. You can find the 6-digit code on the Endpoints page; expand the user for whom the request is being made, and provide them with the listed v1-v2 code.

---

## Send a new Installation Code

If installation codes have expired, you can follow these steps to send new installation codes to users.

### Procedure

- 1 Sign in to the Carbon Black Cloud console.
- 2 On the navigation bar, click **Inventory** and then click **Endpoints**.

- 3 Search for and select the sensors that have expired installation codes.
- 4 Click **Take Action** and click **Send new installation code**.

## Method 2: Install the Sensor on the Endpoint by using the Command Line or Software Distribution Tools

You can install sensors on the command line, or by using a scripted or automated method such as Group Policy or systems management tools.

The latter method is useful when you are installing sensors across a large number of endpoints.

---

**Note** Sensors automatically try to detect proxy settings during initial installation. This should be tested. If the automatic detection does not succeed, you must define the parameters to include the proxy IP address and port in the MSI command line. See [Configure a Proxy](#).

---

**Important** You must have administrator privileges to install the sensor.

---

Follow these procedures in the order listed:

- 1 [Obtain a Company Registration Code](#)
- 2 [Download Sensor Kits](#)
- 3 [Installing Sensors on Endpoints](#)

### Obtain a Company Registration Code

To register new sensors you must obtain a company registration code.

Copy a generated company code and supply it when performing automated sensor installation by software distribution system or imaging.

If you generate a new company code, it invalidates the previous one and cannot be undone. You must regenerate a company code in case of concerns that the current code has been compromised and is being used by unauthorized individuals. If you regenerate the company code, you must update any software distribution tools or any existing installation scripts to use the new code.

If you change the company code and install sensors using the new code, the old sensors continue to operate. Installed sensors are unaffected. Only new installation packages must use the new code.

---

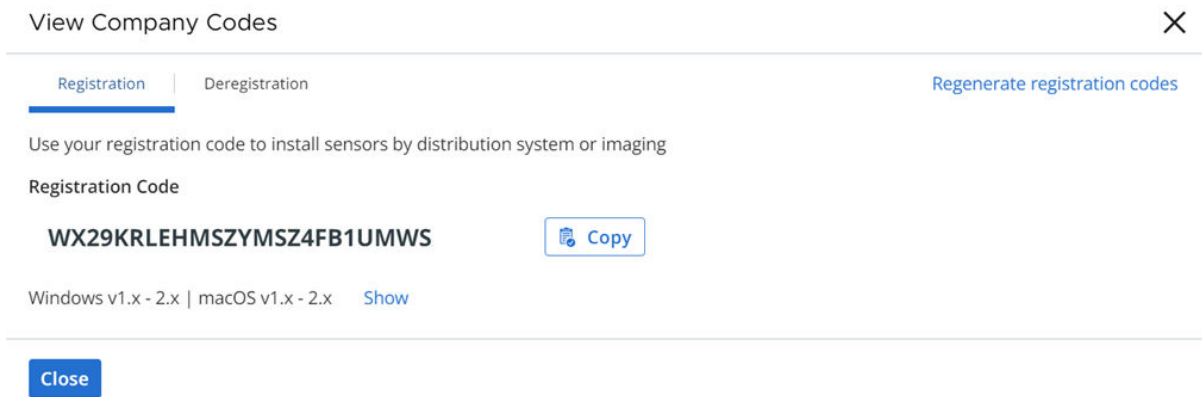
**Note** For 1.x – 2.x Windows or macOS sensor versions, the length of the company registration code is shortened and contains eight characters.

Use the 1.x – 2.x company code to update Windows or macOS sensors prior to version 3.0. The process of supplying the code during sensor installation remains the same. You must update any software distribution tools or any existing installation scripts to use the extended codes.

---

**Procedure**

- 1 On the left navigation bar, click **Inventory > Endpoints** or **Inventory > VM Workloads**.
- 2 Click **Sensor Options > View company codes**.



- 3 Click **Copy** next to the code.

**Obtain a Deregistration Code**

You can obtain a company deregistration code to use when uninstalling Windows or macOS sensors.

The company deregistration code can be regenerated at any time if the current code has been compromised.

To view and supply the company code for uninstalling sensors, locate the **Deregistration** tab, part of the **View Company Codes** window, and copy the deregistration code.

---

**Note** Only macOS and Windows sensors can be uninstalled by using a company deregistration code. [Uninstall a Linux Sensor from an Endpoint](#) by using the command line.

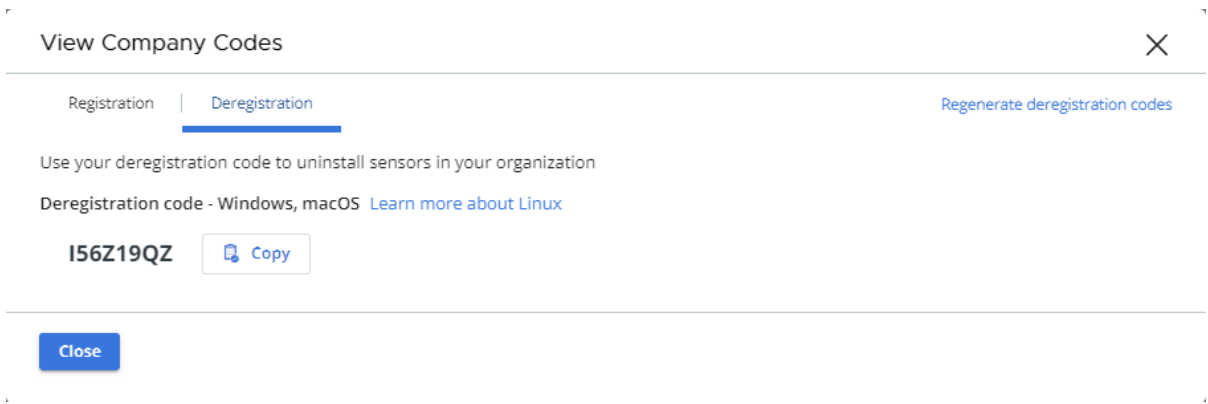
---

**Procedure**

- 1 On the left navigation bar, click **Inventory > Endpoints** or **Inventory > VM Workloads**.
- 2 Click **Sensor Options > View company codes**.



3 Click the **Deregistration** tab.



4 Click **Copy** next to the code.

## Download Sensor Kits

You must download a sensor kit that matches the operating system of the endpoint.

### Procedure

- 1 Sign in to the Carbon Black Cloud console.
- 2 On the navigation bar, click **Inventory** and then click **Endpoints**.
- 3 Click **Sensor Options** and click **Download sensor kits**.

#### 4 Select the appropriate sensor kit version and click the link to download it.

Download Sensor Kits
✕

---

Learn more about sensors from: [Supported Operating Systems](#), [Sensor Release Notes](#), and the [Sensor Installation Guide](#)

OS	SENSOR VERSION	ACTION
Windows 64-bit	3.8.0.535 <span style="float: right;">▼</span>	<a href="#">Download Kit</a>
Windows 32-bit	3.8.0.535 <span style="float: right;">▼</span>	<a href="#">Download Kit</a>
macOS ⓘ 10.14 - 10.15, 11, 12	3.4.4.51 <span style="float: right;">▼</span>	<a href="#">Download Kit</a>
RHEL / CentOS / Oracle Linux	2.13.1.933911 <span style="float: right;">▼</span>	<a href="#">Download Kit</a>
Ubuntu & Debian	2.13.1.933911 <span style="float: right;">▼</span>	<a href="#">Download Kit</a>
SUSE / SLES 12 & 15	2.13.1.933911 <span style="float: right;">▼</span>	<a href="#">Download Kit</a>
Amazon Linux 2	2.13.1.933911 <span style="float: right;">▼</span>	<a href="#">Download Kit</a>

Close

## Installing Sensors on Endpoints

Sensor installation on endpoints varies by operating system and environment.

See the following sections for specific sensor installation instructions:

- [Chapter 2 Installing Linux Sensors on Endpoints](#)
- [Chapter 3 Installing macOS Sensors on Endpoints](#)
- [Chapter 4 Installing Sensors on Endpoints in a VDI Environment](#)
- [Chapter 5 Installing Windows Sensors on Endpoints](#)

# Installing Linux Sensors on Endpoints

# 2

This section describes how to install Linux sensors from the command line.

---

**Important** Before you begin the processes described here, read [Chapter 1 Getting Started with Sensor Installation](#). It contains highly relevant information to help you succeed in your sensor installation.

---

Before you can install sensors, you must perform the following steps:

[Obtain a Company Registration Code](#)

[Download Sensor Kits](#)

The sensor kit is a .tgz with the format `cb-psc-sensor-<DISTRO>-<BUILD-NUMBER>.tgz`.

With the release of the Carbon Black Cloud v2.5.0 Linux sensor, Audit and Remediation and Enterprise EDR are supported on the Linux platform. The Carbon Black Cloud Linux sensor is highly modularized. It can support independent runtime enablement of Enterprise EDR and Audit and Remediation. You can manually customize the installer package to install only desired features. To install Audit and Remediation only, see [Customizing the Carbon Black Cloud Linux feature selection](#).

To configure a proxy for a Linux installation, see [Configure a Proxy for Linux \(all Sensor Versions\)](#).

---

**Note** If the company registration code contains special characters (!, #, \*, \$, etc.) and is not quoted, the installation will immediately terminate. Double quotation marks are not an acceptable substitute to single quotes.

---

Read the following topics next:

- [Unpack the Agent](#)
- [Verify the Unpacked Tar-ball Contents](#)
- [Linux Kernel Requirements for Linux Sensor Versions 2.10+](#)
- [About the Linux Sensor cfg.ini File](#)
- [Linux Installer Command Line Parameters](#)
- [Customize the Linux Feature Selection](#)
- [Install a Linux Sensor on a Single Endpoint](#)

- [Install a Linux Sensor on an Endpoint using the RPM/DPKG Installer](#)
- [Install a Linux Sensor on an Endpoint that Automatically Registers the First Time it is Started](#)

## Unpack the Agent

The first step in installing a Linux sensor on an endpoint is to unpack the agent.

### Procedure

- 1 Create a root-owned temporary install directory named `cb-psc-install` on the endpoint; do not use a shared folder such as `/tmp` or `/var/tmp`:

```
$ sudo mkdir cb-psc-install
```

- 2 Extract the contents of the installer package into the temporary directory you created. Replace `cb-psc-sensor-<DISTRO>-<BUILD-NUMBER>.tgz` with the filename of the installer package.

```
$ sudo tar -C cb-psc-install -xzf cb-psc-sensor-<DISTRO>-<BUILD-NUMBER>.tgz
```

---

**Note** In regards to the `-xzf` option, the `z` is optional.

---

## Verify the Unpacked Tar-ball Contents

With the release of the 2.11.2 Linux sensor, digital-integrity verification of all tar-ball contents is enabled.

Perform the following steps to verify integrity after you unpack the TGZ and before you install the sensor.

### Prerequisites

You need two tools that are usually pre-installed on Linux:

- GnuPG package (for `/usr/bin/gpg` tool)
- SHA256 checksum tool: `/usr/bin/sha256sum`

### Procedure

- 1 Copy the text below to a file named: `public.asc`.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2.0.14 (GNU/Linux)
mQENBFMsJ4kBCACp93MIPVj1NVY7HEZm+gFtRU7lihQr+7lYIXCL59nXSaoniI/T
eihTlGTjWoJ7fTqzstA2Syt+Mmq7VecOVor0mJgBjw1CFXlzApZI1tTnq9Iio6Xs
2fxP08n1kKXQF1G7x62Y7EjJaFAF1fcMvrHPc43CTM455tRW9V5ODETGyt9DByf3
R2w11NzqGUzonElwIKib2zUJ+XSIvIU5Go60t+BDfmJMdTtAxoyZ79b+sTl//lcqBe0WhSX48Fn6CfFzeH84/1CPcf/
i1MB5qE9Vjk6iR2Z9M4xB1YKGUZT/Z1L9yurt
bs3tp5kSajgYrkCYaYkHY/so+E01zbQa99vABEBAAG0JWJpdDlidWlsZCAoYml0
```

```

OWNzKSA8c3VvcG9ydEBiaXQ5LmNvbT6JATgEEwECACIFAlMsJ4kCGwMGCwkIBwMC
BhUIAgkKCwQWAgMBAh4BAheAAoJEEhbsN9qxXcER9IH/i8dg4q4cK1lLLFr8vEi
30I1/kokNCacNdbH0gPV1CiGaVrcmgC1pAZuO8HjyEhFplrWU7rRPFhdLgupN95I
rFY5CJ9r+FO99SJTkhJY7vM/4rSOVTat+ZAJgJ/lk3Q148jUK+vOhKa/9I21ys5N
g7OR0EST7fLBNigKIXgy44Zb5GjzJBAQ1vbGNuErduldrR4lIueFjk6QdbVp8SN1
kD/SgqH1rmBiCeX2YMFcudDT6YQ7DnKfzC+GtKp+Lbs2ZyYH96bIeSNKA008x95f
y0dWEsxyvrsoAvl9zIml5mg2mnLHNxiDV54ABvtPk27TKeZBxlQPWBU3TZCmNkdn
umy5AQ0EUywniQEIALtfcwslAk9pyfgj0GqznkalLrln8KsznYAtUCQUl8odtHLP
QW4puA713glzLLk+IU69SFUHYdUI10I2VP3M9gRWuQQ7NNXaniPXF0xTCrLPPYH7
Y4d7VlK7q/Fu+qP+pobT9RV9Z0hINmm5mYeeNneCqWzFdmdOYqMp592gdqsKA9E6
M70jSszYbL9ZVCENiCM11q+CqciddZkAN0MrOP78w7sMXPqiJ6oRTBDy43GcHf1Z
BwC1ePknHQ1tXrCxY4ns/+nbhNgx5U0CtZMk034Cj75+Auyen2sbsgFj889Gjxoj
SzZ2elWzKbjiC9sZJjI++ENDsH79Vi84u98tp1MAEQEAAyKbHwQYAQIACQUUywn
iQIbDAAKCRBIW7DfasV3BBYBCACMw7aKV2vsVVVQ8GSfe6gjnR5iYc+aoFpoMSRf
5keGk0Tws17Qx1H4CEJTBJRuSol+KHKkR+S2rqc3FfU97WnODx3xPIZlguL2+MUi
LENm8W37QIr3G3vC7Lxens+67Fr367P0c1C7irJxo6I8s5R//
eiUaU5y3CzrTYOzeyS3ZaG3Bmax7EinfR0kcdGE0PuKEJ+qUPoOQPEDgqncrPtxou7ihzGPbWg75en
B6HS7k++NlyRGXQWRK1P2XHZjCUpkFHZJQJwDpnphTqq+2DqJ89+wBf2cvKCFgO
v7EXrlqie7DcHHDhpc1M7ZcSCqTCjbrQTb6KetUJK+WM/Uotx
=0gTd
----END PGP PUBLIC KEY BLOCK----

```

- 2 To verify the included `manifest.sha256` file with the public key, perform the following step. This step creates a `trustdb.gpg` file, which can be safely ignored.

**Note** In the following example output, the "Good signature" line validates the manifest. The WARNING lines can be ignored. The Signature date is the TGZ signing date.

```

$ gpg --no-default-keyring --homedir . \
--keyring public.asc.gpg \
--verify manifest.sha256.asc manifest.sha256

```

**Example output:**

```

gpg: WARNING: unsafe permissions on homedir '/tmp/cb-psc-install'
gpg: Signature made Wed Jun  9 01:49:05 2021 IST
gpg:          using RSA key 485BB0DF6AC57704
gpg: /tmp/cb-psc-install/trustdb.gpg: trustdb created
gpg: Good signature from "bit9build (bit9cs) <support@bit9.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: 1853 62D1 D591 FDFA 0C64 7B58 485B B0DF 6AC5 7704

```

- 3 Check the integrity of the unpacked files: `$ sha256sum -c manifest.sha256`

```

blades/bladesUnpack.sh: OK
blades/cb-psc-lq-0.9.8200-8200-blade.tar.gz: OK
blades/cb-psc-th-0.9.8200-8200-blade.tar.gz: OK
cb-psc-sensor-2.11.2-545096.el6.x86_64.rpm: OK
cb-psc-sensor-2.11.2-545096.el7.x86_64.rpm: OK
cb-psc-sensor-2.11.2-545096.el8.x86_64.rpm: OK
install.sh: OK

```

- 4 Check for unexpected files extracted from the TGZ. You should see the files listed in the verified `manifest.sha256`, `public.asc`, `public.asc.gpg`, `trustdb.gpg`, and the two manifest files. The existence of additional files in the directory indicate that the TGZ was tampered.

## Linux Kernel Requirements for Linux Sensor Versions 2.10+

Depending on the Linux kernel version, the Carbon Black Cloud Linux sensor uses one of two mechanisms to monitor events.

- On kernel versions less than 4.8 (for example, CentOS or RHEL 6.x or 7.x, SLE12SP3, or Ubuntu16), a kernel module is used.
- On kernel versions equal to or greater than 4.8, the BPF feature is used.

In addition, different versions of the sensor may be required depending on the Linux kernel version. See [Linux Operating Systems and Respective Sensors](#) for reference.

---

### Note

- Non-default OEM kernels such as `*-azure` or `*-aws` are not supported unless they are specifically listed in [Carbon Black Cloud Linux Sensor Operating Environment Requirements](#).
  - Linux command examples in this section are run under a bash shell.
- 

You can check the kernel version by running the following command:

```
$ uname -r
```

## Kernel Module-based Sensor

Secure Boot is not supported because the kernel module is not signed. Before installation, disable Secure Boot or sign the kernel module. Otherwise, an attempted installation will result in the sensor entering bypass mode immediately after installation.

## BPF-based Sensor

For the sensor's underlying BPF implementation to work, one or more of the following are required:

- The Linux kernel includes BTF format metadata, which newer kernels provide, and the sensor version is 2.15.0 or later.
- The Linux kernel has BPF features enabled.
- The Linux kernel headers associated with the running kernel are preconfigured.
- The Linux kernel headers associated with the running kernel were manually installed.

We highly recommend that you install the sensor to determine whether the sensor's prerequisites have been met. During installation, the sensor checks BPF functionality and displays an error message if BPF is not functional.

After the sensor is successfully installed, check whether the BPF requisites are still met by running the following command.

```
$ /opt/carbonblack/psc/blades/E51C4A7E-2D41-4F57-99BC-6AA907CA3B40/bpf/event_collector -p
```

A `Successfully initialized BPF Program` message and an exit status of 0 indicates that the check was successful.

If the requirements are not met, you must go through relevant checks and remediation steps. See [Check BPF-based Sensor Requirements](#) and its associated topics.

## Check BPF-based Sensor Requirements

To manually check for <OS Distro> before or after sensor installation, perform the following steps.

### Procedure

- 1 Determine whether the kernel includes BTF metadata.

To check for the Linux kernel file, run the following command:

```
$ ls /sys/kernel/btf/vmlinux
```

- 2 Review the boot configuration file settings to determine whether the kernel has BPF features enabled.

- a To determine whether all of the following flags are set to equal to `y`: `CONFIG_BPF`, `CONFIG_BPF_SYSCALL`, `CONFIG_BPF_JIT`, `CONFIG_BPF_EVENTS`, run the following command:

```
$ egrep '(CONFIG_BPF|CONFIG_BPF_SYSCALL|CONFIG_BPF_JIT|CONFIG_BPF_EVENTS)= ' /boot/config-$(uname -r)
```

- b One of the following flags must be set equal to `y`: `CONFIG_HAVE_BPF_JIT` or `CONFIG_HAVE_EBPF_JIT`:

```
$ egrep '(CONFIG_HAVE_BPF_JIT|CONFIG_HAVE_EBPF_JIT)= ' /boot/config-$(uname -r)
```

- 3 To check whether the kernel headers are preconfigured into the kernel, run the following command. If they are preconfigured, you do not have to install the packages.

```
$ cat /boot/config-$(uname -r) | grep CONFIG_IKHEADERS
```

**Note** On Oracle UEK, there might be no output from running this command.

A result of either `CONFIG_IKHEADERS=m` or `CONFIG_IKHEADERS=y` indicates that you should not need to install any headers for BPF. Any other result means that you must manually install the kernel header packages.

If the kernel headers are preconfigured, a kernel module creates a directory with the headers under `/tmp`.

To check for the existence of this header directory, run the following command:

```
$ ls -ld /tmp/kheaders-$(uname -r)*
```

- 4 For Debian OS, add IPtables as a prerequisite for using the Quarantine feature.

### What to do next

Perform the following Linux variant instructions to check for kernel headers and to install them if necessary.

---

### Note

- You might need to activate subscriptions or configure package repositories to install packages.
  - Linux distributions regularly update the kernel package and do not always keep the old kernel headers package in their repositories. If you cannot find the associated header packages for your kernel, update the system to a newer supported kernel, install the associated kernel header packages, and then boot the new kernel. Only update the kernel as much as needed, and to a version that the sensor supports. See [Linux Operating Systems and Respective Sensors](#) for reference.
  - When you update the kernel, you must also update the kernel header package before you reboot.
- 

## Check Requirements for CentOS, RHEL, Oracle RHCK, and Amazon Linux

To check for kernel headers and install them if necessary on CentOS, RHEL, Oracle RHCK, or Amazon Linux systems, perform the following steps.

When they are properly installed, the required kernel headers are located under `/usr/src/kernels/$(uname -r)/include/`.

### Prerequisites

#### [Check BPF-based Sensor Requirements](#)

### Procedure

- 1 To determine whether kernel headers are installed, run the following command:

```
$ sudo yum list installed kernel-devel-$(uname -r)
```

If the package is installed, the output will be similar to the following:

#### Installed Packages

```
kernel-devel.x86_64 <version>
```

- 2 To determine whether the kernel headers are available to install, run the following command:

```
$ sudo yum list available kernel-devel-$(uname -r)
```



If the package is available, the output will be similar to the following:

#### Available Packages

**kernel-devel.x86\_64 <version> baseos**

- 3 To install any necessary available kernel headers, run the following command:

```
$ sudo yum install -y kernel-devel-$(uname -r)
```

- 4 If the kernel headers packages are not installed and are not available, update the kernel to a supported version and install the matching kernel headers.

---

**Note** This action requires a reboot.

---

- a To list the available kernel and kernel header packages and find matching versions, run the following command:

```
$ sudo yum list available kernel kernel-devel
```

If newer packages are available, the output will be similar to the following:

#### Available Packages

**kernel.x86\_64 <newer-version> baseos**

...

**kernel-devel.x86\_64 <newer-version> baseos**

- b To install a newer supported kernel and kernel header packages with matching versions, run the following command:

```
% sudo yum install -y kernel-<newer-version> kernel-devel-<newer-version>
```

- c Reboot into the new kernel.

## Check Requirements for Oracle UEK

To check for kernel headers and install them if necessary on Oracle UEK systems, perform the following steps.

When they are properly installed, the required kernel headers are located under `/usr/src/kernels/$(uname -r)/include/`.

### Prerequisites

#### Check BPF-based Sensor Requirements

### Procedure

- 1 To determine whether kernel headers are installed, run the following command:

```
$ sudo yum list installed kernel-uek-devel-$(uname -r)
```

If the package is installed, the output will be similar to the following:

### Installed Packages

**kernel-uek-devel.x86\_64 <version>**

- 2 To determine whether the kernel headers are available to install, run the following command:

```
$ sudo yum list available kernel-uek-devel-$(uname -r)
```

If the package is available, the output will be similar to the following:

### Available Packages

**kernel-uek-devel.x86\_64 <version> baseos**

- 3 To install any necessary available kernel headers, run the following command:

```
$ sudo yum install -y kernel-uek-devel-$(uname -r)
```

- 4 If the kernel headers packages are not installed and are not available, update the kernel to a supported version and install the matching kernel headers.

---

**Note** This action requires a reboot.

---

- a To list the available kernel and kernel header packages and find matching versions, run the following command:

```
$ sudo yum list available kernel-uek kernel-uek-devel
```

If newer packages are available, the output will be similar to the following:

### Available Packages

**kernel-uek.x86\_64 <newer-version> baseos**

...

**kernel-uek-devel.x86\_64 <newer-version> baseos**

- b To install a newer supported kernel and kernel header packages with matching versions, run the following command:

```
% sudo yum install -y kernel-uek-<newer-version> kernel-uek-devel-<newer-version>
```

- c Reboot into the new kernel.

## Check Requirements for SUSE and OpenSUSE

To check for kernel headers and install them if necessary on SUSE and OpenSUSE systems, perform the following steps.

When they are properly installed, the required kernel headers are located under `/usr/src/linux-$(uname -r | sed "s/-default//")/include/`.

## Prerequisites

### Check BPF-based Sensor Requirements

#### Procedure

- 1 To determine whether kernel headers are installed, run the following command:

```
$ sudo zypper search -si kernel-default-devel | grep $(uname -r | sed "s/-default//")
```

If the package is installed, the output will be similar to the following. An `i` or `i+` in the left column signifies that the package is installed.

**i+ kernel-default-devel | package | <version> | <arch> | <repository>**

- 2 To determine whether the kernel headers are available to install, run the following command:

```
$ sudo zypper search -s kernel-default-devel | grep $(uname -r | sed "s/-default//")
```

If the package is available, the output will be similar to the following. A `v` in the left column signifies that the package is available.

**v kernel-default-devel | package | 4.12.14-lp150.12.25.1 | x86\_64 | openSUSE-Leap-15.0-Update**

- 3 To install any necessary available kernel headers, run the following command:

```
$ sudo zypper install --oldpackage kernel-default-devel=$(uname -r | sed "s/-default//")
```

- 4 If the kernel headers packages are not installed and are not available, update the kernel to a supported version and install the matching kernel headers.

---

**Note** This action requires a reboot.

---

- a To list the available kernel and kernel header packages and find matching versions, run the following command:

```
$ sudo zypper search -s kernel-default kernel-default-devel | egrep "(^v|^S)"
```

If newer packages are available, the output will be similar to the following:

S	Name	Type	Version	Arch	Repository
v	kernel-default	package	<newer-version>	<arch>	<repo>
...					
v	kernel-default-devel	package	<newer-version>	<arch>	<repo>

- b To install a newer supported kernel and kernel header packages with matching versions, run the following command:

```
% sudo zypper install kernel-default=<newer-version> kernel-default-devel=<newer-version>
```

- c Reboot into the new kernel.

---

**Note** If you cannot find a newer kernel to upgrade to by following this method, you might need to upgrade to a newer service pack or migrate to a newer release.

---

## Check Requirements for Debian

To check for kernel headers and install them if necessary on Debian systems, perform the following steps.

When they are properly installed, the required kernel headers are located under `/usr/src/linux-headers-$(uname -r)/include/`.

### Prerequisites

#### Check BPF-based Sensor Requirements

Add IPtables as a prerequisite for using the Quarantine feature.

### Procedure

- 1 To determine whether kernel headers are installed, run the following command:

```
$ sudo apt list --installed linux-headers-$(uname -r)
```

If the package is installed, the output will be similar to the following. An `i` or `i+` in the left column signifies that the package is installed.

**linux-headers-<version>-generic/...**

- 2 To determine whether the kernel headers are available to install, run the following command:

```
$ sudo apt search linux-headers-$(uname -r)
```

If the package is available, the output will be similar to the following:

**linux-headers-<version>-generic/...**

- 3 To install any necessary available kernel headers, run the following command:

```
$ sudo apt install linux-headers-$(uname -r)
```

- 4 If the kernel header packages are not installed and are not available, update the kernel to a supported version and install the matching kernel headers.

---

**Note** This action requires a reboot.

---

- a To list the available kernel and kernel header packages and find matching versions, run the following commands:

```
$ sudo apt search linux-image
$ sudo apt search linux-headers
```

If newer packages are available, the output will be similar to the following:

**linux-image-<newer-version>-<arch>**

**Linux <newer-version> for 64-bit PCs (signed)**

...

**linux-headers-<newer-version>-<arch>**

**Header files for Linux <newer-version>-<arch>**

...

- b To install a newer supported kernel and kernel header packages with matching versions, run the following commands:

```
% sudo apt install kernel-default=<newer-version>
% kernel-default-devel=<newer-version>
```

- c Reboot into the new kernel.

## Check Requirements for Ubuntu

To check for kernel headers and install them if necessary on Ubuntu systems, perform the following steps.

When they are properly installed, the required kernel headers are located under `/usr/src/linux-headers-$(uname -r)/include/`.

## Prerequisites

### Check BPF-based Sensor Requirements

#### Procedure

- 1 To determine whether kernel headers are installed, run the following command:

```
$ sudo apt list linux-headers-$(uname -r)
```

If the package is installed, the output will be similar to the following. An `i` or `i+` in the left column signifies that the package is installed.

**linux-headers-<version>-generic/...**

- 2 To determine whether the kernel headers are available to install, run the following command:

```
$ sudo apt search linux-headers-$(uname -r)
```

If the package is available, the output will be similar to the following:

**linux-headers-<version>-generic/...**

- 3 To install any necessary available kernel headers, run the following command:

```
$ sudo apt install linux-headers-$(uname -r)
```

- 4 If the kernel headers packages are not installed and are not available, update the kernel to a supported version and install the matching kernel headers.

---

**Note** This action requires a reboot.

---

- a Get the kernel update tool.

```
$ sudo wget https://raw.githubusercontent.com/pimlie/ubuntu-mainline-kernel.sh/master/ubuntu-mainline-kernel.sh
```

- b To list the available kernel versions, run the following command:

```
$ sudo ubuntu-mainline-kernel.sh -r
```

- c To install a newer supported kernel version, run the following command:

```
$ sudo ubuntu-mainline-kernel.sh -i <version>
```

- d To install kernel header packages for the new kernel version, run the following command:

```
$ sudo apt install linux-headers-<version>
```

- e To reboot into the new kernel, run the following command:

```
$ sudo reboot
```

## About the Linux Sensor `cfg.ini` File

The Linux sensor keeps its primary configuration details together with transient state in the `/var/opt/carbonblack/psc/cfg.ini` file.

The `cfg.ini` file is created when the sensor is installed. It changes while the sensor is running, and is used to manage many long term stateful processes such as software upgrades, communication configuration and state, and device registration information.

The sensor normally reads the `cfg.ini` file one time upon startup and writes it one or more time when the sensor needs to update its information. Therefore, you must edit the `cfg.ini` file while the sensor is stopped. Modifications done while the sensor is running are likely to be overwritten by the sensor's next update of the file, and in any case are not visible to the sensor until its next startup. It is advisable to plan what changes to make to reduce the sensor downtime that occurs while editing the file.

The `install.sh` script installs the sensor on an endpoint (see [Install a Linux Sensor on a Single Endpoint](#)). When running this script, you can set `cfg.ini` fields by using the `--prop` option of that script. For example, the following would set the email address for this sensor:

```
./install --prop 'EmailAddress=bill@example.com'
```

---

**Tip** In this example, the parameter is enclosed in single quotation marks. This is not strictly required in this case, but it is a good practice.

---

Table 2-1. Supported Install.sh cfg.ini (--prop) Options

Option	Value	Notes
CompanyCode	String value To access or regenerate a company code, navigate to <b>Endpoints &gt; Sensor Options &gt; Company Codes</b> . You must enclose the company code in single quotes.	The company code identifies the company that owns this machine. Across an organization, all machines must have the same value for this option.  <b>Important</b> Carbon Black does not recommend using the company code, unless you are managing VDI environments and must copy the company code to new instances. The company code value is already set during sensor installation, by providing a registration code to the <code>install.sh</code> script.
GroupName	String value Always enclose this value with quotes if the policy name (group name) includes spaces.	Optional policy name assignment. This field sets the Policy value for this endpoint. This affects what rule sets are applied to this sensor. This can be used to pre-set the policy used by the sensor at install time.  <b>Note</b> It might be easier to manage this in the Carbon Black Cloud console; the backend might change this field depending on changes made in the console. This assignment can also be set during installation: <code>./install.sh --groupname 'SensorGroupName'</code>
EmailAddress	You can set any email address.	This is the point of contact for administering this sensor. The provided address is visible in the Endpoints page in the console.
ProxyServer	<code>server:port</code>	You can set the <code>ProxyServer</code> field to direct sensor network traffic through a proxy server (such as a `squid` proxy server.) You can specify the server IP address.
ProxyPemFile	Provide the full file path and file name of the PEM file. For example: <code>./install.sh --proxy 1.2.3.4:3129 --prop ProxyPemFile=/path/path/my-pem-file.pem</code> .	The PEM file is used to connect to some proxies that use certificate-based authentication. The PEM file is only used if the proxy server is also set. If there is no PEM file, the proxy server connection will be attempted without authentication.



## Linux Installer Command Line Parameters

The following command line parameters are supported by the `install.sh` install utility script starting with Linux sensor 2.12 and later. Parameter values must always be enclosed in single quotes.

To view all command line parameters, run the command together with the `-h` parameter.

Table 2-2.

Parameter	Description
<no parameter>	Only installs the sensor on the endpoint. Does not register or start the sensor.
<code>-b   --bypass</code>	Enable bypass mode (disabled protection) immediately after installation.
<code>'&lt;Company_Code&gt;'</code>	Registers and starts the sensor.
<code>-d   --disable-lr</code>	Disable Live Response for the sensor during start. Live Response is enabled by default.
<code>-g   --groupname '&lt;POLICY_NAME&gt;'</code>	Assign the policy to which the sensor will be added during start.
<code>-h   --help</code>	Displays all command line options.
<code>-p   --proxy '&lt;PROXY_SERVER:PORT&gt;'</code>	Preferred proxy server and port.
<code>-r   --register</code>	This option skips starting the agent and only registers the sensor during installation. Company code must be passed when using this option.

## Customize the Linux Feature Selection

With the release of the Linux 2.15.0+ sensor, Audit and Remediation and Enterprise EDR are supported on the Linux platform and are automatically installed together with the sensor.

### Note

- You cannot install a subset of these components, but you can remove unwanted features after installation.
- The next time you upgrade the sensor, any missing components will be reinstalled.
- For sensor versions earlier than 2.15.0, you can manually customize the installer package.

To remove either component from the platform, run the following commands:

### Procedure

```
1 $rm -fr /opt/carbonblack/psc/blades/<undesired_blade>
```

```
2 $systemctl service cbagentd restart
```

For older distributions, run `service restart cbagentd` instead.

## Install a Linux Sensor on a Single Endpoint

You can install a Carbon Black Cloud Linux sensor on a single endpoint by following this procedure.

### Procedure

- 1 Extract the contents of the installer package into a temporary directory `cb-psc-install`. See [Unpack the Agent](#).
- 2 Install and register the sensor by running the following command; replace '`<COMPANY_CODE>`' with your company registration code:

```
sudo cb-psc-install/install.sh '<COMPANY_CODE>'
```

For Linux sensor versions 2.11.1 onwards, you can specify proxy server details while installing and registering the sensor through command line. Replace the '`<COMPANY_CODE>`' with your company registration code:

```
sudo cb-psc-install/install.sh [-p|--proxy 'proxyhost:proxyport'] '<COMPANY_CODE>'
```

For Linux sensor versions 2.6.0 onwards, you can pass optional parameters from the `cfg.ini` file to the install script. For a full list of parameters and additional information about `cfg.ini`, see [About the Linux Sensor cfg.ini File](#).

---

**Note** `-p|--proxy` is an optional parameter available for Linux sensor versions 2.11.1 onwards. It passes proxy server details for the endpoint to communicate with backend. You can specify the IP address or hostname as part of `proxyhost`.

---

**Note** The Linux sensor only supports a HTTP non-authenticated proxy server through `cfg.ini`.

---

**Table 2-3.**

Proxy Type	IP Format	FDQN Format
HTTP	ip:port	fdqn:port
HTTP	http://ip:port	http://fdqn:port

Example `Cfg.ini` settings:

```
[customer]
ProxyServer=proxy.example.com:3128
```

---

# Install a Linux Sensor on an Endpoint using the RPM/DPKG Installer

You can install a Linux sensor on an endpoint by using this method.

---

**Note** For Linux sensors 2.15.0+

- Blades are intrinsic to the sensor installation package and are automatically installed and configured during the sensor installation. With 2.15.0+, both Audit and Remediation and Enterprise EDR blades are installed.
- 

## Procedure

- 1 Extract the contents of the installer package into an empty directory.
- 2 Install the RPM/DEB package.

RPM:

For x86\_64 machines:

```
$ cd <preferred working directory>
```

```
$ sudo rpm -i cb-psc-install/cb-psc-sensor-<BUILD-NUMBER>.x86_64.rpm
```

For aarch64 machines:

```
$ cd <preferred working directory>
```

```
$ sudo rpm -i cb-psc-install/cb-psc-sensor-<BUILD-NUMBER>.aarch64.rpm
```

DEB:

For x86\_64 machines:

```
$ cd <preferred working directory>
```

```
$ sudo dpkg -i cb-psc-install/cb-psc-sensor-<BUILD-NUMBER>.x86_64.deb
```

For aarch64 machines:

```
$ cd <preferred working directory>
```

```
$ sudo dpkg -i cb-psc-install/cb-psc-sensor-<BUILD-NUMBER>.aarch64.deb
```

- 3 For sensor versions prior to 2.15.0, install the blades.

```
$ sudo cb-psc-install/blades/bladesUnpack.sh
```

- 4 Update the `cfg.ini` file with the v3.x+ company registration code.

```
$ sudo /opt/carbonblack/psc/bin/cbagentd -d '<COMPANY_CODE>'
```

## 5 Start the agent.

For CentOS/RHEL 6:

```
$ service cbagentd start
```

For all other distributions:

```
$ systemctl start cbagentd
```

## Install a Linux Sensor on an Endpoint using the RPM Installer on an Amazon Linux 2 Graviton EC2 Instance

To install a Linux sensor on an Amazon Linux 2 Graviton EC2 instance, perform the following procedure.

### Procedure

- 1 Extract the contents of the Amazon installer package into a temporary directory.
- 2 Extract the contents of `sysroot-glibc-2.28.tar.gz` at `/opt/glibc-2.28`:

```
$ cd <temporary directory>
$ sudo mkdir -p /opt/glibc-2.28
$ sudo tar -xf sysroot-glibc-2.28.tar.gz -C /opt/glibc-2.28
```

- 3 Install the RPM package:

```
$ sudo rpm -i --nodeps cb-psc-install/cb-psc-sensor-<BUILD-NUMBER>.a12.aarch64.rpm
```

- 4 Update the `cfg.ini` file with the v3.x+ company registration code:

```
$ sudo /opt/carbonblack/psc/bin/cbagentd -d '<COMPANY_CODE>'
```

- 5 Start the agent:

```
$ systemctl start cbagentd
```

## Install a Linux Sensor on an Endpoint that Automatically Registers the First Time it is Started

By using this method, the Linux sensor registers the first time it starts up.

### Procedure

- 1 Extract the contents of the installer package into a temporary directory.
- 2 Use the `install.sh` script to install the agent, but do not provide a company code. `$ sudo cb-psc-install/install.sh`
- 3 Update the `cfg.ini` file with the v3.x+ company code. `$ sudo /opt/carbonblack/psc/bin/cbagentd -d '<COMPANY_CODE>'`

## Results

---

**Note** The sensor is configured to register when the sensor starts up. This can occur on the next system boot or by restarting the agent. When the agent starts, the sensor will register itself with the Carbon Black Cloud backend.

---

# Installing macOS Sensors on Endpoints

# 3

This section introduces ways to install macOS sensors on endpoints.

---

**Important** Before you begin the processes described here, read [Chapter 1 Getting Started with Sensor Installation](#). It contains highly relevant information to help you succeed in your sensor installation.

If you use [Method 1: Invite Users to Install Sensors on Endpoints](#) to install the macOS sensor 3.6.1 or later, an **Advanced** option allows you to enable FIPS. Instruct your users not to enable FIPS. This feature is not supported for commercial instances of the Carbon Black Cloud.

---

Before you can install sensors, you must perform the following steps:

[Obtain a Company Registration Code](#)

[Download Sensor Kits](#)

---

**Note** For VMware Workspace One instructions, see [Deploying Carbon Black Cloud Sensor with Workspace ONE UEM](#).

---

**Note** For information about mobile device management (MDM), see:

- [Approve the System Extension via MDM](#)
  - [Grant the Sensor Full Disk Access by using MDM](#)
- 

Read the following topics next:

- [Approve the Kernel Extension \(macOS 10.13 – macOS 11\)](#)
- [Approving the System Extension and Network Extension for macOS 11+](#)
- [Full Disk Access Requirement for the macOS Sensor](#)
- [Restart Requirements for macOS 10.15+](#)
- [Special Considerations for the macOS Sensor on Big Sur](#)
- [Manually Install and Approve the Sensor on macOS 11+](#)
- [macOS Sensor Command Line Install](#)
- [Deploying macOS Sensors on Big Sur and Later by using Jamf Pro](#)
- [Validate a Healthy System Extension Sensor through RepCLI](#)

- [Address the Extension Warning Post-install](#)
- [Resolve macOS Sensor Upgrade or Install Error due to Existing System Extension](#)
- [macOS Services, Utilities, and Uninstaller](#)
- [Installing macOS Sensors on Endpoints by using Workspace ONE UEM](#)

## Approve the Kernel Extension (macOS 10.13 – macOS 11)

For macOS v3.1 sensor installations on macOS 10.13, High Sierra requires initial KEXT approval of the product kernel extension by administrative policy or user.

This requirement is enforced by Apple. It applies to all third-party products that have a driver component. The sensor requires KEXT approval regardless of the previous KEXT approval status.

---

**Note** For macOS Big Sur, user KEXT approval is only part of the requirement; MDM approval is required for KEXT deployment on macOS Big Sur.

---

**Important** Sensor version 3.8.0 and future sensor versions no longer support Kernel Extension approval as well as macOS11 and prior Operating Systems. Customers must use System Extension approval.

---

Carbon Black recommends that you pre-configure endpoints with pre-approved drivers by using MDM policy, netboot, or pre-configured images. This approach simplifies sensor installation, especially during a command line installation. A CLI message occurs during the install, and requires the `-kext` flag to skip and finish the install.

If drivers are not pre-approved before sensor installation, the behavior is as follows:

Command line installation: Installation finalizes and returns success, but logs a warning to installation logs. Because drivers cannot load, the sensor enters Bypass state and reports this state to the cloud. After KEXT is approved, the sensor recovers within one hour and enters the full protection state.

Direct installation is handled similarly to a command line installation, with two differences: (1) sensor installation displays a dialog message that requests the user to approve the KEXT by using system preferences; (2) installer stalls for up to 10 minutes to give the user the opportunity to approve the KEXT.

---

**Note** For information about mobile device management (MDM), see:

- [Approve the System Extension via MDM](#)
  - [Grant the Sensor Full Disk Access by using MDM](#)
- 

## Manually Approve the KEXT

This procedure lets you manually approve the KEXT.

### Procedure

- 1 Install the sensor on the endpoint.
- 2 In **System Preferences**, in the **Security & Privacy** pane, click the **General** tab.
- 3 Authenticate as Administrator.
- 4 Click the **Allow** button for **System software from developer “Carbon Black” was prevented from loading**. The installer will finish running and load the sensor.

## Approve the KEXT via MDM

Use this procedure to approve KEXT using an MDM.

### Procedure

- ◆ Specify the Apple Team ID and KEXT bundle in your configuration profile.

- Apple Team ID: `7AGZNQ2S2T`
- KEXT Bundle ID: `com.carbonblack.defense.kext`

See [How to approve Mac Sensor 3.0 KEXT for Install/Upgrade](#) and Apple Technical Note TN245.

## Approving the KEXT via MDM for Big Sur

The easiest way to distribute the necessary MDM payload to approve the KEXT is to upload the `MDM-KEXT-approval.mobileconfig` file, which is located in the mounted DMG of the installer in the docs folder.

You can also recreate the attached mobileconfig in your MDM tool by specifying the Apple Team ID and the KEXT Bundle ID in your Kernel Extension configuration profile:

- Apple Team ID: `7AGZNQ2S2T`
- KEXT Bundle ID: `com.carbonblack.defense.kext`

To allow the KEXT to load on MacOS Big Sur, the OS either requires a local action from an admin to approve the KEXT after install or a customized reboot command from your MDM to rebuild the Kernel Cache.

Your MDM must support custom XML to use the following method. If your MDM provider does not support custom XML, use the local approval method to run the KEXT.



The easiest way to distribute the necessary MDM command is to upload the `MDM-KEXT-reboot-command.xml` file, which is found in the mounted DMG of the installer in the docs folder. This XML file should be uploaded as a Custom Command and sent to endpoints after KEXT install. The target machine will reboot without warning; this distribution method is a temporary workflow until MDM providers update their reboot protocols to support `RebuildKernelCache`. This command is here:

```
<dict>
  <key>RebuildKernelCache</key>
  <true/>
  <key>KextPaths</key>
  <string>/Library/Extensions/CbDefenseSensor.kext</string>
  <key>RequestType</key>
  <string>RestartDevice</string>
</dict>
```

## Identify Devices with Sensors that do not support the Operating System or need KEXT Approval

These search procedures show you which sensors are running on unsupported operating systems or are not KEXT approved.

### Procedure

- 1 Sign in to the Carbon Black Cloud Console.
- 2 On the navigation bar, click **Inventory** and then click **Endpoints**.
- 3 Change the **Status** filter to **All**, and type the following search query:  
`sensorStates:UNSUPPORTED_OS`
- 4 Use the following search query to help identify devices with sensors that do support the operating system, but with sensor KEXT or System Extension not approved:  
`sensorStates:DRIVER_LOAD_NOT_GRANTED`

## Approving the System Extension and Network Extension for macOS 11+

Beginning with macOS 11 (Big Sur), the sensor utilizes a System Extension and Network Extension (web content filter) for user space operation. In order to suppress client-side notifications to approve their operation, the System Extension and Network Extension should be pre-approved via MDM whenever possible.

For manual installation and sensor approval, see [Manually Install and Approve the Sensor on macOS 11+](#).

### Approve the System Extension via MDM

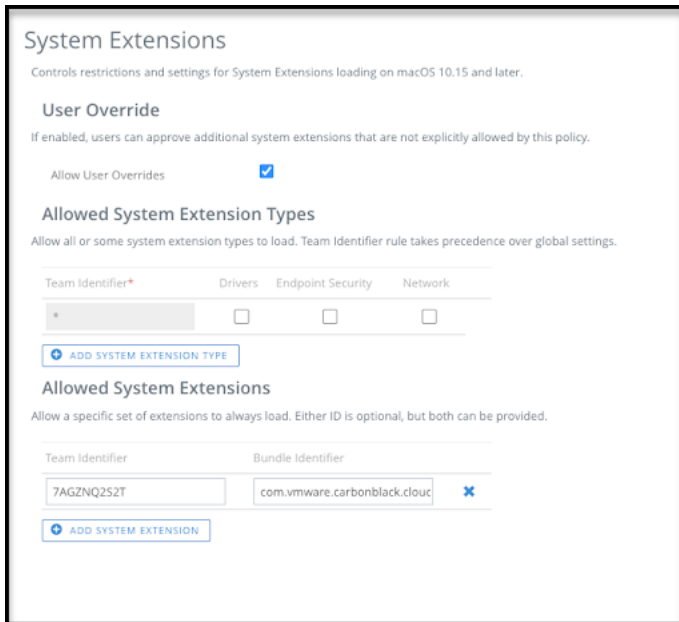
Use this procedure to manually create the correct mobileconfig in your MDM.

MDM begins when you set up a server and distribute your MDM enrollment profile to devices to initiate connecting them. Once the device installs an MDM enrollment profile and connects to the server, it can receive commands from the server. When you remove the MDM enrollment profile from a device, that terminates the device management relationship with the server.

**Procedure**

- ◆ Specify the Apple Team ID and System Extension bundle Identifier in your Allowed System Extension configuration profile:
  - System Extension Types: `Allowed System Extensions`
  - Apple Team ID: `7AGZNQ2S2T`
  - System Extension Bundle ID: `com.vmware.carbonblack.cloud.se-agent.extension`

The Workspace One configuration should look like the following:



The JAMF configuration should look like the following:

## Approve the Network Extension Component of the System Extension via MDM

Use this procedure to grant the System Extension the ability to Filter Network Content via a Web Content Filter configuration profile.

After creating this profile, the profile should be signed to enable distribution via MDM.

### Procedure

- ◆ The fields should be completed exactly as follows. Copy and paste for accuracy.

In the General payload:

- **Payload Scope:** `System`

In the Web Content Filter payload:

- **Filter Type:** `Plug-In`
- **Plug-In Bundle ID:** `com.vmware.carbonblack.cloud.se-agent`
- Check **Enable Socket Filtering**
  - **Filter Data Provider System Extension Bundle ID (macOS):**  
`com.vmware.carbonblack.cloud.se-agent.extension`
  - **Filter Data Provider Designated Requirement (macOS):** `identifier "com.vmware.carbonblack.cloud.se-agent.extension" and anchor apple generic and certificate 1[field.1.2.840.113635.100.6.2.6] /* exists */ and certificate leaf[field.1.2.840.113635.100.6.1.13] /* exists */ and certificate leaf[subject.OU] = "7AGZNQ2S2T"`
- Check **Enable Packet Filtering (macOS)**
  - **Filter Packet Provider System Extension Bundle ID (macOS):**  
`com.vmware.carbonblack.cloud.se-agent.extension`

- **Filter Packet Provider Designated Requirement (macOS):** `identifier`  
`"com.vmware.carbonblack.cloud.se-agent.extension"` and anchor `apple generic`  
 and certificate `1[field.1.2.840.113635.100.6.2.6] /* exists */` and  
 certificate `leaf[field.1.2.840.113635.100.6.1.13] /* exists */` and  
 certificate `leaf[subject.OU] = "7AGZNQ2S2T"`

## Full Disk Access Requirement for the macOS Sensor

As part of user data security enhancements in macOS 10.14.5 and above, you must approve access to protected user and application data. This is required for the sensor to operate at maximum capability (with protection fully enabled, monitoring and protecting all applications and the file system).

This requirement is in addition to SysEXT and NetworkExtension approvals, and does not replace that approval process.

The preferred mechanism for the Full Disk Access approval is through the creation and deployment of a mobile device management (MDM) profile.

Alternatively to the MDM method, Full Disk Access can be granted manually by the endpoint administrator through the **Security & Privacy System Preferences** pane. This option is only recommended for very small deployments where MDM solution may not be available, or for testing on individual devices prior to deployment.

### Grant the Sensor Full Disk Access by using MDM

The easiest way to distribute the necessary Privacy Preference payload is to upload the `MDM-privacyconfig.mobileconfig` file, which is in the mounted DMG of the installer in the `docs` folder.

The following steps recreate the `mobileconfig` in your MDM.

These instructions were created using Apple documentation and were validated in Jamf PRO and WorkspaceONE UEM using sensor version 3.5.0.30. Field names, values, and functionality vary depending on the MDM framework or sensor version.

Granting an application full disk access is accomplished via a Privacy Preferences payload. The Carbon Black Cloud Sensor requires five identifiers in this Privacy payload.

#### Procedure

- 1 Complete the fields exactly as follows. Copy and paste for accuracy.

Identifier: `com.vmware.carbonblack.cloud.daemon`

Identifier Type: `Bundle ID`

Code Requirement:

```
identifier "com.vmware.carbonblack.cloud.daemon" and anchor apple generic
    and certificate 1[field.1.2.840.113635.100.6.2.6] /* exists */ and
    certificate leaf[field.1.2.840.113635.100.6.1.13] /* exists */ and
    certificate leaf[subject.OU] = "7AGZNQ2S2T"
```

App or Service: SystemPolicyAllFiles

Access: Allow

Identifier: com.vmware.carbonblack.cloud.osqueryi

Identifier Type: Bundle ID

Code Requirement:

```
identifier "com.vmware.carbonblack.cloud.osqueryi" and anchor apple generic
    and certificate 1[field.1.2.840.113635.100.6.2.6] /* exists */ and
    certificate leaf[field.1.2.840.113635.100.6.1.13] /* exists */ and
    certificate leaf[subject.OU] = "7AGZNQ2S2T"
```

App or Service: SystemPolicyAllFiles

Access: Allow

Identifier: com.vmware.carbonblack.cloud.se-agent.extension

Identifier Type: Bundle ID

Code Requirement:

```
identifier "com.vmware.carbonblack.cloud.se-agent.extension" and anchor apple generic
    and certificate 1[field.1.2.840.113635.100.6.2.6] /* exists */and
    certificate leaf[field.1.2.840.113635.100.6.1.13] /* exists */ and
    certificate leaf[subject.OU] = "7AGZNQ2S2T"
```

App or Service: SystemPolicyAllFiles

Access: Allow

Identifier: com.vmware.carbonblack.cloud.uninstall

Identifier Type: Bundle ID

Code Requirement:

```
identifier "com.vmware.carbonblack.cloud.uninstall" and anchor apple generic
    and certificate 1[field.1.2.840.113635.100.6.2.6] /* exists */ and
    certificate leaf[field.1.2.840.113635.100.6.1.13] /* exists */ and
    certificate leaf[subject.OU] = "7AGZNQ2S2T"
```

App or Service: SystemPolicyAllFiles

Access: Allow

Identifier: com.vmware.carbonblack.cloud.uninstallerui

Identifier Type: Bundle ID

Code Requirement:

```
identifier "com.vmware.carbonblack.cloud.uninstallerui" and anchor apple
    generic and certificate 1[field.1.2.840.113635.100.6.2.6] /*
exists */ and
    certificate leaf[field.1.2.840.113635.100.6.1.13] /* exists */ and
    certificate leaf[subject.OU] = "7AGZNQ2S2T"
```

App or Service: SystemPolicyAllFiles

Access: Allow

## 2 Verify the Full Disk Access application on endpoints in one of two ways:

- Use the Repcli status command on the individual endpoint:

```
sudo /Applications/VMware\ Carbon\ Black\ Cloud/repcli.bundle/Contents/MacOS/repcli
status
```

Expected values are:

- All five Carbon Black Cloud sensor items that require Full Disk Access are marked as Configured via MDM. For example:

```
Full Disk Access Configurations:
  Repmgr: Configured via MDM
  System Extension: Configured via MDM
  OSQuery: Configured via MDM
  Uninstall Helper: Configured via MDM
  Uninstall UI: Configured via MDM
Sensor State:
  State: Enabled
```

- **Sensor State** is Enabled.
- In the Carbon Black Cloud console, click **Inventory > Endpoints**. For the endpoint in question, the status should read:
  - **Sensor Status:** Active (not in Bypass)
  - No FDA Error is present

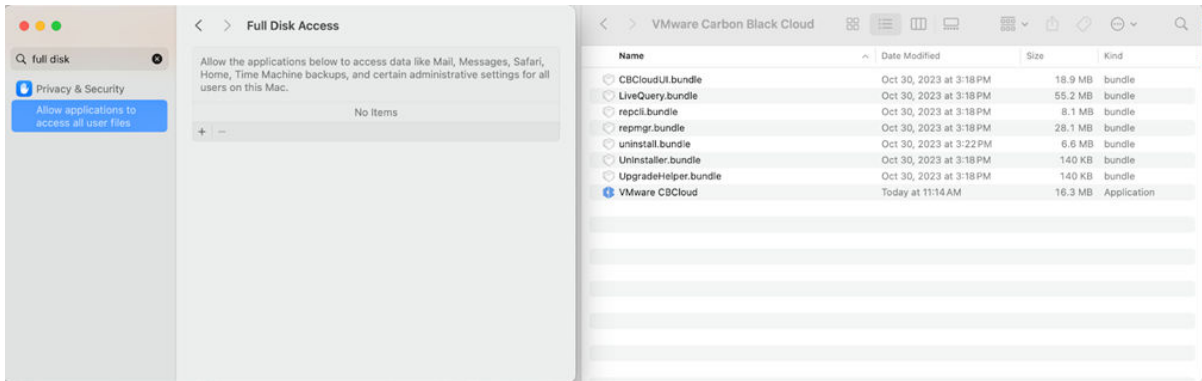
## Manually Grant Sensor Full Disk Access

Use this procedure to manually grant Full Disk Access to macOS sensors.

### Procedure

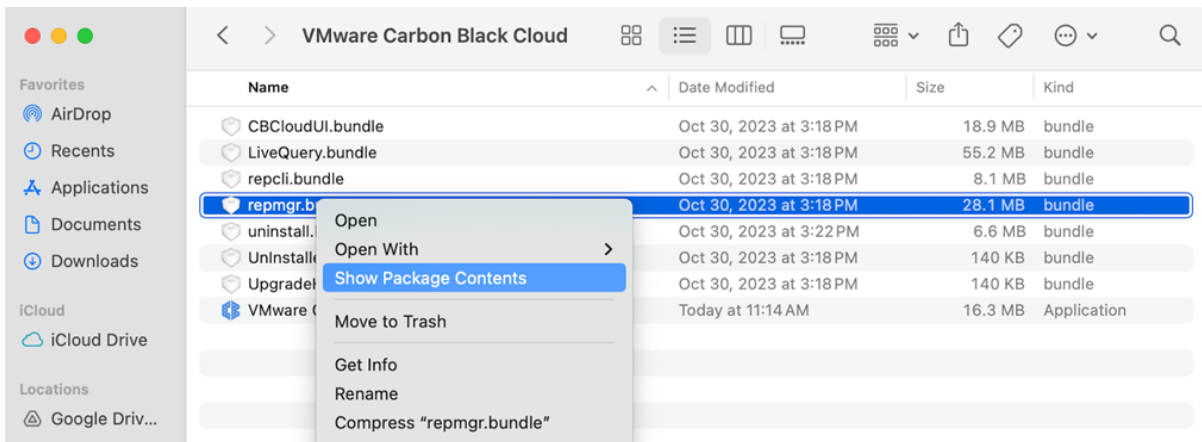
- 1 In System Settings, open the **Security & Privacy** pane and scroll down to **Full Disk Access**.
- 2 Open a new, separate **Finder** window.
- 3 Go to the `/Applications/VMware Carbon Black Cloud/` folder.

- Position this Finder window next to the previously opened **Privacy & Security > Full Disk Access** pane.

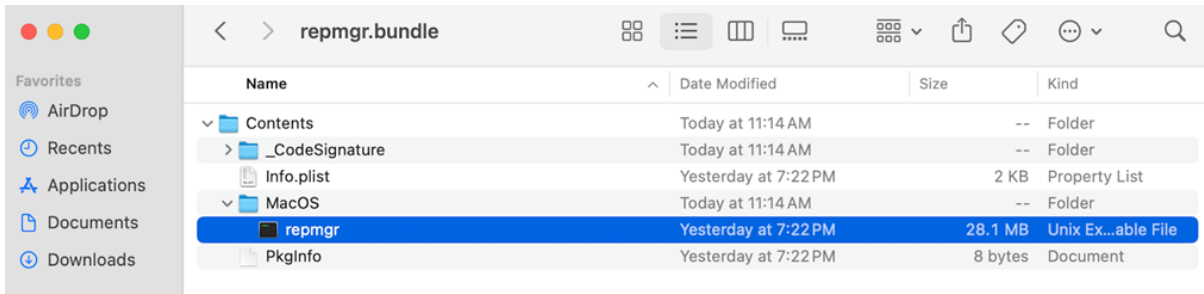


**Important** Due to the pane limitation, you must open the separate Finder window from which to drag-and-drop sensor bundles instead of clicking **+** to add them. The latter method may not support granting by individual binaries.

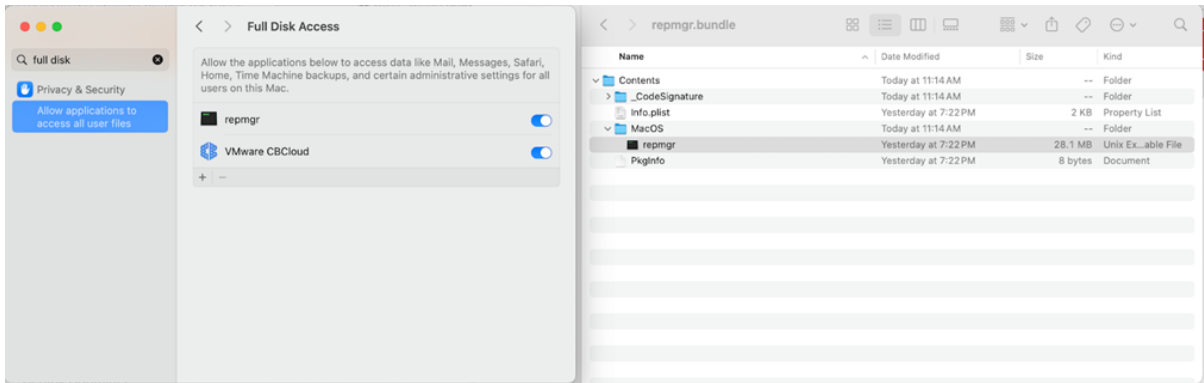
- Drag-and-drop the **VMware CBCloud** bundle folder from the Finder window to the **Full Disk Access** pane and set the slider to **Enabled**. This step grants Full Disk Access to the **CBC SysEXT**.
- Grant the following four executables Full Disk Access by binary file, not by bundle. Navigate to each binary, relative to the `/Applications/VMware\ Carbon\ Black\ Cloud/` top level folder:
  - `repmgr.bundle/Contents/MacOS/repmgr`
  - `LiveQuery.bundle/Contents/MacOS/osqueryi`
  - `UnInstaller.bundle/Contents/MacOS/UnInstaller`
  - `uninstall.bundle/Contents/MacOS/uninstall`
- Starting with the `repmgr` bundle, navigate to the binary by using the Finder right-click **Show Package Contents** menu option.



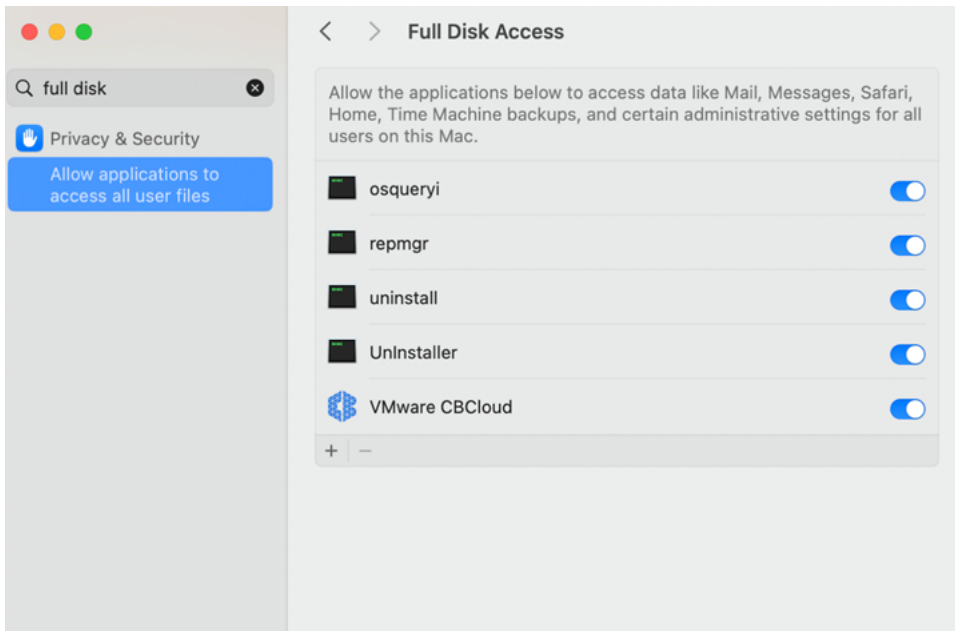
- 8 Navigate to the binary within the bundle Contents/MacOS folders and select the binary file.



- 9 Drag-and-drop the binary to the Full Disk Access pane and set the slider to Enabled.



- 10 Repeat Steps 8 through 9 for the remaining three bundle binaries. The end result should depict the following pane view:



- 11 Verify that Full Disk Access is successfully enabled. It takes approximately one hour for RepCLI to successfully report Full Disk Access changes. To verify the change immediately, restart the endpoint.



- 12 Open a Terminal window and run the RepCLI status command (requires admin password):

```
sudo /Applications/VMware\ Carbon\ Black\ Cloud/repcli.bundle/Contents/MacOS/repcli status
```

RepCLI displays a sensor information report that includes Full Disk Access. Expected output in the **Full Disk Access Configuration** section should read "Configured Manually" for the five sensor items.

```
bit9qa@macos-20D91 MacOS % sudo ./repcli status
General Info:
  Sensor Version: 3.5.2.64
  Kernel Type: System Extension
  System Extension: Running
  Kernel File Filter: Connected
  Background Scan: Standard Scan
  Sensor Restarts: 8
  Last Reset: not set
Full Disk Access Configurations:
  Repmgr: Configured Manually
  System Extension: Configured Manually
  OSQuery: Configured Manually
  Uninstall Helper: Configured Manually
  Uninstall UI: Not Configured
```

## Restart Requirements for macOS 10.15+

Beginning with macOS 10.15, a system reboot is required in the following scenarios.

- Installing the sensor for the first time on macOS 10.15+.
- Installing or upgrading the sensor in KEXT mode on macOS 11.

Endpoints that require a reboot report that state on the Dashboard or Endpoints page; search for `sensorStates:DRIVER_INIT_REBOOT_REQUIRED` on the Endpoints page to find 10.15+ devices that are in bypass mode and require a reboot.

## Special Considerations for the macOS Sensor on Big Sur

This section describes considerations for installing macOS sensors on Big Sur.

### Install a KEXT-enabled Sensor on Big Sur

Use this procedure to install a KEXT-enabled Sensor on Big Sur. On macOS 11, the attended installer defaults to installing a System Extension sensor. To install into KEXT mode, we

recommend using the `cbcloud_install_unattended.sh` install script, which is found in the mounted DMG of the sensor installer in the docs folder.

---

**Note** [Approving the KEXT via MDM for Big Sur](#) is required for KEXT operation on macOS Big Sur. If you do not have an MDM tool, do not attempt KEXT install on macOS Big Sur.

---

A new `-k` flag in `cbcloud_install_unattended.sh` signifies a KEXT sensor install. This flag also works during an update. See [macOS Command Line Parameters](#).

For Kernel Extensions (legacy System Extensions) to run on macOS Big Sur, Apple has added two new restrictions for new installs or updates:

- Kernel Extensions must be pre-approved via MDM.
- Kernel Extensions must be approved manually, and the OS requires a reboot after install.

### Procedure

- 1 Run the `cbcloud_install_unattended.sh` script. Your mount point may be slightly different than what is shown here:

```
sudo /Volumes/CBCloud-3.5.1.19/docs/cbcloud_install_unattended.sh -i /Volumes/CBCloud-3.5.1.19/CBCloud\ Install.pkg -c [Company Registration Code] -k
```

- 2 Before the install finishes, a window appears stating that a System Extension has been updated. Approve this prompt in the **Security & Privacy** pane of **System Preferences**. The KEXT must be pre-approved using an MDM provider prior to installation or this alert will be suppressed by the operating system. See [Approving the KEXT via MDM for Big Sur](#) for more information on this process. For manual Full Disk Access instructions, see [Manually Grant Sensor Full Disk Access](#).

- 3 Restart the OS to finish installing the new KEXT.

## Switch the macOS Sensor Kernel Type on Big Sur during Update

On Big Sur, to switch between kernel types during an update, run the `cbcloud_install_unattended.sh` script with either the `-k` or `-e` flag.

The `-k` flag will force a Kernel Extension sensor. The `-e` flag will force a System Extension sensor. See [macOS Command Line Parameters](#).

Your mount point might be slightly different than what is shown

```
here: sudo /Volumes/CBCloud-3.5.1.19/docs/cbcloud_install_unattended.sh -i /Volumes/CBCloud-3.5.1.19/CBCloud\ Install.pkg -c [Company Registration Code] -k
```

## Toggle between Kernel Extension and System Extension in Big Sur

We highly recommend that you perform the toggle command after you have configured MDM for both Kernel Extension and System Extension in Big Sur.

Your organization's deregistration code is required to run the toggle command. In the Carbon Black Cloud console, go to **Inventory > Endpoints > Sensor Options > View Company Codes**. In this context, the code does not uninstall anything and is used as an administrative code to enable the RepCLI tool. For more information about RepCLI, see *Managing Sensors by using RepCLI* in the User Guide.

#### Procedure

◆ Perform one of the following procedures:

■ To toggle from System Extension to Kernel Extension:

a Run the following command:

```
sudo /Applications/VMware\ Carbon\ Black\ Cloud/repcli.bundle/Contents/  
macOS/repcli setsensorkext [deregistration code]
```

b The KEXT must be manually approved. A window will appear stating that a System Extension has been updated. Approve this prompt in the **Security & Privacy** pane of **System Preferences**.

c Restart the OS.

■ To toggle from Kernel Extension to System Extension:

a Run the following command:

```
sudo /Applications/VMware\ Carbon\ Black\ Cloud/repcli.bundle/Contents/  
macOS/repcli setsensorsysex [deregistration code]
```

b Restart the OS.

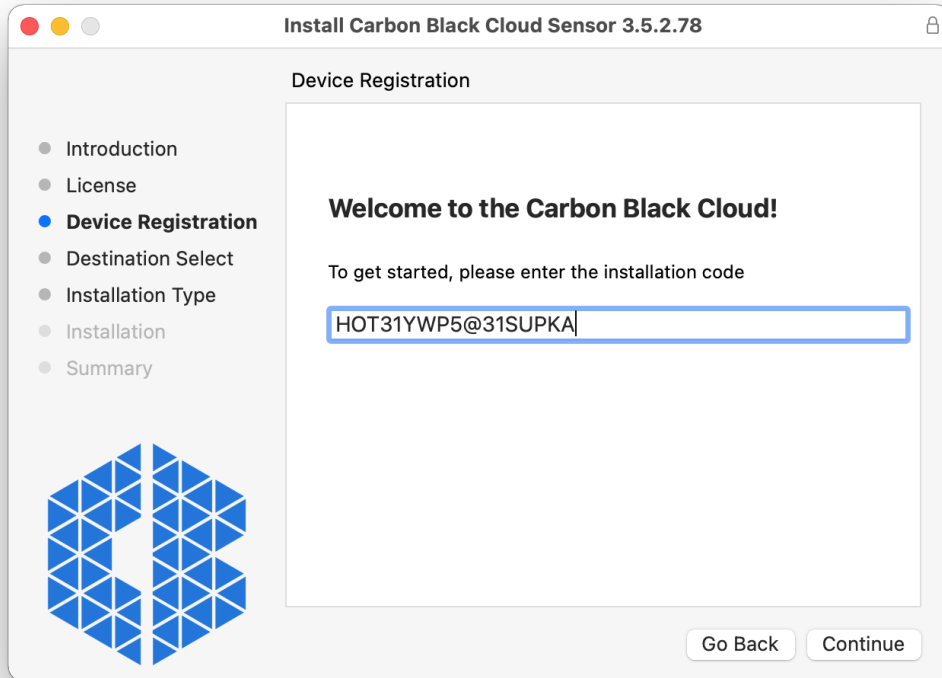
## Manually Install and Approve the Sensor on macOS 11+

This article describes how to manually install and approve the Carbon Black Cloud sensor on macOS Big Sur (macOS 11+).

#### Procedure

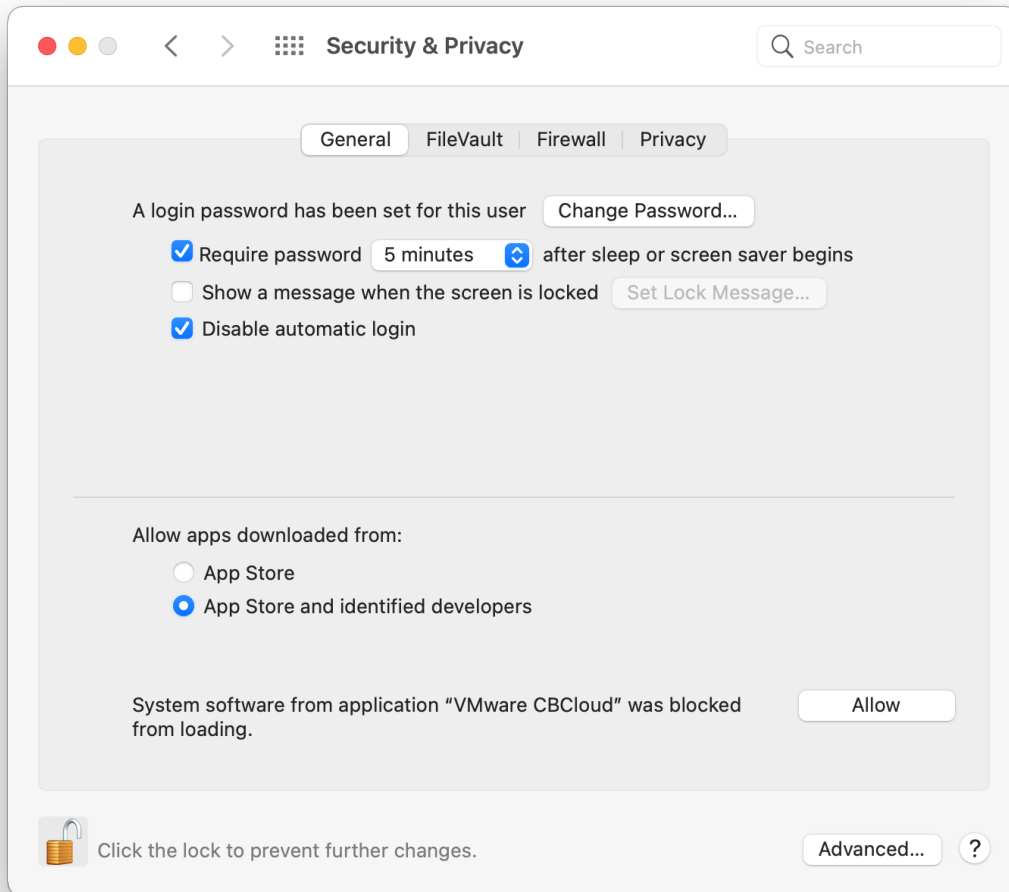
- 1 Start the Carbon Black Cloud installer. The installer will request access to your Desktop folder. Click **OK**.

- 2 Enter the sensor installation code. If the installation code is entered incorrectly, an error message will state that the installer cannot communicate with the Carbon Black Cloud. Check the installation code and try again.



- 3 When the installer finishes running, a message will notify you that you need to approve the Carbon Black Cloud system extension. Click **Open Security Preferences** to open the **Security & Privacy** pane.

- 4 On the **General** tab in the **Security & Privacy** pane, a notification indicates that the VMware CBCloud system extension was blocked from loading. Enter your administrator password to unlock the pane and then click the **Allow** button next to the notification.



---

**Note** The notification will persist until the system extension is approved, even if the endpoint is restarted. Until it is approved, the sensor will be in bypass mode.

---

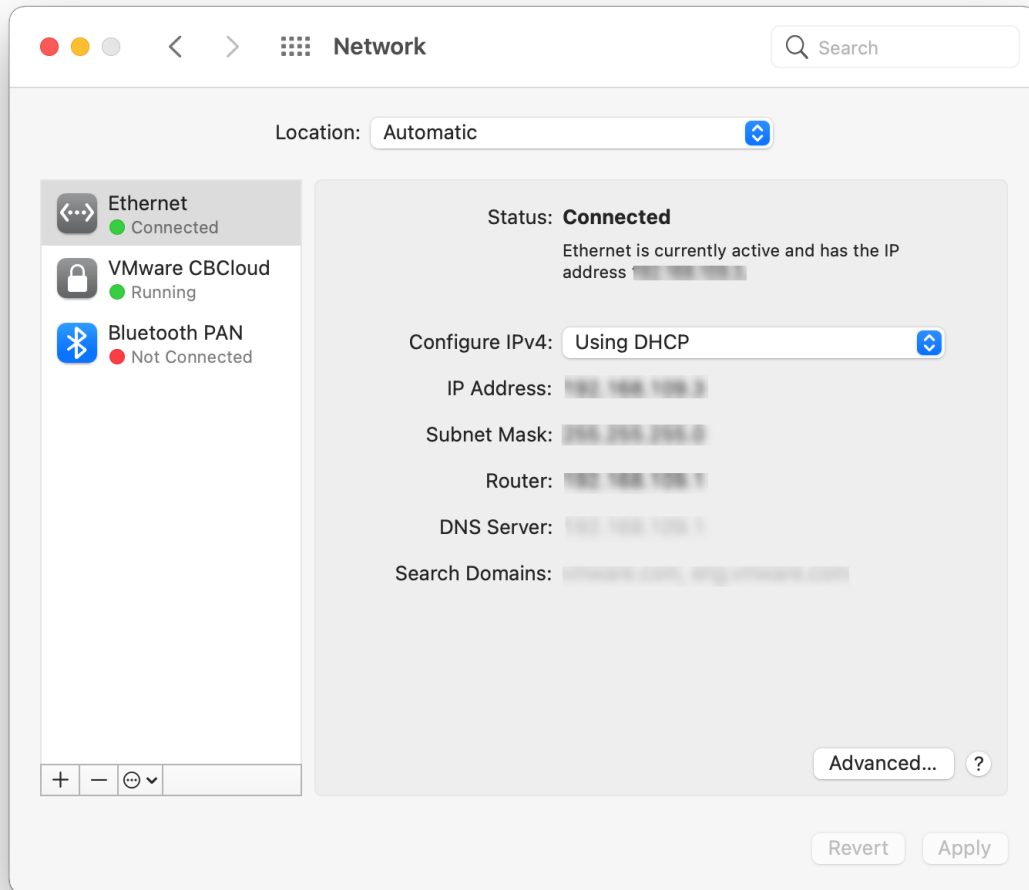
- 5 After the system extension is approved, another notification will state that the Carbon Black Cloud sensor wants to filter network content. This is required for the sensor to report network events to the Carbon Black Cloud console. Click **Allow**.

---

**Note** If you click **Don't Allow** instead of **Allow**, network events will not be sent to the backend. You can restart this prompt by running the `CBCloud.app` in the `/Applications/VMware Carbon Black Cloud` folder.

---

- To verify that the network extension is approved, go to the **Network** pane. An entry for *VMware CBCloud* will appear as a network interface.



## macOS Sensor Command Line Install

The `CBCloud Install.pkg` and `ccloud_install_unattended.sh` scripts are part of the macOS sensor release and are embedded in the CB Defense DMG. Both files are required for command line installations on macOS endpoints.

**Note** Instructions on how to create custom packages for software distribution tools are beyond the scope of this article. Carbon Black provides generic instructions on how to install the `CBCloud Install.pkg` payload on the command line, with the help of the `ccloud_install_unattended.sh` utility script. You can adapt these instructions to a software distribution tool.

## macOS utility script

The utility script can be used in the following ways:

- As-is (passed command line options to customize the install process).
- Modified to hard-code the install options and simplify the installation.
- Used as an example or guide on how to create a custom script.

A common installation method is to use the utility script as-is, push the script and the PKG payload onto the target device (both files can be bundled in a custom package), and then execute the utility script.

---

**Note** For information about mobile device management (MDM), see:

- [Approve the System Extension via MDM](#)
  - [Grant the Sensor Full Disk Access by using MDM](#)
- 

## Extract and Prepare the macOS Install Files

Before you can install sensors, you must extract and prepare the macOS install files.

### Procedure

- 1 Click **CB Cloud DMG** or mount it by using system tools. DMG is mounted to the `/Volumes/CBCloud-X.X.X.X` directory (where X.X.X.X refers to the sensor version).
- 2 Alternatively, use the `hdiutil` command to mount the downloaded sensor release disk image; for example: `hdiutil attach /path/to/CBCloud_Installer_mac_X.X.X.X`
- 3 Extract the `CBCloud Install.pkg` file from the mounted volume `/Volumes/CBCloud-X.X.X.X` directory. The `.pkg` file is the sensor installer payload.
- 4 Extract the `cbcloud_install_unattended.sh` utility script from the `/Volumes/CBCloud-X.X.X.X/docs/` directory.
- 5 The mounted volume can be unmounted because it is not needed for the remainder of the steps. You can unmount it by using `Finder`, or by running the following command: `hdiutil eject /Volumes/CBCloud-X.X.X.X`
- 6 Use the extracted `CBCloud Install.pkg` and `cbcloud_install_unattended.sh` files to create a custom package that is compatible with your software distribution tool or install the two files directly onto the target macOS device.

### Results

---

**Note** `cbcloud_install_unattended.sh` and the `CBCloud Install.pkg` payload must be extracted from the same major and minor version of released DMG file to ensure compatibility between the utility and the installer payload. If the two files do not originate from the same release, the installation might fail.

---

Typically, the extracted `cbcloud_install_unattended.sh` and the `CBCloud_Install.pkg` files are pushed to the target server endpoints. They can be used to create a custom installation bundle that is compatible with a specific software distribution tool.

---

**Note** You must always wrap the company registration code in single quotation marks. Double quotation marks are not an acceptable substitute to single quotes.

---

## Perform a macOS Sensor Command Line Installation

Follow this procedure to install a macOS sensor from the command line.

### Procedure

- 1 Extract the files from a sensor release DMG file.
- 2 Optionally, create a custom wrapper package bundle that is compatible with the selected software distribution tool. The custom package embeds the `CBCloud_Install.pkg` file, together with a utility to set up options and start the sensor PKG installation.
- 3 Install the sensor installer on the endpoint by using the supported options.

## macOS Command Line Parameters

The following common command line parameters are supported by the `cbcloud_install_unattended.sh` utility script. The parameters are passed on to the installer.

---

**Note** Parameter values must always be enclosed in single quotes.

---

To view all command line parameters, run the command together with the `-h` parameter.

Parameter	Required or Optional	Description
<code>-c</code> <code>COMPANY_CODE</code>	Required	Company registration code.
<code>-d</code>	Optional	Enter bypass mode (disabled protection) immediately after installation. You can enable protection at a later time. This mode is only recommended for test situations.
<code>-e</code>	Optional	Forces System Extension install on macOS Big Sur (the sensor will default to this mode on Big Sur and does not need to be explicitly specified).
<code>-g</code> <code>POLICY_NAME</code>	Optional	Specify a policy to which the sensor will be added.
<code>-h</code>		Displays all command line options, including advanced options that are not documented here.  Refer to the built-in help in the <code>cbcloud_install_unattended.sh</code> utility script for currently supported installation options.
<code>-i</code> <code>PKG_FILE</code>	Required	Absolute path to the PKG installer payload.



Parameter	Required or Optional	Description
-k	Optional	Forces Kernel Extension install on macOS Big Sur (pre-approvals must be in place).
-o	Optional	Username/email address override. Used during registration and for identifying the device.
-p PROXY_SERVER:PORT	Optional	Preferred Proxy server and port; for example: -p '10.5.6.7:54443' Multiple proxy servers can be provided and separated by semi-colons; for example: - p '10.5.6.8:54443;10.5.6.7:54443' If a proxy server/port are not specified but are required, the sensor will attempt proxy auto-detection. See <a href="#">Configure a Proxy</a> .
-s	Optional	Background scan enabled ("on") or disabled ("off"). Default is enabled. Cloud policy overrides this setting.
-t	Optional	File upload limit (in MB). Default is no limit.
-u	Optional	Disable auto-update. Auto-update is enabled by default.
-v	Optional	Show version of this script. Major and minor versions should match the version of the Carbon Black Cloud package being deployed.
-x PROXY_USER:PASSWORD	Optional	Proxy credentials to use for the proxy server, if required. These apply whether the proxy server is auto-detected or specified. Example: -x 'proxy_user:proxy_password' If proxy credentials are not specified, but are required by the proxy server, the macOS sensor will attempt to detect and use proxy credentials that are stored in the keychain that match the detected or specified proxy server.
--skip-kext-approval-check=1	Optional	Allows for >=3.1 sensor install/upgrade to run on macOS >=10.13 even if KEXT approval has not been done prior to the install/upgrade. KEXT approval can be deferred until after the sensor install/upgrade.  <b>Warning</b> macOS 11.0 requires a KEXT MDM pre-approval to install a KEXT sensor.
--disable-live-response=1	Optional	Disable Live Response.
--disable-sysextnetwork-extension	Optional	Disable network extension. Only available in System Extension mode on macOS 11 and later. Network Extension is enabled by default.
--disable-upgrade-jitter=1	Optional	Disable auto-update jitter.

## Obfuscation of command line inputs

Endpoint users might input sensitive data into the command line. The obfuscation of command line inputs protects against unauthorized users accessing the data in plain text in the sensor `.log` files and the sensor databases. You can obfuscate command line inputs by using the following argument in the unattended install script: `--enable-hide-command-lines=1`

The setting enables the obfuscation of command line input in sensor .log files and databases. The data in the Carbon Black Cloud console is not obfuscated.

## macOS Command Line Install Examples

Review the following examples for macOS command line installations.

The following commands should be on a single line.

The following examples assume that the required files are installed to the target device /tmp/ directory.

### To run a command line install with required parameters

```
sudo /tmp/cbcloud_install_unattended.sh -i '/tmp/CBCloud Install.pkg' -c 'XYZ'
```

### To specify a policy for the sensor

```
sudo /tmp/cbcloud_install_unattended.sh -i '/tmp/CBCloud Install.pkg' -c 'XYZ' -g 'Monitored'
```

## Deploying macOS Sensors on Big Sur and Later by using Jamf Pro

You can use Jamf Pro to deploy the Carbon Black Cloud macOS sensor on macOS systems that are running Big Sur.

We recommend that you use the latest macOS sensor version for your deployment. The configuration documented here was tested using the following versions:

- macOS Big Sur 11.4
- Jamf Pro 10.30
- macOS sensor 3.5.3.82

---

**Note** We offer this procedure as guidance only. VMware does not provide official support of Jamf software.

---

The basic deployment workflow is as follows:

- 1 Create and deploy a configuration profile.
- 2 Deploy a distribution policy that subsequently deploys a package that you created in Jamf Pro to a temporary location. The policy then runs a script that contains the installer package location and the company registration. Optionally, the script can install the sensor in Kernel extension mode. System Extension is the default mode.

To deploy the macOS sensor by using Jamf Pro, perform the following steps:

- 1 Download a macOS sensor kit and obtain a company registration code.
- 2 Create a package using Jamf Composer.

- 3 Obtain, prepare, and upload the installation script.
- 4 Create and deploy a configuration profile.
- 5 Create and assign smart computer groups.

## Obtain and Prepare the Sensor

To obtain and prepare the sensor kit for deployment, perform the following procedure.

### Procedure

- 1 Follow these steps to download the latest macOS sensor kit: [Download Sensor Kits](#).
- 2 Mount the DMG file that you downloaded in Step 1 and locate the `CBCloud Install.pkg` file.
- 3 Follow these steps to find and make note of the company registration code: [Obtain a Company Registration Code](#).

---

**Important** Do not generate a new company registration code; simply make a note of the current code.

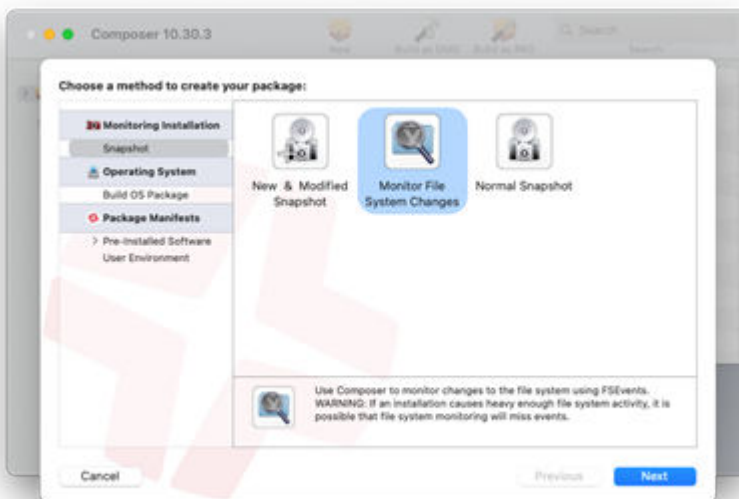
---

## Create a Package by using Jamf Composer

Create a package for the installer in a temporary location on the endpoint.

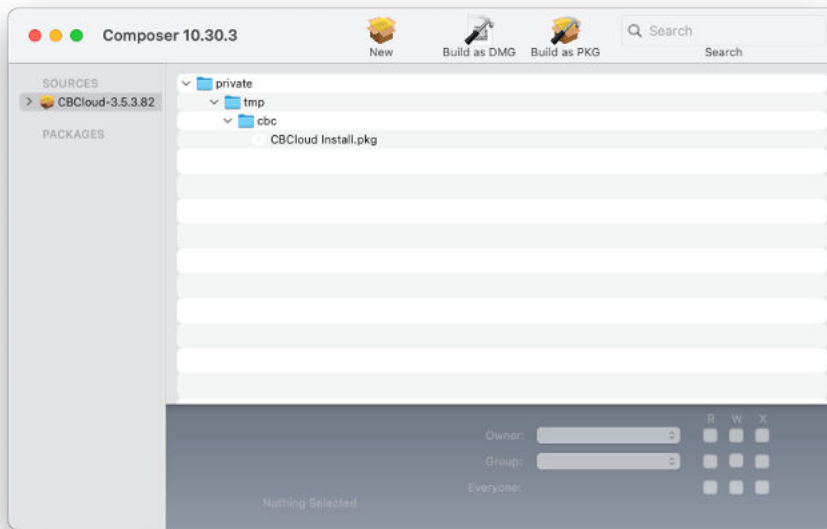
### Procedure

- 1 Open Jamf Composer and go to **Preferences**. Remove `/private/tmp` from the **Exclusion List** and save the settings.
- 2 Create a new package using your preferred method - in this case, **Monitor File System Changes**. Click **Next**.



- 3 Enter an appropriate package name and click **Next**.

- 4 Create the file path `/private/tmp/cbc`. Copy the `CBCloud Install.pkg` file from the downloaded sensor kit into this directory. Click **Create Package Source** in Jamf Composer.



- 5 Make sure that `/private/tmp/cbc` is the only file path included in the package.
- 6 Click **Build as DMG** and save.

## Modify the Installation Script

Modify the `cbcloud_install_unattended.sh` script for the macOS sensor deployment.

### Procedure

- 1 In the mounted DMG, locate the `cbcloud_install_unattended.sh` file in the `docs` folder.
- 2 Copy `cbcloud_install_unattended.sh` to a location where it can be modified.
- 3 Open `cbcloud_install_unattended.sh` in a plain text editor and locate the following lines (beginning on line 49):

```
#options
CBC_INSTALLER=""
COMPANY_OR_USER_CODE=""
```

- 4 Set `CBC_INSTALLER` as the temporary location of the `CBCloud Install.pkg` that you established in [Create a Package by using Jamf Composer](#).
- 5 Modify the `COMPANY_OR_USER_CODE` to be the company registration code. For example:

```
CBC_INSTALLER="/private/tmp/cbc/CBCloud Install.pkg"
COMPANY_OR_USER_CODE="3TAB99SW2021"
```

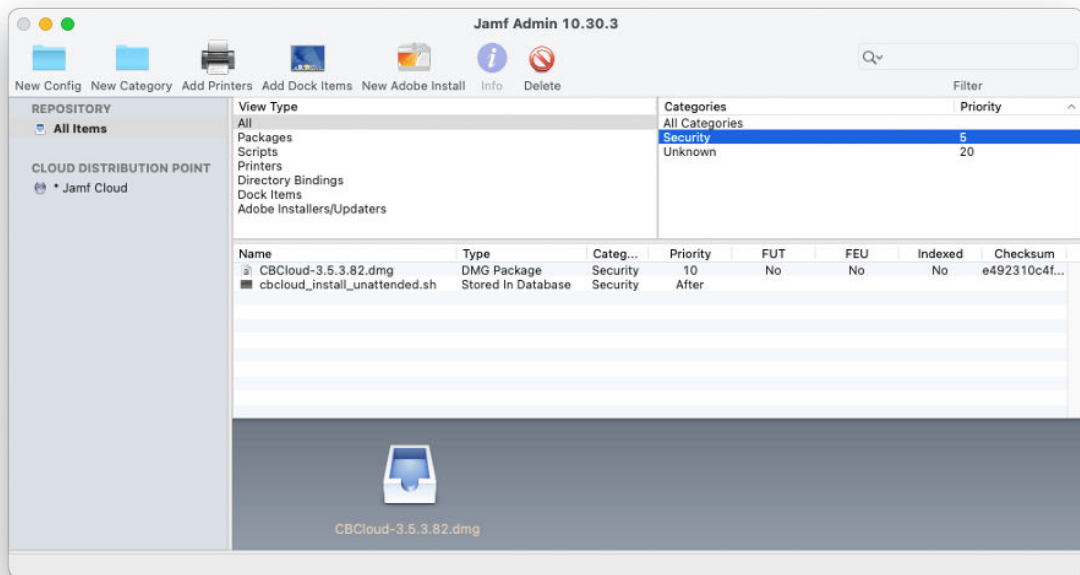
- 6 Save the changes.

## Upload macOS Sensor DMG and Installation Script to Jamf Pro

Use Jamf Admin to upload the macOS Sensor DMG and Installation Script to Jamf Pro.

### Procedure

- 1 Start Jamf Admin.
- 2 Add the DMG and installation script files to the repository.
- 3 Set the `cbcloud_install_unattended.sh` file **Priority** to **After**. Example:



## Creating a Configuration Profile

To allow the deployment of Carbon Black Cloud macOS sensors on macOS Big Sur systems, create a configuration profile. Include the System Extension or Kernel Extension payloads as required for your implementation.

### Configure Configuration Profile General Settings

To configure the **General** settings in the configuration profile, define the following parameters.

#### Procedure

- 1 Provide a descriptive **Name** for the profile. For example, "Carbon Black Cloud Settings - System Extension".
- 2 Add a brief **Description** to the profile.
- 3 Set the **Category** for your organization.
- 4 Set the **Level** to **Computer Level**.

## 5 Set the Distribution Method to Install Automatically.

### Set Privacy Preferences Policy Control in the Configuration Profile

To ensure full functionality of the macOS sensor, enter each App Access sub-payload from the following table. For all sub-payloads, the **Identifier Type** is `Bundle ID`, and the **Application or Service** is `SystemPolicyAllFiles` with **Access** set to `Allow`.

Identifier	Code Requirement
<code>com.vmware.carbonblack.cloud.daemon</code>	<pre> identifier "com.vmware.carbonblack.cloud.daemon" and anchor apple generic and certificate 1[field.1.2.840.113635.100.6.2.6] /* exists */ and certificate leaf[field.1.2.840.113635.100.6.1.13] /* exists */ and certificate leaf[subject.OU] = "7AGZLNQ2S2T" </pre>
<code>com.vmware.carbonblack.cloud.se-agent.extension</code>	<pre> identifier "com.vmware.carbonblack.cloud.se-agent.extension" and anchor apple generic and certificate 1[field.1.2.840.113635.100.6.2.6] /* exists */ and certificate leaf[field.1.2.840.113635.100.6.1.13] /* exists */ and certificate leaf[subject.OU] = "7AGZLNQ2S2T" </pre>
<code>com.vmware.carbonblack.cloud.osqueryi</code>	<pre> identifier "com.vmware.carbonblack.osqueryi" and anchor apple generic and certificate 1[field.1.2.840.113635.100.6.2.6] /* exists */ and certificate leaf[field.1.2.840.113635.100.6.1.13] /* exists */ and certificate leaf[subject.OU] = "7AGZLNQ2S2T" </pre>
<code>com.vmware.carbonblack.cloud.uninstall</code>	<pre> identifier "com.vmware.carbonblack.cloud.uninstall" and anchor apple generic and certificate 1[field.1.2.840.113635.100.6.2.6] /* exists */ and certificate leaf[field.1.2.840.113635.100.6.1.13] /* exists */ and certificate leaf[subject.OU] = "7AGZLNQ2S2T" </pre>
<code>com.vmware.carbonblack.cloud.uninstallerui</code>	<pre> identifier "com.vmware.carbonblack.cloud.uninstallerui" and anchor apple generic and certificate 1[field.1.2.840.113635.100.6.2.6] /* exists */ and certificate leaf[field.1.2.840.113635.100.6.1.13] /* exists */ and certificate leaf[subject.OU] = "7AGZLNQ2S2T" </pre>

## Enable System Extension Payloads in the Configuration Profile

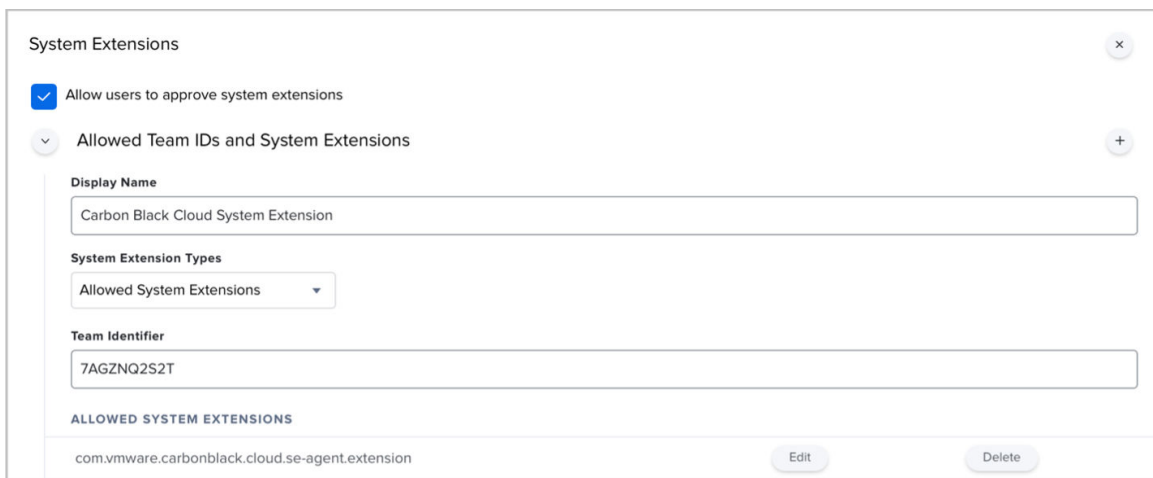
If you are deploying the macOS sensor in System Extension mode, include a System Extensions payload. You can optionally include both System Extension and Kernel Extension payloads to provide flexibility.

### Procedure

- 1 To optionally enable users to approve extensions that are not included in this payload, select **Allow Users to approve system extensions**.
- 2 Define the following settings in **Allowed Team IDs and System Extensions**:
  - **Display Name**: Carbon Black Cloud System Extension
  - **System Extension Types**: **Allowed System Extensions**
  - **Team Identifier**: 7AGZLNQ2S2T
  - **Allowed System Extensions**:

```
com.vmware.carbonblack.cloud.se-agent.extension
```

For example:



## Enable Kernel Extension Payloads in the Configuration Profile

If you are deploying the macOS sensor in Kernel Extension mode (macOS 10.14 - macOS 11), include an Approved Kernel Extension payload. You can optionally include both System Extension and Kernel Extension payloads to provide flexibility.

### Procedure

- 1 Select **Allow Users to approve system extensions**.
- 2 Optionally select **Allow standard users to approve legacy Kernel Extensions (macOS 11 or later)**: This selection is dependent on your environment.

### 3 Define the following settings in **Allowed Team IDs and Kernel Extensions**:

- **Approved Team ID Display Name:** VMware Carbon Black
- **Approved Team ID:** 7AGZMQ2S2T
- **Approved Kernel Extension Display Name:** Carbon Black Cloud Sensor
- **Approved Kernel Extension Kernel Extension Bundle ID:**

```
com.carbonblack.defense.kext
```

For example:

**Note** This payload pre-approves the Kernel Extension. The user must still enable the Kernel Extension in the **Security & Privacy** section of **System Preferences** on the endpoint after you have installed the sensor. The user must then reboot the endpoint.

## Set the Content Filter in the Configuration Profile

To enable the configuration deployment without requiring user approval of the network extension, create the following payload.

### Procedure

#### 1 Set the **Filter Name**:

```
VMware Carbon Black Cloud Network Extension Filter.
```

#### 2 Set the **Identifier**:

```
com.vmware.carbonblack.cloud.se-agent
```

#### 3 Set the **Socket Filter Bundle Identifier**:

```
com.vmware.carbonblack.cloud.se-agent.extension
```



#### 4 Set the **Socket Filter Designated Requirement:**

```
identifier "com.vmware.carbonblack.cloud.se-agent.extension" and anchor apple generic
and certificate 1[field.1.2.840.113635.100.6.2.6] /* exists */ and certificate
leaf[field.1.2.840.113635.100.6.1.13] /* exists */ and certificate leaf[subject.OU] =
"7AGZNQ2S2T"
```

#### 5 Set the **Network Filter Bundle Identifier:**

```
com.vmware.carbonblack.cloud.se-agent.extension
```

#### 6 Set the **Network Filter Designated Requirement:**

```
identifier "com.vmware.carbonblack.cloud.se-agent.extension" and anchor apple generic
and certificate 1[field.1.2.840.113635.100.6.2.6] /* exists */ and certificate
leaf[field.1.2.840.113635.100.6.1.13] /* exists */ and certificate leaf[subject.OU] =
"7AGZNQ2S2T"
```

#### 7 Save the configuration profile.

Example:

**Content Filter**  
Settings configured: 4

**Filter Name**  
Display name of the filter in the app and on the device  
VMware Carbon Black Cloud Network Extension Filter  
Required

**Identifier**  
Identifier for the filter plug-in  
com.vmware.carbonblack.cloud.se-agent  
Required

**Socket Filter**  
**Socket Filter Bundle Identifier** Bundle identifier of the socket filter provider system extension  
com.vmware.carbonblack.cloud.se-agentExtension  
Required

**Socket Filter Designated Requirement** Designated requirement of the socket filter provider system extension  
identifier "com.vmware.carbonblack.cloud.se-agent.extension" and anchor apple generic and certificate [field.1.2.840.113635.100.6.2.6] /\* exists \*/ and certificate leaf[field.1.2.840.113635.100.6.1.13] /\* exists \*/ and certificate leaf[subject.OU] = "7AGZNQ2S2T"  
Required

**Network Filter**  
**Network Filter Bundle Identifier** Bundle identifier of the network filter provider system extension  
com.vmware.carbonblack.cloud.se-agent.extension  
Required

**Network Filter Designated Requirement** Designated requirement of the network filter provider system extension  
identifier "com.vmware.carbonblack.cloud.se-agent.extension" and anchor apple generic and certificate [field.1.2.840.113635.100.6.2.6] /\* exists \*/ and certificate leaf[field.1.2.840.113635.100.6.1.13] /\* exists \*/ and certificate leaf[subject.OU] = "7AGZNQ2S2T"  
Required

## Create a Software Distribution Policy

In Jamf Pro, create a software distribution policy.

## Procedure

- 1 Provide a descriptive policy **Display Name**.
- 2 Set the **Trigger** to **Recurring Check-in**.

Example:

**General**

**Display Name** Display name for the policy

Carbon Black Cloud Sensor 3.5.3.82 (System Extension)

**Enabled**

**Category** Category to add the policy to

Security

**Trigger** Event(s) to use to initiate the policy

**Startup**  
When a computer starts up. A startup script that checks for policies must be configured in Jamf Pro for this to work

**Login**  
When a user logs in to a computer. A login hook that checks for policies must be configured in Jamf Pro for this to work

**Logout**  
When a user logs out of a computer. A logout hook that checks for policies must be configured in Jamf Pro for this to work

**Network State Change**  
When a computer's network state changes (e.g., when the network connection changes, when the computer name changes, when the IP address changes)

**Enrollment Complete**  
Immediately after a computer completes the enrollment process

**Recurring Check-in**  
At the recurring check-in frequency configured in Jamf Pro

- 3 On the **Packages** tab, select the Carbon Black Cloud DMG file that you built using Jamf Composer.
- 4 Set the **Distribution Point** to **Each computer's default distribution point**.
- 5 Set **Action** to **Install**.
- 6 On the **Scripts** tab, select the `cbcloud_install_unattended.sh` file that you modified. Set the priority to **After**. No parameter values are required.
- 7 Save the policy.

## Create and Assign Smart Computer Groups

Create and assign two smart computer groups for the macOS sensor deployment.

You must set the scope of two smart computer groups to the configuration profile and software policy. This scope ensures that the configuration profile is deployed to the endpoint before you install the Carbon Black Cloud macOS sensor. If you deploy the Carbon Black Cloud sensor without having deployed the configuration profile, the user receives approval prompts. If the prompts are not approved, the sensor is not fully functional.

---

**Note** Before you can select the Carbon Black Cloud Settings Configuration Profile in the second smart group (steps 5 through 10), you must first deploy the configuration profile to a macOS endpoint and update the endpoint inventory. After Jamf Pro recognizes the installed configuration profile, the profile becomes a selectable option when you are creating the smart computer group.

---

#### Procedure

- 1 On the Jamf Pro dashboard, on the **Computers** tab, click **Smart Computer Groups** and then click **+ New**.
- 2 On the **Computer Group** tab, enter a name such as “macOS Big Sur Computers”.
- 3 Click the **Criteria** tab and set the following criteria:
  - **Criteria:** Operating System
  - **Operator:** like
  - **Value:** 11
- 4 Save the smart computer group.
- 5 On the Jamf Pro dashboard, on the **Computer** tab, click **Smart Computer Groups** and then click **+ New**.
- 6 On the **Computer Group** tab, enter a name such as “Carbon Black Cloud Settings”.
- 7 Click the **Criteria** tab and set the following criteria:
  - **Criteria:** Profile Name
  - **Operator:** Has
  - **Value:** Click the ... menu and select the Carbon Black Cloud Settings – System Extension Configuration Profile.
- 8 Save the smart computer group.
- 9 For the configuration profile, assign the “**macOS Big Sur computers**” smart computer group to the scope.

- 10 For the policy, assign the “Carbon Black Cloud Settings” smart computer group to the scope. After the smart computer groups are assigned, deployment begins.

---

**Important** If deploying the Kernel Extension, you must approve it after it is installed in **System Preferences>Security & Privacy**. This is an Apple-imposed requirement. This approval is not required if deploying the System Extension. We recommend that you ask the user to approve this extension by using a notification in the deployment policy.

---

## Validate a Healthy System Extension Sensor through RepCLI

Follow this procedure to validate a healthy System Extension Sensor on Big Sur.

### Procedure

- ◆ Run the following command:

```
sudo /Applications/VMware\ Carbon\ Black\ Cloud/repcli.bundle/Contents/macOS/repcli
status
```

For more information about RepCLI, see *Managing Sensors by using RepCLI* in the User Guide.

The following results are expected:

#### General Info

- Kernel Type: System Extension
- System Extension: Running
- Kernel File Filter: Connected

#### Sensor State

- State: Enabled
- SvcStable: Yes (Might take a few minutes after install to reach SvcStable)
- Cloud Status:
- Registered: Yes

```

Last login: Tue Aug 25 07:59:46 on
~ % sudo /Applications/VMware\ Carbon\ Black\ Cloud/repcli.bundle/Contents/MacOS/repcli status
>Password:
General Info:
  Sensor Version: 3.5.0.117
  Kernel Type: System Extension
  System Extension: Running
  Kernel File Filter: Connected
  Background Scan: Standard Scan
  Sensor Restarts: 1
  Last Reset: not set
Sensor State:
  State: Enabled
  Details:
    LiveResponse:NoSession
    LiveResponse:NoKillSwitch
    LiveResponse:Disabled
    FullDiskAccess:NotEnabled
  SveStable: Yes
  Boot Count: 3
  First Boot After OS Upgrade: No
  Service Uptime: 3336500 ms
  Service Waketime: 62500 ms
Cloud Status:
  Server Address:
  Registered: Yes
  Next Check-In: 3 sec
  Private Logging: Disabled
  Next Cloud Upgrade: None
  MDM Device ID: 564DE49A-1B61-F8E0-F19F-799DD1CAAF39

```

What to do next

---

**Note** For information about mobile device management (MDM), see:

- [Approve the System Extension via MDM](#)
  - [Grant the Sensor Full Disk Access by using MDM](#)
- 

## Address the Extension Warning Post-install

A post-install warning can occur after installing a macOS sensor. This procedure resolves the problem.

### Procedure

- 1 Download the installer: macOS.
- 2 When prompted to approve the CB Defense kernel extension, click **OK**.
- 3 When the **System Extension Blocked** message appears, click **Open Security Preferences**.
- 4 Click **Allow** next to **System software from developer “Carbon Black, Inc.” was blocked**.
- 5 Double-click the Carbon Black icon and copy/paste the installation code from a text editor.

What to do next

---

**Note** For information about mobile device management (MDM), see:

- [Approve the System Extension via MDM](#)
  - [Grant the Sensor Full Disk Access by using MDM](#)
-

## Resolve macOS Sensor Upgrade or Install Error due to Existing System Extension

Prior to macOS sensor 3.8, you could not uninstall the System Extension without first disabling System Integrity Protection (SIP). This topic presents a solution to that problem.

In some cases, the installation or upgrade of a macOS 3.51+ sensor fails with the following error message in the Preinstall log:

```
failed to uninstall system extension
program terminated with error code: 4096
```

If you check `Systemextensionsctl`, Carbon Black Cloud errantly displays as installed:

```
sudo Systemextensionsctl list

--- com.apple.system_extension.endpoint_security
enabled active teamID bundleID (version) name [state]
* * 7AGZNQ2S2T com.vmware.carbonblack.cloud.se-agent.extension (3.5.2fc76/3.5.2fc76)
com.vmware.carbonblack.cloud.se-agent.extension [activated enabled]
```

To correct this issue without disabling SIP, perform the following procedure.

### Procedure

- 1 Download a macOS 3.8+ sensor DMG to the affected assets.
- 2 Go to the `docs/` directory of the sensor DMG.
- 3 Run `CBCloud Cleanup Tool.pkg` and follow the onscreen instructions.

---

**Note** The `CBCloud Cleanup Tool.pkg` will not remove the system extension if the asset is in a healthy, installed state.

---

Upon successful completion, the system extension is in the `[Terminated waiting to uninstall on reboot]` state. A reboot is not required.

- 4 Proceed to install or upgrade the sensor.

## macOS Services, Utilities, and Uninstaller

This section lists the macOS sensor services, utilities, and uninstaller files that reside on the endpoints.

### macOS installed services for 3.5.0 and lower

- Sensor Driver Bundle: `/System/Library/Extensions/CBDefenseSensor.kext`
- Sensor Service: `/Applications/Confer.app/Contents/MacOS/repmgr`

- Sensor UI: `/Applications/Confer.app/Contents/MacOS/CBDefense`

## macOS installed services for 3.5.1 and higher

- Sensor Driver Bundle: `/Applications/VMware Carbon Black Cloud/`
- Sensor data directories: `/Library/Application Support/com.vmware.carbonblack.cloud/`
- Sensor Service: `/Applications/VMware Carbon Black Cloud/repmgr.bundle/Contents/MacOS/repmgr`
- Sensor UI: `/Applications/VMware Carbon Black Cloud/CBCloudUI.bundle/Contents/MacOS/CBCloudUI`

## macOS installed utilities

- Uninstaller helper: `/VMware Carbon Black Cloud/uninstall.bundle/Contents/MacOS/uninstall`
- Upgrade helper: `/VMware Carbon Black Cloud/UpgradeHelper.bundle/Contents/MacOS/UpgradeHelper`
- RepCLI: `/VMware Carbon Black Cloud/repcli.bundle/Contents/MacOS/repcli`
- CB Cloud Cleanup Tool: available with macOS sensor version 3.8.0 and compatible with all earlier macOS sensor versions

## macos uninstaller

- 3.X: `/Applications/Confer.app/uninstall`
- 1.X sensor: `/Applications/Confer.app/uninstall.sh`
- `CLI_USERS=sid` #Required, needed to interact with sensor locally

## Installing macOS Sensors on Endpoints by using Workspace ONE UEM

This section describes how to deploy the Carbon Black Cloud macOS sensor through Workspace ONE UEM.

---

**Note** The steps can vary based on the specific version of macOS, Carbon Black Cloud macOS sensor, and Workspace ONE UEM that you are using. This content was created using macOS Big Sur 11.1, Workspace ONE UEM 2101, and the Carbon Black Cloud macOS sensor version 3.5.1.19.

---

## Extension Types

Starting with macOS 11, the Carbon Black Cloud macOS sensor (v3.5.1) operates by default in user-space by using System Extensions (user-space) instead of Kernel Extensions (KEXTs) that were used in prior versions. As a result, some functionality is temporarily unavailable when using the sensor in System Extension mode on macOS 11 and later versions. Using the sensor in KEXT mode achieves the same functionality on macOS 11 as seen in older operating systems. Differences in functionality are listed at [Carbon Black Cloud macOS Sensor Operating Environment Requirements](#).

## Prerequisites

- Workspace ONE UEM with permissions to manage devices and applications
- Carbon Black Cloud console access and admin account credentials
- An endpoint running macOS to test the integration

## Prepare to Install macOS Sensors

Before deploying the Carbon Black Cloud sensor for macOS on your endpoint you must obtain an installation code and download the sensor installer Carbon Black Cloud. As Workspace ONE administrator, you use the registration code to connect your endpoint to the respective Carbon Black Cloud environment tenant.

There are two methods to retrieve the sensor installer and the installation code.

- You can send a sensor installation request through the Carbon Black Cloud console. Then, you receive an email invitation that contains the installer download link and a unique single use installation code. For more details, see [Method 1: Invite Users to Install Sensors on Endpoints](#).
- You can use the company registration code and download a sensor kit that matches the operating system of the endpoint. For more details, see [Obtain a Company Registration Code](#) and [Download Sensor Kits](#).

By default, the Carbon Black Cloud macOS sensor version 3.5.1.19 and later installs System Extensions on macOS Big Sur 11.0 and later.

## Deploying the Carbon Black Cloud sensor for macOS Manually with System Extensions

By default, the Carbon Black Cloud macOS sensor version 3.5.1.19 and later installs System Extensions on macOS Big Sur 11.0 and later.

As a Workspace ONE administrator, you must create a macOS device profile by using the Workspace ONE UEM console and then deploy the Carbon Black Cloud sensor for macOS with System Extensions.



## Prerequisites

- Check for updates on Carbon Black Cloud sensor for macOS version 3.5.1.19 and later.
  - For updates on Carbon Black Cloud sensor for macOS version 3.6.1.10 and 3.6.2.110, see *macOS Sensor Release Notes*, part of the *Carbon Black Cloud documentation*.
  - For updates on Carbon Black Cloud sensor for macOS version 3.5.1.19 to 3.6.1.10 see [Carbon Black Cloud macOS Sensor Release Notes \(Carbon Black Cloud Community\)](#).
- Make sure you are familiar with smart groups. For information on creating smart groups with the UEM console, see *Getting Started* → *Console Basics* → *Assignment Groups*, part of the *VMware Workspace ONE UEM Console Documentation*.
- If you plan to deploy the Carbon Black Cloud sensor in KEXT mode, VMware recommends submitting the applicable kernel extension IDs for approval by Workspace ONE UEM before installing or upgrading the macOS sensor version 3.5 and later. For details, see [Approve the Kernel Extension \(macOS 10.13 – macOS 11\)](#).
- To deploy the Carbon Black Cloud sensor in System Extension mode, VMware recommends submitting the applicable system extension IDs for approval by Workspace ONE UEM before installing or upgrading the macOS sensor version 3.5 and later. For details, see [Approving the System Extension and Network Extension for macOS 11+](#).

## Procedure

### 1 [Creating a Configuration Profile](#)

To allow the deployment of Carbon Black Cloud macOS sensors on macOS Big Sur systems, create a configuration profile. Include the System Extension or Kernel Extension payloads as required for your implementation.

### 2 [Manually Install and Approve the Sensor on macOS 11+](#)

After publishing the device configuration profile, manually install and approve the Carbon Black Cloud sensor on macOS Big Sur (macOS 11+).

### 3 [Confirm the Carbon Black Cloud Sensor Installed on the macOS Device](#)

You can use the RepCLI output to verify that the Carbon Black Cloud sensor for macOS is installed successfully on the device. For more information about RepCLI, see *Managing Sensors by using RepCLI* in the User Guide.

## Creating a Configuration Profile

To allow the deployment of Carbon Black Cloud macOS sensors on macOS Big Sur systems, create a configuration profile. Include the System Extension or Kernel Extension payloads as required for your implementation.

## What to read next

### Procedure

#### 1 [Configure Device Profile General Settings](#)

To configure the **General** settings in the device configuration profile and ensure that the Carbon Black sensor has the appropriate permissions granted prior installation, define the following parameters in the Workspace ONE UEM console.

#### 2 [Enable Kernel Extension Payloads in the Configuration Profile](#)

If you are deploying the macOS sensor in Kernel Extension mode (macOS 10.14 - macOS 11), include an approved Kernel Extension payload. You can optionally include both System Extension and Kernel Extension payloads to provide flexibility.

#### 3 [Enable System Extension Payloads in the Configuration Profile](#)

If you are deploying the macOS sensor in System Extension mode, include a System Extensions payload. You can optionally include both System Extension and Kernel Extension payloads to provide flexibility.

#### 4 [Set Privacy Preferences Policy Control in the Configuration Profile](#)

For the macOS sensor to operate at full functionality on an endpoint, the sensor must have full disk access on the endpoint. This payload grants the macOS sensor full disk access.

#### 5 [Set the Content Filter in the Configuration Profile](#)

To enable the configuration deployment without requiring user approval of the network extension, configure the custom settings payload.

### Configure Device Profile General Settings

To configure the **General** settings in the device configuration profile and ensure that the Carbon Black sensor has the appropriate permissions granted prior installation, define the following parameters in the Workspace ONE UEM console.

### Procedure

#### 1 Provide a descriptive **Name** for the profile.

For example, **Carbon Black Settings**.

#### 2 Add a brief **Description** to the profile.

#### 3 Set the **Assignment Type** to **Auto**.

#### 4 Populate the **Smart Groups** text box with the smart groups you used for deploying the Carbon Black Cloud sensor installer to macOS Big Sur 11.0 and later.

### Enable Kernel Extension Payloads in the Configuration Profile

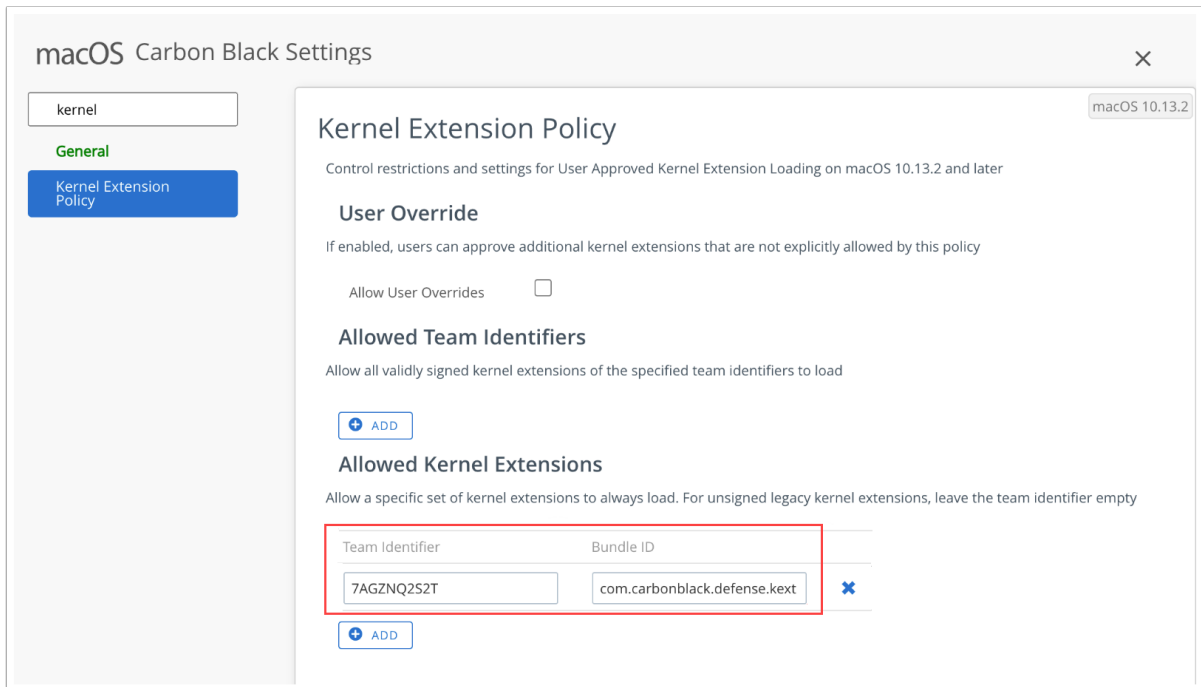
If you are deploying the macOS sensor in Kernel Extension mode (macOS 10.14 - macOS 11), include an approved Kernel Extension payload. You can optionally include both System Extension and Kernel Extension payloads to provide flexibility.

## Procedure

- 1 Enter **kernel** in the search text box of the device configuration profile.
- 2 Select the `Kernel Extension Policy` payload option and click **Configure**.
- 3 Enter the Carbon Black Cloud team identifier and the Carbon Black Cloud kernel extension bundle ID.

For example:

- **Team identifier:** `7AGZNQ2S2T`
- **Bundle ID:** `com.carbonblack.defense.kext`



**Note** This payload pre-approves the Kernel Extension. The user must still enable the Kernel Extension in the **Security & Privacy** section of **System Preferences** on the endpoint after you have installed the sensor. The user must then reboot the endpoint.

## Enable System Extension Payloads in the Configuration Profile

If you are deploying the macOS sensor in System Extension mode, include a System Extensions payload. You can optionally include both System Extension and Kernel Extension payloads to provide flexibility.

## Procedure

- 1 Enter **system** in the search text box of the device configuration profile.

- 2 Select the `System Extensions` payload option and click **Configure**.

**Note** If you are deploying the Sensor in Kernel Extension mode, pre-staging the System Extensions settings prepares you for a later migration from Kernel Extension to System Extensions.

- 3 Click **Add System Extension**.
- 4 Enter the Carbon Black Cloud team identifier and the Carbon Black Cloud system extension bundle ID.

For example:

- **Team Identifier:** `7AGZNQ2S2T`
- **Bundle ID:** `com.vmware.carbonblack.cloud.se-agent.extension`

macOS Add a New Apple macOS Profile

system

General

System Extensions

System Extensions macOS 10.15

Controls restrictions and settings for System Extensions loading on macOS 10.15 and later.

**User Override**

If enabled, users can approve additional system extensions that are not explicitly allowed by this policy.

Allow User Overrides

**Allowed System Extension Types**

Allow all or some system extension types to load. Team Identifier rule takes precedence over global settings.

Team Identifier*	Drivers	Endpoint Security	Network
*	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

+ ADD SYSTEM EXTENSION TYPE

**Allowed System Extensions**

Allow a specific set of extensions to always load. Either ID is optional, but both can be provided.

Team Identifier	Bundle Identifier
7AGZNQ2S2T	com.vmware.carbonblack.cloud

+ ADD SYSTEM EXTENSION

SAVE AND PUBLISH CANCEL

### Set Privacy Preferences Policy Control in the Configuration Profile

For the macOS sensor to operate at full functionality on an endpoint, the sensor must have full disk access on the endpoint. This payload grants the macOS sensor full disk access.

To ensure full functionality of the macOS sensor, enter each App Access sub-payload from the following table. For all sub-payloads, the **Identifier Type** is `Bundle ID`, and the **Application or Service** is `SystemPolicyAllFiles` with **Access** set to `Allow`.

Identifier	Code Requirement
<code>com.vmware.carbonblack.cloud.daemon</code>	<pre> identifier "com.vmware.carbonblack.cloud.daemon" and anchor apple generic and certificate 1[field.1.2.840.113635.100.6.2.6] /* exists */ and certificate leaf[field.1.2.840.113635.100.6.1.13] /* exists */ and certificate leaf[subject.OU] = "7AGZNQ2S2T" </pre>
<code>com.vmware.carbonblack.cloud.se-agent.extension</code>	<pre> identifier "com.vmware.carbonblack.cloud.se- agent.extension" and anchor apple generic and certificate 1[field.1.2.840.113635.100.6.2.6] /* exists */ and certificate leaf[field.1.2.840.113635.100.6.1.13] /* exists */ and certificate leaf[subject.OU] = "7AGZNQ2S2T" </pre>
<code>com.vmware.carbonblack.cloud.osqueryi</code>	<pre> identifier "com.vmware.carbonblack.osqueryi" and anchor apple generic and certificate 1[field.1.2.840.113635.100.6.2.6] /* exists */ and certificate leaf[field.1.2.840.113635.100.6.1.13] /* exists */ and certificate leaf[subject.OU] = "7AGZNQ2S2T" </pre>
<code>com.vmware.carbonblack.cloud.uninstall</code>	<pre> identifier "com.vmware.carbonblack.cloud.uninstall" and anchor apple generic and certificate 1[field.1.2.840.113635.100.6.2.6] /* exists */ and certificate leaf[field.1.2.840.113635.100.6.1.13] /* exists */ and certificate leaf[subject.OU] = "7AGZNQ2S2T" </pre>
<code>com.vmware.carbonblack.cloud.uninstallerui</code>	<pre> identifier "com.vmware.carbonblack.cloud.uninstallerui" and anchor apple generic and certificate 1[field.1.2.840.113635.100.6.2.6] /* exists */ and certificate leaf[field.1.2.840.113635.100.6.1.13] /* exists */ and certificate leaf[subject.OU] = "7AGZNQ2S2T" </pre>

### Set the Content Filter in the Configuration Profile

To enable the configuration deployment without requiring user approval of the network extension, configure the custom settings payload.

## Procedure

- 1 Enter **custom** in the search text box of the device configuration profile.
- 2 Select **Custom Settings** and click **Configure**.
- 3 Copy and paste the following custom XML for the sensor's network extension.

This setup grants System Extensions the ability to Filter Network Content by using a Web Content Filter configuration profile.

Example of a custom settings XML.

```
<dict>
  <key>FilterDataProviderBundleIdentifier</key>
  <string>com.vmware.carbonblack.cloud.se-agent.extension</string>
  <key>FilterDataProviderDesignatedRequirement</key>
  <string>identifier "com.vmware.carbonblack.cloud.se-agent.extension" and anchor apple
generic and certificate 1[field.1.2.840.113635.100.6.2.6] /* exists */ and certificate
leaf[field.1.2.840.113635.100.6.1.13] /* exists */ and certificate leaf[subject.OU] =
"7AG2NQ2S2T"</string>
  <key>FilterPacketProviderBundleIdentifier</key>
  <string>com.vmware.carbonblack.cloud.se-agent.extension</string>
  <key>FilterPacketProviderDesignatedRequirement</key>
  <string>identifier "com.vmware.carbonblack.cloud.se-agent.extension" and anchor apple
generic and certificate 1[field.1.2.840.113635.100.6.2.6] /* exists */ and certificate
leaf[field.1.2.840.113635.100.6.1.13] /* exists */ and certificate leaf[subject.OU] =
"7AG2NQ2S2T"</string>
  <key>FilterPackets</key>
  <true/>
  <key>FilterSockets</key>
  <true/>
  <key>FilterType</key>
  <string>Plugin</string>
  <key>PayloadDisplayName</key>
  <string>Web Content Filter</string>
  <key>PayloadIdentifier</key>
  <string>com.apple.webcontent-filter.71C289AC-7ACF-44BC-AB5E-580736C634DF</string>
  <key>PayloadType</key>
  <string>com.apple.webcontent-filter</string>
  <key>PayloadUUID</key>
  <string>71C289AC-7ACF-44BC-AB5E-580736C634DF</string>
  <key>PayloadVersion</key>
  <integer>1</integer>
  <key>PluginBundleID</key>
  <string>com.vmware.carbonblack.cloud.se-agent</string>
  <key>UserDefinedName</key>
  <string>Carbon Black Network Extension Filter</string>
</dict>
```

- 4 Save the configuration profile.

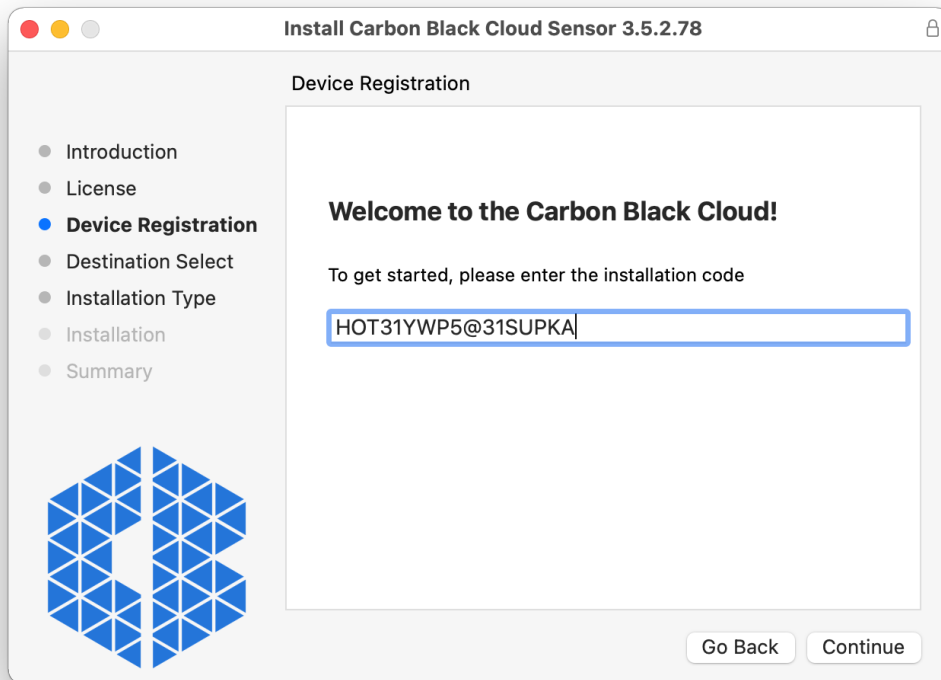
## Manually Install and Approve the Sensor on macOS 11+

After publishing the device configuration profile, manually install and approve the Carbon Black Cloud sensor on macOS Big Sur (macOS 11+).

## Procedure

- 1 Start the Carbon Black Cloud installer and click **OK** when the installer requests access to your Desktop folder.
- 2 Enter the sensor installation code.

If you enter the installation code incorrectly, an error message states that the installer cannot communicate with the Carbon Black Cloud. Check the installation code and try again.

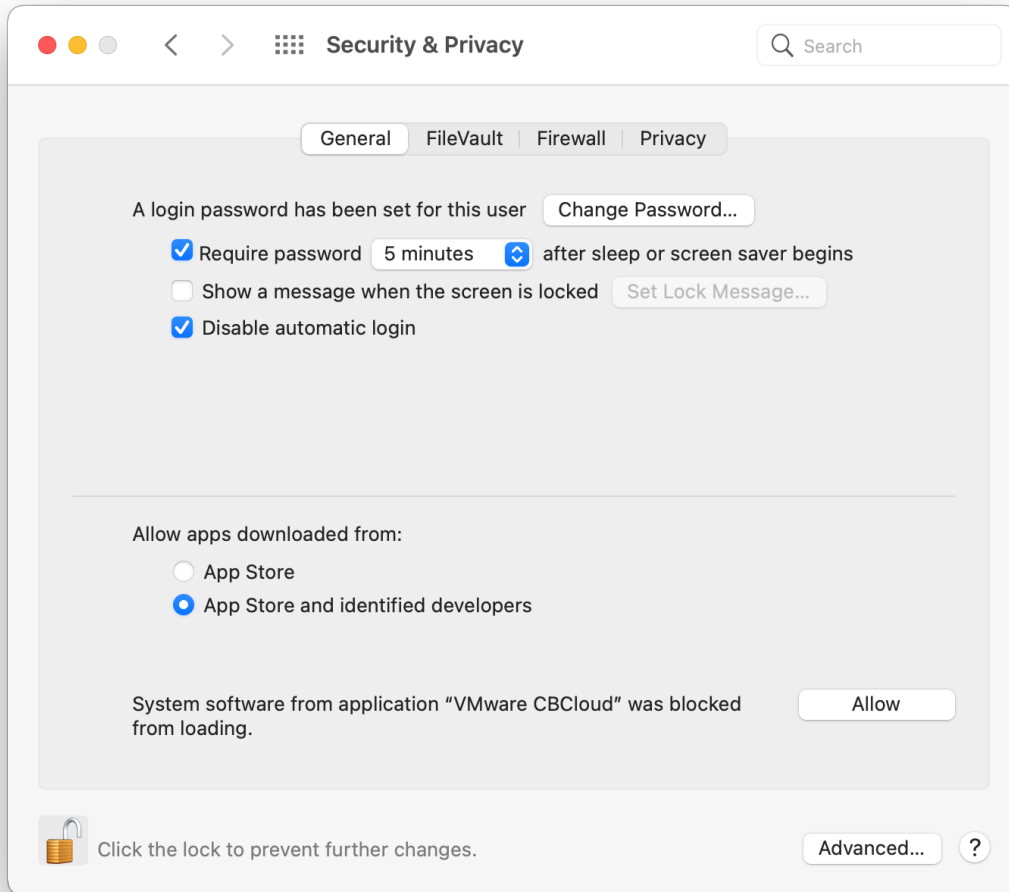


When the installer finishes running, a message notifies you that you must approve the Carbon Black Cloud system extension.

- 3 Click **Open Security Preferences** to open the **Security & Privacy** pane.

On the **General** tab in the **Security & Privacy** pane, a notification indicates that the `VMware CBCloud` system extension was blocked from loading.

- 4 Enter your administrator password to unlock the pane and then click the **Allow** button next to the notification.




---

**Note** The notification persists, regardless of device restart, and the sensor remains in bypass mode until System Extensions approval.

---

After the System Extension is approved, another notification states that the VMware Carbon Black Cloud sensor wants to filter network content.

- 5 Click **Allow**.

Enabling Network Extensions is required for the sensor to report network events to the Carbon Black Cloud console.

---

**Note** If you click **Don't Allow** instead of **Allow**, network events do not reach the backend. You can restart this prompt by running the `CBCloud.app` in the `/Applications/VMware Carbon Black Cloud` folder.

---



## Confirm the Carbon Black Cloud Sensor Installed on the macOS Device

You can use the RepCLI output to verify that the Carbon Black Cloud sensor for macOS is installed successfully on the device. For more information about RepCLI, see *Managing Sensors by using RepCLI* in the User Guide.

### Procedure

- 1 Open the `Terminal.app` on the enrolled macOS device and enter the following command:

```
cd /Applications/VMware\ Carbon\ Black\ Cloud/repcli.bundle/Contents/MacOS
```

- 2 Enter the following command:

```
sudo ./repcli status
```

**Note** If prompted, enter the administrative password.

- 3 Observe the values for `System Extension status`, `sensor State`, and `Cloud Status` lines.

A successful deployment lists the values as follows.

Option	Description
System Extension:	Running
State:	Enabled
Cloud Status:	Registered

### What to do next

- If the System Extensions are not loading, make sure that you staged the correct profile payloads.
- If the Kernel Extensions are not loading in macOS Big Sur, you must rebuild the kernel cache.

### Rebuild the Kernel Extension Cache

If you installed the Carbon Black Cloud sensor for macOS in Kernel Extension mode and the Kernel Extensions are not loading, you must rebuild the Kernel Extension cache.

### Procedure

- 1 In the Workspace ONE UEM admin console, navigate to **Devices > List View** and select the macOS device that needs the kernel cache rebuilt.
- 2 Click the **More Actions** drop-down menu and select **Custom Command**.
- 3 Paste the following command in the **Command XML** text box and add the full list of `KextPaths` into the array.

```
<dict>
  <key>RebuildKernelCache</key>
  <true/>
```

```

<key>KextPaths</key>
<array>
  <string>/Library/Extensions/CbDefenseSensor.kext</string>
  <string>/Library/Extensions/SomeOtherExtension.kext</string>
</array>
<key>RequestType</key>
<string>RestartDevice</string>
</dict>

```

**Note** When you specify the `<key>KextPaths</key>`, you must include the Carbon Black Cloud Kernel Extension path, as well as any other paths you want to include in the Kernel Cache Rebuild.

- Optional. Paste the following command in the **Command XML** text box without the `<key>KextPaths</key>` and array values, if they are unknown.

```

<dict>
  <key>RebuildKernelCache</key>
  <true/>
  <key>RequestType</key>
  <string>RestartDevice</string>
</dict>

```

**Note** If you do not specify the `<key>KextPaths</key>`, macOS attempts to rebuild the cache with any known Kernel Extensions. For example, from Apps that have launched before and attempted to load a Kernel Extension.

- Click **Send**.

## Deploying the Carbon Black Cloud sensor for macOS as Managed Application

As a Workspace ONE administrator, you can create a non-store, managed application for macOS in Workspace ONE. You must supply the icon file, the installer (`dmg` or `pkg`), and the metadata file. Additionally, the sensor kit deployment package structure requires modification to the metadata (`PLIST`) file. Then, you can deploy the Carbon Black Cloud sensor to an enrolled macOS device.

### Prerequisites

- Use the Workspace ONE Intelligent Hub for macOS to determine, based on the metadata file, if the managed application is installed and if the installed application is the correct version.
- Become familiar with the Workspace ONE Intelligent Hub app. For details, see *Overview of VMware Workspace ONE Intelligent Hub*, which is part of *VMware Workspace ONE UEM Documentation*.
- Before configuring the sensor kit deployment, you must generate the required icon and metadata file with the Workspace ONE Admin Assistant application.

- Make sure you are familiar with the VMware Workspace ONE Admin Assistant tool. For more information, see *Workspace ONE Admin Assistant → Admin Assistant for macOS*, which is part of *VMware Workspace ONE Productivity Apps Documentation*.

## Procedure

### 1 Set Up the Application Installer

To correctly distribute the Carbon Black Cloud sensor for macOS as a managed application, you must parse the sensor kit and modify the `PLIST` file.

### 2 Install the Carbon Black Cloud Sensor for macOS as a Managed Application

As a Workspace ONE administrator, after you modify the `PLIST` file you can deploy the Carbon Black Cloud sensor to an enrolled macOS device.

### 3 Confirm the Carbon Black Cloud Sensor for macOS Installed as Managed Application

There are few ways that you can validate if the Carbon Black Cloud sensor for macOS has been successfully installed as managed application on your macOS device.

## Set Up the Application Installer

To correctly distribute the Carbon Black Cloud sensor for macOS as a managed application, you must parse the sensor kit and modify the `PLIST` file.

## Procedure

- 1 Open the Workspace One Admin Assistant app and upload the downloaded sensor kit from the Carbon Black Cloud console (`confer_installer_mac-<version>.dmg`).
- 2 When parsing completes, click the **Reveal in Finder** button.
- 3 Expand the `CBCloud Install-<version>` folder and right-click the `CBCloud Install-<version>.plist` file.

- 4 Click **Open With** and select an editor of your choice.

For example, TextEdit, Xcode, or vim.

- 5 Modify the `PLIST` file by adding the following XML snippet:

```
<key>installs</key>
  <array>
    <dict>
      <key>CFBundleIdentifier</key>
      <string>com.vmware.carbonblack.cloud.se-agent</string>
      <key>CFBundleName</key>
      <string>VMware CBCloud</string>
      <key>CFBundleShortVersionString</key>
      <string>3.5.1fc19</string>
      <key>CFBundleVersion</key>
      <string>3.5.1fc19</string>
      <key>minosversion</key>
      <string>10.15</string>
      <key>path</key>
```

```

    <string>/Applications/VMware Carbon Black Cloud/VMware CBCloud.app</string>
    <key>type</key>
    <string>application</string>
    <key>version_comparison_key</key>
    <string>CFBundleShortVersionString</string>
  </dict>
</array>

```

**Note** The version displays with "fc" included. Examples:

Sensor Version	PLIST File Sensor Version Value
3.5.2.76	3.5.2fc76
3.6.2.110	3.6.2fc110
3.7.2.77	3.7.2fc77
3.7.3.30	3.7.3fc30

- 6 Replace the `CFBundleShortVersionString` and `CFBundleVersion` values if they are different for the particular sensor version that you are deploying.
- 7 Optional. Generate the `installs` array in either way:
  - Export the `VMware CBCloud.app` from the installer package and run `VMware CBCloud.app` through the Workspace ONE Admin Assistant app.
  - If you installed the Carbon Black sensor kit on the machine with Workspace ONE Admin Assistant, copy `VMware CBCloud.app` to your `~/Downloads` directory (`cp -R /Applications/VMware\ Carbon\ Black\ Cloud\VMware CBCloud.app ~/Downloads`) and parse `~/Downloads/VMware CBCloud.app` through the Workspace ONE Admin Assistant app.

The PLIST generated in this instance contains the appropriate `installs` array information.

- 8 Save and close the modified PLIST in the editor of your choice.

## Install the Carbon Black Cloud Sensor for macOS as a Managed Application

As a Workspace ONE administrator, after you modify the PLIST file you can deploy the Carbon Black Cloud sensor to an enrolled macOS device.

### Add an Application

Use the Workspace ONE UEM admin console to deploy an internal macOS application for the sensor.

### Prerequisites

To have the Workspace ONE UEM natively process macOS metadata, install the Workspace ONE UEM Admin Assistant for macOS tool. For description of the Admin Assistant tool and

how to install it, see *Introduction to Workspace ONE Admin Assistant for macOS*, part of the *VMwareWorkspace ONE UEM* documentation.

#### Procedure

- 1 Select **Resources** from the **Getting Started** wizard.
- 2 Unfold **Apps** and select **Native > Internal**.
- 3 Select **Application File** from the **Add** drop-down menu.
- 4 Click **Upload**, choose the local file `confer_installer_mac-<version>.dmg` that you generated with the VMware Workspace ONE UEM Admin Assistant Tool, and save it.
- 5 Upload the metadata file by choosing the `CbDefense Install-<version>.plist` file that you generated with the Workspace ONE Admin Assistant, and save it.
- 6 Add an image for the app install by dragging the `CBCloud Install.png` graphic to the Workspace ONE UEM console.

#### Define Pre-Install and Uninstall Scripts

You can provide a pre-install and post-install scripts to populate a configuration file that is consumed by the Carbon Black Cloud sensor kit installation.

You must configure the **Scripts** settings to run the installation and uninstallation of the Carbon Black Cloud sensor macOS application. By providing pre-install scripts and post-install scripts, you can perform additional configuration tasks or install additional items without the need of repacking the application or software. Simply paste the script and Workspace ONE UEM formats it to be used by Munki.

#### Procedure

- 1 Select the **Scripts** tab.

- 2 Paste one of the following scripts into the **Pre-Install Script** text box and replace the Code value with your Registration Code.

Each of the scripts includes the bare minimum required information — the Registration Code, for installing the Carbon Black Cloud sensor for macOS. The advanced pre-install script contains additional options for customizing the sensor installation. Replace them with your own values.

Option	Description
<b>Basic Pre-Install Script For System Extension Install</b>	<pre data-bbox="646 514 1101 787">#!/bin/bash PATH="/var/cbcloud-install" /bin/mkdir -p "\$PATH" /usr/bin/touch "\$PATH/cfg.ini" /bin/cat &gt; "\$PATH/cfg.ini" &lt;&lt;- EOM [customer] Code=&lt;REGISTRATION_CODE&gt; DisableSysextnetworkExtension=false KernelType=2 EOM</pre> <p data-bbox="630 819 766 840">For example:</p> <pre data-bbox="646 861 1101 1155">#!/bin/bash PATH="/tmp/cbcloud-install" /bin/mkdir -p "\$PATH" /usr/bin/touch "\$PATH/cfg.ini" /bin/chmod 644 "\$PATH/cfg.ini" /bin/cat &gt; "\$PATH/cfg.ini" &lt;&lt;- EOM [customer] Code=12345 DisableSysextnetworkExtension=false KernelType=2 EOM</pre>
<b>Basic Pre-Install Script For Kernel Extension Install</b>	<pre data-bbox="646 1207 1101 1480">#!/bin/bash PATH="/var/cbcloud-install" /bin/mkdir -p "\$PATH" /usr/bin/touch "\$PATH/cfg.ini" /bin/cat &gt; "\$PATH/cfg.ini" &lt;&lt;- EOM [customer] Code=&lt;REGISTRATION_CODE&gt; DisableSysextnetworkExtension=false KernelType=1 EOM</pre>
<b>Advanced Pre-Install Script</b>	<pre data-bbox="646 1522 1292 1900">PATH="/var/cbcloud-install" /bin/mkdir -p "\$PATH" /usr/bin/touch "\$PATH/cfg.ini" /bin/cat &gt; "\$PATH/cfg.ini" &lt;&lt;- EOM [customer] Code=&lt;REGISTRATION_CODE&gt; ProxyServer=&lt;PROXY_SERVER&gt; ProxyServerCredentials=&lt;PROXY_CREDS&gt; LastAttemptProxyServer=&lt;LAST_ATTEMPT_PROXY_SERVER&gt; PemFile=&lt;customer.pem&gt; AutoUpdate=&lt;true false&gt; AutoUpdateJitter=&lt;true false&gt; InstallBypass=&lt;true false&gt; FileUploadLimit=&lt;FILE_UPLOAD_LIMIT&gt;</pre>

Option	Description
	<pre> GroupName=&lt;GROUP_NAME&gt; EmailAddress=&lt;USER_NAME&gt; BackgroundScan=&lt;true false&gt; RateLimit=&lt;RATE_LIMIT&gt; ConnectionLimit=&lt;CONNECTION_LIMIT&gt; QueueSize=&lt;QUEUE_SIZE&gt; LearningMode=&lt;LEARNING_MODE&gt; &lt;POC=1&gt; CbLRKill=&lt;true false&gt; HideCommandLines=&lt;true false&gt; DisableSysextnetworkExtension=&lt;true false&gt; KernelType=&lt;1 2&gt; #1=KEXT,2=SysExt EOM </pre>

- 3 Select **Uninstall Script** as the uninstall method.
- 4 Paste the script and populate the Deregistration Code into the **Uninstall Script** text box.

```

#!/bin/sh
/Applications/VMware\ Carbon\ Black\ Cloud/uninstall.bundle/Contents/MacOS/uninstall -y -c
<Deregistration_Code>

```

### Set Deployment Options

By setting the deployment options you define the applications or processes that can prevent the installation from completing successfully.

#### Procedure

- 1 Click the **Deployment** tab.
- 2 Select **No** for the **Blocking Applications** option.  
The end user must not close any Carbon Black Cloud applications. This is all handled by the Workspace ONE Intelligent Hub and the Carbon Black Cloud sensor installer.
- 3 If deploying the sensor with System Extensions, select **None** from the **Restart Action** drop-down menu. If deploying the sensor using KEXTs, choose the appropriate restart action.
- 4 Click **Save and Assign**.

### Configure the Assignment

Before assigning the configurations to the sensor application installer, you must set your distribution and restriction preferences.

#### Procedure

- 1 In the **Distribution** page, enter a name for the distribution.  
For example, *All Macs*.
- 2 Select the **Assignment Groups** containing the devices that must receive the Carbon Black Cloud sensor.

- 3 Select **Auto** for the **App Delivery Method** option.
- 4 Determine if you want the user to see the Carbon Black application in their App Catalog. It can remain inactive.
- 5 Click **Restrictions** and enable **Remove on Unenroll** and **Desired State Management**.
- 6 **Create** the assignment.
- 7 Optional. Locate the **Exclusions** tab, add exclusions to the assignments, and adjust the priority for the assignments.
- 8 **Save** the assignment.
- 9 Review the assigned device and click **Publish**.

## Confirm the Carbon Black Cloud Sensor for macOS Installed as Managed Application

There are few ways that you can validate if the Carbon Black Cloud sensor for macOS has been successfully installed as managed application on your macOS device.

### Verify Carbon Black Cloud Sensor for macOS Installed as Managed Application with Workspace ONE UEM

You can use the Workspace ONE UEM admin console to verify that the Carbon Black Cloud sensor for macOS has installed as a managed application on the assigned devices.

#### Procedure

- 1 Go to **Devices > List View**
- 2 Select a device and click the **Apps** tab.
- 3 Locate the Carbon Black Cloud sensor for macOS in the list of applications.

#### Results

The Carbon Black Cloud sensor is installed as a managed application on the devices you previously assigned.

### Verify Carbon Black Cloud Sensor for macOS Installed as Manage Application on the Device

You can check if the Carbon Black Cloud sensor for macOS application has been installed successfully on your macOS device.

#### Procedure

- 1 Open **Finder > Application** on the enrolled macOS device.
- 2 Ensure that the `VMware Carbon Black Cloud` folder is present and contains the **VMware CBCloud** sensor application and its related bundles.
- 3 Optionally check if the Confer menulet exists in the menu bar of the device.



## Verify Carbon Black Cloud Sensor for macOS Installed as Managed Application in the Installation Log

You can check if the Carbon Black Cloud sensor for macOS application has been installed successfully on your macOS device by viewing the installation logs.

### Procedure

- 1 Open the `Terminal.App` on the enrolled macOS device and enter the following command:

```
tail -f -n +1 /Library/Application\ Support/AirWatch/Data/Munki/Managed\ Installs/Logs/ManagedSoftwareUpdate.log
```

- 2 Browse the log file for the line stating that the `Install of CbDefense Install.pkg` was successful.

If the line states that the sensor appears to not be installing, or is installing repeatedly, you must adjust the metadata of the `PLIST` file to include an `installs` array.

# Installing Sensors on Endpoints in a VDI Environment

# 4

This section describes how to install sensors through the command line or software distribution tools in a Virtual Desktop Infrastructure (VDI) environment.

---

**Important** Before you begin the processes described here, read [Chapter 1 Getting Started with Sensor Installation](#). It contains highly relevant information to help you succeed in your sensor installation.

---

Before you install sensors, perform the following steps:

[Obtain a Company Registration Code](#)

[Download Sensor Kits](#)

For firewall and proxy information, see [Chapter 13 Configuring Carbon Black Cloud Communications](#).

Read the following topics next:

- [Creating Multiple Golden or Primary Images](#)
- [Carbon Black Windows Sensors with Horizon Virtual Desktops](#)
- [Carbon Black Linux Sensors with VMware Horizon Virtual Desktops](#)
- [Carbon Black Windows Sensors with Citrix Virtual Desktops](#)
- [Carbon Black Windows Sensors with vSphere Clients](#)
- [Carbon Black Linux Sensors with vSphere Clients](#)

## Creating Multiple Golden or Primary Images

This topic describes caveats and steps to follow when creating (cloning) multiple golden or primary images. It is pertinent to all VDIs.

To create multiple primary images, you must uninstall the sensor from the cloned primary image. Then, install a new sensor. The second primary image will receive a base device ID on its own, thus severing the dependency of the original primary image.

If you do not follow this recommendation and if the cloned primary image is deregistered because of inactivity and policy settings, sensors on the VDIs will be uninstalled and the VDIs will be unprotected.

## Create Multiple Golden or Primary Images

This section provides general steps to follow when creating (cloning) multiple golden or primary images.

### Procedure

- 1 Clone the golden or primary image.
- 2 Start the cloned image device.
- 3 Log in to the Carbon Black Cloud console.
- 4 Click **Inventory > Endpoints** or click **Inventory > VM Workload**.
- 5 Verify that the cloned image has a different DeviceID from the original image.

---

**Note** If the cloned image does not have a different DeviceID, run the `RepCLI reregister now` command on the cloned image and then verify that it has a new DeviceID.

For more information about RepCLI, see *Managing Sensors by using RepCLI* in the User Guide.

- 6 Uninstall the sensor from the cloned image.

---

**Note** Do not uninstall the sensor if the cloned image device does not have a different DeviceID.

- 7 Install the sensor on the cloned image device.

## Carbon Black Windows Sensors with Horizon Virtual Desktops

This section describes how to deploy Carbon Black Cloud Windows 3.6+ sensors on Horizon full clones and Horizon instant clones 7.13, Horizon 2012 or later.

Before you install the Carbon Black Cloud Windows sensor, make sure that you are following standard Horizon best practices, including optimizing the golden image. The versions of the Carbon Black Cloud sensor and Horizon that you are using must be compatible and supported. See the [VMware Interoperability Matrix](#).

Review the following requirements and implement the recommended best practices. If the best practices included here are not the preferred method for deployment, an alternative configuration that uses the `OFFLINE_INSTALL` switch is supported. This is useful for organizations who want to create a golden image and clone it to offline computers. See [Windows Sensor Supported Commands](#).

See [Chapter 4 Installing Sensors on Endpoints in a VDI Environment](#) for preliminary instructions.

## Carbon Black Windows Sensor Policy Setting Recommendations for Horizon Instant Clones

We recommend that the golden image be assigned a different policy from its clones. Use Asset Groups or Sensor Groups to avoid the clones inheriting the golden image.

---

**Important** Asset Groups is available to Carbon Black Cloud customers on 27 November 2023. Carbon Black recommends that you upgrade from Sensor Groups to Asset Groups as soon as it is operationally feasible for your organization. Sensor Groups will be phased out by 01 December 2024. See [Asset Groups and Sensor Groups](#) in the *User Guide*.

---

To get started, we recommend that you duplicate the Standard policy rules to the instant clone policy. We then recommend the following specific policy settings for Horizon instant clones.

### General Tab

- **Name** – For easy identification, we recommend giving the policy a name that distinguishes the sensors as Instant Clones.
- **Description** – This policy is optimized for Horizon instant clones. Special considerations improve performance and provide a strong base of reputation, behavioral, and targeted prevention.
- **Target Value** – Medium

### Sensor Tab

- **Display sensor message in system tray** - Enable this setting and add a message similar to this sample text: "Virtual Desktops Policy - Contact *someone@example.com* with any questions and concerns. Provide context regarding the issue and any available replication steps."

### Prevention Tab - Permissions

- **Bypass rules (exclusions)** – Policy-level bypass rules help achieve stability in a VDI environment.

Each organization must understand the trade-offs between performance and security. Carbon Black recommends the use of exclusions. Work with stakeholders to review risks and benefits (performance versus visibility) and apply the bypass rules as needed.

Carbon Black Cloud provides exclusions for supported methods as examples. Please review the applications that are installed in the VDI environment and apply any required bypass rules.

The following examples are based on public documentation for Carbon Black solutions. Additional bypass rules might be needed.

### Bypass rules best practices

```
**\Program Files\VMware\**
**\SnapVolumesTemp**
```

```

**\SVROOT**
**\SoftwareDistribution\DataStore**
**\System32\Spool\Printers**
**\ProgramData\VMware\VDM\Logs**
**\AppData\VMware\**

```

## Blocking and Isolation

Best practices recommend applying **Blocking and Isolation** rules to address specific attack surfaces.

### Local Scan tab

- **On Access File Scan Mode** – Enabled
- **Allow Signature Updates** – Disabled

This setting is circumstantial. For short-lived clones, it is recommended to have **Allow Signature Updates** set to Disabled and have **On Access File Scan Mode** set to Enabled. The policy of the golden image would have both of these settings Enabled. This setup makes sure that clones can still perform AV scans using the signature packs that came from the golden image, without incurring the cost of updating the signature pack on each clone. If the clones are expected to be long-lived it is recommended to have both settings set to Enabled (to avoid the use of outdated signature packs).

### Sensor tab

- **Run Background Scan** – Disabled. To optimize performance, it is recommended to complete a background scan on the golden image and subsequently have the background scan disabled on the policy that is assigned to the clones.
- **Scan files on network drives** – Disabled
- **Scan execute on network drives** – Enabled
- **Delay execute for Cloud scan** – Enabled. This critical setting serves as the sole point of reference for pre-execution reputation lookups. If it is disabled, endpoints must rely on **Application at Path** and **Deny List** rules for pre-execution prevention.
- **Hash MD5** – Disabled. The sensor always calculates the SHA-256.
- **Auto-deregister VDI clone sensors that have been inactive for** – Because instant clones are generally short-lived, it is recommended to Enable this setting to remove any instant clones that have been inactive for the specified duration.

---

**Note** Carbon Black recommends setting an interval of at least 24 hours to ensure that sensors do not get de-registered during common maintenance windows from Carbon Black or your environment.

---

## Install the Carbon Black Windows Sensor on Horizon Instant Clones

To install a Carbon Black Cloud Windows sensor on Horizon instant clones, perform the following procedure. These instructions apply to both instant clone pools only.

**Important** Previous installation use of a post-synchronization script (batch file) is no longer necessary. If you are upgrading to Horizon 7.13+ from a previous Horizon version, you must remove the batch file that had previously been inserted into the golden image. Failure to remove the script will cause multiple re-registrations of the same device.

Do not run `repcli reregister now` or `repcli reregister onrestart` commands on the golden image. Either command turns the golden image into a clone, which might deregister the golden image if `autoderegister` is set and a time-out has occurred. Deregistration of the golden image results in clones being unable to reregister.

The instant clone agent now sets the following registry value to a unique GUID when `IT/replica/clone nga` customization begins. Each clone has a unique value:

```
Key: HKLM\Software\VMware, Inc.\ViewComposer\ga\AgentIntegration
Type: REG_SZ
Value: CustomizationStarted
```

### Prerequisites

See [Carbon Black Windows Sensor Policy Setting Recommendations for Horizon Instant Clones](#) before installing the sensor.

### Procedure

- 1 Create the golden image VM for the clone pool deployment. Perform required Windows updates and install the required [VMware Tools](#) and Horizon Agent.
- 2 Install the sensor on the golden image:
  - If you are using Horizon versions 7.13+ or 8.0+ and Carbon Black Cloud sensor 3.6+, no additional configuration is required. In this case, the sensor uses a Horizon Agent-provided registration key to perform reregistration on the clone:

```
msiexec.exe /q /i <Sensor Installer Path> /L*v msi.log COMPANY_CODE="XYZABC"
CLI_USERS=<UserGroupSid> POLICY_NAME="<NAME Virtual Policy>"
```

- If you are using a Horizon version Pre-7.13, 8.0 and Windows sensor 3.7 MR2+, add the "AUTO\_REREGISTER\_FOR\_VDI\_CLONES=3" install flag:

```
msiexec.exe /q /i <Sensor Installer Path> /L*v msi.log COMPANY_CODE="XYZABC"
CLI_USERS=<UserGroupSid> AUTO_REREGISTER_FOR_VDI_CLONES=3 POLICY_NAME="<NAME Virtual
Policy>"
```

**Note** <Sensor Installer Path>: Replace this value with the location of the sensor MSI file; for example, c:\tmp\installer\_win-64-3.8.0.627.msi.

CLI\_USERS= <UserGroupSid>: This parameter on the golden image enables RepCLI usage on the clones. The value is the Security Identifier (SID) of the user account/group that will run RepCLI commands on the clones.

POLICY\_NAME: Indicates the policy name that has the necessary exclusions and configurations to apply to the golden image. For Carbon Black Cloud sensors that are on versions prior to 3.8, use the GROUP\_NAME parameter instead.

See [Installing Windows Sensors on Endpoints](#) and [Windows Sensor Supported Commands](#). For more information about RepCLI, see *Managing Sensors by using RepCLI* in the User Guide.

- 3 Optional (Recommended). Complete a background scan on the golden image to optimize clone performance.
  - a In the Carbon Black Cloud console, click **Enforce > Policies**, select the policy, and click the **Sensor** tab.
  - b Select the **Run background scan** option and select **Expedited** scanning.
  - c Click **Save**.
  - d You can track scan progress by running the `repcli status` command. The output will be similar to the following:

```
General Info:
  Sensor Version[3.7.0.1473 - Sep 29 2021 - 20:34:38]
  Local Scanner Version[ - ]
  Disk Filter Version[3.7.0.1473]
  CbShared[104365] Policy[1269] FileAnalysis[386] Proto[548]
  Sensor State[Enabled]
  Details[LiveResponse:NoSession, LiveResponse:NoKillSwitch, LiveResponse:Disabled,
  SvcStable]
  DeviceHash[31dbad895ab7161f1f53bed2f4e3fa49ac64de98935b03752b53a407f65d9ea2]
  DeviceID[26365289]
  VirtualGuestToHostCommsStatus[Disconnected]
  ExternalIdentity[Not Available]
  Kernel File Filter[Connected]
  LastUser[Device\user]
  Background Scan [Complete]
  Total Files Processed[52581] Current Directory[None]
```

#### 4 Optional. Configure cache persistence for improved performance.

The persistent cache setting (`FileCachePersistenceState=3`) saves significant CPU and disk IO resources by reusing calculated hashes on clones. This feature is available with Windows sensor 3.8+. In addition, pruning parameters (available with Windows sensor 3.7+) improve VDI performance.

Persistent cache depends on the secure storage of the golden image snapshot files and assumes that no modifications are made to the snapshot while the golden image is offline. When enabling this setting, secure the golden image and storage infrastructure to an equivalent level, or to a higher level than the guest OS. We recommend that you limit physical and administrative access to the golden image and storage infrastructure, and regularly check your audit logs.

- a In the left navigation bar in the console, click **Inventory>Endpoints** or **Inventory>Workloads**.
- b Select the endpoint, click **Take Action**, and click **Enable bypass**. Confirm the action.
- c To confirm that the endpoint is in bypass mode, run the following command: `repcli status`.
- d As a best practice, make a backup of the `cfg.ini` file into another directory. For Windows sensor versions 3.6 and earlier, `cfg.ini` is located at `C:\Program Files\Confer\cfg.ini`. For Windows sensors 3.7+, `cfg.ini` is located at `C:\ProgramData\CarbonBlack\DataFiles\cfg.ini`.
- e Edit `cfg.ini`. Add the following parameters:
  - `RepDbPruneCountdownMs=14400000` (Default is 5 minutes; modified to 4 hours). This setting defines the interval after which the first pruning attempt is initiated.
  - `PruneDeletedFilesSleepInterval=14400000` (Default is 30 minutes; modified to hours). This setting defines the delay for subsequent attempts after a successful pruning attempt.
  - `PruneDeletedFilenameRowCount=500` (Default is 100). This setting defines the maximum number of rows to prune in one attempt.
  - `FileCachePersistenceState=3` (Default is 0). This setting enables cache persistence on instant clones.
- f Reboot the golden image.
- g Log into the golden image and run common applications to populate the cache.
- h In the left navigation bar in the console, click **Inventory>Endpoints** or **Inventory>Workloads**.
- i Select the endpoint, click **Take Action**, and click **Disable bypass**. Confirm the action.
- j Delete the backup file you created in Step 4d.
- k Shut down the golden image.



- 5 Reboot the golden image to apply full ransomware protections (3.7+ sensors). You can skip this step if you completed Step 4.
- 6 Take a snapshot of the golden image.
- 7 In the Horizon console, create an instant clone pool using the golden image and snapshot.
- 8 After the pool becomes available in the Horizon console, check in the Carbon Black Cloud console to verify that the newly created instant clones are registered with a new Device ID and are assigned the correct policy. The endpoint inherits the policy from the golden image unless you have previously created Asset groups or Sensor groups and the installed sensor matches a group's criteria. Manual policy assignment post-installation overrides the inheritance.

## Horizon Instant Clones and Carbon Black Windows Sensor Installation Known Issues and Mitigation

During the instant clone pool creation, a temporary full clone of the golden image known as the “internal template” (with a device name `itxxxxxx`) powers on and has network access. The sensor on that internal template device will probably connect to the Carbon Black Cloud with the same Device ID as the golden image. This connection results in the golden image being overwritten by the `itxxxxxx` device in the Carbon Black Cloud console.

---

**Note** This issue is resolved with the Carbon Black Cloud Windows sensor 3.7MR2.

---

When the golden image is powered on, the sensor on the golden image re-connects to the backend and overwrites the `itxxxxxx` device. In addition to the duplicate devices overwriting each other's data on the backend, this can lead to the backend sending a re-register request to the golden image. This causes the golden image to be considered a VDI by the backend, which could cause the golden image to deregister due to inactivity.

The duplicate device scenario can also expose a group membership issue where the golden image is no longer in the expected policy group. The negative implications of the internal template having the duplicate device ID as the golden image are as follows:

- The internal template's events and activities can intermingle with the golden images.
- The golden image's device name in the console might change.
- If you are using MSM to assign device policy by device name, make sure that golden image and internal template names are accounted for.

We recommend that you deploy instant clones with the golden image powered off. This recommendation will not eliminate the internal template duplicate Device ID scenario, but it will mitigate the downside of having a duplicate Device ID.

Provisioning of instant clones with the setting **Provision all machines up front** can cause a measurable increase in CPU utilization on the hosts and the provisioning clones during the initial provisioning operation. This is a result of the Carbon Black sensor registration occurring on all the provisioning instant clone at the same time.

Provisioning of instant clones with the setting **Provision machines on demand** is unlikely to cause an increase in CPU utilization because the instant clones are provisioned at different times.

As a result of higher CPU load, during initial instant clone provisioning the customization might time-out for the clones while executing the post-synchronization batch file. This can cause an error state for the clones. The error state is due to the 20 seconds timeout limit for executing the post synchronization scripts on the clones, but the failed clones will auto-recover. The 20 seconds default timeout of the Post Synchronization script can be adjusted by modifying the following registry key in the golden image. This reduces the instant clones provisioning failure rate. The maximum suggested value is 120000ms.

```
HKLM\System\CCS\Services\vmware-viewcomposer-ga
Type: DWORD
Value Name: ExecScriptTimeout
Units: milliseconds
Sample:
##Updated timeout value from 20000 to 120000 .
HKLM\System\CCS\Services\vmware-viewcomposer-ga
Type: DWORD
Value Name: ExecScriptTimeout
Units: 120000
```

## Carbon Black Windows Sensor Policy Setting Recommendations for Horizon Full Clones

We recommend that the golden image be in a separate policy from its clones. Use sensor groups to avoid the clones inheriting the golden image policy.

---

**Important** Asset Groups is available to Carbon Black Cloud customers on 27 November 2023. Carbon Black recommends that you upgrade from Sensor Groups to Asset Groups as soon as it is operationally feasible for your organization. Sensor Groups will be phased out by 01 December 2024. See [Asset Groups and Sensor Groups](#) in the *User Guide*.

---

To get started, we recommend that you duplicate the Standard policy rules to the full clone policy. We then recommend the following specific policy settings for Horizon full clones.

### General Tab

- **Name** – For easy identification we recommend giving the policy a name that distinguishes the sensors as Full Clones.
- **Description** – This policy is optimized for Horizon full clones. Special considerations improve performance and provide a strong base of reputation, behavioral, and targeted prevention.
- **Target Value** – Medium

## Sensor Tab

- **Display sensor message in system tray** - Enable this setting and add a message similar to this sample text: "Virtual Desktops Policy - Contact *someone@example.com* with any questions and concerns. Provide context regarding the issue and any available replication steps."

## Prevention Tab - Permissions

- **Bypass rules (exclusions)** – Policy-level bypass rules help achieve stability in a VDI environment.

Each organization must understand the trade-offs between performance and security. Carbon Black recommends the use of exclusions. Work with stakeholders to review risks and benefits (performance versus visibility) and apply the bypass rules as needed.

Carbon Black Cloud provides exclusions for supported methods as examples. Please review the applications that are installed in the VDI environment and apply any required bypass rules.

The following examples are based on public documentation for Carbon Black solutions. Additional bypass rules might be needed.

## Bypass rules best practices

```
**\Program Files\VMware**,
**\Program Files*\cloudvolumes\agent\svhook64.dll,
**\SnapVolumesTemp**,
**\SVROOT**,
**\SoftwareDistribution\DataStore**,
**\System32\Spool\Printers**,
**\ProgramData\VMware\VDM\Logs**,
**\AppData\VMware**
```

## Blocking and Isolation

Best practices recommend applying **Blocking and Isolation** rules to address specific attack surfaces.

## Local Scan tab

- **On Access File Scan Mode** – Enabled
- **Allow Signature Updates** – Enabled

Full clones are rarely recreated from the golden image, so they effectively never receive signature updates. Enable **Allow Signature Updates** for full clones.

## Sensor tab

- **Run Background Scan** – Disabled. To optimize performance, it is recommended to complete a background scan on the golden image and then subsequently have the background scan disabled on the policy assigned to the clones.

- **Scan files on network drives** – Disabled
- **Scan execute on network drives** – Enabled
- **Delay execute for Cloud scan** – Enabled. This critical setting serves as the sole point of reference for pre-execution reputation lookups. If it is disabled, endpoints must rely on **Application at Path** and **Deny List** rules for pre-execution prevention.
- **Hash MD5** – Disabled. The sensor always calculates the SHA-256.
- **Auto-deregister VDI sensors that have been inactive for** – Disable this setting to prevent unintentional uninstall of the sensor.

## Behavior of a Quarantined Instant Clone VM

Carbon Black Cloud Quarantine prevents suspicious activity and malware from affecting the rest of your network by isolating the affected asset from the network. As a result of this network isolation, the Horizon agent communication from an instant clone VM to the Horizon connection server is cut off. To consider the full implications of this scenario, please see the following KB: [Best Practices when Using Endpoint Detection and Response \(EDR\) Tools with Horizon \(95512\)](#).

## Install Carbon Black Windows Sensors on Horizon Full Clones

If you are installing Carbon Black Cloud Windows sensors into an environment that is comprised of Horizon full clones only, use the following installation method.

---

**Important** Previous installation use of a post-synchronization script (batch file) is no longer necessary. If you are upgrading to Horizon 7.13+ from a previous Horizon version, you must remove the batch file that had previously been inserted into the golden image. Failure to remove the script will cause multiple re-registrations of the same device.

Do not run `repcli reregister now` or `repcli reregister onrestart` commands on the golden image. Either command turns the golden image into a clone, which might deregister the golden image if autoderegister is set and a time-out has occurred. Deregistration of the golden image results in clones being unable to reregister.

---

### Prerequisites

See [Carbon Black Windows Sensor Policy Setting Recommendations for Horizon Full Clones](#) before installing the sensor.

### Procedure

- 1 Create the golden image VM for the clone pool deployment. Perform required Windows updates and install the required [VMware Tools](#) and Horizon Agent.

## 2 Install the sensor on the golden image:

- If you are using Horizon versions 7.13+ or 8.0+ and Carbon Black Cloud sensor 3.6+, no additional configuration is required. In this case, the sensor uses a Horizon Agent-provided registration key to perform reregistration on the clone:

```
msiexec.exe /q /i <Sensor Installer Path> /L*v msi.log COMPANY_CODE="XYZABC"
CLI_USERS=<UserGroupSid>POLICY_NAME="<NAME Virtual Policy>"
```

**Note** The instant clone agent now sets the following registry value to a unique GUID when IT/replica/clone nga customization begins. Each clone has a unique value:

```
Key: HKLM\Software\VMware, Inc.\ViewComposer\ga\AgentIntegration
Type: REG_SZ
Value: CustomizationStarted
```

- If you are using a Horizon version Pre-7.13, 8.0 and Windows sensor 3.7 MR2+, add the `"AUTO_REREGISTER_FOR_VDI_CLONES=3"` install flag:

```
msiexec.exe /q /i <Sensor Installer Path> /L*v msi.log COMPANY_CODE="XYZABC"
CLI_USERS=<UserGroupSid> AUTO_REREGISTER_FOR_VDI_CLONES=3 POLICY_NAME="<NAME Virtual
Policy>"
```

**Note** *<Sensor Installer Path>*: Replace this value with the location of the sensor MSI file; for example, `c:\tmp\installer_win-64-3.8.0.627.msi`.

*CLI\_USERS= <UserGroupSid>*: This parameter on the golden image enables RepCLI usage on the clones. The value is the Security Identifier (SID) of the user account/group that will run RepCLI commands on the clones.

*Policy\_NAME*: Indicates the policy name that has the necessary exclusions and configurations to apply to the golden image. For Carbon Black Cloud sensors that are on versions prior to 3.8, use `GROUP_NAME` parameter instead.

See [Installing Windows Sensors on Endpoints](#) and [Windows Sensor Supported Commands](#). For more information about RepCLI, see *Managing Sensors by using RepCLI* in the User Guide.

- 3 Optional (Recommended). Complete a background scan on the golden image to optimize clone performance.
  - a In the Carbon Black Cloud console, click **Enforce > Policies**, select the policy, and click the **Sensor** tab.
  - b Select the **Run background scan** option and select **Expedited** scanning.
  - c Click **Save**.

- d You can track scan progress by running the `repcli status` command. The output will be similar to the following:

```
General Info:
  Sensor Version[3.7.0.1473 - Sep 29 2021 - 20:34:38]
  Local Scanner Version[ - ]
  Disk Filter Version[3.7.0.1473]
  CbShared[104365] Policy[1269] FileAnalysis[386] Proto[548]
  Sensor State[Enabled]
  Details[LiveResponse:NoSession, LiveResponse:NoKillSwitch, LiveResponse:Disabled,
  SvcStable]
  DeviceHash[31dbad895ab7161f1f53bed2f4e3fa49ac64de98935b03752b53a407f65d9ea2]
  DeviceID[26365289]
  VirtualGuestToHostCommsStatus[Disconnected]
  ExternalIdentity[Not Available]
  Kernel File Filter[Connected]
  LastUser[Device\user]
  Background Scan [Complete]
  Total Files Processed[52581] Current Directory[None]
```

- 4 Optional: Update the device signature of the golden image by running the `repcli updateAVSignature` command.
- 5 Reboot the golden image to apply full ransomware protections (Windows sensor versions 3.7+).
- 6 Shut down the golden image. In the Horizon console, convert the golden image into the template VM. Create a full clone pool using the golden VMTemplate.
- 7 New full clones will register with a new Device ID in the Carbon Black Cloud console after the pool becomes available. Confirm that newly provisioned clones have registered and are assigned the correct policy.

## Install Carbon Black Windows Sensors in Horizon Full and Instant Clone Mixed Environments

Use the following procedure to install Carbon Black Cloud Windows sensors in an environment that has both full and instant Horizon clones.

### Procedure

- 1 Create the golden image for the full clone pool. As per the Horizon documentation, perform the required steps, including installing [VMware Tools](#) and Horizon Agent. Do not install the Carbon Black Cloud sensor on the golden image.
- 2 Put the Carbon Black Cloud sensor MSI on the golden image (preferably in the System Root directory).
- 3 Prepare the Customization Specification that will be used to create the full clone pool.

- 4 Add the following sensor installation command into Customization Specification commands:

```
msiexec.exe /q /i <Sensor Installer Path > /L*v msi.log COMPANY_CODE="XYZABC"
CLI_USERS=<UserGroupSid> GROUP_NAME="<NAME Virtual Policy>"
```

**Note** For Horizon Pre-7.13, 8.0 and Windows sensors 3.7MR2+, add the following parameter to enable automatic reregistration of clones: `AUTO_REREGISTER_FOR_VDI_CLONES=3`. If you are using an older sensor version, use the `BASE_IMAGE=1` parameter instead.

*< Sensor Installer Path >* : Replace this value with the location of the sensor MSI file; for example, `c:\tmp\installer_win-64-3.6.0.1941.msi`.

`CLI_USERS=`*UserGroupSid*: This parameter enables RepCLI usage on the clones. The value is the Security Identifier (SID) of the user account/group that will run RepCLI commands on the clones.

`GROUP_NAME`: Indicates the policy name that has the necessary exclusions and configurations to apply to the clones.

See [Chapter 5 Installing Windows Sensors on Endpoints](#) and [Windows Sensor Supported Commands](#). For more information about RepCLI, see *Managing Sensors by using RepCLI* in the User Guide.

- 5 Deploy the full clone pool from the golden image VMTemplate by using the Customization Specification.

The cloned VMs are registered to the Carbon Black Cloud console. If enabled by policy, a background scan is run on each cloned VM after the pool is provisioned. Note that the background scan can cause performance issues, depending on how many VMs exist per host.

- 6 Enable sensor settings to deregister inactive VMs. This setting provides operational and management benefits to instant clone VMs.
- a In the Carbon Black Cloud console, go to **Inventory>Endpoints>Sensor Options>Sensor Settings** or **Inventory>Workloads>Sensor Options>Sensor Settings** or **Inventory>VDI Clones>Sensor Options>Sensor Settings**.
  - b For instant clones, enable the following options and set the timeframes to ensure automatic clean-up of inactive, deregistered instant clones. Do not enable these options for full clones.
    - **Delete sensors that have been deregistered for...**
    - **Deregister VDI sensors that have been inactive for...**

- 7 Enable the **Signature Update** setting on **Enforce>Policy>Assigned Policy>Local Scan>Signature Update**.

---

**Note** This installation method requires a different golden image for the full clone pool than for the instant clone pool.

With the 3.7+ Windows sensor, a reboot is needed on VDI clones to fully apply new ransomware protections.

---

## Horizon Linked-Clones and Carbon Black 3.6+ Windows Sensor Best Practices

Carbon Black supports Horizon linked-clones; however, linked-clones are scheduled for extended support and end-of-life. Linked-clones are deprecated in Horizon 8. Carbon Black recommends migrating to Horizon version 7.13 and later versions, with the Carbon Black Cloud Windows 3.6 sensor or later.

See [Horizon Linked-Clones](#).

You can manage the performance impact of the Carbon Black Cloud sensor with linked-clones by ensuring the following:

- 1 The correct permission bypass (exclusions) are in place at the Policy level.
- 2 A background scan is completed on the golden image VM prior to using vCenter Server to take a snapshot of the golden image VM.

A background scan takes several hours to complete. There is a significant performance benefit from running it on the golden image. Completing the background scan enables the sensor to gather cloud reputation for hashes found on the golden image. This removes the need for the linked-clones to delay execution to pull reputation when those hashes eventually run.

## Carbon Black Linux Sensors with VMware Horizon Virtual Desktops

This section describes how to deploy Carbon Black Cloud Linux sensors on Horizon virtual desktops.

See [Chapter 4 Installing Sensors on Endpoints in a VDI Environment](#) for preliminary instructions.

Before you install Linux sensors in a Horizon VDI environment, confirm that your environment meets the minimum requirements.

- Carbon Black Cloud sensors: Linux sensor v2.12 and later
- Horizon 8.1 and later
- OS distributions and versions must be supported by the Linux sensor and Horizon



## Carbon Black Linux Sensor Policy Setting Recommendations for Horizon Golden Images

We recommend that Carbon Black Cloud console administrators create specific policies to manage a Horizon golden image.

After a policy is applied to the golden image, all clones inherit this policy unless otherwise directed by membership in an Asset Group or a Sensor Group.

---

**Important** Asset Groups is available to Carbon Black Cloud customers on 27 November 2023. Carbon Black recommends that you upgrade from Sensor Groups to Asset Groups as soon as it is operationally feasible for your organization. Sensor Groups will be phased out by 01 December 2024. See [Asset Groups and Sensor Groups](#) in the *User Guide*.

---

We recommend the following policy settings for a Horizon golden image.

- Duplicate the Standard policy and make the following changes on the **Sensor** tab.
- **Run Background Scan** – For optimal clone performance, run the background scan on the golden image. This pre-populates the sensor cache with the reputation of files that are currently on the system and improves clone performance. A background scan takes some time to complete, and not all users want to wait for the scan when creating a new image. For performance sensitive customers, the extra wait time might be worth it if the image is deployed at scale. Turn this setting OFF after the background scan is complete.
- **Auto-deregister VDI clone sensors that have been inactive for** – Enable this setting to remove any instant clones that been inactive for the specified duration. Set the timeframe to remove inactive VMs.

---

**Note** Carbon Black recommends setting an interval of at least 24 hours to ensure that sensors do not get de-registered during common maintenance windows from Carbon Black or your environment.

---

**Note** Previously, golden image sensors could be inadvertently uninstalled by auto-deregistration settings. This is no longer possible because the backend will not deregister any device that is the golden image for a clone.

---

## Horizon Golden Image Considerations for Carbon Black Linux Sensors

This article contains recommendations for installing the Carbon Black Linux sensor on the Horizon golden image.

- Make sure that the golden image never registers as a clone or gets deregistered.

- For optimal performance, allow the background scan to complete on the golden image before creating clones.

---

**Important** Do not run `"/opt/carbonblack/psc/bin/cbagentd -R"` on the golden image. This command converts the golden image into a clone, which might de-register the golden image if auto-deregister of VDI clone sensors is set and timeout has reached. The deregister of golden image results in failure of clone registration.

---

## Horizon Instant Clone Considerations for Carbon Black Linux Sensors

This article describes how a Horizon instant clone receives a Device ID and is assigned a policy.

---

**Important** Asset Groups is available to Carbon Black Cloud customers on 27 November 2023. Carbon Black recommends that you upgrade from Sensor Groups to Asset Groups as soon as it is operationally feasible for your organization. Sensor Groups will be phased out by 01 December 2024. See [Asset Groups and Sensor Groups](#) in the *User Guide*.

---

- 1 The endpoint requests a new Device ID.
- 2 The new Device ID is identified as a virtual desktop on the backend.
- 3 The endpoint inherits the policy from the golden image unless one of the following conditions applies. Manual policy assignment post-installation overrides the inheritance.
  - You have previously created Asset Groups and the installed sensor matches an Asset Group's criteria.
  - You have previously created Sensor Groups and the installed sensor matches a Sensor Group's criteria.

## Install the Carbon Black Cloud Linux Sensor on a Horizon Golden Image and Create Instant Clones

Use the following procedure to install the Carbon Black Cloud Linux sensor on a Horizon golden image and create instant clones.

### Prerequisites

See the following topics:

- [Chapter 4 Installing Sensors on Endpoints in a VDI Environment](#)
- [Carbon Black Linux Sensors with VMware Horizon Virtual Desktops](#)

### Procedure

- 1 Create the golden image for the clone pool deployment. Perform required Linux updates, dependency installation for Horizon, and Horizon agent installation.
- 2 Install the Linux sensor on the golden image by following the steps in [Installing Linux Sensors on Endpoints](#).

- 3 Confirm that configuration properties are set to enable the automatic identification and registration of a Horizon instant clone:
  - a The features are enabled by default. If `EnableAutoReregisterForVDIClones` does not exist in `/var/opt/carbonblack/psc/cfg.ini`, then you do not need to do anything and can proceed directly to Step 4. If the following property exists in the configuration file, confirm that it is set to one of the following values to enable automatic registration:
    - `EnableAutoReregisterForVDIClones=3`. Enables BIOS UUID and MAC Address hash change-based automatic registration. Recommended and default setting.
    - `EnableAutoReregisterForVDIClones=2`. Enables BIOS UUID change-based automatic registration.
  - b If the `EnableAutoReregisterForVDIClones` property is not set to the correct value, perform the following steps:
    - 1 Run `service cbagentd stop` OR `systemctl stop cbagentd`.
    - 2 Edit the `/var/opt/carbonblack/psc/cfg.ini` file to set the correct value.
    - 3 Run `service cbagentd start` OR `systemctl start cbagentd`.

---

#### Note

- To disable automatic registration, set the value of `EnableAutoReregisterForVDIClones` to 1.
  - When you install a Carbon Black Cloud Linux sensor 2.15+ on the Golden Image, Carbon Black automatically adds the `EnableAutoReregisterForVDIClones` property to the `/var/opt/carbonblack/psc/cfg.ini` file. The assigned value of the `EnableAutoReregisterForVDIClones` property depends on your platform. For example, `EnableAutoReregisterForVDIClones=1` for non-AWS, Azure, or GCP VMware platforms.
- 
- 4 Allow the background scan to complete on the golden image to optimize clone performance. Run the following command to determine whether the background scan has completed:

```
cat /var/opt/carbonblack/psc/blades/E51C4A7E-2D41-4F57-99BC-6AA907CA3B40/th.ini | grep LocalScanRunning
```

If `LocalScanRunning` is true, the background scan is ongoing.

- 5 Power off the golden image.
- 6 Take a snapshot of the golden image.

---

**Important** Do not run the `/opt/carbonblack/psc/bin/cbagentd -R` command on the golden image. This command turns the golden image into a clone, which might deregister pre-existing clones.

---

- 7 In the Horizon console, create an instant clone pool using the golden image and the snapshot created in Step 6.
- 8 After the pool becomes available in the Horizon console, verify that the newly created instant clones are registered with a new Device ID in the Carbon Black Cloud console.

## Carbon Black Windows Sensors with Citrix Virtual Desktops

This section describes how to deploy Carbon Black Cloud Windows sensors on Citrix virtual desktops.

---

### Note

---

- Linux sensors are not supported on Citrix virtual desktops.
- Windows sensor is not supported when installed on Citrix or Unidesk Application Layers. It is only supported on the OS layer, not the application layer.

## Citrix Golden Image Considerations for Carbon Black Sensors

This topic contains recommendations for installing the Carbon Black Cloud sensor on a Citrix golden image.

- For optimal clone performance, allow the background scan to complete on the golden image before creating clones.
- You can run the `repcli status` command to view the background scan status. For example:

```
General Info:
  Sensor Version[3.7.0.1473 - Sep 29 2021 - 20:34:38]
  Local Scanner Version[ - ]
  Disk Filter Version[3.7.0.1473]
  CbShared[104365] Policy[1269] FileAnalysis[386] Proto[548]
  Sensor State[Enabled]
  Details[LiveResponse:NoSession, LiveResponse:NoKillSwitch, LiveResponse:Disabled,
  SvcStable]
  DeviceHash[31dbad895ab7161f1f53bed2f4e3fa49ac64de98935b03752b53a407f65d9ea2]
  DeviceID[26365289]
  VirtualGuestToHostCommsStatus[Disconnected]
  ExternalIdentity[Not Available]
```

```
Kernel File Filter[Connected]
LastUser[Device\user]
Background Scan [Complete]
Total Files Processed[52581] Current Directory[None]
```

**Important** Do not run `repcli reregister now` or `repcli reregister onrestart` commands on the golden image. Either command turns the golden image into a clone, which might deregister the golden image if `autoderegister` is set and a time-out has occurred. Deregistration of the golden image results in clones being unable to reregister.

For more information about RepCLI, see *Managing Sensors by using RepCLI* in the User Guide.

## Carbon Black Policy Setting Recommendations for Citrix Golden Images

We recommend that Carbon Black Cloud console administrators create specific policies to manage a Citrix golden image.

All clones inherit the policy from the golden image unless otherwise directed by membership in an Asset Group or Sensor Group.

**Important** Asset Groups is available to Carbon Black Cloud customers on 27 November 2023. Carbon Black recommends that you upgrade from Sensor Groups to Asset Groups as soon as it is operationally feasible for your organization. Sensor Groups will be phased out by 01 December 2024. See [Asset Groups and Sensor Groups](#) in the *User Guide*.

We recommend the following policy settings for a Citrix golden image.

### Prevention Tab - Permissions

- **Bypass rules (exclusions)** – Policy-level bypass rules help achieve stability in a VDI environment.

Each organization must understand the trade-offs between performance and security. Carbon Black recommends the use of exclusions. Work with stakeholders to review risks and benefits (performance versus visibility) and apply the bypass rules as needed.

Carbon Black Cloud provides exclusions for supported methods as examples. Please review the applications that are installed in the VDI environment and apply any required bypass rules.

The following examples are based on public documentation for Citrix solutions. Additional bypass rules might be needed.

## Citrix bypass rules best practices

```
**\Program Files*\Citrix\**
    **\AppData\Local\Temp\Citrix\HDXRTConnector\*\*.txt
    **\*.vdiskcache
    **\System32\spoolsv.exe
```

**Note** Additional bypass rules might be required. For example, some organizations do not want to bypass `winlogon.exe`. This is a Citrix recommendation for any AV solution because a common problem with VDIs that use AV is longer login times. This bypass rule helps restore the expected experience.

## Prevention

### Blocking and Isolation

Best practices recommend applying **Blocking and Isolation** rules to address specific attack surfaces. To get started, we recommend that you duplicate the Standard policy rules to the Virtual Desktops policy.

### Local Scan tab

- **On Access File Scan Mode** – Disabled
- **Allow Signature Updates** – Enabled

### Sensor tab

- **Run Background Scan** – For optimal clone performance, run the background scan on the golden image. This pre-populates the sensor cache with the reputation of files that are currently on the system and improves clone performance. A background scan takes some time to complete, and not all users want to wait for the scan when creating a new image. For performance sensitive customers, the extra wait time might be worth it if the image is deployed at scale.
- **Scan files on network drives** – Disabled
- **Scan execute on network drives** – Enabled
- **Delay execute for Cloud scan** – Enabled. This critical setting serves as the sole point of reference for pre-execution reputation lookups. If it is disabled, endpoints must rely on **Application at Path** and **Deny List** rules for pre-execution prevention.
- **Hash MD5** – Disabled. The sensor always calculates the SHA-256.
- **Auto-deregister VDI sensors that have been inactive for** – Disable this setting to prevent unintentional uninstall of the sensor.

**Note** Previously, Carbon Black Cloud could automatically deregister golden image machines due to inactivity. Carbon Black Cloud no longer leverages time-based deregistration for any VM that has a child.

## Install the Sensor on a Citrix Golden Image

Use the following procedure to set up the Carbon Black Cloud sensor on a Citrix golden image virtual machine (VM).

### Procedure

- 1 Create the golden image VM.
- 2 Install the sensor on the golden image using the following command:

```
msiexec.exe /q /i <Sensor Installer Path> /L*v msi.log COMPANY_CODE="XYZABC"
CLI_USERS=<UserGroupSid> GROUP_NAME="<NAME Virtual Policy>"
```

### Note

- For Windows sensors 3.7MR1, add the `AUTO_REREGISTER_FOR_CITRIX=true` parameter to the command line.
- To establish a golden image-clone relationship with 3.8+ Windows sensors for Citrix VDIs, add both the `AUTO_REREGISTER_FOR_CITRIX=true` and `AUTO_REREGISTER_FOR_VDI_CLONES=1` parameters to the command line.

*< Sensor Installer Path >* : Replace this value with the location of the sensor MSI file; for example, `c:\tmp\installer_win-64-3.6.0.1941.msi`.

`CLI_USERS=UserGroupSid`: This parameter on the golden image enables RepCLI usage on the clones. The value is the Security Identifier (SID) of the user account/group that will run the `reregister now` command on the clone.

`GROUP_NAME`: Indicates the policy name that has the necessary exclusions and configurations to apply to the golden image.

See [Chapter 5 Installing Windows Sensors on Endpoints](#) and [Windows Sensor Supported Commands](#). For more information about RepCLI, see *Managing Sensors by using RepCLI* in the User Guide.

- 3 Complete an expedited background scan on the golden image to optimize clone performance.
  - a In the Carbon Black Cloud console, click **Enforce > Policies**, select the policy, and click the **Sensor** tab.
  - b Select the **Run background scan** option and select **Expedited** scanning.
  - c Click **Save**.
  - d You can track scan progress by running the `repcli status` command. The output will be similar to the following:

```
General Info:
  Sensor Version[3.7.0.1473 - Sep 29 2021 - 20:34:38]
  Local Scanner Version[ - ]
  Disk Filter Version[3.7.0.1473]
```

```

CbShared[104365] Policy[1269] FileAnalysis[386] Proto[548]
Sensor State[Enabled]
Details[LiveResponse:NoSession, LiveResponse:NoKillSwitch, LiveResponse:Disabled,
SvcStable]
DeviceHash[31dbad895ab7161f1f53bed2f4e3fa49ac64de98935b03752b53a407f65d9ea2]
DeviceID[26365289]
VirtualGuestToHostCommsStatus[Disconnected]
ExternalIdentity[Not Available]
Kernel File Filter[Connected]
LastUser[Device\user]
Background Scan [Complete]
Total Files Processed[52581] Current Directory[None]

```

- 4 Apply the clone policy to the golden image. For recommendations on clone policy settings, see [Carbon Black Policy Setting Recommendations for Citrix Clones](#).
- 5 Take a snapshot of the golden image.

---

**Note** Previously, the Carbon Black Cloud could automatically deregister golden image machines due to inactivity. The Carbon Black Cloud no longer leverages time-based deregistration for any VM that has a child.

---

## Citrix Clone Considerations for Carbon Black Windows Sensors

This topic describes how a Citrix clone receives a Device ID and is assigned to a policy.

---

**Important** Asset Groups is available to Carbon Black Cloud customers on 27 November 2023. Carbon Black recommends that you upgrade from Sensor Groups to Asset Groups as soon as it is operationally feasible for your organization. Sensor Groups will be phased out by 01 December 2024. See [Asset Groups and Sensor Groups](#) in the *User Guide*.

---

The `reregister` command is needed to register new clones with a unique Device ID.

When `reregister now` is run, a clone performs the following operations:

- 1 The endpoint requests a new Device ID.
- 2 The new Device ID is identified as a VDI endpoint on the backend.
- 3 The endpoint inherits the policy from the primary image unless one of the following conditions applies. Manual policy assignment post-installation overrides the inheritance.
  - You have previously created Asset Groups and the installed sensor matches an Asset Group's criteria.
  - You have previously created Sensor Groups and the installed sensor matches a Sensor Group's criteria.

---

**Note** With the 3.7+ Windows sensor, a reboot is needed on VDI clones to fully apply new ransomware protections.

---



## Carbon Black Policy Setting Recommendations for Citrix Clones

We recommend that Carbon Black Cloud console administrators create specific policies to manage Citrix clones.

After a policy is applied to the golden image, all clones inherit this policy unless otherwise directed by membership in an Asset Group or a Sensor Group.

---

**Important** Asset Groups is available to Carbon Black Cloud customers on 27 November 2023. Carbon Black recommends that you upgrade from Sensor Groups to Asset Groups as soon as it is operationally feasible for your organization. Sensor Groups will be phased out by 01 December 2024. See [Asset Groups and Sensor Groups](#) in the *User Guide*.

---

We recommend the following policy settings for Citrix clones.

### General Tab

- **Name** – Virtual Desktops – “Virtual Desktops” was previously a prescribed policy name. You can now put VMs into any policy name, and support VMs in different policies. This allows you to segregate clones from physical machines, and have different settings for each type.
- **Description** – This policy is optimized for Citrix clones. Special considerations improve performance and provide a strong base of reputation, behavioral, and targeted prevention.
- **Target Value** – Medium

### Sensor Tab

- **Display sensor message in system tray** - Enable this setting and add a message similar to this sample text: "Virtual Desktops Policy - Contact someone@example.com with any questions and concerns. Provide context regarding the issue and any available replication steps."

### Prevention Tab - Permissions

- **Bypass rules (exclusions)** – Policy-level bypass rules help achieve stability in a VDI environment.

Each organization must understand the trade-offs between performance and security. Carbon Black recommends the use of exclusions. Work with stakeholders to review risks and benefits (performance versus visibility) and apply the bypass rules as needed.

Carbon Black Cloud provides exclusions for supported methods as examples. Please review the applications that are installed in the VDI environment and apply any required bypass rules.

The following examples are based on public documentation for Citrix solutions. Additional bypass rules might be needed.

---

**Note** Additional bypass rules might be required. For example, some organizations do not want to bypass `winlogon.exe`. This is a Citrix recommendation for any AV solution because a common problem with VDIs that use AV is longer login times. This bypass rule helps restore the expected experience.

---

## Citrix bypass rules best practices

```
**\Program Files*\Citrix\**,
    **\AppData\Local\Temp\Citrix\HDXRTConnector\*\*.txt,
    **\*.vdiskcache,
    **\System32\spoolsv.exe
```

## Prevention

### Blocking and Isolation

Best practices recommend applying **Blocking and Isolation** rules to address specific attack surfaces. To get started, we recommend that you duplicate the Standard policy rules to the Virtual Desktops policy.

### Local Scan tab

- **On Access File Scan Mode** – Disabled
- **Allow Signature Updates** – Disabled

It is a best practice to disable **Allow Signature Updates** for clones. The local scan feature adds network overhead and augments resource utilization. The Carbon Black Cloud can pull reputation and enforce policy in real time from the Cloud because most VDI environments maintain 99% uptime.

However, you can install the signature pack to the golden image. This installation avoids the performance penalty of running updates on each clone, but allows the clones to have some offline protection. Malware that can be identified by the signature pack on the golden image is detected and blocked independent of Cloud activity.

Installing updates to a golden image works well for clones because the clones are frequently recreated from the golden image and thereby inherit the updates.

### Sensor tab

- **Run Background Scan** – To optimize performance, most VDI vendors recommend disabling any background scan of the file system. Operating under the expectation that the golden image is free of malware, and the clones maintain consistent connectivity to the Cloud, it is not recommended to utilize the background scan feature. Reputation is derived from the Cloud at execution when necessary, per policy configuration. See the following **Delay Execute for Cloud** scan recommendation.

- **Scan files on network drives** – Disabled
- **Scan execute on network drives** – Enabled
- **Delay execute for Cloud scan** – Enabled. This critical setting serves as the sole point of reference for pre-execution reputation lookups. If it is disabled, endpoints must rely on **Application at Path** and **Deny List** rules for pre-execution prevention.
- **Hash MD5** – Disabled. The sensor always calculates the SHA-256.
- **Auto-deregister VDI sensors that have been inactive for** – Enable this setting to remove any clones that been inactive for the specified duration.

## Citrix MCS and Carbon Black Windows Sensor

This topic describes the necessary steps to deploy Citrix MCS with the Carbon Black Cloud Windows 3.7MR1+ sensor.

---

**Note** With the Windows 4.0+ Sensor, if the parent hostname is modified in Private or Maintenance mode for a Citrix golden device, the sensor does not go through a reregistration process and continues to work using its existing device ID when `CitrixMCSModeCheckEnabled` is true.

The Windows 4.0+ Sensor includes the following configuration property that you can change by editing `cfg.ini`. The default setting is `Disabled`.

```
CitrixMCSModeCheckEnabled, false, "Enable/Disable citrix mcs device mode check for auto reregister."
```

---

### Procedure

- 1 Run the following command to install the Windows 3.7MR1+ sensor:

```
msiexec.exe /q /i <Sensor Installer Path> /L*v msi.log COMPANY_CODE="XYZABC"  
CLI_USERS=<sid> GROUP_NAME="<NAME Virtual Policy>" AUTO_REREGISTER_FOR_CITRIX=true
```

*< Sensor Installer Path >* : Replace this value with the location of the sensor MSI file; for example, `c:\tmp\installer_win-64-3.6.0.1941.msi`.

---

**Note** To establish a golden image-clone relationship with 3.8+ Windows sensors for Citrix VDIs, add both the `AUTO_REREGISTER_FOR_CITRIX=true` and `AUTO_REREGISTER_FOR_VDI_CLONES=1` parameters to the command line.

---

`CLI_USERS=sid`: This parameter enables RepCLI usage. The value is the Security Identifier (SID) of the user account/group that will run RepCLI commands .

`GROUP_NAME`: Indicates the policy name to apply.

See [Chapter 5 Installing Windows Sensors on Endpoints and Windows Sensor Supported Commands](#). For more information about RepCLI, see *Managing Sensors by using RepCLI* in the User Guide.

- 2 Check policy recommendations at [Carbon Black Policy Setting Recommendations for Citrix Clones](#).
- 3 Shut down the Golden Device.
- 4 Take a snapshot of the Golden Device.
- 5 Create a Citrix Machine Catalog based on the Golden Device snapshot.
- 6 Create a Citrix Delivery Group based on the newly created Citrix Machine Catalog.
- 7 Verify that Citrix clone devices are visible in the Carbon Black Cloud console.

---

**Important** Do not perform manual reregistration on the Golden Device.

Remove any previous BAT scripts or other reregister mechanisms that you have set up. Leaving such mechanisms in place can cause sensors to reregister more than once.

---

## Citrix PVS and Carbon Black Windows Sensor

This topic describes the necessary steps to deploy Citrix PVS with the Carbon Black Cloud Windows 3.7MR1+ sensor.

**Note** Carbon Black supports a single designated parent device to mount vDisk in Private Maintenance mode for a Citrix golden image. Prior to the Windows 4.0 sensor release, if a non-designated parent that has a different hostname is used to mount vDisk in Private Maintenance mode for a Citrix golden image, the sensor goes through a reregistration process and is marked as a clone.

With the Windows 4.0+ Sensor, if a non-designated parent that has a different hostname is used to mount vDisk in Private or Maintenance mode for a Citrix golden image, the sensor does not go through a reregistration process and continues to work using its existing device ID.

The Windows 4.0+ Sensor includes the following configuration property that you can change by editing `cfg.ini`. The default setting is `Enabled`.

```
CitrixPVSMODECHECKENABLED, true, "Enable/Disable citrix pvs device mode check for auto reregister."
```

---

### Procedure

- 1 Put the vDisk into Private mode and access vDisk from the *Designated Parent Device*.
- 2 Run the following command to install the Windows 3.7MR1+ sensor:

```
msiexec.exe /q /i <Sensor Installer Path> /L*v msi.log COMPANY_CODE="XYZABC"  
CLI_USERS=<UserGroupSid> GROUP_NAME="<NAME Virtual Policy>" AUTO_REREGISTER_FOR_CITRIX=true
```

`< Sensor Installer Path >` : Replace this value with the location of the sensor MSI file; for example, `c:\tmp\installer_win-64-3.6.0.1941.msi`.

`CLI_USERS=`*UserGroupSid*: This parameter enables RepCLI usage. The value is the Security Identifier (SID) of the user account/group that will run RepCLI commands .

GROUP\_NAME: Indicates the policy name to apply.

---

**Note** To establish a golden image-clone relationship with 3.8+ Windows sensors for Citrix VDIs, add both the `AUTO_REREGISTER_FOR_CITRIX=true` and `AUTO_REREGISTER_FOR_VDI_CLONES=1` parameters to the command line.

---

See [Chapter 5 Installing Windows Sensors on Endpoints](#) and [Windows Sensor Supported Commands](#). For more information about RepCLI, see *Managing Sensors by using RepCLI* in the User Guide.

- 3 Shut down the Designated Parent Device.
- 4 Put the vDisk into Shared mode.
- 5 Start Citrix clone devices with vDisk in Shared mode.
- 6 Verify that Citrix clone devices are visible in the Carbon Black Cloud console.

---

**Important** Do not use clone devices to access vDisk in Private mode. Always use vDisk in Private mode or Maintenance mode from the Designated Parent Device. Do not perform manual reregistration on the Parent Device.

Remove any previous BAT scripts or other reregister mechanisms that you have set up. Leaving such mechanisms in place can cause sensors to reregister more than once.

---

## Citrix PVS and Carbon Black Windows 3.7 Sensor

This topic describes the necessary steps to deploy Citrix PVS with the Carbon Black Cloud Windows 3.7 sensor. Earlier sensor versions are not supported.

### Procedure

- 1 Put the vDisk into Private mode and access vDisk from the *Designated Parent Device*.
- 2 Install or upgrade to the Windows 3.7 sensor.
- 3 In the Carbon Black Cloud console, click **Inventory>Endpoints** or **Inventory>Workloads**.
- 4 Select the endpoint, click **Take Action**, and then click **Enable bypass**. Confirm the action.
- 5 Edit `C:\ProgramData\CarbonBlack\DataFiles\cfg.ini` and append the following statement. A separate script to re-register the agent is not required after specifying this parameter in the `cfg.ini` file.

```
AutoReRegisterForCitrix=true
```

---

**Note** If you are upgrading to the 3.7 sensor from a previous sensor version, add: `HostNameAsOfLastReregister=<HOSTNAME>`. Replace HOSTNAME with the hostname of the Designated Parent Device.

---

- 6 Apply the new configuration by using the following RepCLI command. For more information about RepCLI, see *Managing Sensors by using RepCLI* in the User Guide.

```
RepCLI updateconfig
```

- 7 In the Carbon Black Cloud console, click **Inventory>Endpoints** or **Inventory>Workloads**.
- 8 Select the endpoint, click **Take Action**, and then click **Disable bypass**. Confirm the action.
- 9 Shut down the Designated Parent Device.
- 10 Put the vDisk into Shared mode.
- 11 Start Citrix clone devices with vDisk in Shared mode.
- 12 Verify that Citrix clone devices are visible in the Carbon Black Cloud console.

---

**Important** Do not use clone devices to access vDisk in Private mode. Always use vDisk in Private mode or Maintenance mode from the Designated Parent Device. Do not perform manual reregistration on the Parent Device.

Remove any previous BAT scripts or other reregister mechanisms that you have set up. Leaving such mechanisms in place can cause sensors to reregister more than once.

---

## Carbon Black Windows Sensors with vSphere Clients

This article describes how to install Carbon Black Cloud Windows sensors on the command line or through software distribution tools in a VMware vSphere environment to enable the automatic identification and registration of vSphere clones.

### Prerequisites

To get started, see [Chapter 4 Installing Sensors on Endpoints in a VDI Environment](#).

Make sure that your environment meets the minimum requirements:

- Carbon Black Cloud Windows sensor 3.7 MR2 and later
- vCenter Server 6.7 or later
- Host with ESXi 6.7 or later connected to the vCenter server
- Windows VM running OS versions that are supported by the sensor
- Carbon Black Host Module installed and running on the hosts on which VMs are deployed

### Procedure

- 1 Install the Windows sensor on VMs using the following command:

```
msiexec.exe /q /i <Sensor Installer Path> /L*v msi.log COMPANY_CODE="XYZABC"  
CLI_USERS=<UserGroupSid> AUTO_REREGISTER_FOR_VDI_CLONES=3 GROUP_NAME="<Virtual Policy>"
```

< *Sensor Installer Path* > : Replace this value with the location of the sensor MSI file; for example, `c:\tmp\installer_win-64-3.7.0.1503.msi`.

`CLI_USERS=<UserGroupSid>`: This parameter enables RepCLI usage on the clones. The value is the Security Identifier (SID) of the user account/group that will run RepCLI commands on the clones.

`GROUP_NAME`: Indicates the policy name that has the necessary exclusions and configurations.

See [Chapter 5 Installing Windows Sensors on Endpoints](#) and [Windows Sensor Supported Commands](#). For more information about RepCLI, see *Managing Sensors by using RepCLI* in the User Guide.

2 (Optional) Confirm that the required configuration properties are set correctly to enable the automatic identification and reregistration of vSphere clones:

- a Use the `repcli configprops` command to verify that the configuration properties have the expected values:

**Table 4-1. Configuration Properties**

Configuration Property and Value	Usage
<code>VHostEnabled=1</code>	<p><b>Controls communication with Host User World. Supported values:</b></p> <p>0=False</p> <p>1=True (default)</p>
<code>EnableAutoReregisterFor VDIclones=3</code>	<p><b>Controls the auto-reregistration feature. Supported values:</b></p> <p>1=Disable auto-reregistration for VDI clones (default)</p> <p>2=Make reregister decision based on BIOS UUID change only</p> <p>3=Make reregister decision based on BIOS UUID and MAC HASH change (required)</p>
<code>EnableExternalIdsChange DetectionForVDIclones=1</code>	<p><b>Controls Host User World-based auto-reregistration feature. Supported values:</b></p> <p>0=False</p> <p>1=True (default)</p>
<code>IncludeExternalIds InMsgsToBackend=1</code>	<p><b>Select if external identifiers should be sent in messages to the backend. Supported values:</b></p> <p>0=False</p> <p>1=True (default)</p>

- b If any of these configuration properties have different values, edit `C:\ProgramData\CarbonBlack\DataFiles\cfg.ini` to change them. To disable automatic reregistration, set `EnableExternalIdsChangeDetectionForVDIclones` to 0. After you have made your changes, run the `RepCLI updateconfig` command to immediately update the `cfg.ini` file.

---

**Note** You must put the sensor into bypass mode before you can edit `cfg.ini`. As a best practice, make a backup of `cfg.ini` into another directory before you edit it in a plain text editor. After you have edited `cfg.ini`, take the sensor out of bypass mode. For more details about editing `cfg.ini`, see [How To Change ConfigProps Via Cfgi.ini](#).

---



- 3 Issue the `repcli status` command to verify that the sensor can connect to the host module and can query external identifiers. For example:

```
cmd> repcli status
General Info:
  Sensor Version[3.7.0.1500 - Oct  8 2021 - 14:41:39]
  Local Scanner Version[ - ]
  Sensor State[Enabled]
  DeviceHash[e56223a76e00...]
  DeviceID[11223344]
  VirtualGuestToHostCommsStatus [Connected]
  ExternalIdentity[ee8fc1c7-
bd1e-4b10-9f32-04825a8b136e::501052cb-1d88-24b6-963d-9625a6c39f1e]
```

- 4 Clone the VM through vSphere or by using APIs (managed object browser, pyVmomi, etc.). The sensor running on the cloned VM should reregister. You can check the same by running `repcli status` on the clone VM and ensuring that the updated external identifiers have persisted and the Device ID has changed.

## Carbon Black Linux Sensors with vSphere Clients

This article describes how to install Carbon Black Cloud Linux sensors through the command line or software distribution tools in a vSphere environment to enable the automatic identification and registration of VC clones.

### Prerequisites

To get started, see [Chapter 4 Installing Sensors on Endpoints in a VDI Environment](#).

Make sure that your environment meets the minimum requirements:

- Carbon Black Cloud Linux sensor v2.12 and later
- Host with ESXi 6.7 or later
- vCenter Server 6.7 or later
- OS distributions and version must be supported by the Linux sensor

### Procedure

- 1 Follow the steps described in [Enable Host User World](#) to install the HostUW module on the ESXi host.
- 2 Install the Linux sensor on the primary VM. See [Chapter 2 Installing Linux Sensors on Endpoints](#).

- 3 Confirm that configuration properties are set to enable the automatic identification and registration of a vSphere clone. For more information about `cfg.ini`, see [About the Linux Sensor `cfg.ini` File](#).
  - a The features are enabled by default. If the following properties do not exist in `/var/opt/carbonblack/psc/cfg.ini`, then you do not need to do anything and can proceed directly to Step 4. If the following properties exist in the configuration file, confirm that they are set to the following correct values to enable automatic registration.
    - `VHostEnabled=true`. Enables the automatic registration feature.
    - `EnableExternalIdsChangeDetectionForVDIClones=true`. Enables HostUW-based automatic registration.
    - `IncludeExternalIdsInMsgsToBackend=true`. Enables external identifiers in messages sent to the Cloud backend.

---

**Note** To optionally *disable* automatic registration, set the value of `EnableExternalIdsChangeDetectionForVDIClones=false`.

---

- b If any property is not set to the correct value, perform the following steps:
    - 1 Run `service cbagentd stop` OR `systemctl stop cbagentd`
    - 2 Edit the `/var/opt/carbonblack/psc/cfg.ini` file to set the correct values.
    - 3 Run `service cbagentd start` OR `systemctl start cbagentd`.
- 4 Clone the primary VM.

# Installing Windows Sensors on Endpoints

# 5

This section describes how to install Carbon Black Cloud Windows sensors on the command line or through software distribution tools.

---

**Important** Before you begin the processes described here, read [Chapter 1 Getting Started with Sensor Installation](#). It contains highly relevant information to help you succeed in your sensor installation.

---

Before you can install sensors, perform the following steps:

[Obtain a Company Registration Code](#)

[Download Sensor Kits](#)

You can optionally verify the Windows sensor signatures before installing the sensor. See [Verifying Windows Sensor Digital Signatures](#).

If you are installing Windows sensors v3.5 or later, you can install sensors offline. This is useful for organizations who want to create a primary image and clone it to offline computers. This option is only available if you are installing sensors on the command line, or by using software distribution tools. See also [Windows Sensor Supported Commands](#).

---

**Note** With the release of the Windows 3.6 sensor, you can supply either the installation code (obtained via email — see [Invite Users to install Sensors](#)) or the company code (obtained via the console — see [Obtain a Company Registration Code](#)).

For VMware Workspace One installation instructions, see [Deploying Carbon Black Cloud Sensor with Workspace ONE UEM](#).

---

**Important** The 3.6 Windows sensor leverages a content management system to enable dynamic configuration of prevention features. Prior to installing or updating to 3.6, if you have restrictive firewall policies active in your environment, you might need to add a new firewall/proxy exclusion for the sensor to be fully functional. See [Configure a Firewall](#).

---

Read the following topics next:

- [Verifying Windows Sensor Digital Signatures](#)
- [Windows Sensor Rollback](#)
- [Local Scan Settings and the AV Signature Pack](#)

- [Windows Sensor Command Line Parameters](#)
- [Windows Sensor Supported Commands](#)
- [Windows Command Line Install on Endpoints — Examples](#)
- [Windows Sensor Log Files and Installed Services](#)
- [Installing Windows Sensors on Endpoints by using Group Policy](#)
- [Installing Windows Sensors on Endpoints by using SCCM](#)
- [Installing Carbon Black Cloud Sensor for Windows by Using Workspace ONE UEM](#)

## Verifying Windows Sensor Digital Signatures

You can optionally verify digital signatures of Windows sensor installation files.

### Prepare to Verify Windows Sensor Digital Signatures

Perform the following steps to prepare to verify Windows sensor digital signatures.

#### Procedure

- 1 Download the [Microsoft Windows SDK](#).
- 2 Install all components of the SDK.

---

**Note** SignTool is usually installed under `C:\Program Files (x86)\Windows Kits\10\bin`, but the exact location depends on the version of the SDK and your operating system. For example, it can be installed in any of the following (or other) locations:

- `C:\Program Files (x86)\Windows Kits\10\App Certification Kit\signtool.exe`
  - `C:\Program Files (x86)\Windows Kits\10\bin\x86\signtool.exe`
  - `C:\Program Files (x86)\Windows Kits\10\bin\x64\signtool.exe`
- 

- 3 Add the location of the `signtool` binary to your PATH environment variable.
  - a Press the Windows key.
  - b Type `env`.
  - c Click **Edit the System Environment Variables**.
  - d Click **Environmental Variables**.
  - e Select **Path** and click **Edit**.

- f At the end of the existing value, add the `Signtool` location. A semicolon (;) must separate the old value from the new value. For example:
  - old value = `%USERPROFILE%\AppData\Local\Microsoft\WindowsApps;`
  - new value = `%USERPROFILE%\AppData\Local\Microsoft\WindowsApps;C:\Program Files (x86)\Windows Kits\10\App Certification Kit\`
- g Click **OK** three times to save your changes and exit the editor.

## Verify the Signature of a Windows Sensor Install Package

Run the following procedure to verify the signature of a Windows sensor install package.

### Procedure

- 1 Open a command prompt window.
- 2 Run the following command, where *\$file\_to\_verify* is the name of the install package:

```
signtool.exe verify /pa /hash SHA256 /all $file_to_verify
```

- The `/pa` parameter instructs `Signtool` to check for code signing.
- An optional `/hash SHA256` parameter instructs `Signtool` to only check the SHA256 signatures.
- The `/all` parameter instructs `Signtool` to check all signatures on the file.

## Verify Multiple Files Digital Signatures

You can follow this procedure to verify multiple Windows sensor files. This procedure generally applies to installed products/packages.

### Prerequisites

You must know which files can be verified. Typically, files that cannot be verified change during the use of the product, such as configuration files or JIT compiled files.

### Procedure

- 1 Create a file that contains a list of files to verify, one file name per line. The following example includes relevant files for a x64 install package:

```
C:\users\user_name\desktop\cbd-setup64-3.8.0.276.msi
C:\program files\confer\api-ms-win-core-console-l1-1-0.dll
C:\program files\confer\api-ms-win-core-datetime-l1-1-0.dll
C:\program files\confer\api-ms-win-core-debug-l1-1-0.dll
C:\program files\confer\api-ms-win-core-errorhandling-l1-1-0.dll
C:\program files\confer\api-ms-win-core-file-l1-1-0.dll
C:\program files\confer\api-ms-win-core-file-l1-2-0.dll
C:\program files\confer\api-ms-win-core-file-l2-1-0.dll
C:\program files\confer\api-ms-win-core-handle-l1-1-0.dll
```

```

C:\program files\confer\api-ms-win-core-heap-l1-1-0.dll
C:\program files\confer\api-ms-win-core-interlocked-l1-1-0.dll
C:\program files\confer\api-ms-win-core-libraryloader-l1-1-0.dll
C:\program files\confer\api-ms-win-core-localization-l1-2-0.dll
C:\program files\confer\api-ms-win-core-memory-l1-1-0.dll
C:\program files\confer\api-ms-win-core-namedpipe-l1-1-0.dll
C:\program files\confer\api-ms-win-core-processenvironment-l1-1-0.dll
C:\program files\confer\api-ms-win-core-processthreads-l1-1-0.dll
C:\program files\confer\api-ms-win-core-processthreads-l1-1-1.dll
C:\program files\confer\api-ms-win-core-profile-l1-1-0.dll
C:\program files\confer\api-ms-win-core-rtlsupport-l1-1-0.dll
C:\program files\confer\api-ms-win-core-string-l1-1-0.dll
C:\program files\confer\api-ms-win-core-synch-l1-1-0.dll
C:\program files\confer\api-ms-win-core-synch-l1-2-0.dll
C:\program files\confer\api-ms-win-core-sysinfo-l1-1-0.dll
C:\program files\confer\api-ms-win-core-timezone-l1-1-0.dll
C:\program files\confer\api-ms-win-core-util-l1-1-0.dll
C:\program files\confer\api-ms-win-crt-conio-l1-1-0.dll
C:\program files\confer\api-ms-win-crt-convert-l1-1-0.dll
C:\program files\confer\api-ms-win-crt-environment-l1-1-0.dll
C:\program files\confer\api-ms-win-crt-filesystem-l1-1-0.dll
C:\program files\confer\api-ms-win-crt-heap-l1-1-0.dll
C:\program files\confer\api-ms-win-crt-locale-l1-1-0.dll
C:\program files\confer\api-ms-win-crt-math-l1-1-0.dll
C:\program files\confer\api-ms-win-crt-multibyte-l1-1-0.dll
C:\program files\confer\api-ms-win-crt-private-l1-1-0.dll
C:\program files\confer\api-ms-win-crt-process-l1-1-0.dll
C:\program files\confer\api-ms-win-crt-runtime-l1-1-0.dll
C:\program files\confer\api-ms-win-crt-stdio-l1-1-0.dll
C:\program files\confer\api-ms-win-crt-string-l1-1-0.dll
C:\program files\confer\api-ms-win-crt-time-l1-1-0.dll
C:\program files\confer\api-ms-win-crt-utility-l1-1-0.dll
C:\program files\confer\concrtdll.dll
C:\program files\confer\msvcpl140.dll
C:\program files\confer\ucrtbase.dll
C:\program files\confer\vccorlib140.dll
C:\program files\confer\vcruntime140.dll
C:\program files\confer\BladeRunner.exe
C:\program files\confer\CbNativeMessagingHost.exe
C:\program files\confer\RepCLI.exe
C:\program files\confer\RepMgr.exe
C:\program files\confer\RepUtils.exe
C:\program files\confer\RepUx.exe
C:\program files\confer\RepWAV.exe
C:\program files\confer\RepWmiUtils.exe
C:\program files\confer\RepWSC.exe
C:\program files\confer\Uninstall.exe
C:\program files\confer\VHostComms.exe
C:\program files\confer\blades\livequery\osqueryi.exe
C:\program files\confer\blades\livequery\exts\cbc_plugin_extension.ext.exe
C:\program files\confer\blades\livequery\exts\cbosqext.dll
C:\program files\confer\scanner\apcfile.dll
C:\program files\confer\scanner\apchash.dll
C:\program files\confer\scanner\avupdate.dll
C:\program files\confer\scanner\msvcr120.dll

```

```
C:\program files\confer\scanner\savapi.dll
C:\program files\confer\scanner\scew.dll
C:\program files\confer\scanner\scanhost.exe
C:\program files\confer\scanner\upd.exe
```

## 2 Create a batch file that contains the following text:

```
@echo off
set FILE=list_of_files
set numFiles=0
set numGoodSigs=0
setlocal ENABLEDELAYEDEXPANSION

for /f "delims== tokens=1,2" %%G in (%FILE%) do (
  if not exist "%%G\*" (
    set /a numFiles=numFiles+1
    (signtool verify /all /hash SHA256 /pa "%%G") && (set /a numGoodSigs=numGoodSigs+1)
  )

  @echo. & @echo.
)

set /a numBadSigs=numFiles-numGoodSigs

echo %numFiles% files checked
echo %numGoodSigs% verified files
echo %numBadSigs% UNverifiable files
```

## 3 Change the value of *FILE* in the batch file to specify the file that you created in Step 1.

## 4 Run the batch file.

### Results

A summary of how many files could or could not be verified is written at the end of the output. For example:

```
File: C:\program files\confer\api-ms-win-core-console-l1-1-0.dll
Index  Algorithm  Timestamp
=====
0      sha1      Authenticode
1      sha256    RFC3161

Successfully verified: C:\program files\confer\api-ms-win-core-console-l1-1-0.dll

67 files checked
67 verified files
0 UNverifiable files
```

## Windows Sensor Rollback

With the Carbon Black Cloud Windows 3.6 sensor and later, if a failure occurs during an initial install or uninstall, the endpoint will be returned to the state it was in prior to the attempt.

If a failure occurs during initial installation of the sensor, the sensor will rollback any changes made to the system. This includes files, services, and registry artifacts that were removed, thereby leaving the system in a clean state to reattempt installation. This rollback does not harm other services or files on the endpoint.

If a failure occurs during the uninstall of a sensor, the sensor will roll back any changes including files, services, and registry artifacts. This rollback does not harm other services or files on the endpoint. The endpoint continues to check into the console and is controlled through its designated policy.

With the Carbon Black Cloud Windows 3.7 sensor and later, rollback is supported for upgrade scenarios. If a failure occurs during sensor upgrades, the endpoint will be returned to the state it was in prior to the upgrade attempt by rolling back any changes including files, services, and registry artifacts. This rollback does not harm other services or files on the endpoint. The endpoint continues to check into the console and is controlled through its designated policy.

The following table describes types of rollbacks that different sensor versions support:

**Table 5-1. Rollbacks supported by sensor versions**

	3.5 and earlier	3.6	3.7 and later
Fresh install	Not supported	Supported	Supported
Uninstall	Not supported	Supported	Supported
Upgrade	Not supported	Not supported	Supported

The following table describes the expected final sensor state when failures occur while upgrading from different sensor versions:

**Table 5-2. Sensor states**

Upgrading from	Upgrading to	Upgrade failure point	System state after upgrade	Additional explanation
3.5	3.7 or above	Uninstall of 3.5	System could be left with partial 3.5	3.5 did not support uninstall rollback
3.5	3.7 or above	Install of 3.7	System left with no sensor	3.5 did not support upgrade rollback
3.6	3.7 or above	Uninstall of 3.6	System left with 3.6	3.6 supports uninstall rollback
3.6	3.7 or above	Install of 3.7	System left with no sensor	3.6 did not support upgrade rollback
3.7 or above	3.7 or above	Uninstalling old	System left with older version	Uninstall rollback supported
3.7 or above	3.7 or above	Installing new	System left with older version	Upgrade rollback supported



## Local Scan Settings and the AV Signature Pack

The AV Signature Pack is not packaged with the sensor installation, but should be downloaded and installed automatically after sensor installation based on policy settings.

---

**Note** The local scan feature is only available for Windows sensors 2.0 and later.

---

The AV Signature Pack requires approximately 120MB at rest. During run time, 400MB is required because a second copy is created; the scan continues to function while signatures are being updated. After the update is complete, the old signatures are deleted. At least 200MB of memory is required to run the local scan.

Signature file updates are ON by default via a policy setting. You might encounter high bandwidth utilization upon sensor installation due to the initial signature file download. Subsequent updates following the initial install of the AV Signature Pack are differential. Therefore, setting a regular update schedule ensures that every subsequent update remains small.

To avoid network saturation during sensor installation, we recommend that you install sensors in small batches.

### Disable Automatic Signature Updates and use the Standalone Installer

Use the following procedure to disable automatic signature updates.

#### Procedure

- 1 Sign in to the Carbon Black Cloud console.
- 2 Click **Enforce**, click **Policies**, and select the policy.
- 3 Click the **Local Scan** tab and disable **Signature Updates**. Click **Save**.
- 4 Install the sensors. Make sure the sensors are assigned a policy that has signature updates disabled (steps 1 and 2). Wait at least 10 seconds before you run the signature pack installer.
- 5 Click **Endpoints**, click **Sensor Options**, and click **Download sensor kits**.
- 6 Download the AV Signature Pack.
- 7 Run the following command under a user account that has full administrator rights by using system management software:

```
CbDefenseSig-YYYYMMDD.exe /silent
```

---

**Note** You can run the installation command through Live Response.

---

- On the **Local Scan** tab on the Policies page, enable **Signature Updates**. After you save the changes to the selected policy, sensors in that policy begin to download the AV Signature Pack from Carbon Black Cloud servers in the next 5-60 minutes.

By default, updates download every 4 hours with a staggered update window of 4 hours. You can change these settings on the **Local Scan** tab of the Policies page.

## To Update the AV Signature Pack by using the RepCLI Command

You can update the AV signature pack by using the RepCLI command.

### Procedure

- Log into the machine with a user account that matches the AD User or Group SID that was configured at the time of sensor install.
- Open a command prompt window with administrative privileges.
- Change the directory to `C:\Program Files\Confer`.
- Type the following command: `repcli UpdateAvSignature`

### Results

If the command is successful, the message **The request of AV signature update has been accepted** displays on the command line.

---

**Note** Active Directory-based SID authentication is not required to run the `repcli UpdateAVSignature` command. For more information about RepCLI, see *Managing Sensors by using RepCLI* in the User Guide.

---

## Windows Sensor Command Line Parameters

The following command line parameters are used during a Windows command line sensor installation.

Parameter	Required or Optional	Description
<code>/q</code>	Required	If you install without using this parameter, the user is prompted for an installation code.
<code>/i</code>	Required	This parameter tells the MSI to install.
<code>/L*</code>	Optional	Creates an MSI install log file.
<code>/L*vxx</code>	Optional	Creates a verbose MSI install log file. This is recommended over the <code>/L*</code> parameter because it provides more information to troubleshoot installation problems.

## Windows Sensor Supported Commands

You can use the following command line parameters during a Windows sensor install.

Using commands or command line parameters other than those listed in the following table can cause the installation to fail.

Carbon Black Cloud supports automatic detection of proxy settings; however, it does not prompt for or pass the machine's credentials for use in proxy authentication, if enabled. If proxy authentication is required for your environment, use the command line options to specify `PROXY_SERVER=value`, `PROXY_USER=value`, and `PROXY_PASSWD=value`. See [Configure a Proxy](#).

### Note

- Command options are case-sensitive.
- With Windows sensors 3.3.0 and later, you can use the `RepCLI` command line tool to locally administer the sensor. For more information about `RepCLI`, see *Managing Sensors by using RepCLI* in the User Guide.

Command Options	Values	Supported with Updates	Notes
<code>AUTO_CONFIG_MEM_DUMP=value</code>	true/false	Yes	Provides the ability to stop the sensor from automatically configuring the memory dump settings in the registry when set to false. Available for Windows sensors 3.5+.
<code>AUTO_REREGISTER_FOR_CITRIX=value</code>	true/false	Yes	Default is false. When set to true, this setting enables auto-reregistration for Citrix PVS and MCS clones. Available for Windows sensors 3.7MR1+.
<code>AUTO_REREGISTER_FOR_VDI_CLONES=value</code>	4 - Checks for Hostname change (available from 3.8+) 3 - Checks for BIOS UUID + MAC HASH changes (preferred) 2 - Checks for BIOS UUID change 1 - Auto Reregister disabled	Yes	Default for Windows sensor 3.7MR2 is 1. Default for Windows sensor 3.8+ is 3. Sets auto-reregistration functionality for Horizon and vSphere VDI clones. Available for Windows sensors 3.7MR2+.
<code>AUTO_UPDATE=value</code>	1/0 or true/false	No	Default is true (enable auto update). This setting toggles whether the sensor will accept backend-pushed upgrade requests. Turning this off will prevent the update from being pushed from the backend.

Command Options	Values	Supported with Updates	Notes
BACKGROUND_SCAN=value	1/0 or true/false	No	Default is true. This setting toggles whether sensor will do an inventory of what hashes exist on the machine. Not applicable to Audit and Remediation Standalone.
BASE_IMAGE=value	1/0 or true/false	No	Default is false; the installed image is a base image that can be cloned to child images. This option is not supported for Non-Persistent VDI installations, but is currently used for Persistent VDI installations.
BYPASS=value	1/0 or true/false	No	Default is false; setting it to true will enable bypass mode In bypass mode, the sensor does not send any data to the cloud: the sensor functions in a passive manner and does not interfere with or monitor the applications on the endpoint. Installing the sensor in bypass mode enables thorough testing for interoperability issues.
CBLR_KILL=value	1/0	No	A value of 1 disables Live Response functionality for the sensor. The default value is 0.  <b>Note</b> Not reversible without reinstalling the sensor.
CLI_USERS=sid	SID value for authenticated users group	No	Use this field to enable the RepCLI tool. Any member in the specified user group can use the authenticated RepCLI commands.
COMPANY_CODE=value	Company registration code	No	Required for command line installations.
CURL_CRL_CHECK	1/0	Yes	Default is 1 (enabled). See <a href="#">Disable CURL CRL CHECK</a>

Command Options	Values	Supported with Updates	Notes
<code>CURL_CRL_REVOKE_BEST_EFFORT</code>	1/0	Yes	<p>Default value is <b>False</b>.</p> <p>When set to <b>False</b>, upon failing to receive revocation information for the SSL certificate, the sensor rejects connection with the backend server or the MITM proxy.</p> <p>When set to <b>True</b>, the sensor will make a best effort attempt to verify the SSL certificate but will not reject the connection if revocation information can't be obtained due to firewall or other network restrictions.</p> <p><b>Note</b> When <code>CURL_CRL_CHECK=0</code> is set, CRL validation is skipped altogether and the <code>CURL_CRL_REVOKE_BEST_EFFORT</code> setting is not considered, however the sensor will always verify that the certificate is trusted in the local certificate store regardless of whether revocation checking is enabled.</p>
<code>DELAY_SIG_DOWNLOAD=value</code>	1/0	No	Default is delay signature/definition download. We recommend that you do not change the default value.
<code>FILE_UPLOAD_LIMIT=value</code>	4-byte integer representing number of megabytes	No	Example: value of 3 is a limit of 3*1024*1024 bytes; default value is 5.
<code>FORCE_STATIC_PROXY_USE</code>	true/false	Yes	<p>When set to <b>True</b>, the sensor will always use a proxy to reach the Carbon Black Cloud backend and will not use the existing fallback mechanism of trying with other options if the configured proxy is not reachable. The default value is <b>False</b>.</p> <p><b>Note</b> If this property is set after sensor installation, a reboot is required.</p>
<code>GROUP_NAME=value</code>	String value	No	<p>Optional policy name assignment. Enclose this value with double quotes if the policy name includes spaces. Use this parameter for Windows sensors 3.7 and earlier. For Windows sensors 3.8+, use the <code>POLICY_NAME</code> parameter instead.</p>
<code>HIDE_COMMAND_LINES</code>	1/0	No	Obfuscates command line inputs. Default is 0.

Command Options	Values	Supported with Updates	Notes
LAST_ATTEMPT_PROXY_SERVER	Value example: 10.101.100.99:8080	No	Optional. Sensor will attempt cloud access by using this setting when all other methods fail (including dynamic proxy detection).
OFFLINE_INSTALL=value	1/0 or true/false	No	Optional. Default is false. This parameter allows you to install sensors when the endpoint is offline. The sensor connects with the Carbon Black Cloud backend and accesses a policy when network connectivity is restored. The sensor is in a bypass state until the sensor can access the policy. This option is only available for Windows sensors v3.5 and later.
POLICY_NAME=value	String value	No	Optional policy name assignment. Enclose this value with double quotes if the policy name includes spaces. Use this parameter for Windows sensors 3.8+. For Windows sensors 3.7 and earlier, use the GROUP_NAME parameter instead.
PROXY_PASSWD=value	Proxy password	No	Optional.
PROXY_SERVER=value	server:port	No	Optional.
PROXY_USER=value	Proxy username	No	Optional.
USER_EMAIL=value	Email address Example: user@example.com	No	Optional. <b>Note</b> This option is not valid for Windows Sensor versions 4.0+.
VDI=value	1/0 or true/false	No	This option is deprecated in sensor versions 3.4+. Default is false.
VHOST_COMMS=value	true/false	Yes	Disables the VHostComms helper utility when set to false.

## Obfuscation of Command Line Inputs

Endpoint users might input sensitive data into the command line. The obfuscation of command line inputs protects against unauthorized users accessing the data in plain text in the sensor `.log` files and the sensor databases.

There are three ways to obfuscate command line inputs:

- **Command line** - `HIDE_COMMAND_LINES=1`
- **RepCLI** - `hideCmdLines [0|1]`
- **Set** `HideCommandLines=true` **in** `cfg.ini`

The setting enables the obfuscation of command line input in sensor .log files and databases. The data in the Carbon Black Cloud console is not obfuscated. For more information about RepCLI, see *Managing Sensors by using RepCLI* in the User Guide.

## Windows Command Line Install on Endpoints — Examples

Review the following examples of Windows sensor installations.

---

**Important** Asset Groups is available to Carbon Black Cloud customers on 27 November 2023. Carbon Black recommends that you upgrade from Sensor Groups to Asset Groups as soon as it is operationally feasible for your organization. Sensor Groups will be phased out by 01 December 2024. See [Asset Groups and Sensor Groups](#) in the *User Guide*.

---

The following commands should be on a single line. For documentation formatting reasons, they may appear here on several lines.

**Note** If the company code contains special characters (!, #, @, \$, etc.), you must wrap the company code in double quotation marks. For example: `COMPANY_CODE="XXXXDKIHWKH@ORFXXXX"`.

---

### Base Install using a Company Registration Code

```
msiexec /q /i C:\Users\UserFolderName\Desktop\installer_vista_win7_win8-32-3.3.0.953.msi /L*
log.txt COMPANY_CODE=XYZ
```

In this basic install example, no policy is specified; therefore, the sensors are assigned to either the Standard policy, or to a policy that an Asset Group or Sensor Group specifies (if Asset Groups or Sensor Groups are defined and the sensors match group criteria).

### Base Install into a Specific Policy

```
msiexec /q /i C:\Users\UserFolderName\Desktop\installer_vista_win7_win8-32-3.3.0.953.msi /L*
log.txt COMPANY_CODE=XYZ POLICY_NAME=Phase1
```

Using the POLICY\_NAME (policy assignment option) assigns the sensor to the specified policy. To use Asset Groups or Sensor Groups to determine a policy assignment, omit this option.

**Note** For Windows sensors prior to 3.8+, replace POLICY\_NAME with GROUP\_NAME.

---

### Configure RepCLI Authenticated User AD Group

```
msiexec /q /i C:\temp\installer_vista_win7_win8-32-2.0.4.9.msi /L* log.txt COMPANY_CODE=XYZ
CLI_USERS=S-1-2-34-567
```

**Note** For more information about RepCLI, see *Managing Sensors by using RepCLI* in the User Guide.

---

## Windows Sensor Log Files and Installed Services

The following log files and installed services reside on endpoints that have a Windows sensor installed.

### Windows log files

Use the `/L* log.txt` command line option to obtain an MSI log that shows the Windows installation process. A `confer-temp.log` file is also generated in `C:\Users\Username\AppData\Local\Temp` that shows the sensor registration attempts to the cloud. These two log files are required for troubleshooting installation and update issues.

The Windows 3.6 sensor stores some log files in `Program Files` and some log files in `ProgramData`. Previous versions of the sensor stored logs in in the `\Program Files\Confer\Logs\` directory. Carbon Black will continue to move all log files to `ProgramData` to align with Microsoft guidelines. You must have administrative privileges to access the log files in `ProgramData`.

- `\Program Files\Confer\Logs\`
- `\ProgramData\CarbonBlack\Logs\`

### Windows installed services

- Main sensor service: `RepMgr64.exe`, `RepMgr32.exe`, `Scanhost.exe` (if local scanning is enabled)
- Utility: `RepUtils32.exe`, `RepWmiUtils32.exe`
- UI: `RepUx.exe`

## Installing Windows Sensors on Endpoints by using Group Policy

To install sensors by using Group Policy, make a batch file to pass the parameters to an edited `.msi` file.

By default, Group Policy installs software on startup; therefore, you must reboot the endpoint to install the sensor.

### Create a Microsoft Installer Transform (.MST) File

To create an `.mst` file, perform the following procedure.

#### Procedure

- 1 Sign in to the Carbon Black Cloud console.
- 2 On the navigation bar, click **Inventory** and then click **Endpoints**.



- 3 Click **Sensor Options** and then click **Download sensor kits**. Download the `.msi` file for the Windows sensor .
- 4 Download and install the Orca installer; see [https://msdn.microsoft.com/en-%20us/library/windows/desktop/aa370557\(v=vs.85\).aspx](https://msdn.microsoft.com/en-%20us/library/windows/desktop/aa370557(v=vs.85).aspx).
- 5 Right-click the `.msi` file that you downloaded in Step 3 and click **Edit with Orca**.
- 6 Click **Transform > New Transform**.
- 7 Create additional **Property** table entries. Under **Tables > Property**, right-click in a blank space and then click **Add row**.
- 8 Click **Property** and enter "COMPANY\_CODE". Click **Value** and enter the company registration code for your organization. (See [Obtain a Company Registration Code](#).)
- 9 If you are installing in a VDI Environment, see [Chapter 4 Installing Sensors on Endpoints in a VDI Environment](#) for additional parameters.
- 10 Click **Transform > Generate Transform** and save the file as an `.mst` file. Use a file name that is easily recognizable.

---

**Note** Carbon Black recommends that you create a verbose `.msi` install log file to help troubleshoot Group Policy installation or update issues.

---

## Automatically Create a Windows Installer .MSI Log

Use this procedure to create an `.msi` log.

### Procedure

- 1 Open the Group Policy editor and expand **Computer Configuration > Administrative Templates > Windows Components**.
- 2 Select **Windows Installer** and double-click **Logging** or **Specify the types of events Windows Installer records in its transaction log**, depending on the Windows version.
- 3 Select **Enabled**.
- 4 In the **Logging** textbox, type `voicewarmupx`.
- 5 Click **Save Changes**

This setting creates an `.msi` install log for all users in the GPO in the `c:\Windows\Temp\` folder on the system volume.

- 6 To enable Windows Installer `.msi` log using the registry:
  - a Open Regedit.
  - b Go to registry key  
`HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\Installer`.

- c Set the **Logging** registry value to `voicewarmupx`.

---

**Note** If Group Policy is configured to automatically create a Windows Installer `.msi` log, the registry value `voicewarmupx` should match the value that is configured in Group Policy.

---

## Install Sensors by using Group Policy

You can install sensors via Group Policy by using the `.msi` file that you previously created.

### Procedure

- 1 Click **Start > Administrative Tools > Group Policy Management**.
- 2 Click **Software Settings > Software Installation > New > Package**.
- 3 Select the `.msi` file that you downloaded in the [Create a Microsoft Installer Transform \(.MST\) File](#) procedure.
- 4 Under **Deployment Method**, click **Advanced**.
- 5 Add a package name that identifies the sensor (for example, WinSensor34). For 32 bit `.msi` files only: in the **Deployment** tab, click **Advanced** and uncheck **Make this 32-bit x86 application available to Win64 machines**. Click **OK**.
- 6 Click the **Modifications** tab and click **Add**. Select the `.mst` file and click **Save**.
- 7 If you use a script to force a reboot to install software, run the script.
- 8 Check the Carbon Black Cloud console periodically to verify that sensor information is populating and that the sensors are checking in regularly.

### Note

- The path to the `.msi` and `.mst` files must be available through a network share that is accessible from everywhere in your network, and to which everyone has at least read permissions.
  - For additional optional installation properties, see [Windows Sensor Supported Commands](#).
  - Active Directory does not support command line parameters. You must make a batch file to pass the parameters or package to an edited `.msi` file. Upon the next system restart, a drive is mounted and the installation is scheduled. The installation failure rate when using Active Directory is usually higher than with other software management tools.
  - By default, Group Policy installs software on startup, You can force an install/reboot by using a script. Consider the restart requirement when you deploy sensors via Group Policy.
- 

## Installing Windows Sensors on Endpoints by using SCCM

You can install Windows sensors by using System Center Configuration Manager (SCCM).

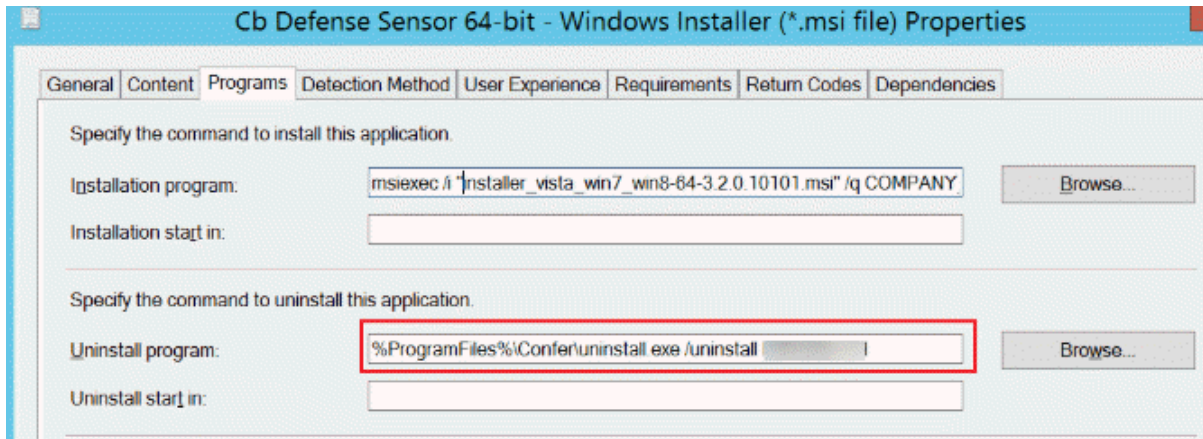
## Add the Sensor Application to SCCM

As a first step in installing a Windows sensor by using SCCM, perform the following procedure.

### Procedure

- 1 Open SCCM Configuration Manager.
- 2 In the **Software Library**, click **Overview > Application Management > Applications**.
- 3 Right-click **Applications** and click **Create Application**.
- 4 On the **General** page, select **Automatically detect information about this application from installation files**:
  - **Type**: Windows Installer (\*.msi file)
  - **Location**: Accessible share that contains the sensor .msi file
- 5 Click **Next**. On the **Import Information** page, a message displays: **Application information successfully imported from the Windows Installer**. Click **Next**.
- 6 On the **General Information** page, add the required COMPANY\_CODE install parameter and any other optional install parameters. See [Windows Sensor Supported Commands](#) for options. Click **Next**. On the **Summary** page, click **Next**.
- 7 On the **Completion** page, view the application details and click **Close**.
- 8 In the **Software Library**, right-click **Cb Defense Sensor Application** and click **Properties**.
- 9 Click the **Deployment Type** tab. Click the deployment type for CB Defense and click **Edit**. Note that the CB Defense type applies to Carbon Black Cloud Endpoint Standard, Carbon Black Cloud Enterprise EDR, and Carbon Black Cloud Audit and Remediation.

- 10 Click the **Programs** tab. If the **Require code to uninstall sensor** is enabled for the sensor policy and you want to be able to uninstall the sensor using SCCM, change the uninstall command from `msiexec /x "installer_vista_win7_win8-xx-x.x.x.xxxx.msi"` to `%ProgramFiles%\Confer\uninstall.exe /uninstall <Company Deregistration Code>`.



**Note** For security best practices, Carbon Black recommends that you regenerate the uninstall code after it has been used by SCCM.

Sensors that are offline at the time a new uninstall code is generated may not honor the new code. Previous uninstall codes may be required for these offline sensors.

- 11 Click the **Detection Method** tab. Select the configured detection rule and click **Edit Clause**. Change the **Setting Type** to **File System**
- 12 Select **The file system setting must satisfy the following rule to indicate the presence of this application**.
- 13 Set **Path** to `%ProgramFiles%\Confer` and **File or Folder name** to `RepUx.exe`.

- 14 Configure **MSI Property Version, Operator Greater than or equal to**. The **Version** is the currently installed sensor version.

The screenshot shows the 'Detection Rule' dialog box. The 'Setting Type' is set to 'File System'. Under 'Specify the file or folder to detect this application', the 'Type' is 'File', the 'Path' is '%ProgramFiles%\Confer', and the 'File or folder name' is 'RepUx.exe'. A 'Browse...' button is next to the path field. Below this, there is a checkbox for 'This file or folder is associated with a 32-bit application on 64-bit systems.' Two radio buttons are present: the first is 'The file system setting must exist on the target system to indicate presence of this application', and the second is 'The file system setting must satisfy the following rule to indicate the presence of this application', which is selected. Under the selected rule, the 'Property' is 'Version', the 'Operator' is 'Greater than or equal to', and the 'Value' is '3.2.0.10101'. 'OK' and 'Cancel' buttons are at the bottom right.

- 15 Click **OK** three times to save **Detection Rule**, **Detection Method**, and **Deployment Type**.

## Deploy the Sensor Application using SCCM

To install a Windows sensor by using SCCM, follow this procedure.

### Procedure

- 1 Open SCCM Configuration Manager.
- 2 In the **Software Library**, click **Overview > Application Management > Applications**.
- 3 Select the CB Defense application, and click **Deploy**.
- 4 On the **General** page, for the **Collection** field, click **Browse**. From the dropdown menu, select **Device Collections** and select a collection of devices. Click **Next**.
- 5 On the **Content** page, click **Add** to add a distribution point. Click **Next**.

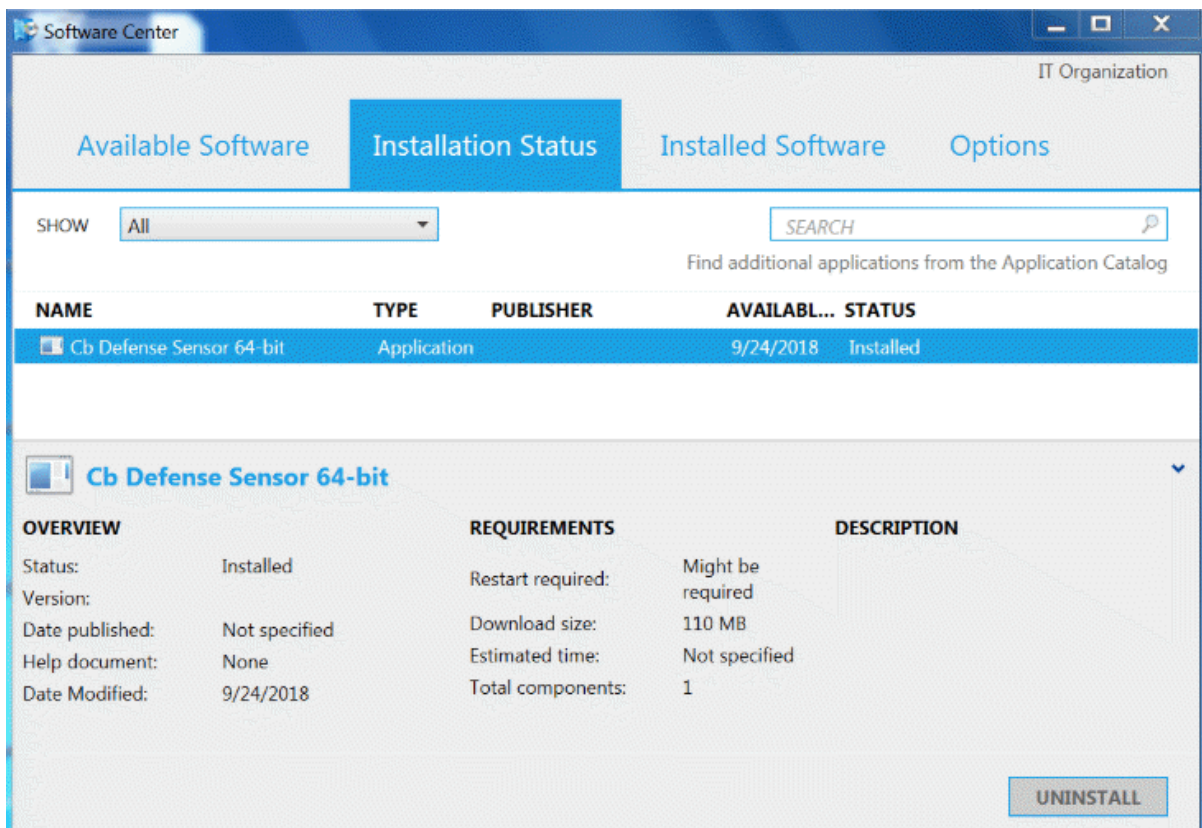
- 6 On the **Deployment Settings** page, set **Action** to **Install**, set **Purpose** to **Required**, and click **Next**.
- 7 On the **User Experience** page, set your deployment preferences and click **Next**.
- 8 On the **Alerts** page, set your alert preferences and click **Next**.
- 9 On the **Summary** page, review and confirm all settings and click **Next**.
- 10 On the **Completion** page, click **Close**.

## Verify that the Sensor Application was Deployed via SCCM

After installing the Windows sensor by using SCCM, perform the following procedure to verify the deployment.

### Procedure

- 1 In SCCM Configuration Manager, select the **CB Defense Sensor Application**, click the **Deployments** tab, and check **Compliance** status.
- 2 On the target device, open the Software Center and view the **Installation Status** or **Installed Software** tab.



# Installing Carbon Black Cloud Sensor for Windows by Using Workspace ONE UEM

As a Workspace ONE administrator, you can deploy the Carbon Black Cloud sensor for Windows as a managed application with Workspace ONE UEM and thus, silently deploy the sensor across all your managed devices that are running Windows 10.

## What to read next

### Procedure

- 1 [Deploy the Carbon Black Cloud Sensor for Windows as Managed Application in Workspace ONE UEM](#)

Perform the following procedure to add the Carbon Black Cloud sensor as an internal application and configure the deployment options in Workspace ONE UEM.

- 2 [Verify that Carbon Black Cloud Sensor for Windows Installed as Managed Application with Workspace ONE UEM](#)

You can use the Workspace ONE UEM admin console to verify that the Carbon Black Cloud sensor has been installed as a managed application on the assigned devices.

## Deploy the Carbon Black Cloud Sensor for Windows as Managed Application in Workspace ONE UEM

Perform the following procedure to add the Carbon Black Cloud sensor as an internal application and configure the deployment options in Workspace ONE UEM.

### Prerequisites

- Workspace ONE UEM with permissions to manage devices and applications.
- Carbon Black Cloud console access and admin account credentials.
- A device running Windows 10 to test the integration.
- Follow the instructions in [Obtain a Company Registration Code](#) and [Download Sensor Kits](#) to obtain your company registration code and download the Windows sensor kit.

### Procedure

- 1 Open the Workspace ONE UEM admin console.
- 2 Add the Carbon Black Cloud sensor for Windows as an internal application by uploading the Carbon Black Cloud sensor MSI installation file.

- 3 Navigate to the **Deployment Options** tab and define how the Workspace ONE UEM must install the sensor application on the device.

The following options under the **How to Install** section are set by the Workspace ONE administrator and the rest of the options are set automatically by Workspace ONE UEM.

- a Populate the **Install Command** text box with the following command: `msiexec /i "installer_vista_win7_win8-64-3.5.0.1523.msi" /qn COMPANY_CODE=<REPLACE WITH YOUR REGISTRATION CODE> .`
- b Optional. To obtain the installation log file for troubleshooting purposes, add `/L*vx <file name>` to the above command.

```
msiexec /i "installer_vista_win7_win8-64-3.5.0.1523.msi" /qn /L*vx <file name>
COMPANY_CODE=<REPLACE WITH YOUR REGISTRATION CODE>
```

- c Set the **Admin Privileges** to **YES** if not so already.

The Carbon Black Cloud sensor requires admin privileges for installation.

- 4 Click the **Save & Assign** button.
- 5 Assign the Carbon Black Cloud sensor application to the **Assignment Groups**, which represent the devices that must have the sensor installed.

#### What to do next

Confirm that the sensor application installation completed successfully.

## Verify that Carbon Black Cloud Sensor for Windows Installed as Managed Application with Workspace ONE UEM

You can use the Workspace ONE UEM admin console to verify that the Carbon Black Cloud sensor has been installed as a managed application on the assigned devices.

#### Procedure

- 1 Go to **Devices > List View**
- 2 Select a device and click the **Apps** tab.
- 3 Locate the Carbon Black Cloud Sensor 64-bit in the list of applications.

#### Results

The Carbon Black Cloud sensor is installed as a managed application on the devices that you previously assigned.



# Searching for Sensors

# 6

On any **Inventory** page in the Carbon Black Cloud console, you can search for specific sensors by any criteria that exists in the list of sensors. For example, you can search for specific devices, users, or operating systems on the **Inventory > Endpoints** or **Inventory > Workloads** page.

## Search for Sensors Based on Operating System

The following table provides examples of valid operating system search queries. They are not case-sensitive.

**Note** Operating system versions listed in the following table are examples only; other operating system versions are accepted.

Table 6-1. Example Operating System Search Queries

Linux	macOS	Windows
CentOS 7.9-2009	MAC	Windows
RHEL 7.8	OS X	Windows Server
Amazon 2.0	10.14.6	Windows 10
Debian 9.13	10.15.7	x64
Ubuntu 19.10	10.14.* where * is a wildcard	x86
OpenSUSE Leap 15.2		
SLES 12 SP2		

## Search for Sensors Based on User Name

Create a search query using one of the following search terms:

Search Term	Description
loginUserName	Currently logged-in user
email	User who installed the sensor

## Search for Sensors Based on Last Check-in

This search query requires a millisecond epoch timestamp. Various epoch timestamp converters are available online.

Create a search query using an epoch timestamp:

```
Find Sensors that have checked in since the timestamp
lastContact:>{Epoch_Timestamp}

Find Sensors that have not checked in since the timestamp
lastContact:<{Epoch_Timestamp}
```

### Examples

- To find sensors that have checked in since 21-Apr-2021 at 12:00:00AM GMT

```
lastContact:>1618963200000
```

- To find sensors that have not checked in since 21-Apr-2021 at 12:00:00am GMT

```
lastContact:<1618963200000
```

## Search for Sensors Based on Device Name

This section describes various ways to search for sensors based on device names. Note that special characters in device names act as delimiters and not as part of a string. In this context, special characters are:

```
- ~ ( ) [ ] { } ^ | & " :
```

Create a search query using one any of the following search criteria:

- Searching on a device name works as a plain-text search across all fields.

```
Example search: Win-10-Laptop-0123
Results: Win AND 10 AND Laptop AND 0123 in any field
```

- Use specific fields to search on specific information.

```
Example search: name:Win-10-Laptop-0123
Results: Win AND 10 AND Laptop AND 0123 in the Device Name only
```

- Use more than one instance of the `name:` field to search for all parts of the device name.

```
Example search: name:Win name:10 name:Laptop name:0123
Results: Win AND 10 AND Laptop AND 0123 in the Device Name only
```

- Use negation (-, AND NOT) to exclude criteria where additional devices are returned.

```
Example search 1: name:Win name:10 -name:Desktop -name:012345 -name:.domain.org  
Results: Device Names including Win AND 10 and excluding Desktop AND 012345
```

- Use other available fields to narrow results.

```
Example search: name:Win name:10 name:Laptop loginUserName:"Jane Doe"  
Results: Device Names including Win AND 10 AND Laptop and where the User field shows Jane  
Doe
```

## Search for Sensors Based on Domain\Hostname

To search for a domain\hostname, you must escape the \ character. For example:

```
Domain\\Hostname
```

# Updating Sensors on Endpoints

# 7

It is important that you keep your sensor versions up-to-date. There are several ways to update sensors.

- Update sensors on selected endpoints through the Carbon Black Cloud console. See:
  - [About Updating Sensors on Endpoints through the Console](#)
  - [Update Sensors through the Console](#)
- Reinstall the sensors.
- Use third party tools such as SCCM or GPO; see:
  - [Update Sensors on Endpoints that were Deployed by using SCCM](#)
  - [Update Sensors on Endpoints by using Group Policy](#)
- Update a sensor by double-clicking the new installer package or by issuing a command on the command line. Standard command line options are applicable. Command line options from the first install persist across updates.
  - To update a Windows sensor through the command line, see [Update Windows Sensors on Endpoints through the Command Line](#).
  - To update a Linux sensor through the command line, see [Update Linux Sensors on Endpoints through the Command Line](#).
  - To update a macOS sensor through the command line, simply reinstall the sensor. See [Chapter 3 Installing macOS Sensors on Endpoints and macOS Sensor Command Line Install](#).

Read the following topics next:

- [About Updating Sensors on Endpoints through the Console](#)
- [Update Windows Sensors on Endpoints through the Command Line](#)
- [Update Sensors on Endpoints by using Group Policy](#)
- [Update Sensors on Endpoints that were Deployed by using SCCM](#)
- [Update Linux Sensors on Endpoints through the Command Line](#)
- [View Progress of Sensor Updates](#)

- [Sensor Status and Details — Asset Groups](#)
- [Sensor Status and Details — Sensor Groups](#)
- [Endpoint Filters](#)
- [Bypass Reasons](#)

## About Updating Sensors on Endpoints through the Console

You can add up to 250,000 sensors to an upgrade request (job) in the Carbon Black Cloud console. After a Sensor Update Status (SUS) job is created, it can remain in a pending state while other jobs are being processed.

Sensor updates are prioritized by the date of the request, from oldest to newest. When there are less than 500 sensors eligible for update in all currently processing jobs, oldest jobs are processed first.

---

**Note** When all 500 processing slots are used for sensors within the same job, the SUS provides 10 extra processing slots filled for smaller jobs.

---

An organization can have multiple sensors update jobs at the same time.

The number of concurrent updates is the lesser of 25% of the total organization size or 500. For example, an organization that has 100 total sensors would hint up to 25 sensors to update at a time, and an organization that has 100,000 sensors would hint up to 500 sensors to update at a time. When an individual sensor completes its update process successfully or returns an error, a new sensor can be added to the processing queue to be updated.

The system attempts to upgrade up to 500 sensors at a time, and only considers sensors that are in a processing state (not pending). A job can only be promoted from pending to processing if at least one of its sensors has checked in within the last 30 minutes.

The processor runs every five minutes to see how many openings there are currently in the queue. It adds eligible sensors to the queue and sends hints for eligible sensors that are already in the queue. Sensors must have checked in within the last 30 minutes to be considered, and then must check in again after they are assigned a position in the queue.

SUS waits four hours before clearing any openings in a cancelled job. If a cancelled job had sensors, sensors that are in the processing state fill those openings in the queue.

---

**Note** Processing updates automatically timeout after two weeks. Timeouts occur even if the sensor has been hinted for an update, but the sensor has not successfully completed the update.

---

To monitor the status of sensor updates, see [View Progress of Sensor Updates](#).

---

**Tip** You can also use the [Sensor Update Services API](#) to perform batch sensor updates automatically across your organization. This API can update large quantities of devices — up to 250,000 — without putting your network at risk.

---

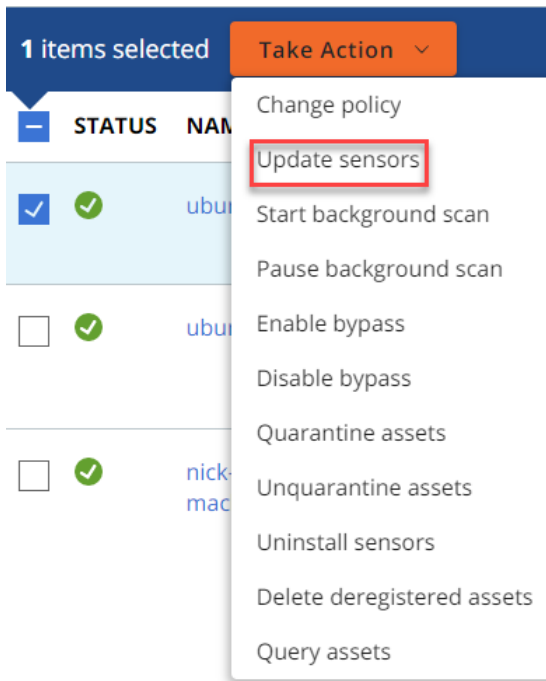
## Update Sensors through the Console

You can update sensors through the Carbon Black Cloud console.

**Important** If you are updating to the Windows 3.6 sensor, see *Configure a Firewall* in the Sensor Installation Guide.

### Procedure

- 1 On the left navigation pane, click **Inventory**.
- 2 Select the requisite asset that you want to update:
  - **Endpoints**
  - **AWS**
  - **VM Workloads**
  - **VDI Clones**
- 3 Search for and select the sensors to update.
- 4 Click **Take Action > Update Sensors**.



- 5 Confirm the number of sensors to update.

**Tip** You can choose to update all sensors on the page matching the search. Select **Update sensors on all X Assets** where *X* is the number of devices that match the search.

## Update Sensors



- Update sensors on the **1** selected assets  
 Update sensors on all **10012** assets matching the search

PLATFORM	VERSION
Windows	3.9.1.2464 

Sensor update name

**Update Sensors**

Cancel

- Select the sensor version from the **Version** dropdown menu.
- Provide a unique **Sensor Update Name** of up to 100 characters.

**Note** Jobs without a sensor update name use "--" in the Carbon Black Cloud console. Carbon Black recommends providing a unique sensor update name for searchability.

- Click **Update Sensors**.

### Results

After you initiate the sensor updates, you can view the progress of the updates on the **Sensor Update Status** tab.

**Note** For more information about the **Sensor Update Status** tab, see [View Progress of Sensor Updates](#).

## Update Windows Sensors on Endpoints through the Command Line

You can update Carbon Black Cloud Windows sensors through the command line.

**Note** Not all command line options apply to a Windows sensor update; see [Windows Sensor Supported Commands](#) for details.

### Procedure

- Sign into the endpoint.

- 2 Download the updated sensor file; see [Download Sensor Kits](#).
- 3 Open an administrative command prompt.
- 4 Run the update command; for example:

```
%SYSTEMROOT%\system32\msiexec.exe" /qN /i
"C:\temp\installer_vista_win7_win8-32-2.0.4.9.msi" /L*v+ "%SYSTEMROOT%\temp\cb-installer-
version.log
```

**Note** In this example, the `L*v+` option creates a verbose log in `c:\windows\temp\` that will append to any existing log rather than overwriting and replacing it.

## Update Sensors on Endpoints by using Group Policy

If you deploy sensors by using Group Policy, you must remove the existing sensor from the current Group Policy before you can perform a sensor update using Group Policy, the Carbon Black Cloud console, SCCM, manual updates, etc.

### Procedure

- ◆ Use one of the following procedures to remove a sensor from a Group Policy or update sensors using Group Policy?
  - To remove the existing sensor from Group Policy
    - a Click **Start > Administrative Tools > Group Policy Management** and select the Group Policy Object (GPO).
    - b Click **Computer Configuration > Policies > Software Settings > Software Installation**.
    - c Right-click the CB Defense Sensor package and click **All Tasks > Remove...**
    - d Select **Allow users to continue to use the software but prevent new installations** and click **OK**.

**Note** The previous procedure removes the `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Group Policy\AppDataMgmt\{Cb Defense GUID}` registry key without uninstalling the current version of the sensor. To confirm that the registry key is removed, open `Regedit` and go to `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Group Policy\AppDataMgmt`. Search for "CB Defense", "PSC Sensor", or "Carbon Black Cloud". If no results are found, the key is removed.

**Note** If you are updating from Windows sensor version 3.2.x.x, read [GPO upgrade fails on sensor version 3.2.x.x](#).

- To update sensors by using Group Policy
  - a Follow the preceding procedure to remove the sensor from its existing Group Policy.



- b Force a Group Policy update on all endpoints.
- c Use the following instructions to update the sensors: [Install Sensors by using Group Policy](#).

## Update Sensors on Endpoints that were Deployed by using SCCM

If you deployed sensors by using System Center Configuration Manager (SCCM), you can configure SCCM to allow alternate methods of updating the sensors.

### Procedure

- 1 Open SCCM and go to the **Software Library**.
- 2 Click **Overview > Application Management > Applications > Carbon Black**.
- 3 Click the **Deployment Type** tab and select the **Deployment Type** that is configured for the sensor.
- 4 Click the **Detection Method** tab, click the configured detection rule, and click **Edit Clause**.
- 5 Change the **Setting Type** to **File System**.
- 6 Set **Path** to `%ProgramFiles%\Confer`.
- 7 Set **File or Folder name** to `RepUx.exe`.
- 8 Select **The file system setting must satisfy the following rule to indicate the presence of this application**.
- 9 Configure **MSI Property Version** operator to **Greater than or equal to**. **Version** should be the currently installed sensor version.
- 10 Click **OK** three times to save the configuration.

## Update Linux Sensors on Endpoints through the Command Line

You can update Carbon Black Cloud Linux sensors through the command line.

### Procedure

- 1 Sign into the endpoint.
- 2 Download the updated sensor file; see [Download Sensor Kits](#).

- 3 For Linux sensor versions prior to 2.15, unpack the agent tar ball into: `/var/opt/carbonblack/psc/pkgs/upgrade_staging/` If you have not previously updated the sensor, this folder does not exist and you must create it.

For Linux sensors 2.15+, you can unpack the tarball and run the commands from any directory.

- 4 Run the update script:

RPM:

```
$rpm -U cb-psc-sensor-xxx.rpm
```

---

### Note

- For the RHEL sensors kit, you must specify the rpm package that corresponds to the distro version that you are installing.
- Run the following command to upgrade sensors that are deployed on an Amazon Linux 2 Graviton EC2 instance:

```
$ rpm -U cb-psc-sensor-<BUILD-NUMBER>.al2.aarch64.rpm --nodeps
```

---

el6 → centos/rhel/oracle 6.0-6.x

el7 → centos/rhel/oracle 7.0-7.x

el8 → centos/rhel/oracle 8.0-8.x

el9 → centos/rhel/oracle 9.0-9.x

DEB:

```
$dpkg --force-confold -i cb-psc-sensor-xxx.deb
```

- 5 Verify the following:

- Agent is upgraded — `/opt/carbonblack/psc/bin/cbagentd -v` to make sure that the agent matches the version you installed.
- Kernel or BPF module is loaded.
  - Kernel module: Run the following command and verify that there is a 1 in the right column of the output. This shows that the kernel module is loaded and enabled. Other versions of the kernel might display as disabled; this is acceptable.

Command: `lsmod | grep event_collector`

Sample output: `event_collector_2_x_yyyyyy zzzzz 1`

- BPF module: Run the following command and verify that the grep returns a single result with the command `event_collector`.

Command: `ps -e | grep event_collector`

Sample output: `85150 ? 00:00:05 event_collector`

- Check agent-blade details on the Endpoints page in the console:
  - Updated agent details are displayed
  - Agent checks in with the server at regular intervals
- For both Kernel and BPF, to see if the agent was successfully upgraded, you can check the build by running the following command:

```
cat /var/opt/carbonblack/psc/log/log.txt | grep -i upgrade
```

This command works for either server upgrade or manual upgrade.

## View Progress of Sensor Updates

You can view the **Sensor Update Status** tab on pages of the **Inventory** sections, such as the **AWS**, **Endpoints**, **VM Workloads**, and **VDI Clones** pages.

- 1 On the left navigation pane, click **Inventory**.
- 2 Select the category of inventory to update, for example:
  - **AWS**
  - **Endpoints**
  - **VM Workloads**
  - **VDI Clones**
- 3 Click the **Sensor Update Status** tab.

### Sensor Update Status

Use the search bar on the left navigation pane to find a sensor update name. Click the sensor name to show the details in the right navigation pane. The top right navigation pane displays details about the sensor update name, including the status of the update, the requestor, and the target sensor versions.

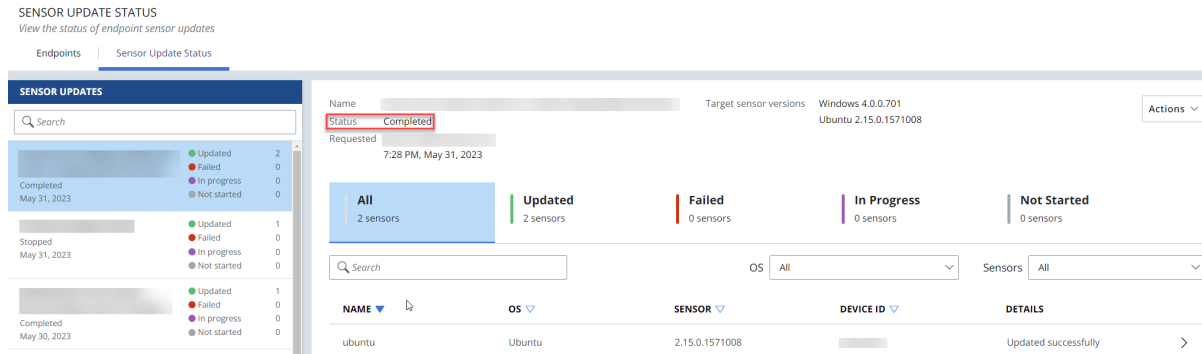
You can search for a sensor update name using the right navigation pane.

---

**Note** In this search box, you can only search for a device name. A device ID is not searchable.

---

Filter the sensor updates by OS or sensor version using the drop-down menus.



The **Status** bar displays the status of the sensor name update:

- **Not Started:** The sensor update has not started.
- **Initializing:** The sensor update is initializing. You can view the sensors when the initialization completes by refreshing. The wait time can be several minutes.
- **In Progress:** The sensor update is currently in progress.
- **Stopping:** The sensor update is stopping. The wait time can be up to several hours and you can view the change by refreshing.
- **Stopped:** The sensor update was canceled.
- **Completed:** The sensor update has fully completed.

**Note**

- Processing updates automatically timeout after two weeks. Timeouts occur when the sensor has not successfully completed the update. Typically, sensors that have not updated due to a timeout show a "Sensor unresponsive" error.
- You cannot restart a stopped update. You must initiate a new sensor update.

## View Results of Sensor Updates

The status of each sensor displays in the following tabs:

- **All:** Provides a list of all sensors within a sensor update.
- **Updated:** Provides a list of all sensors that are successfully updated to a targeted sensor version.
- **Failed:** Provides a list of all sensors that have failed to update along with the reason for failure.
- **In Progress:** Provides a list of all sensors that are currently being updated to target sensor version.
- **Not Started:** Provides a list of all sensors where the update has not started.

## Endpoint Details

To view the endpoint details of a sensor update, do one of the following:

- Double-click the sensor update.
- Click the > to the right of the **Details** column.

The **Endpoint Details** page displays:

---

### ENDPOINT DETAILS

Device ID	[REDACTED]
Sensor version	2.15.0.1571008
Last check-in	11:44:34 pm May 31, 2023
Installed by	[REDACTED]
Internal IP	[REDACTED]
External IP	[REDACTED]
Registered	7:23:19 pm May 31, 2023
Live response status	--
OS	LINUX
OS version	Ubuntu 21.04 x64
Uninstall code	[REDACTED]

## Actions Menu

Use the **Actions** drop-down menu to rename, export, or stop the sensor:

- **Rename:** You can rename the sensor update name while the sensor update is still in progress.
- **Export:** You can export all sensor details in a sensor update to a CSV file.

---

### Note

- The CSV export is available in the **Notifications** section for download.
  - Use the CSV file to view full results of your sensor update. The file contains useful information about your updates, including the device IDs of all requested sensors, their initial and updated sensor versions, and the reason for any update failure.
- 
- **Stop:** You can stop all sensors that are currently **Not Started** in the sensor update. These sensors move to the **Failed** tab and have a **Stopped by user** failure message.

---

**Note** If you stop a sensor update, any sensors that are **In Progress** continue to update to your selected sensor version.

---

## View Failed Sensors and Errors

If an update contains failures, click the > on the right of the row in the table to view a summary of failure reasons. Possible reasons for sensor failure can include:

- **Sensor unresponsive:** The sensor was offline or failed to check in with the system during the timeframe of the update.
- **No sensor found:** No sensor was found. The sensor might be deregistered.
- **Stopped by user:** The update request was stopped by a user before the sensor updated.
- **Update error:** The sensor failed to update to the targeted version.
- **FAILURE\_INELIGIBLE:** K8s sensors must be updated from the Clusters page.
- **FAILURE\_VALIDATE\_SHA256:** Sensor must be 2.15+ before upgrading to the targeted sensor version.

## Reviewing Audit Logs

If you create or rename a sensor update, a message displays in the **Audit Log**. You can view the **Audit Log** in the Carbon Black Cloud console. On the left navigation pane, click **Settings > Audit Log**.

The **Audit Log** message displays the requested sensor update to the selected versions.

4:06:08 pm Jun 1, 2023

Requested sensor update to the selected versions: [SensorVersion(type=WINDOWS, version=3.8.0.677)] (ID: 6d33229b-89f8-4468-ab97-7a2fea993e7)

## Sensor Status and Details — Asset Groups

The Endpoints page in the console displays sensor status and details. This topic describes this page for users who are using Asset Groups to manage assets.

The **Endpoints** tab on the Endpoints page displays all deployed sensors by default.

STATUS	NAME	OS VERSION	POLICY	ASSET GROUPS	S...	SENSOR	VULNERABILITY	LAST CHECK-IN	ACTIONS
<input type="checkbox"/>		MAC OS X 10.9.0	Domain Controllers (Rank 2) Assigned manually	Domain Controllers	--	2.0.1.4	--	1:47:27 pm Aug 9, 2022	
<input type="checkbox"/>		MAC OS X 10.9.0	Domain Controllers (Rank 2)	7	--	2.0.1.4	--	1:47:27 pm Aug 9, 2022	
<input type="checkbox"/>		MAC OS X 10.9.0	Domain Controllers (Rank 2)	7	--	2.0.1.4	--	1:47:26 pm Aug 9, 2022	
<input type="checkbox"/>		Windows 10 x64	Standard (Rank 1) Assigned by default	--	■	2.0.1.4	--	1:47:22 pm Aug 9, 2022	

You can limit which sensors to display by using the **Filters** options in the left pane. See [Endpoint Filters](#).

To export the table data into a CSV file, click the **Export** button in the upper right section of the page.








You can define which columns display in the results table. Click **Configure Table** at the bottom of the page to hide or display columns.

The resulting sensor data displays in the following columns

## Status

The **Status** column indicates the state of a sensor and any administrator actions that have been taken on the sensor. This column can contain multiple icons to indicate the sensor state.

Table 7-1. Sensor Status

Icon	Status	Description
	Active	Sensor has checked in within the last 30 days.
	Bypass	Sensor has been put into Bypass mode by an administrator. All policy enforcement on the device is disabled and the sensor does not send data to the cloud. Sensors also enter Bypass mode briefly during a sensor update. See <a href="#">Bypass Reasons</a> .
	Deregistered	Sensor has been deregistered or uninstalled; it will persist on the Endpoints page in this state until it is removed.
	Errors	Sensor is reporting errors.
	Inactive	Sensor has not checked in within the last 30 days.
	Pending install	Sensor has not been installed following an installation request email sent to a user.
No icon	Pending update	Sensor is pending an update.
	Quarantine	Sensor has been put into Quarantine mode. It is isolated from the network to mitigate the spread of potentially malicious activity.  <b>Note</b> Quarantine is not supported for Linux sensors before version 2.13.
No icon	Sensor out of date	Sensor is not using the current available sensor release version and is eligible for update.

## Name

The **Name** column represents the Device ID of the asset.

## OS Version

The **OS Version** column lists the version of the operating system that is running on the asset.

## Policy

The **Policy** column lists the group to which the sensor belongs (if any), how its policy was assigned, and the name of the assigned policy. If a sensor is not a member of an asset group and was manually assigned a policy, it is listed as **Manually assigned**. If the sensor metadata does not match any group criteria, it is listed as **Unassigned**.

## Asset Groups

The **Asset Groups** column displays the number of asset groups to which the asset belongs. If the asset belongs to multiple asset groups, a number indicates the number of asset groups in which the asset is a member. Click the number to view the corresponding group membership.

View Assets
✕

Showing 2 total assets in 'Test'

NAME	OS	TYPE	SOURCE
Asset Group Test 1	Windows 10 x64	Endpoint	Manually added
Asset Group Test 2	Windows	Endpoint	Group criteria

Showing 1-2 of 2
Items per page 
Jump to page 
< 1 >

## Sensor

The **Sensor** column lists the sensor version that is running on the asset.

## Sensor Gateway

The **Sensor Gateway** column lists the sensor gateway, if applicable.






## Last Check-in

The **Last Check-in** column displays the last time and date that the sensor checked in with the Cloud.

## Signature Status

The **Signature Status** column displays an icon that represents the status of each sensor signature version.

**Table 7-2. Signature Version Status**

Signature Status	Icon	Description
Up to date		Signature version is current. The installed signature version was released within 7 days of the current date.
Out of date		Signature version is out of date. The installed signature version has not been released within 7 days of the current date.
Not Available		Signature version is not yet reported or is unidentifiable. Signatures can display as not reported if the local scan is not configured or if the sensor encountered an error after the local scan was configured.
Not Applicable	No icon	Unidentifiable sensor signature version. This presents for macOS and Linux sensors.

## Target Value

The **Target Value** column lists the target value of the endpoint. This value can be Critical, High, Medium, or Low.

## User

The **User** column displays user data based on the OS and the sensor version.

- macOS 3.3.2+ versions display the last active user logged in to the device.
- Windows 3.5+ versions display the last active user logged in every 8 hours; if there is no interactive user logged in within the 8 hour window, a noninteractive user name can appear.
- Previous macOS and Windows versions display the user who installed the sensor.
- Linux versions are intentionally left blank because multiple, simultaneous logged-in users and desktop users are possible.

## Actions

The **Actions** column lists the actions that you can perform on the endpoint.



Click the  icon to investigate any events that have occurred on the endpoint.

Click the > icon to open an **Endpoint Details** panel that provides more details about the selected endpoint.

InvestorRelationsPC



**ENDPOINT DETAILS**



Device ID 41043845  
 Internal IP  
 External IP  
 Registered 12:13:25 pm May 11, 2023  
 Last check-in 12:13:25 pm May 11, 2023  
 Signature version --  
 Sensor 1.0.1.91  
 Installed by  
 Live response status  
 OS MAC  
 OS version MAC OS X 10.9.0  
 Uninstall code  
 Sensor Gateway --

**ASSET GROUPS**

ASSET GROUP ▾	POLICY ▾	POLICY RANK ▾
C-Suite	VIP effective policy	6
EUC - macOS workstations	Standard Policy for macOS Machines	9

**ACTIONS**

**Update sensor**

Update the sensor on InvestorRelationsPC

Update

**Quarantine asset**

Isolate InvestorRelationsPC on network to mitigate risk

Quarantine

## Sensor Status and Details — Sensor Groups

The Endpoints page in the console displays sensor status and details. This topic describes this page for users who are using Sensor Groups to manage assets.

The **Endpoints** tab on the Endpoints page displays all deployed sensors by default.

STATUS	NAME	USER	OS	GROUP/POLICY	S...	SENSOR	T	LAST CHECK-IN	ACTIONS
				Manually Assigned			Low	7:30:19 pm Nov 28, 2021	
				Manually Assigned			Low	7:31:21 pm Nov 28, 2021	
				Unassigned Standard			Medium	7:32:15 pm Nov 27, 2021	
				Manually Assigned			Low	3:31:14 am Nov 27, 2021	

You can limit which sensors to display by using the **Filters** options in the left pane. See [Endpoint Filters](#).

To export the table data into a CSV file, click the **Export** button in the upper right section of the page.

You can define which columns display in the results table. Click **Configure Table** at the bottom of the page to hide or display columns.

The resulting sensor data displays in the following columns by default.


## Status

The **Status** column indicates the state of a sensor and any administrator actions that have been taken on the sensor. This column can contain multiple icons to indicate the sensor state.

Table 7-3. Sensor Status

Icon	Status	Description
	Active	Sensor has checked in within the last 30 days.
	Bypass	Sensor has been put into Bypass mode by an administrator. All policy enforcement on the device is disabled and the sensor does not send data to the cloud. Sensors also enter Bypass mode briefly during a sensor update. See <a href="#">Bypass Reasons</a> .
	Deregistered	Sensor has been deregistered or uninstalled; it will persist on the Endpoints page in this state until it is removed.
	Errors	Sensor is reporting errors.
	Inactive	Sensor has not checked in within the last 30 days.
	Pending install	Sensor has not been installed following an installation request email sent to a user.
No icon	Pending update	Sensor is pending an update.

Table 7-3. Sensor Status (continued)

Icon	Status	Description
	Quarantine	Sensor has been put into Quarantine mode. It is isolated from the network to mitigate the spread of potentially malicious activity.  <b>Note</b> Quarantine is not supported for Linux sensors before version 2.13.
No icon	Sensor out of date	Sensor is not using the current available sensor release version and is eligible for update.

## Name

The **Name** column represents the Device ID of the endpoint.

## User

The **User** column displays user data based on the OS and the sensor version.

- macOS 3.3.2+ versions display the last active user logged in to the device.
- Windows 3.5+ versions display the last active user logged in every 8 hours; if there is no interactive user logged in within the 8 hour window, a noninteractive user name can appear.
- Previous macOS and Windows versions display the user who installed the sensor.
- Linux versions are intentionally left blank because multiple, simultaneous logged-in users and desktop users are possible.

## OS

The **OS** column lists the operating system that is running on the endpoint.

## Group/Policy

The **Group/Policy** column lists the group to which the sensor belongs (if any), how its policy was assigned, and the name of the assigned policy. If a sensor is not a member of a sensor group and was manually assigned a policy, it is listed as **Manually assigned**. If the sensor metadata does not match any group criteria, it is listed as **Unassigned**.

## Signature




The **Signature** column displays an icon that represents the status of each sensor signature version.

---

**Note** This feature is only available for Windows sensors.

---

Table 7-4. Signature Version Status

Signature Status	Icon	Description
Up to date		Signature version is current. The installed signature version was released within 7 days of the current date.
Out of date		Signature version is out of date. The installed signature version has not been released within 7 days of the current date.
Not Available		Signature version is not yet reported or is unidentifiable. Signatures can display as not reported if the local scan is not configured or if the sensor encountered an error after the local scan was configured.
Not Applicable	No icon	Unidentifiable sensor signature version. This presents for macOS and Linux sensors.

## Sensor

The **Sensor** column lists the sensor version that is running on the endpoint.

## Target

The **Target** column lists the target value of the endpoint. This value can be Critical, High, Medium, or Low.

## Last Check-in

The **Last Check-in** column displays the last time and date that the sensor checked in with the Cloud.

## Actions


The **Actions** column lists the actions that you can perform on the endpoint.



Click the  icon to investigate any events that have occurred on the endpoint.

Click the > icon to open an **Endpoint Details** panel that provides more details about the selected endpoint.

### ENDPOINT DETAILS



Device ID

Internal IP

External IP

Registered 7:30:18 pm Nov 29, 2021

Last check-in 7:30:19 pm Nov 29, 2021

Signature Version:

Installed by

Live response status

OS WINDOWS

OS version

Uninstall code

---

### ACTIONS

**Update sensor**  
Update the sensor on

**Quarantine asset**  
Isolate  on  network to mitigate risk

**Enable Bypass**  
Disable policy enforcement on

## Endpoint Filters

You can define which sensors display on the **Inventory>Endpoints** page by using the filters in the left pane.










The following filters are available on the **Inventory>Endpoints** page. You can combine filters for a more granular search.

**Note** The **Asset Group** filter only displays if you have upgraded to Asset Groups from Sensor Groups. This is a highly recommended upgrade - see *Asset Groups and Sensor Groups* in the User Guide.

FILTERS		Clear	⋮	⏪
+	Sensor Status (9)			
+	Sensor Version (4)			
+	OS (3)			
+	OS Version (3)			
+	Signature Status (4)			
+	Policy (7)			
+	Policy Assigned By (3)			
+	Golden Image Status (2)			
+	Asset Group (61)			
+	Host-Based Firewall Status (4)			
+	Subnet (12)			

Filter	Description
Sensor Status	The Status column indicates the state of a sensor and any administrator actions that have been taken on the sensor. This column can contain multiple icons to indicate the sensor state. For details about sensor states, see <a href="#">Sensor Status and Details — Asset Groups</a> or <a href="#">Sensor Status and Details — Sensor Groups</a> .
Sensor Version	You can filter the installed sensors by sensor version.
OS	You can filter sensors based on the device operating system, such as Windows or Linux.
OS Version	You can filter sensors based on the version of the device operating system, such as Windows 10 x64.



Filter	Description															
Signature Status	<p>The status of each sensor signature version displays in the <b>Sig</b> column.</p> <table border="1"> <thead> <tr> <th>Signature Status</th> <th>Icon</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Up to date</td> <td></td> <td>The sensor signature files are up-to-date if the signature version installed is released within 7 days of the current date.</td> </tr> <tr> <td>Out of date</td> <td></td> <td>The sensor signature files show as out-of-date one week after being disabled, until the updates are reenabled.</td> </tr> <tr> <td>Not Available</td> <td></td> <td>The sensor signature version is not yet reported if the local scan is not configured, or if the sensor encountered an error after local scan was configured, such as a connectivity issue</td> </tr> <tr> <td>Not Applicable</td> <td>No icon</td> <td>Unidentifiable sensor signature version. This is present for macOS and Linux sensors that are not supported.</td> </tr> </tbody> </table>	Signature Status	Icon	Description	Up to date		The sensor signature files are up-to-date if the signature version installed is released within 7 days of the current date.	Out of date		The sensor signature files show as out-of-date one week after being disabled, until the updates are reenabled.	Not Available		The sensor signature version is not yet reported if the local scan is not configured, or if the sensor encountered an error after local scan was configured, such as a connectivity issue	Not Applicable	No icon	Unidentifiable sensor signature version. This is present for macOS and Linux sensors that are not supported.
Signature Status	Icon	Description														
Up to date		The sensor signature files are up-to-date if the signature version installed is released within 7 days of the current date.														
Out of date		The sensor signature files show as out-of-date one week after being disabled, until the updates are reenabled.														
Not Available		The sensor signature version is not yet reported if the local scan is not configured, or if the sensor encountered an error after local scan was configured, such as a connectivity issue														
Not Applicable	No icon	Unidentifiable sensor signature version. This is present for macOS and Linux sensors that are not supported.														
Policy	<p>The Standard policy filter lists sensors that are:</p> <ul style="list-style-type: none"> <li>■ Newly deployed and are assigned the Standard policy by default.</li> <li>■ Do not meet a group's criteria and are assigned the default Standard policy.</li> </ul>															
Policy Assigned By	You can filter your view by assets that have had their policy assigned by Group, Default, or Manually Assigned.															
Golden Image Status	You can filter the displayed sensors based on their type: not a golden image, or as a golden image with clones.															
Asset Group	Filter by Asset Group, if activated. The Unassigned group filter shows only sensors which metadata does not match any group criteria.															
Host-Based Firewall Status	<p>Filter options are:</p> <ul style="list-style-type: none"> <li>■ Not enabled</li> <li>■ Active</li> <li>■ Errors</li> <li>■ Warning</li> </ul>															
Subnet	IP subnet.															

## Configure Results Table

You can further streamline your results table by clicking the **Configure Table** button at the bottom of the table, selecting which columns to display, and then clicking **Apply**.

The screenshot displays the Carbon Black Cloud interface. On the left, a 'FILTERS' sidebar lists various categories: Sensor Status (9), Sensor Version (5), OS (3), OS Version (3), Signature Status (4), Policy (7), Golden Image Status (2), Asset Group (5), Host-Based Firewall Status (4), and Subnet (13). The main area shows a table with columns for selection, status, and name. A 'Columns' modal is open, allowing users to select or hide columns. The 'Visible' section includes Status, Name, User, Target Value, OS, OS Version, Policy, Asset Groups, and Signature Status. The 'Hidden' section includes AD Distinguished Name, AD Organizational Unit, Subnet, Sensor, Vulnerability, Last Check-in, and AD Domain. An 'Apply' button is at the bottom of the modal. A red arrow points to the 'Configure Table' button located at the bottom of the table area, next to the text 'Showing 1-50 of 499'.

To export your filtered search results, click **Export**.

## Bypass Reasons

You can view the reason an asset goes into a bypass mode in the Carbon Black Cloud console.

The following table lists reasons for an asset to go in a bypass mode, and the remediation actions that you can perform. You can use a search value associated with a bypass reason to filter assets matching the bypass reason.

Search value of the bypass reason	Display value of the bypass reason	Description	Action to resolve bypass
sensorStates:"CSR_ACTION"	Bypass (Admin action)	The Carbon Black Cloud console instructs the sensor to go into a bypass mode.  Relates to sensors supporting Windows, Linux, and macOS.	Use the Carbon Black Cloud console or a local action to remove the sensor from the bypass state.
sensorStates:"REPUX_ACTION"	Bypass (Local action)	A local action instructs the sensor to enter bypass mode. For example, enable bypass locally on the sensor: <ul style="list-style-type: none"> <li>■ By elevating a command prompt and executing the command <code>"C:\Program Files\Confer\Uninstall.exe" / bypass 1 &lt;UninstallCode&gt;</code></li> <li>■ By logging into the asset with credentials configured at sensor installation, launching a command prompt, and executing the command <code>repcli bypass 1</code> from the <code>C:\Program Files\Confer</code> directory .</li> <li>■ By using the policy setting <b>Allow user to disable protection</b>. For more about this setting, see <i>Sensor Policy Settings</i> in the User Guide.</li> <li>■ By executing the command for installing the sensor with the option <code>bypass=1</code>.</li> </ul> Relates to sensors supporting Windows and macOS.	Use the Carbon Black Cloud console or a local action to remove the sensor from the bypass state.
<ul style="list-style-type: none"> <li>■ sensorStates:"UNSUPPORTED_OS" OR</li> <li>■ sensorStates:"OS_VERSION_MISMATCH"</li> </ul>	Error (Unsupported OS)	The installed sensor does not support the operating system.  Relates to sensors supporting macOS and Linux.	Upgrade the sensor or the operating system to a supported version. For information on the sensor operating environment requirements, see the applicable Operating Environment Requirements (OER).

Search value of the bypass reason	Display value of the bypass reason	Description	Action to resolve bypass
<ul style="list-style-type: none"> <li>■ sensorStates:"DRIVER_LOAD_NOT_GRANTED AND</li> <li>■ sensorStates:"DRIVER_USERSPACE</li> </ul>	Error (System ext. not approved)	The Carbon Black Cloud macOS sensor's System Extension is not approved. Relates to sensors supporting macOS.	Approve the System Extension that the sensor utilizes. See <i>Approving the System Extension and Network Extension for macOS 11+</i> in the Sensor Installation guide.
<ul style="list-style-type: none"> <li>■ sensorStates:"DRIVER_LOAD_NOT_GRANTED " AND</li> <li>■ sensorStates:"DRIVER_KERNEL"</li> </ul>	Error (Kernel ext. not approved)	The Carbon Black Cloud macOS sensor requires a Kernel Extension approval, regardless of the previous Kernel Extension approval status. Relates to sensors supporting macOS.	Approve the Kernel Extension. See <i>Approve the Kernel Extension (macOS 10.13 – macOS 11)</i> in the Sensor Installation Guide.
sensorStates:"REMGR_INIT_ERROR"	Error. Contact Broadcom Carbon Black Support)	The sensor cannot connect to the event_collector. Relates to sensors supporting Linux.	Check that the Linux distribution is supported. For version compatibility, see <a href="#">Carbon Black Cloud Linux Sensor Operating Environment Requirements</a> . If the distribution is supported, contact Carbon Black Support.
sensorStates:"KERNEL_HEADERS_NOT_INSTALLED"	Error (Missing kernel headers)	The sensor's Extended Berkeley Packet Filter (eBPF) probe cannot load because the Linux kernel headers, matching the running kernel version, are not installed. Also, the sensor might be running an unsupported OS Kernel version. Relates to sensors supporting Linux.	Verify that the kernel headers are installed. See <i>Linux Kernel Requirements for Linux Sensor Versions 2.10+</i> in the Sensor Installation Guide. For version compatibility, see <a href="#">Carbon Black Cloud Linux Sensor Operating Environment Requirements</a> .
sensorStates:"DRIVER_INIT_REBOOT_REQUIRED"	Error (Reboot required)	The asset requires a reboot to initialize the driver. Relates to sensors supporting macOS.	If a reboot does not resolve this issue, contact Broadcom Carbon Black Support.
sensorStates:"DRIVER_LOAD_PENDING"	Error (Extension load pending)	Loading extension is pending. Relates to sensors supporting macOS.	If a reboot does not resolve this issue, contact Broadcom Carbon Black Support.
sensorStates:"DRIVER_INIT_ERROR"	Error (Extension Error)	Driver fails to load properly. Relates to sensors supporting Windows, macOS, and Linux.	If a reboot does not resolve this issue, contact Broadcom Carbon Black Support.

<b>Search value of the bypass reason</b>	<b>Display value of the bypass reason</b>	<b>Description</b>	<b>Action to resolve bypass</b>
sensorStates:"SENSOR_UP GRADE_IN_PROGRESS"	Error (Update in progress)	The sensor is being updated on the asset. Relates to sensors supporting Windows.	Resolves immediately after the sensor update completes.
N/A	Bypass (Contact Broadcom Carbon Black Support)	Device is in bypass for an unknown reason.	Contact Broadcom Carbon Black Support for assistance.

# Uninstalling Sensors from Endpoints



You can uninstall sensors from the Carbon Black Cloud console or directly at the endpoint.

Read the following topics next:

- [Uninstall Sensors from the Endpoint by using the Console](#)
- [Require Codes to uninstall Sensors at an Endpoint](#)
- [Uninstall a Linux Sensor from an Endpoint](#)
- [Uninstall a 3.5+ macOS Sensor from an Endpoint](#)
- [Uninstall a pre-3.5.1 macOS Sensor from an Endpoint](#)
- [Uninstall a Windows sensor from an Endpoint](#)
- [Delete Deregistered Sensors from Endpoints](#)

## Uninstall Sensors from the Endpoint by using the Console

You can uninstall macOS and Windows sensors via the Carbon Black Cloud console.

---

**Note** You cannot uninstall Linux sensors via the Carbon Black Cloud console. You must uninstall Linux sensors by using the command line as explained in [Uninstall a Linux Sensor from an Endpoint](#).

---

### Procedure

- 1 Sign in to the Carbon Black Cloud console.
- 2 On the navigation bar, click **Inventory** and then click **Endpoints**.
- 3 Search for and select the sensors to uninstall.
- 4 Click **Take Action** and then click **Uninstall**.

### Results

After you uninstall a sensor, it persists on the **Endpoints** page as a deregistered sensor until you delete it. See [Delete Deregistered Sensors from Endpoints](#).

# Require Codes to uninstall Sensors at an Endpoint

If you have deployed v3.1 or later sensors, you can protect the action of uninstalling the sensor at the endpoint by requiring a unique, randomly-generated code. This setting is enabled per policy, and is recommended for security purposes. The uninstall code is case-sensitive.

## Procedure

### 1 To require a code to uninstall a sensor at an endpoint:

- a Sign in to the Carbon Black Cloud console, click **Enforce**, and then click **Policies**.
- b Select the policy.
- c On the **Sensor** tab, select the **Require code to uninstall sensor** checkbox and then click **Save**

After you have enabled this setting, a user must have an individual device uninstall code or a company deregistration code to uninstall the sensor at the endpoint. No code is required to uninstall sensors from within the Carbon Black Cloud console.

An individual device uninstall code is automatically generated when a sensor is registered with the Carbon Black Cloud.

### 2 To view a sensor uninstall code at an endpoint:

- a Sign in to the Carbon Black Cloud console.
- b On the navigation bar, click **Inventory** and then click **Endpoints**.
- c Click the > next to the sensor to view the uninstall code.

You can also generate a company deregistration code, and use this code to uninstall any sensor in your organization.

---

**Caution** The company deregistration code can be used to uninstall all sensors in your organization. If you do not want this capability, do not generate the company deregistration code.

---

### 3 To generate a company deregistration code:

- a Sign in to the Carbon Black Cloud console.
- b On the navigation bar, click **Inventory** and then click **Endpoints**.
- c Click **Sensor Options > View company codes**.
- d Select the **Deregistration** tab and copy the deregistration code. If the code does not exist, click **Regenerate deregistration codes**.

---

**Note** Only macOS and Windows sensors can be uninstalled with a company deregistration code. [Uninstall a Linux Sensor from an Endpoint](#) by using the command line.

---

## Uninstall a Linux Sensor from an Endpoint

You can use this procedure to uninstall a Linux Sensor from an endpoint.

---

**Note** After you run the command, the sensor remains listed in the **Registered Devices** list on the **Endpoints** page in the console until you click **Take Action > Uninstall**.

---

Run the following command:

- For CentOS, RHEL, SUSE, Oracle Linux, Rocky Linux, or Amazon Linux:

```
$ sudo rpm -e cb-psc-sensor
```

- For Ubuntu or Debian:

```
$ sudo dpkg --purge cb-psc-sensor
```

---

**Note** After you uninstall the Linux sensor on the following OS distributions: CentOS, RHEL, Suse, Amazon Linux, the sensor log files together with a copy of `cfg.ini` and directories such as `/opt/carbonblack` and `/var/opt/carbonblack` are not deleted from the endpoint.

---

## Uninstall a 3.5+ macOS Sensor from an Endpoint

Use the following procedures to uninstall 3.5+ macOS sensors from endpoints.

### Procedure

- ◆ Perform one of the following uninstall procedures:
  - To perform a command line uninstall of the sensor:
    - a Run the following command: `sudo /Applications/VMware\ Carbon\ Black\ Cloud/uninstall.bundle/Contents/MacOS/uninstall -y`
    - b If the **Require code to uninstall sensor** option is enabled, run the following command: `sudo /Applications/VMware\ Carbon\ Black\ Cloud/uninstall.bundle/Contents/MacOS/uninstall -y -c <Uninstall Code>`
    - c If the sensor was installed in KEXT mode, you must reboot the endpoint to fully remove the unloaded KEXT.
  - To perform an attended uninstall of the sensor:
    - a Mount the `CBCloud.dmg` and double-click **CBCloud Uninstall**.
    - b Proceed through the uninstallation prompts. You must authenticate as admin.
    - c If the sensor was installed in KEXT mode, you must reboot the endpoint to fully remove the unloaded KEXT.



## Uninstall a pre-3.5.1 macOS Sensor from an Endpoint

Use this procedure to uninstall pre-3.5.1 sensors from a macOS endpoint.

By default, this mode is interactive and requires a confirmation prompt unless you specify the `-y` parameter. To view all command line parameters, run the command by specifying the `-h` parameter.

### Procedure

- 1 Open **Terminal** with elevated privileges.
- 2 Type `sudo /Applications/Confer.app/uninstall -y` and click **Enter**.

If you require a device uninstallation code or a company deregistration code, enter it as part of the command; for example:

```
sudo /Applications/Confer.app/uninstall -y -c 35BQCCYX
```

## Uninstall a Windows sensor from an Endpoint

This procedure describes how to uninstall a Windows sensor from an endpoint.

---

**Note** You can uninstall multiple sensors by using batch files or system management tools.

---

### Procedure

- 1 Open a command prompt window with administrative privileges.
- 2 Go to the `Confer` directory.
- 3 Run the following command; if you require a device uninstallation code or a company deregistration code, enter it as part of the command; for example: `uninstall.exe /uninstall 35EQCCYG`

### Results

The `Confer` directory and log files remain after the sensor is uninstalled. The `Confer` directory is removed after an uninstall and a reboot.

## Uninstall Windows Sensors from an Endpoint by using Group Policy

You can use Group Policy to uninstall Windows sensors by following this procedure.

### Procedure

- 1 Click **Start > Administrative Tools > Group Policy Management** and go to **Software Installation**.
- 2 In the **Results** pane, right-click the CB Defense Sensor application, click **All Tasks**, and then click **Remove**.

- 3 In the **Remove Software** dialog box, select **Immediately uninstall the software from users and computers** and click **OK**.

### Results

The application is removed the next time a user logs on or restarts the computer.

---

**Caution** The sensor does not support uninstall using Group Policy if "Require code to uninstall sensor" is enabled. See <https://community.carbonblack.com/t5/Knowledge-Base/PSC-Sensor-uninstalled-without-de-registration-code/ta-p/84736>.

---

You can also uninstall a Windows sensor by using Group Policy **Software Installation > Results > Deployment**; however Carbon Black does not recommend this option.

## Enable SCCM to Uninstall a Windows Sensor from an Endpoint

You can enable SCCM to uninstall a Windows sensor.

On the **Programs** tab in SCCM, if the **Require code to uninstall sensor** is enabled for the sensor policy and you want to uninstall the sensor using SCCM, change the uninstall command from `msiexec /x"installer_vista_win7_win8-xx-x.x.x.xxxx.msi"` to `%ProgramFiles%\Confer\uninstall.exe /uninstall <Company Deregistration Code>`.

## Delete Deregistered Sensors from Endpoints

Use this procedure to delete deregistered sensors.

### Procedure

- ◆ Use one of the following procedures to delete deregistered sensors from an endpoint:
  - **To manually delete deregistered sensors on an endpoint**
    - a Sign in to the Carbon Black Cloud console.
    - b On the navigation bar, click **Inventory** and then click **Endpoints**.
    - c Filter the list of sensors to show only devices that have deregistered sensors.
    - d Select the sensors to delete.
    - e Click **Take Action** and then click **Delete deregistered devices**. You are prompted to confirm the deletion.
  - **To automatically delete deregistered sensors on an endpoint**
    - a Sign in to the Carbon Black Cloud console.
    - b On the navigation bar, click **Inventory** and then click **Endpoints**.
    - c Click **Sensor Options** and then click **Sensor settings**.
    - d Select **Delete sensors that have been deregistered for** and set the time frame. Click **Save**.

# Managing Sensors for VM Workloads

# 9

You can secure VMware workloads in your data center using Carbon Black Cloud. VMware workloads require Windows 3.6+ and Linux 2.9+ sensor versions.

Read the following topics next:

- [Installing Sensors on VM Workloads](#)
- [Update Sensors for Workloads from the Console](#)
- [Update Linux Sensors on Workloads through the Command Line](#)
- [Uninstall Linux Sensors from Workloads](#)
- [Uninstall Windows Sensors from Workloads](#)
- [Delete Deregistered Sensors from Workloads](#)

## Installing Sensors on VM Workloads

You install sensors on eligible VM workloads from the Carbon Black Cloud console.

The Carbon Black Cloud console allows you to view which deployed Virtual machine (VM) workloads in your data center are available for sensor installation. This data is available in the **Eligibility** column, part of the **Inventory > VM Workloads > Not Enabled** tab. The **Eligibility** column contains also the workloads that are not eligible for sensor installation and the once that need to upgrade to a supported OS version.

Eligibility Column	Description
Eligible	The eligible VM workloads have the appropriate version of the VMware Tools with the Carbon Black launcher and you can install sensors on them.
Not eligible	<p>VM workloads not eligible for sensor installation have the required version of the VMware Tools or Carbon Black launcher unavailable. To minimize your deployment efforts, a lightweight Carbon Black launcher is available with the VMware Tools. Carbon Black launcher must be available on the Windows and Linux virtual machines (VMs).</p> <ul style="list-style-type: none"> <li>■ For Windows VMs, the Carbon Black launcher is packaged with the VMware Tools. To receive the launcher for your workloads, you must install or upgrade VMware Tools to version 11.2 or later.</li> <li>■ For Linux VMs, you must manually install the launcher available at VMware Tools Operating System Specific Packages (OSPs). To learn more, visit <a href="#">Carbon Black Cloud Workload Guide</a>.</li> </ul> <p>After the launcher is available, you can proceed to install sensors on your workloads inventory.</p> <ul style="list-style-type: none"> <li>■ If VMs are offline, you cannot proceed with the installation. Go to the vCenter Server and power on the VMs.</li> </ul>
Not supported	Carbon Black Cloud Workload Plug-in does not support the Operating System (OS) or the OS version. Upgrade to the supported OS or version as per the system requirements.

Use the auto-install sensors feature to automatically install the latest version of the sensor on eligible VM workloads. The workload must have a connected appliance and be detected on a vCenter to use this feature.

There are no rule definitions: every eligible workload will automatically be triggered for sensor installation. This feature requires a one-time consent on the **Inventory > VM Workloads > Manage Sensor Settings** popup.

**Manage Sensor Settings**
✕

---

Auto-install sensors on eligible VM workloads

Delete sensors that have been deregistered for -- ▾

Deregister VDI sensors that have been inactive for -- ▾

Deregister VM workload sensors that have been inactive for -- ▾

---

Save

Cancel

## Prepare Your Workloads Environment for Sensor Installation

To prepare your environment for installing sensors on your deployed VM workloads you register the Carbon Black Cloud Workload Appliance with the vCenter Server and connect the appliance to the Carbon Black Cloud. Carbon Black launcher must be available on the VMs.

### Procedure

- 1 Set up your [Carbon Black Cloud Workload appliance](#).

Carbon Black Cloud Workload Appliance must be online and connected to the Carbon Black Cloud via an API key to receive a sensor. You confirm appliance connectivity in two ways:

- In the Carbon Black Cloud console, check for available VM workloads on the **Inventory > VM Workloads > Not Enabled** tab.
- In the Carbon Black Cloud console, go to the **Settings > API Access > API Keys** page and click the appliance name to view connection status.

- 2 Enable Carbon Black Cloud through a lightweight Carbon Black launcher to install a sensor for VM workloads.

- For Windows VMs, Carbon Black launcher is packaged with [VMware Tools](#). You must install or upgrade VMware Tools to version 11.2.0 or later to obtain the launcher.
- For Linux VMs, you must manually install the launcher from VMware Tools Operating System Specific Packages (OSPs). Download and install Carbon Black launcher for your guest operating system from the package repository at <http://packages.vmware.com/>. For detailed instructions, see [Carbon Black Launcher for Linux VMs](#).

## Install Sensors on VM Workloads

Use this procedure to install sensors on VM workloads through the Carbon Black Cloud console. You can use the configuration file to specify the proxy server that a Carbon Black launcher and a Carbon Black sensor can use after the installation completes.

### Prerequisites

- Make sure you have configured firewall correctly. For more information, see [Configure a Firewall](#).
- Make sure you are familiar with the command line installation options. For more information, see [Windows Sensor Supported Commands](#).
- The only supported proxy connection for the Carbon Black launcher and the Carbon Black sensor is the unauthenticated HTTP tunneling proxy.
- To obtain the Carbon Black launcher for Windows VMs with proxy support, install or upgrade VMware Tools to version 11.3.0 or later.

### Procedure

- 1 Sign in to the Carbon Black Cloud console.

- 2 On the navigation bar, select **Inventory > VM Workloads**.
- 3 Click the **Not Enabled** tab and select eligible workloads.

Eligible workloads are running a supported OS and have a correct version of the VMware Tools with the Carbon Black launcher.

ELIGIBILITY	INSTALL STATUS	NAME	OS	VMWARE TOOLS	ADDED
Eligible	Not started	Name: [redacted]	Microsoft Windows 10 (64-bit)	11333	6:23:35 am Aug 9, 2021
Eligible	Not started	Name: [redacted]	Microsoft Windows Server 2012 (64-bit)	11328	6:19:36 am Mar 16, 2021

- 4 Click the **Take Action** drop-down menu and select **Install sensors**.
- 5 Select the sensor version to install.

### Install Sensors

Install sensors on 1 selected workload(s)

**SENSOR VERSION**  
Learn more in [Sensor Release Notes](#) and [Sensor Install Guide](#)

OS	SENSOR VERSION
Windows 64-bit	3.7.0.1375 <span style="float: right;">▼</span>

**SENSOR CONFIGURATION FILE**  
[Download a template](#) | Learn more in [Sensor Install Guide](#)  
Authenticated proxy is not supported as part of configuration setting

Upload File (file format: ini, txt, conf, cfg)

Install
Cancel

- 6 Optional. Download and update the sensor configuration file.

By default, the INI file contains the following configurations that are mandatory for the successful installation of your Windows and Linux sensors.

Command Options	Values	Description/Notes
EncodedCompanyCode=value	String	For sensor version 3.0+ an encoded company code is required. The encoded company code is encoded with both - the 8-digit code and backend server.
CompanyCode=value	String	The company registration code you must acquire for command line installations.
BackendServer=value	String	The backend URL.

To customize the Windows sensor installation, you can add the following optional parameters during sensor install.

**Note** Windows is the only supported operating system for sensor install customization. Currently, you cannot customize the installation of Linux sensors.

Command Options	Values	Description/Notes
ConfigureMemoryDumpSettings=value	true/false Default value is true.	When false, it prevents the sensor from automatically configuring the memory dump settings in the registry. Available for Windows sensors 3.5 and later.
AutoReRegisterForCitrix=value	true/false Default value is false.	When true, it enables auto-reregistration for Citrix PVS and MCS clones. Available for Windows sensors 3.7MR1 and later.
EnableAutoReregisterForVDIClones=value	4 - Checks for Hostname change (available from 3.8+) 3 - Checks for BIOS UUID and MAC HASH changes (preferred) 2 - Checks for BIOS UUID change 1 - Disables Auto Reregister	Sets the auto-reregistration functionality for Horizon and vSphere VDI clones. <ul style="list-style-type: none"> <li>■ For Windows sensor 3.7MR2, the default value is 1.</li> <li>■ For Windows sensor 3.8 and later, the default value is 3.</li> </ul> Available for Windows sensors 3.7MR2 and later.
AutoUpdate=value	1/0 or true/false Default value is true.	Toggles whether the sensor will accept backend-pushed upgrade requests. When false, it prevents the update from being pushed from the backend.

Command Options	Values	Description/Notes
BackgroundScan=value	1/0 or true/false Default value is true.	Toggles whether the sensor does an inventory of what hashes exist on the machine.  Not applicable to Audit and Remediation Standalone.
InstallBypass=value	1/0 or true/false Default value is false.	When true, it enables bypass mode.  The sensor functions in a passive manner and does not interfere with or monitor the applications on the endpoint.  Installing the sensor in bypass mode enables thorough testing for interoperability issues.  For information on sensor bypass mode, see the User Guide.
CbLRKill=value	1/0 Default value is 0.	When 1, it disables Live Response functionality for the sensor.  <b>Note</b> To enable Live Response, reinstall the sensor.
AuthenticatedCLIUsers=value	SID value for authenticated users group	Enables the RepCLI tool. Any member in the specified user group can use the authenticated RepCLI commands.
ConnectionLimit=value	Number of connections per hour By default, there is no limit.	Optional.
CurlCrlCheck=	1/0 Default value is 1.	When 0, it disables CRL check during an initial sensor installation. For more information, see <a href="#">Disable CURL CRL CHECK</a> .
DelaySigDownload=value	1/0 Default value is 1.	We recommend that you keep the delay signature/definition download option enabled.
FileUploadLimit=value	4-byte integer representing number of megabytes Default value is 5.	Example: value of 3 is a limit of 3*1024*1024 bytes.
GroupName=value	String	Optional policy name assignment. Enclose this value with double quotes if the policy name includes spaces. <ul style="list-style-type: none"> <li>■ For Windows sensors 3.7 and earlier, use this parameter.</li> <li>■ For Windows sensors 3.8 and later, use the <code>PolicyName</code> parameter instead.</li> </ul>



Command Options	Values	Description/Notes
PolicyName=value	String	Optional policy name assignment. Enclose this value with double quotes if the policy name includes spaces. <ul style="list-style-type: none"> <li>■ For Windows sensors 3.8 and later, use this parameter.</li> <li>■ For Windows sensors 3.7 and earlier, use the <code>GroupName</code> parameter instead.</li> </ul>
HideCommandLines=value	1/0 Default value is 0.	Obfuscates command line inputs.
LastAttemptProxyServer=value	String Example: 10.101.100.99:8080	Optional. Sensor attempts Cloud access by using this setting when all other methods fail (including dynamic proxy detection).
LearningMode=value	Number of hours after sensor install to limit event types. By default, disabled.	Optional. Reduces the load on the backend by dropping some report types after initial install. Generally, more reports are sent to the backend soon after sensor install, because the sensor reports on newly detected hashes. Learning mode reports only on file and process behavior while the sensor is detecting hashes. Reporting of API, registry, and network behavior is dropped during this period.
OfflineInstall=value	1/0 or true/false Default value is false.	Optional. Allows you to install sensors when the endpoint is offline. The sensor connects with the Carbon Black Cloud backend and accesses a policy when network connectivity is restored. The device is in a bypass state until the sensor can access the policy. For Windows sensors 3.5 and later.
ProxyServerCredentials=user:password=value	Proxy password and username	Optional.
ProxyServer=value	server:port	Optional.
QueueSize=value	Event backlog Default value for Endpoint Standard is 100MB.	Optional. This value does not include SSL overhead.
RateLimit=value	KB per hour Default value is No Limit.	Optional.

Command Options	Values	Description/Notes
EmailAddress=value	Example: user@example.com	Optional.
VHostEnabled=value	true/false Default value is true.	When false, disables the VHostComms helper utility.

## 7 Click **Install**.

You see a **Sensor installation submitted** notification and the install status for the VM changes to **In Progress**.

It takes up to 5 minutes for the installation to complete.

### Results

After the sensor installs, it appears on the **Enabled** tab.

## Update Sensors for Workloads from the Console

Use this procedure to update sensors for Workloads from the Carbon Black Cloud Console.

It is important that you keep your sensor versions up-to-date.

There are two ways to update sensors:

- You can update sensors on selected workloads through the console. You can select up to 10,000 sensors to update at one time. After you initiate sensor updates, the selected sensors receive the message to update the next time that they check in with the Carbon Black Cloud backend. The system allows up to 500 concurrent updates. When an individual sensor completes its update process, a new sensor is signaled to start its update.
- You can reinstall the sensors.

### Procedure

- 1 Sign into the Carbon Black Cloud console.
- 2 On the navigation bar, click **Inventory** and then click **Workloads**.
- 3 Search for and select the sensors to update.
- 4 Click **Take Action** and then click **Update Sensors**.
- 5 Confirm the number of sensors to update.
- 6 Select the sensor version from the **Version** dropdown menu.
- 7 Click **Update**

## Update Linux Sensors on Workloads through the Command Line

You can update Linux sensors through the command line.

**Procedure**

- 1 Sign into the workload.
- 2 Unpack the agent tar ball into: `*/var/opt/carbonblack/psc/pkgsg/upgrade_staging/*`. If you have not previously updated the sensor, this folder does not exist and you must create it.
- 3 Run the update script from the `/var/opt/carbonblack/psc/pkgsg/upgrade_staging` location:

RPM:

```
$rpm -U cb-psc-sensor-xxx.rpm
```

---

**Note** For the RHEL sensors kit, you must specify the rpm package that corresponds to the distro version that you are installing.

---

el6 --> centos/rhel/oracle 6.0-6.x

el7 --> centos/rhel/oracle 7.0-7.x

el8 --> centos/rhel/oracle 8.0-8.x

DEB:

```
$dpkg --force-confold -i cb-psc-sensor-xxx.deb
```

- 4 Verify the following:
  - Agent is updated - `/opt/carbonblack/psc/bin/cbagentd -v` to make sure that the agent matches the version you installed.
  - Kernel or BPF module is loaded
    - Kernel module: Run the following command and verify that there is a 1 in the right column of the output. This shows that the kernel module is loaded and enabled. Other versions of the kernel might display as disabled; this is acceptable.  
  
 Command: `lsmod | grep event_collector`  
  
 Sample output: `event_collector_2_x_yyyyyy zzzzz 1`
    - BPF module: Run the following command and verify that the grep returns a single result with the command `event_collector`.  
  
 Command: `ps -e | grep event_collector`  
  
 Sample output: `85150 ? 00:00:05 event_collector`
  - Check agent-blade details on the Workloads page in the server console:
    - Updated agent details are displayed
    - Agent checks in with the server at regular intervals

## Uninstall Linux Sensors from Workloads

Use this procedure to use the command line to uninstall Linux sensors from Workloads.

---

**Note** After you run the following command, the sensor remains listed in the **Registered Devices** list on the **Workloads** page in the console until you click **Take Action > Uninstall**.

Run the following command from the location where the installer kit was unpacked:

- For CentOS, RHEL, SUSE or Amazon Linux: `$ sudo rpm -e cb-psc-sensor`
- For Ubuntu: `$ sudo dpkg --purge cb-psc-sensor`

---

### Results

After you uninstall a sensor, it persists on the **Workloads** page as a deregistered sensor until you delete it. See [Delete Deregistered Sensors from Workloads](#)

## Uninstall Windows Sensors from Workloads

You can uninstall Windows sensors via the Carbon Black Cloud console.

---

**Note** You cannot uninstall Linux sensors via the Carbon Black Cloud console. You must uninstall Linux sensors by using the command line as explained in [Uninstall Linux Sensors from Workloads](#).

---

### Procedure

- 1 Sign in to the Carbon Black Cloud console.
- 2 On the navigation bar, click **Inventory** and then click **Workloads**.
- 3 On the **Enabled** tab, select the sensors to uninstall.
- 4 Click **Take Action** and then click **Uninstall**. You are prompted to confirm the action.

### Results

After you uninstall a sensor, it persists on the **Workloads** page as a deregistered sensor until you delete it. See [Delete Deregistered Sensors from Workloads](#)

## Delete Deregistered Sensors from Workloads

You can delete deregistered sensors from Workloads manually or automatically.

### Procedure

- ◆ To delete deregistered sensors from Workloads, use one of the following procedures.
  - **To manually delete deregistered sensors from workloads:**
    - a Sign in to the Carbon Black Cloud console.
    - b On the navigation bar, click **Inventory** and then click **Workloads**.

- c On the **Enabled** tab, filter the list of sensors to show only devices that have deregistered sensors.
  - d Select the sensors to delete.
  - e Click **Take Action** and then click **Delete deregistered assets**. You are prompted to confirm the deletion.
- **To automatically delete deregistered sensors from workloads:**
- a Sign in to the Carbon Black Cloud console.
  - b On the navigation bar, click **Inventory** and then click **Workloads..**
  - c Click **Sensor Options** and then click **Manage Sensor Settings**.
  - d Select **Delete sensors that have been deregistered for** and set the time frame. Click **Save**.

# Installing Sensors on AWS Workloads

# 10

After you have onboarded your AWS account into Carbon Black Cloud and have associated all the AWS workloads (EC2 instances) with this account, you can install sensors on the running EC2 instances. To install the Carbon Black sensor, you can use an installation script or the Carbon Black Cloud console.

Read the following topics next:

- [Install Sensors on AWS Workloads by Using Sensor Installation Scripts](#)
- [Install Sensors on AWS Workloads by Using the Console](#)

## Install Sensors on AWS Workloads by Using Sensor Installation Scripts

As a cloud security admin, you can secure your AWS workloads (EC2 instances) at the time of rollout through sensor installation scripts for the AWS Userdata, Ansible, Chef, or Puppet configuration management tools.

You can log in to the EC2 instance and run the sensor installation script commands directly into that instance, but it is a time consuming process. For efficiency, use the Carbon Black Cloud console to download the customized sensor install script and install it as part of the instance initialization.

### Procedure

- 1 On the navigation bar, click **Inventory > AWS**.
- 2 In the **AWS Workloads** page, click the **Sensor Options** drop-down menu and select **Download sensor install scripts**.

The **Download Sensor Install Scripts** window displays.

- 3 Locate the OS version for your instance and use the **Sensor Version** drop-down menu to select the related sensor version to install.

These scripts are customized with pre-populated Org Keys and selected platform details.

- 4 Click **Download Scripts**.

## 5 Unzip the downloaded package.

The sensor installation folders are available for each of the configuration management tools.

If you select the `aws-userdata` folder, it contains one script for Unix-based platforms and one PowerShell script for Windows.

## 6 Use the script that is relevant to your configuration management tool.

The following steps show how to create an EC2 instance with a Userdata script running as a part of the instance initialization.

- a Click **Launch instances** in the AWS Management Console, select an IAM template, and select an instance type.
- b Locate the **Step 3: Configure Instance Details > Advanced Details > User data** option and upload the `aws-userdata` script **As file**.
- c To tag your instance, go to **Step 5: Add Tags** and define the key-value pairs.

For example:

Key	Value
Name	latestSensorInstalled
Priority	P2

- d Click **Launch > Launch Instance**.

The sensor installation starts as part of the instance initialization.

## 7 Optional: Create Auto Scaling Groups using the same `aws-userdata` script for easier sensor installations in frequently used images.

### Results

After the sensor installs, the instance displays on the **Enabled** tab.

## Install Sensors on AWS Workloads by Using the Console

Use the following procedure to install sensors on your eligible AWS workloads (EC2 instances) through the Carbon Black Cloud console.

You can install sensors on eligible EC2 instances through the Carbon Black Cloud console. The AWS Systems Manager's `sendCommand` API is used to run the sensor installation script remotely on the AWS EC2 instances.

## Prerequisites

- To prepare your onboarded AWS account for installing sensors on your running EC2 instances, you must create additional resources on the AWS account.
  - Enable the AWS services by running the event stream script in the AWS CLI of the AWS account. For information about enabling the event stream, see the User Guide.
  - The EC2 instance upon which you install the sensor must be an SSM-managed node. A managed node is any machine that is configured for AWS Systems Manager. For more information, see [Systems Manager prerequisites](#).
- Install sensors on eligible EC2 instances. You can view which EC2 instances in your onboarded AWS accounts are available for sensor installation by referring to the **Not Enabled** tab.

Eligibility column	Description
Eligible	The eligible EC2 instances run the supported operating system (OS) and have SSM connectivity. You can install sensors on them.
Not eligible	<p>The EC2 instances that are not eligible for sensor installation are either without AWS SSM connectivity or are running an unsupported OS. To minimize your installation effort, Carbon Black Cloud creates an AWS SSM document in your AWS account with a preconfigured sensor installation script.</p> <ul style="list-style-type: none"> <li>■ You must run the AWS services setup script (event stream script), which creates the SSM document and other required resources on your AWS account.</li> <li>■ The EC2 instance must be an SSM-managed instance. It must be running an SSM agent and have the appropriate IAM instance role.</li> <li>■ To proceed with the sensor installation, the EC2 instance must be in a running state.</li> </ul>
Not supported	The Carbon Black Cloud sensor does not support the OS or the OS version. You must upgrade to the supported OS or OS version as per the system requirements.

## Procedure

- 1 On the left navigation bar, click **Inventory > AWS**.
- 2 Click the **Not Enabled** tab and select eligible EC2 instances.
- 3 Select **Install sensors** from the **Take Action** drop-down menu.
- 4 Select the sensor version to install.
- 5 Click **Install**.

A **Sensor installation submitted** notification displays. The install status for the instance changes to **In Progress**.

It takes up to five minutes for the installation to complete.



## Results

After the sensor installs, it displays on the **Enabled** tab.

# Setting up the Container Security Environment

# 11

This section describes how to prepare your environment for securing Kubernetes with Carbon Black Container.

Follow these basic steps to set up your container environment: for Carbon Black Container security:

- 1 Make sure your Kubernetes environment meets the supported Operating Environment Requirements for the Kubernetes Sensor. See [Kubernetes Sensor OER](#).
- 2 Add users and assign user roles so that the appropriate people can install, configure, and manage Carbon Black Container security features.
- 3 Optionally review and manually install the Operator and Agent. (This deployment happens automatically during Step 4.)
- 4 Add Kubernetes clusters to the Carbon Black Cloud console and install a Kubernetes sensor into each Kubernetes cluster that you want to protect.
- 5 Optionally install Containerized Sensors for non-Kubernetes environments.
- 6 Download, add, and configure a CLI client to scan local images.

You will then be ready to create scopes and policies to manage your containers.

Read the following topics next:

- [Roles and Users for Containers](#)
- [Adding Clusters and Installing Kubernetes Sensors](#)
- [Check the Kubernetes Sensor Status and Health](#)
- [Installing a Containerized Sensor](#)
- [Setting up CLI Client for Image Scanning](#)
- [Carbon Black Container Operator Technical Reference](#)

## Roles and Users for Containers

You can add users and assign appropriate roles for their work in Containers.

By setting up and managing users and their roles, you give the users access to the Carbon Black Cloud console and Containers security functionality.

---

**Note** This section specifically describes setting up users and user roles for Containers. For information about managing all Carbon Black Cloud users and their roles, see [Managing Users](#) and [User Roles](#).

---

## Using and Creating Roles for Containers

Every Carbon Black Cloud console user is assigned to a role that defines permissions. The role is assigned when you create the new user account; this assignment can be modified at any time.

Carbon Black Cloud includes four Kubernetes-related pre-defined roles that you can assign to users (or you can create custom roles: see [Add a Container Role](#)).

- Kubernetes SecOps View Only
- Kubernetes SecOps
- Kubernetes DevOps
- Kubernetes Security Developer

**Kubernetes Security DevOps** are responsible for the Kubernetes workload posture.

Responsibilities include setting up clusters, scopes, and security policies for Kubernetes workloads. Security DevOps can monitor the health of the Kubernetes environment, investigate workloads and violations, and take appropriate actions.

## Role Definitions and Recommendations

The following table describes Carbon Black Cloud permissions and recommendations for user roles for Containers.

Table 11-1. User Roles/Permissions Matrix - by Role

Role	Description	Permissions	Workflow
<b>Kubernetes SecOps View Only</b>	Monitors environment. Cannot take any actions.	<ul style="list-style-type: none"> <li>■ View Notifications</li> <li>■ View Kubernetes Security</li> <li>■ View Images</li> <li>■ View Workloads</li> </ul>	N/A
<b>Kubernetes SecOps</b>	<p>Assess and control the workload's attack surface from build to runtime. Focus on detecting, responding to, and preventing container runtime threads —can quickly detect runtime threads.</p> <p>This role is appropriate for SOC Analysts.</p>	<ul style="list-style-type: none"> <li>■ Dismiss Alerts</li> <li>■ View and Manage Alerts, Notes, and Tags</li> <li>■ View and Manage Notifications</li> <li>■ View and Manage API Keys</li> <li>■ Manage Users</li> <li>■ View and Manage Kubernetes Security</li> <li>■ View Images</li> <li>■ Manage Image Exceptions</li> </ul>	<ol style="list-style-type: none"> <li>1 Monitor and analyze Containers. See <a href="#">Monitoring and Analyzing Containers</a>.</li> <li>2 Take action and remediate security issues. See <a href="#">Investigating and Remediating Container Security Issues</a>.</li> <li>3 Triage alerts. See <a href="#">Triaging Kubernetes Alerts</a>.</li> </ol>

Table 11-1. User Roles/Permissions Matrix - by Role (continued)

Role	Description	Permissions	Workflow
<b>Kubernetes DevOps</b>	<p>Assess and control the workload's attack surface from build to runtime. Troubleshooting and remediation of security issues.</p> <p>Responsible for determining the Kubernetes workload posture. Responsibilities include setting up Kubernetes policies, scopes, and clusters in the Carbon Black Cloud console. Security DevOps can monitor the health of the Kubernetes environment, investigate workloads and violations, and take appropriate actions.</p>	<ul style="list-style-type: none"> <li>■ Dismiss Alerts</li> <li>■ View and Manage Notifications</li> <li>■ View and Manage API Keys</li> <li>■ Manage Users</li> <li>■ View and Manage Kubernetes Security</li> <li>■ View Images</li> <li>■ Manage Image Exceptions</li> </ul>	<ol style="list-style-type: none"> <li>1 Set up user roles and manage users. See <a href="#">Roles and Users for Containers</a>.</li> <li>2 Add clusters to the console and install Kubernetes Sensors. See <a href="#">Adding Clusters and Installing Kubernetes Sensors</a>.</li> <li>3 Configure Containers. See <a href="#">Configuring Container Security</a>.</li> <li>4 Monitor and analyze Containers. See <a href="#">Monitoring and Analyzing Containers</a>.</li> <li>5 Triage alerts. See <a href="#">Triaging Kubernetes Alerts</a>.</li> <li>6 Take action and remediate security issues. See <a href="#">Investigating and Remediating Container Security Issues</a>.</li> </ol>
<b>Kubernetes Security Developer</b>	<p>Inspects a single container for security posture and compliance.</p>	<ul style="list-style-type: none"> <li>■ View and Manage Kubernetes Security</li> <li>■ View Images</li> <li>■ Manage Image Exceptions</li> </ul>	<ol style="list-style-type: none"> <li>1 Monitor and analyze Kubernetes workloads. See <a href="#">Monitoring Kubernetes Workloads</a>.</li> <li>2 Triage alerts. See <a href="#">Triaging Kubernetes Alerts</a>.</li> </ol>

## Add a Container Role

To add a new role for Containers work, perform the following procedure.

### Procedure

- 1 On the left navigation pane, click **Settings > Roles**.
- 2 In the upper right of the page, click **Add Role**.
- 3 Enter a unique name and description for the new role. Special characters are not allowed.


- 4 Optionally, select a role from the **Copy permissions from** dropdown to use an existing role as a template. This allows you to add and remove permissions from an existing set of role permissions.



See [Using and Creating Roles for Containers](#) for more information about Container role permissions.

- 6 Click **Save**.

---

**Tip** Click the **Duplicate**  icon next to the role in the table to make a copy of that role. Use copied roles to easily make minor adjustments for new roles.

---

#### What to do next

- Use the icons to the right of your new role in the table to duplicate, edit, export, or delete the role.
- [Create a User Account for Containers](#)

## Create a User Account for Containers

To create a new user account for Containers work, perform the following procedure.

#### Prerequisites

We recommend that you study the available user roles before you create a user for Containers. Users are granted specific permissions based on their assigned role. Pre-defined user roles are available for selection. If existing roles do not suffice for your environment, you can create custom roles. See [Using and Creating Roles for Containers](#).

#### Procedure

- 1 On the left navigation pane, click **Settings > Users**.
- 2 In the upper right of the page, click **Add User**.
- 3 Enter the details for the new user including name, email address, and role.
- 4 Click **Save**.

#### Results

- An email is sent to the input email address. The email prompts the user to log in and create a password.
- Added usernames display after the users have confirmed their login credentials.

## Adding Clusters and Installing Kubernetes Sensors

To enable Carbon Black Container, you must install one Carbon Black Kubernetes Sensor for each Kubernetes cluster. To do so, you must add a cluster to the console.



A Kubernetes extension called *Operator* and a custom resource definition are used to deploy the Kubernetes Sensor. Operators consist of set of controllers that deploy and manage user-defined components and report on their health. You define the components with a custom resource definition.

The Carbon Black operator deploys the Kubernetes Sensor inside the cluster and manages its lifecycle. The data in the custom resource file defines which features are enabled for the sensor. The essential steps of the sensor deployment procedure are:

- Setup and install the Carbon Black Operator
- Deploy the Carbon Black Agent on top of the Operator.
- Allow access to the Carbon Black Cloud console
- Configure the Kubernetes Sensor and scanner

---

### Note

- The **Add Cluster** wizard walks you through these steps in [Add a Cluster and Install the Kubernetes Sensor](#).
  - A technical overview and separate deployment instructions for the Operator and Agent are included in [Carbon Black Container Operator Technical Reference](#). You generally do not need to separately install these components, but the background information and deployment content is added here for your convenience.
- 

## Add a Cluster and Install the Kubernetes Sensor

To add a cluster to the Carbon Black Cloud console and install the Kubernetes Sensor into that cluster, perform the following procedure.

### Prerequisites

Before you begin, open both the Carbon Black Cloud console and a terminal window.

### Procedure

- 1 On the left navigation pane of the console, do one of the following depending on your system configuration and role:
  - If you have the Kubernetes Security DevOps or SecOps role and your system has the Containers Security feature only, click **Inventory > Clusters**.
  - If you have any other role and your system has Container security and other Carbon Black Cloud features, click **Inventory > Kubernetes > Clusters**.
- 2 In the upper right of the page, click **Add Cluster**.

### 3 Add the **Cluster Detail** information.

**Add Cluster** ✕

1 — 2 — 3 — 4

CLUSTER DETAIL      AUTHENTICATION      SENSOR      FINISH SETUP

**CLUSTER DETAIL** [Cluster setup guide](#)

\* Cluster name  Cluster group

> Cluster labels

**Next** **Cancel**

- a Enter a unique cluster name using lowercase letters, numbers, and hyphens. The name cannot contain a colon (:) symbol.
- b Type or select an existing cluster group to help specify resources in scopes and policies. The cluster group is also used for observing the network activity map of your clusters.

When no group is provided, the cluster is added to the *default* group.

- c Optionally add cluster labels. A label consists of a key and a value. You can add multiple labels.

#### 4 Click **Next**.

#### 5 Provide a dedicated API key to establish the communication between your Kubernetes cluster and the console.

- Click **Generate a new API key** and enter an API key name that is unique to your Carbon Black Cloud organization.

Add Cluster
✕

**AUTHENTICATION** [Cluster setup guide](#)

New API Key
Existing API Key

⚠ Record this API Secret Key in a secure location for later reference. This information will not be stored in Carbon Black.

API ID	API Secret Key
VYUIRF76AL	HZPFPMZEEK3S5MGMIM1M3CWF

Next
Back
Cancel

- Click **Use existing API key** and select an existing API key.

**Important** Do not reuse keys between clusters. Use a separate Carbon Black Cloud API key for each cluster.

- 6 Select the version of the Kubernetes Sensor to install on your cluster. The latest sensor version is set by default.
- 7 Under **Advanced Settings**, optionally set up a proxy server or a private container registry.
  - **Proxy server** can include a proxy URL or remain empty. The field is empty by default.
  - **Private container registry** can include a private registry URL or remain empty. The field is empty by default. For important information about using a private container registry, see [Private Container Registry](#).

Add Cluster
✕

### SENSOR

[Cluster setup guide](#)

**\* Sensor version**

Version main (latest) ▾

Features include:

- K8s security posture management for workload risk
- Automated scans of running images for vulnerability, malware and secrets
- Runtime workload and container protection with threat detection and prevention

**Advanced Settings**

Proxy server (HTTP/HTTPS)

Private container registry [?](#)

Next
Back
Cancel

## Note

- 8 On the **Finish Setup** page, select **Kubectl** or **Helm Charts**.
- 9 Copy and run each command in sequence into your terminal:

Add Cluster
✕

### FINISH SETUP

[Cluster setup guide](#)

Deployment tool Kubectl Helm Charts

Run these commands in your terminal and click **Done**

1 — Detect K8s version and install appropriate operator Bash ▾

```
curl -s https://setup.dev.containers.carbonblack.io/develop/operator-apply.sh | bash
```

2 — Create secret for this cluster to connect with Carbon Black

```
kubectl create secret generic cbcontainers-access-token --namespace cbcontainers-dataplane --from-literal=accessToken=HZPFPMZEEK355HGMI1H3CWF/VYU1RF76AL
kubectl create secret generic cbcontainers-company-code --namespace cbcontainers-dataplane --from-literal=companyCode=47H1CAW7HR45WR4S32G03R14U1G5Y
```

3 — Apply cluster configuration and install sensor [View YAML details](#)

```
kubectl apply -f https://setup.dev.containers.carbonblack.io/cr-e841ab46-733d-4ac3-b1ed-890aeffa2e49
```

Done
Back
Cancel

- 10 In the console, click **Done**.
- 11 Refresh the console browser page to view the new cluster.

The cluster status will be **Pending install**.

It takes up to 5 minutes for the cluster to stabilize during the initial setup. During this time, the status might display an error. Wait three to five minutes after submitting the install request to verify the correct status.

## Results

After completing the setup procedure successfully, the status changes to **Running**.

## What to do next

- 1 [Check the Kubernetes Sensor Status and Health](#)
- 2 [Download a CLI Client](#)
- 3 [Add and Configure a CLI Client](#)

## Private Container Registry

You can use a private container registry to reduce traffic costs or to provide application teams with a source of verified container images.

To use Carbon Black Container security through a private container registry, Carbon Black Container provides you with a script to simplify the task of mapping, downloading, and storing the container images.

Before you deploy the sensor:

- Upload and tag the container images to your site
- Configure the registry

For configuration information and instructions, see <https://github.com/octarinesec/octarine-operator/tree/main/charts> (external link).

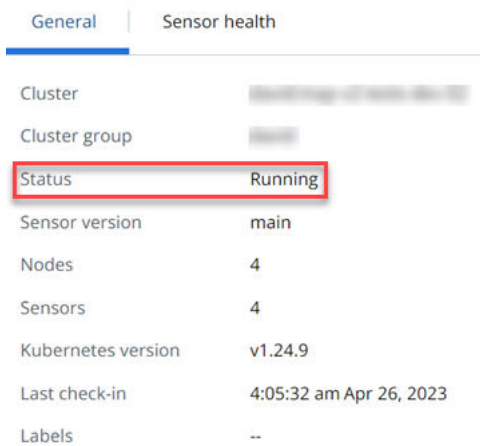
## Check the Kubernetes Sensor Status and Health

To view the status of a Kubernetes Sensor in a cluster, perform the following procedure.

### Procedure

- 1 On the left navigation pane of the console, do one of the following depending on your system configuration and role:
  - If you have the Kubernetes Security DevOps or SecOps role and your system has the Containers Security feature only, click **Inventory > Clusters**.
  - If you have any other role and your system has Container security and other Carbon Black Cloud features, click **Inventory > Kubernetes > Clusters**.

- 2 On the Kubernetes Clusters page, click the **Clusters** tab and then click the **General** tab.
- 3 In the left pane, you can filter the list of displayed clusters by:
  - Status
  - Sensor Version
  - Operator Version
  - Cluster Label Key
  - Cluster Label Value
- 4 In the Clusters panel, you can search for a cluster, and you can select a displayed cluster name to view sensor health data.
- 5 Select the cluster and view **Status** in the right panel.



**Table 11-2. Kubernetes Sensor Status**

Status	Description
Critical	No activity has been detected from any cluster components for more than 24 hours
Error	A critical component is down or the status cannot be detected
Pending install	Cluster setup is in progress
Running	All components are up and running without errors
Warning	A non-critical component is down, or the status cannot be detected

- 6 Click the **Sensor health** tab.

To expand any entry, click the arrow  icon on the left. For example:

General | **Sensor health**

---

**Deployment**

- ✓ cbcontainers-runtime-resolver
  - ✓ cbcontainers-runtime-resolver-b9f7674c9-2dnwb
- ✓ cbcontainers-hardening-enforcer
  - ✓ cbcontainers-hardening-enforcer-6f9585dfd4-mq8l5
- ✓ cbcontainers-hardening-state-reporter
  - ✓ cbcontainers-hardening-state-reporter-77fb77cf7c-4ksfb
- ✓ cbcontainers-monitor
  - ✓ cbcontainers-monitor-64b49fc687-lvcc7

**Operator**

- ✓ cbcontainers-operator
  - ✓ cbcontainers-operator-555f9fb769-jdbrj

**Webhook**

- ✓ cbcontainers-hardening-enforcer (validating)
- ✓ cbcontainers-hardening-enforcer (mutating)

---

**DaemonSet**

- ✓ cbcontainers-node-agent
  - ✓ cbcontainers-node-agent-c84v4
  - ✓ cbcontainers-node-agent-dzmp9
  - ✓ cbcontainers-node-agent-hcm8q
  - ✓ cbcontainers-node-agent-mdgf9

## Installing a Containerized Sensor

The Containerized Sensor is an agent that includes both Carbon Black EDR and Image Scanning capabilities. It is used for non-Kubernetes container environments.

The sensor runs as a container, and provides container context to the regular Carbon Black EDR capabilities. This context is known as Cloud Native Detection and Response (CNDR). The sensor scans the containers on the node for vulnerabilities, malware, and secrets.

## Required Dependencies

Before you install the Containerized Sensor, make sure that the following requirements are satisfied:

- The sensor is installed as a privileged container on the host network. The installing user must have permissions that allow the sensor to be installed as a privileged container on the host network, as well as the permissions to mount root folders and unix sockets to the container.
- Carbon Black Container
- Carbon Black EDR
- 2GB of memory.
- An API key that has these settings:

Setting	Description
Access Level type	Set to Custom and select KUBERNETES_SECURITY_DATAPLANE.
Access Token	Record the provided API ID and API Secret Key in the format of API Secret Key/API ID, and use it as the access token.

See [Create and Manage an API key](#) for more information.

## Set up a Containerized Sensor

Before you can install the containerized sensor, you must set it up.

The Containerized Sensor includes both Carbon Black EDR and Image Scanning capabilities. It is used for non-Kubernetes container environments.

### Prerequisites

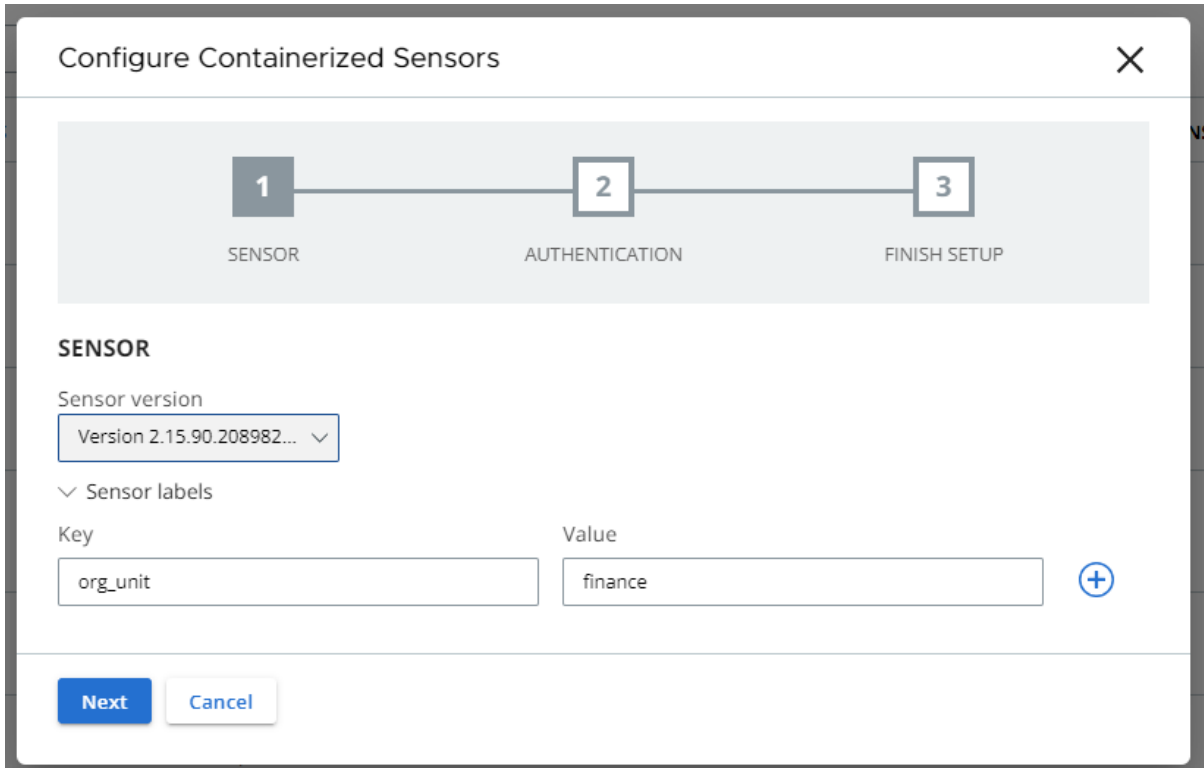
See [Installing a Containerized Sensor](#).

### Procedure

- 1 On the left navigation pane, click **Inventory > Endpoints**.
- 2 In the upper right corner, click **Sensor Options** and select **Configure containerized sensors**.
- 3 Select the sensor version. The sensor version is in the format: `a.b.c.d-e.f`, where `a.b.c.d` pertains to the original Linux sensor version and `e.f` is the version of the image scanning agent. For example: 2.14.0.12349-1.3.



- 4 Click **Show** next to **Sensor Labels** and enter a key and value; for example, `org_unit` and `finance`.



- 5 Click **Next**.

- On the Authentication page, click **Use existing API key** and select the pre-existing API key from the drop down menu.

**Configure Containerized Sensors**

1  SENSOR — 2 AUTHENTICATION — 3 FINISH SETUP

**AUTHENTICATION** [Cluster setup guide](#)

API KEY NAME	API ID	ACCESS LEVEL TYPE
hardening api key	XXXXXXXXXX	Kubernetes DevOps
knox-k8s	XXXXXXXXXX	KUBERNETES_SECURITY_DATAPLANE
knox-cli	XXXXXXXXXX	Container Image CLI tool
ben-gke-cndr	XXXXXXXXXX	KUBERNETES_SECURITY_DATAPLANE

Showing 1-20 of 54      Items per page       Jump to page       < 1 2 3 >

- Click **Next**.
- On the Finish Setup page, download the configuration settings as either a JSON or YAML file.
- Click **Done** to finish setup and close the setup wizard.

#### What to do next

[Install a Containerized Sensor](#)

## Install a Containerized Sensor

After you set up a Containerized Sensor, you can install it.

The Containerized Sensor includes both Carbon Black EDR and Image Scanning capabilities. It is used for non-Kubernetes container environments.

## Prerequisites

See:

- [Installing a Containerized Sensor](#)
- [Set up a Containerized Sensor](#)

## Procedure

- 1 Run the container image `cbartifactory/cb-containers-sensor:{sensor-version}` together with your selected sensor version.
- 2 Attach these volume mounts to the container:
  - a Container runtime unix socket. Currently only supports docker - `/var/run/docker.sock:/var/run/docker.sock:ro`
  - b Host root path - `:/var/opt/root`
  - c Host hostname - `/etc/hostname:/etc/hostname`
  - d Host boot folder - `/boot:/boot`
  - e Host operating system identification data - `/etc/os-release:/etc/os-release`
  - f Carbon Black Metadata Mount - `/var/opt/carbonblack:/var/opt/carbonblack`
- 3 During sensor setup, the setup wizard provided these environment variables:

Environment Variable	Description
CBC_ACCOUNT	Your Carbon Black Organization Key.
CBC_ACCESS_TOKEN	API key with appropriate permissions.
CB_COMPANY_CODES	Your Carbon Black Company Codes.
CBC_API_HOST	Your Carbon Black environment API host.
HOST_ROOT_PATH	The mounted location of the root path.
CONTAINER_REPORTER_HOSTNAME_FILEPATH	The mounted location of the hostname path.
CONTAINER_REPORTER_LABELS	Key Value labels used to identify the host. For example: <code>key1=value1, key2=value2.</code>

- 4 (Optional) You can configure the sensor image with additional advanced environment variables:

Environment Variable	Description
CONTAINER_REPORTER_HOST	Value you can set as the container's hostname. You can set the hostname instead of <code>CONTAINER_REPORTER_HOSTNAME_FILEPATH</code> . If both values are set, this variable takes priority. If this value is set, you can delete the hostname volume mount.
ENDPOINT	Value of the host's container-runtime endpoint Unix socket. This value is set to <code>docker's /var/run/docker.sock</code> by default.  <b>Note</b> Currently only the docker container runtime is supported.
CONTAINER_RUNTIME	The name of the host container runtime. This value is set to <code>docker-daemon</code> by default.  <b>Note</b> Currently only docker container runtime is supported.
SCANNER_CLI_FLAGS_ENABLE_SECRET_DETECTION	Boolean flag to enable/disable container scanning secret detection. This value is set to <code>true</code> (enabled) by default.
SCANNER_CLI_FLAGS_IGNORE_BUILD_IN_REGEX	Boolean flag to determine whether to ignore filenames' built-in regexes and scan every file for secrets. This value is set to <code>false</code> by default.
SCANNER_CLI_FLAGS_SCAN_BASE_LAYERS	Boolean flag used to decide whether to scan the image base layers for secrets. This value is set to <code>false</code> by default.
SCANNER_CLI_FLAGS_SKIP_DIRS_OR_FILES	List of files and directories (in Regexes) to ignore when detecting secrets. This value is set to <code>empty</code> by default.
SCANNER_CLI_FLAGS_CONCURRENT_FILE_LIMIT	Number of files to scan at one time for secrets. This value is set to <code>200</code> by default. You can increase or decrease this number to determine the speed of the scan. If the number is higher, the service requires more resources (memory and CPU).
DISABLE_SCANNER	Boolean flag to disable the container scanner capability. This value is set to <code>false</code> by default.
DISABLE_SENSOR	Boolean flag to disable CNDR capability. This value is set to <code>false</code> by default.

- 5 Install the sensor:

- Using a docker compose file: [Install a Containerized Sensor on a Docker Client.](#)
- On an AWS ECS cluster: [Install a Containerized Sensor on an ECS Cluster.](#)

## Install a Containerized Sensor on a Docker Client

You can run the Carbon Black Containerized Sensor on a host that has the Docker client to detect and enforce EDR and Container Scanning capabilities. Additionally, the Containerized Sensor can detect vulnerabilities, malware, and secrets in the runtime in a Docker container.

### Prerequisites

You must have the following products and information:

- Linux Host with docker installed
- Carbon Black Cloud Container
- Carbon Black EDR
- API key with appropriate permissions
- See:
  - [Installing a Containerized Sensor](#)
  - [Set up a Containerized Sensor](#)
  - [Install a Containerized Sensor](#)

### Procedure

- 1 Add the environment variables you received from the setup wizard you ran in [Set up a Containerized Sensor](#) to the `docker-compose.yaml` file.

```
version: "3.3"
services:
  sensor:
    pid:host
    network_mode: host
    image: docker.io/cbartifactory/cb-containers-sensor:{sensor-version}
    privileged: true
    environment:
      # fill environment variables here
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock:ro
      - /boot:/boot
      - /var/opt/carbonblack:/var/opt/carbonblack
      - /etc/os-release:/etc/os-release
      - /:/var/opt/root
      - /etc/hostname:/etc/hostname
```

- 2 Deploy the agent container by running the following command:

```
docker-compose up -d
```

## Install a Containerized Sensor on an ECS Cluster

You can run the Carbon Black Containerized Sensor on an ECS cluster to detect and enforce EDR and Container Scanning capabilities. Additionally, the Containerized Sensor can detect vulnerabilities, malware, and secrets in the runtime in an ECS Cluster.

### Prerequisites

You must have the following products and information:

- ECS Cluster
- Carbon Black Carbon Black Cloud Container
- Carbon Black EDR
- API key with appropriate permissions
- See:
  - [Installing a Containerized Sensor](#)
  - [Set up a Containerized Sensor](#)
  - [Install a Containerized Sensor](#)

### Procedure

- 1 Register the agent task definition and update it with the relevant environment variables from the setup wizard you ran in [Set up a Containerized Sensor](#):

```
{
  "family": "cbcontainers-daemon",
  "pidMode": "host",
  "networkMode": "bridge",
  "executionRoleArn": "<arn role with ec2 deployment permissions>",
  "containerDefinitions":
  [
    {
      "name": "host-container-scanner",
      "image": "docker.io/cbartifactory/cb-containers-sensor:{sensor-version} >",
      "cpu": 512,
      "memory": 1024,
      "privileged": true,
      "environment":
      [
        // fill environment variables list here
      ],
      "mountPoints":
      [
        {
          "sourceVolume": "dockersock",
          "containerPath": "/var/run/docker.sock"
        },
        {
          "sourceVolume": "hostname",
```

```

        "containerPath": "/etc/hostname"
    },
    {
        "sourceVolume": "boot",
        "containerPath": "/boot"
    },
    {
        "sourceVolume": "cb-data-dir",
        "containerPath": "/var/opt/carbonblack"
    },
    {
        "sourceVolume": "os-release",
        "containerPath": "/etc/os-release"
    },
    {
        "sourceVolume": "root",
        "containerPath": "/var/opt/root"
    }
],
"healthCheck": {
    "command": [
        "CMD-SHELL",
        "cat /tmp/ready || exit 1"
    ],
    "interval": 60,
    "timeout": 15,
    "retries": 3,
    "startPeriod": 60
}
],
"volumes":
[
    {
        "name": "dockersock",
        "host":
        {
            "sourcePath": "/var/run/docker.sock"
        }
    },
    {
        "name": "hostname",
        "host":
        {
            "sourcePath": "/etc/hostname"
        }
    },
    {
        "name": "boot",
        "host":
        {
            "sourcePath": "/boot"
        }
    },
    {

```

```

    "name": "cb-data-dir",
    "host":
    {
      "sourcePath": "/var/opt/carbonblack"
    }
  },
  {
    "name": "os-release",
    "host":
    {
      "sourcePath": "/etc/os-release"
    }
  },
  {
    "name": "root",
    "host":
    {
      "sourcePath": "/"
    }
  }
],
"requiresCompatibilities":
[
  "EC2"
]
}

```

- 2 Register the agent task definition by using the AWS ECS user interface or the AWS CLI:

```

aws ecs register-task-definition --cli-input-json file://cbcontainers-daemon.json --region
<region-to-apply-at>

```

- 3 (Optional) To write agent logs to AWS CloudWatch, add the `logConfiguration` section inside the container definition element in the task definition:

```

{
  "logConfiguration":
  {
    "logDriver": "awslogs",
    "options":
    {
      "awslogs-group": "cbcontainers-agent",
      "awslogs-region": "<region>",
      "awslogs-stream-prefix": "cbcontainers-agent"
    }
  }
}

```

Add the `cbcontainers-agent` `awslogs-group` and add the `logs:CreateLogStream` and `logs:PutLogEvents` Actions to the ECS Role Policy.



- 4 To run the agent, create a service to run the task: `cbcontainers-daemon-svc`:

```
aws ecs create-service \
  --region <region-to-apply-at> \
  --cluster <your-cluster-name> \
  --service-name cbcontainers-daemon-svc \
  --launch-type EC2 \
  --task-definition cbcontainers-daemon \
  --scheduling-strategy DAEMON
```

- 5 To run the agent as an ECS task, add a role with the following permissions in the `executionRoleArn` section of the task definition:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeTags",
        "ecs:CreateCluster",
        "ecs:DeregisterContainerInstance",
        "ecs:DiscoverPollEndpoint",
        "ecs:Poll",
        "ecs:RegisterContainerInstance",
        "ecs:StartTelemetrySession",
        "ecs:UpdateContainerInstancesState",
        "ecs:Submit*",
        "ecr:GetAuthorizationToken",
        "ecr:BatchCheckLayerAvailability",
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage"
      ],
      "Resource": "*"
    }
  ]
}
```

- 6 (Optional) To write agent logs to AWS CloudWatch, add the Actions `logs:CreateLogStream` and `logs:PutLogEvents` to the Actions list.

---

**Note** To write the containers logs, the policy must have cloudwatch access and permissions to pull images and run ECS tasks.

---

## Validate the Container Image Signature

To verify the security and integrity of the container image, you can validate the container signature.

During verification, use this public key:

```
-----BEGIN PUBLIC KEY-----
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE1ivoAvFrHG91m01ecsBN1juDOP5
6kGA7G5M0WnOS2zc5qNPQSN1fzwOc/EgEIskERJY/NMmCjq0rcZzzKgfxQ==
-----END PUBLIC KEY-----
```

## Prerequisites

Before you can verify the container image signing, you must download the [cosign tool](#).

## Procedure

- 1 Download the containerized sensor image: `cbartifactory/cb-containers-sensor` using an image management tool, such as `docker`.
- 2 Run the signature verification command using the public key above:

```
cosign verify --key container-signing-key.pub cbartifactory/cb-containers-
sensor:<sensor-version>
```

## Results

An example of a successful verification:

```
Verification for docker.io/cbartifactory/cb-containers-sensor:<sensor-version> --
The following checks were performed on each of these signatures:
- The cosign claims were validated
- Existence of the claims in the transparency log was verified offline
- The signatures were verified against the specified public key
[
  {
    "critical": {
      "identity": {
        "docker-reference": "docker.io/cb/cbartifactory/cb-containers-sensor"
      },
      "image": {
        "docker-manifest-digest":
"sha256:ala0dfe211c0fdbca68f6cb7629e79f3d9775891584daddc8aff5050237911"
      },
      "type": "cosign container image signature"
    },
    "optional": {
      "Bundle": {
        "SignedEntryTimestamp":
"MEUCIBiIc38wiBow7FT09yIlanYEki248tu4kYcJYr3dSwRUkAiEA9R9pK6SnTaTnhPKmK592n0keUGj8mdxTIA1Fc75j7
i4=",
        "Payload": {
          "body":
"eyJhcGlWZXJzaW9uIjoia0dfe211c0fdbca68f6cb7629e79f3d9775891584daddc8aff5050237911"

```

```

KT01XcDFSRT13T1FvMmEwZEJOMGMxVFRCWGJrOVRNbnBqTlhGT1VGR1RUakZtZW5kUF15OUZaMFZKYzJ0RlVrcFpMMDVOY
1VOcWNUQnlZMxA2ZWt0blpuaFJQVDBLTfMwdExTMUZUa1FnVUZWQ1RFbERJRXRGV1MwdExTMHRDZz09In19fX0=",
    "integratedTime": 1699443190,
    "logIndex": 48394752,
    "logID": "c0d23d6ad406973f9559f3ba2d1ca01f84147d8ffc5b8445c224f98b9591801d"
  }
}
}
}
]

```

## Setting up CLI Client for Image Scanning

To include image scanning in your continuous integration script, configure and use the Carbon Black Cloud CLI Client (`cbctl`). This client is available for Linux and macOS.

You can install the CLI client on a Dev/Sec/Ops machine, or you can include it in a CI/CD pipeline — for example, Jenkins or Gitlab. The CLI client requires an Internet connection to Carbon Black Cloud and access to your container registries.

Carbon Black CLI Client performs an image scan for known vulnerabilities and enforces security or compliance rules. The CLI Client performs the following tasks:

- **Vulnerabilities scanning of container images.**

Container images are matched against a known vulnerabilities database. The image details include operating system and non-operating system packages, libraries, licenses, binaries, and metadata. The vulnerabilities scan result is included in the image metadata.

- **Enforcing standards for container images.**

To evaluate policy violations, the image scan results are matched against a specific policy that is configured for the CLI scope. The CLI run fails the build pipeline step if policy violations are detected. The violation of policy rules is added to the image metadata together with image rule exceptions.

- **Enforcing standards for Kubernetes workloads.**

Kubernetes workloads are matched against a Kubernetes hardening policy to evaluate the workload compliance for security risks. By leveraging the information from both image vulnerabilities and workload configuration, a complete picture of the workload risk exposure is available.

The CLI client presents the following interface and command options:

```

$ cbctl
A client CLI for image scanning, and instrumenting Carbon Black services.

Usage:
  cbctl [command]

Available Commands:
  auth          Set auth for cbctl
  completion    generate the autocompletion script for the specified shell
  config        Manage Carbon Black configuration
  help          Help about any command
  image         Commands related to image analysis
  k8s-object    Commands related to k8s-object analysis
  user         Manage cbctl user profiles
  version       Show the cli tool version and build info

Flags:
  -c, --config string          config file (default "/home/slist/.cbctl/.cbctl.yaml")
  --debug string[="/home/slist/.cbctl/debug.log"]  enable debug log (default "/home/slist/.cbctl/debug.log")
  -h, --help                  help for cbctl
  --plain-mode                display ui on plain mode
  -u, --user-profile string    user profile

Use "cbctl [command] --help" for more information about a command.
$

```

## Secrets File Detection

If secret detection is enabled, Carbon Black Cloud detects all text files in an image. Files can be ignored; these are specified by using CLI flags. System files are ignored by default to reduce scan time.

**Note** To enable or disable secrets detection, see [Add a Cluster and Install the Kubernetes Sensor](#).

Table 11-3. CLI Flags

Flag	Description	Default Setting
<code>enableSecretDetection</code>	Indicates whether the scan should scan for secrets	False
<code>skipDirsOrFiles</code>	Files or directories to not scan for secrets	N/A
<code>scanBaseLayers</code>	Indicates whether the scan should scan the base layers for secrets	False
<code>ignoreBuildInRegex</code>	Indicates whether the scan should ignore the build-in regexes of files	False

## Download a CLI Client

Add configured CLI instances to enable local scanning of images, workload vulnerability assessment, and CI integration. The CLI instance scans container images and reports their health to the Carbon Black Cloud console.

## Procedure

- 1 On the left navigation pane of the console, do one of the following depending on your system configuration and role:
  - If you have the Kubernetes Security DevOps or SecOps role and your system has the Containers Security feature only, click **Inventory > Clusters**.
  - If you have any other role and your system has Container security and other Carbon Black Cloud features, click **Inventory > Kubernetes > Clusters**.
- 2 Click the **CLI Config** tab.
- 3 In the upper right of the page, click **Download CLI**.
- 4 Select and download the CLI client for your operating system (macOS or Linux).

### Download CLI

OS	VERSION	DETAILS	ACTION
CLI client (Mac)	v1.9.2	MD5SUM	ca8eb6bdb0f825a2ed4ce329e869f256
		SHA1SUM	295338b3b54bf53bb99beb800aa78d250b80e816
		SHA256SUM	c0d51bbbe7247d91c1f5647f714b6c3683c93887aafef311a3077bf270b883e0
CLI client (Linux)	v1.9.2	MD5SUM	ff954bdec199d856b1bed37c5335059f
		SHA1SUM	f6d151e8d745248c47d05621d5b0c11cac1c21b7
		SHA256SUM	9f3dfe307f02c139c8ef3a22a25807245fe61a11f2094462e2bf57d3dc2a2b20

Close

- 5 Click **Close**.

## What to do next

### Add and Configure a CLI Client

## Add and Configure a CLI Client

To set up a CLI instance for image scanning, perform the following procedure.

Add configured CLI instances to enable local scanning of images, workload vulnerability assessment, and CI integration. The CLI instance scans container images and reports their health to the Carbon Black Cloud console.

## Prerequisites

### Download a CLI Client

Before you begin, open both the Carbon Black Cloud console and a terminal window.

## Procedure

- 1 On the left navigation pane of the console, do one of the following depending on your system configuration and role:
  - If you have the Kubernetes Security DevOps or SecOps role and your system has the Containers Security feature only, click **Inventory > Clusters**.
  - If you have any other role and your system has Container security and other Carbon Black Cloud features, click **Inventory > Kubernetes > Clusters**.
- 2 Click the **CLI Config** tab.
- 3 In the upper right of the page, click **Add CLI**.
  - a Enter a unique name for this CLI instance (different from the API key name).  
Use lowercase characters, numbers, and hyphens only. The name helps identify and manage the CLI in the console.
  - b Enter the build step name (for example, development, production, compliance) to be used as the default field for CLI runs.  
  
Build steps are used as reference IDs in build-phase scopes to establish a connection with related configured CLIs. The build step parameter is used to match a scope in Carbon Black Cloud, and consecutively to apply the policy for that scope. The default scope is stored in the configuration file.

---

#### Note

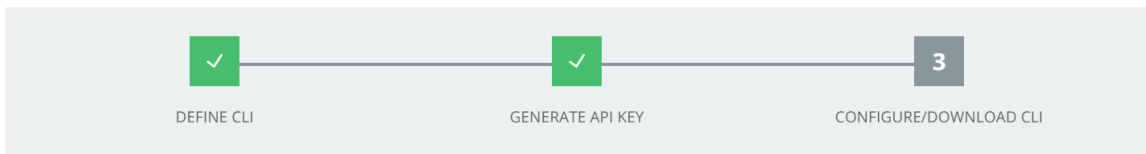
  - The default build step is not unique. Multiple CLI instances can use the same default scope. The **Default build step** cannot be modified after the initial setup, unless you directly edit the configuration file.
  - If a scan is invoked without a build step parameter, the default build step from the configuration file is used.
  - Create a Build Phase scope using this value on the **Kubernetes > Scopes** page in **Build steps**. See [Kubernetes Scopes](#).
  - You must use the CLI `validate` command.

---

  - c Add an optional description (recommended).
- 4 Click **Next**.
- 5 Enter a unique API key name and click **Generate**.
- 6 Click **Next**.

## 7 Copy and run the following command in your terminal window.

```
mkdir -p ~/.cbctl
cat > ~/.cbctl/.cbctl.yaml <<EOF
active_user_profile: cbctl_default
cbctl_default:
  cb_api_id: UHSZCDKMI1
  cb_api_key: 4AYGEJ1T9ILZTQ6VZG9TTEH8
  org_key: EWRTY2PK
  saas_url: https://defense-dev01.cbdtest.io/containers
  default_build_step: ix-test
EOF
```

Add CLI ✕

### CONFIGURE CLI CLIENT

[CLI setup guide](#)

Copy this command into your terminal and run it

```
mkdir -p ~/.cbctl
cat > ~/.cbctl/.cbctl.yaml <<EOF
active_user_profile: cbctl_default
cbctl_default:
  cb_api_id: UHSZCDKMI1
  cb_api_key: 4AYGEJ1T9ILZTQ6VZG9TTEH8
  org_key: EWRTY2PK
  saas_url: https://defense-dev01.cbdtest.io/containers
  default_build_step: ix-test
EOF
```

### DOWNLOAD CLI CLIENT

Already have the CLI client? Skip this step and click **Done**

- + CLI client (Mac) v1.9.2 [Download](#)
- + CLI client (Linux) v1.9.2 [Download](#)




8 If you did not already download the CLI client, you can select and download the CLI instance binary file now, and run it in your build environment.

9 Click **Done**.

### Results

You can operate the configured CLI Client in a terminal to observe the results from vulnerabilities scans on your container images.

### What to do next

To run the Image Scanning CLI API, see [Container Security API and Integrations](#).

To monitor the Vulnerabilities scan for container images that are deployed on Kubernetes, go to the **Inventory > Kubernetes > Container Images** page.

To see the image scanning results for container images that are in particular repositories but not yet deployed, go to the **Inventory > Kubernetes > Container Images** page and click the **Image Repos** tab.

## Carbon Black Container Operator Technical Reference

The Carbon Black Container Operator runs within a Kubernetes cluster. The Container Operator is a set of controllers that deploy and manage the Carbon Black Container components.

The Operator handles the following actions:

- Deploys and manages the Carbon Black Container product, including the configuration and the image scanning for Kubernetes security.
- Automatically fetches and deploys the Carbon Black Container private image registry secret.
- Automatically registers the Carbon Black Container cluster.
- Manages the Carbon Black Container validating webhook and dynamically manages the admission control webhook to avoid possible downtime.
- Monitors and reports agent availability to the Carbon Black Cloud console.

The Carbon Black Container Operator uses the operator-framework to create a GO operator that is responsible for managing and monitoring the Carbon Black Container components deployment.

To review the Operator compatibility matrix, see [Kubernetes Sensor Operator Distributions and Kubernetes Version](#).

---

**Note** We recommend that you deploy the Operator by using the **Add Cluster** wizard (see [Add a Cluster and Install the Kubernetes Sensor](#)). However, this technical reference section of the *User Guide* also includes manual Operator and Agent installation instructions.

---

### Manually Deploy the Container Operator

To manually deploy the Carbon Black Container Operator, perform the following procedure.

These instructions use an Operator image. To deploy the Operator without using an image, see [Container Operator Developer Instructions](#).

#### Prerequisites

Your cluster must be running Kubernetes 1.18+.



## Procedure

- ◆ You can initiate the Operator deployment in two ways:

- **Script:**

```
export OPERATOR_VERSION=v6.0.2
export OPERATOR_SCRIPT_URL=https://setup.containers.carbonblack.io/$OPERATOR_VERSION/
operator-apply.sh
curl -s $OPERATOR_SCRIPT_URL | bash
```

{OPERATOR\_VERSION} is of the format "v{VERSION}".

- **Source code:**

- a Clone the GIT project and deploy the operator from the source code.

By default, the Operator uses `CustomResourceDefinitions v1`, which requires Kubernetes 1.16+. You can also deploy an Operator by using `CustomResourceDefinitions v1beta1` (deprecated in Kubernetes 1.16, removed in Kubernetes 1.22).

- b Create the Operator image:

```
make docker-build docker-push IMG={IMAGE_NAME}
```

- c Deploy the Operator resources:

```
make deploy IMG={IMAGE_NAME}
```

## What to do next

### [Manually Deploy the Container Agent](#)

## Uninstall the Container Operator

To uninstall the Carbon Black Container Operator, perform the following procedure.

## Procedure

- ◆ To uninstall the Carbon Black Container Operator, run the following command:

```
export OPERATOR_VERSION=v6.0.2
export OPERATOR_SCRIPT_URL=https://setup.containers.carbonblack.io/$OPERATOR_VERSION/
operator-apply.sh
curl -s $OPERATOR_SCRIPT_URL | bash -s -- -u
```

This command deletes the Carbon Black Container custom resource definitions (CRDs) and instances.

## Manually Deploy the Container Agent

To manually deploy the Carbon Black Container Agent, perform the following procedure.

## Prerequisites

### Manually Deploy the Container Operator

#### Procedure

- 1 Apply the Carbon Black Container API token secret:

```
kubectl create secret generic cbcontainers-access-token \
--namespace cbcontainers-dataplane --from-literal=accessToken=\
{API_Secret_Key}/{API_ID}
kubectl create secret generic cbcontainers-company-code --namespace cbcontainers-dataplane
--from-literal=companyCode=RXXXXXXXXXXG\!XXXX
```

- 2 Apply the Carbon Black Container Agent custom resource:

Deploy `cbcontainersagents.operator.containers.carbonblack.io` to prompt the Operator to deploy the dataplane components:

```
apiVersion: operator.containers.carbonblack.io/v1
kind: CBContainersAgent
metadata:
  name: cbcontainers-agent
spec:
  account: {ORG_KEY}
  clusterName: {CLUSTER_GROUP}:{CLUSTER_NAME}
  version: {AGENT_VERSION}
  gateways:
    apiGateway:
      host: {API_HOST}
    coreEventsGateway:
      host: {CORE_EVENTS_HOST}
    hardeningEventsGateway:
      host: {HARDENING_EVENTS_HOST}
    runtimeEventsGateway:
      host: {RUNTIME_EVENTS_HOST}
```

---

**Note** See also [Custom Resources Definitions](#).

---

## OpenShift

The Carbon Black Container Operator and Agent require elevated permissions to operate properly. However, this requirement violates the default `SecurityContextConstraints` on most OpenShift clusters, thereby causing the components to fail to start.

You can resolve this issue by applying the following custom security constraint configurations on the cluster. This action requires cluster administrator privileges.

```
kind: SecurityContextConstraints
apiVersion: security.openshift.io/v1
metadata:
  name: scc-anyuid
runAsUser:
```

```

  type: MustRunAsNonRoot
allowHostPID: false
allowHostPorts: false
allowHostNetwork: false
allowHostDirVolumePlugin: false
allowHostIPC: false
allowPrivilegedContainer: false
readOnlyRootFilesystem: true
seLinuxContext:
  type: RunAsAny
fsGroup:
  type: RunAsAny
supplementalGroups:
  type: RunAsAny
users:
- system:serviceaccount:cbcontainers-dataplane:cbcontainers-operator
- system:serviceaccount:cbcontainers-dataplane:cbcontainers-enforcer
- system:serviceaccount:cbcontainers-dataplane:cbcontainers-state-reporter
- system:serviceaccount:cbcontainers-dataplane:cbcontainers-monitor
- system:serviceaccount:cbcontainers-dataplane:cbcontainers-runtime-resolver
---
kind: SecurityContextConstraints
apiVersion: security.openshift.io/v1
metadata:
  name: scc-image-scanning # This probably needs to be fixed in the actual deployment
runAsUser:
  type: RunAsAny
allowHostPID: false
allowHostPorts: false
allowHostNetwork: false
allowHostDirVolumePlugin: false
allowHostIPC: false
allowPrivilegedContainer: false
readOnlyRootFilesystem: false
seLinuxContext:
  type: RunAsAny
fsGroup:
  type: RunAsAny
supplementalGroups:
  type: RunAsAny
allowedCapabilities:
- 'NET_BIND_SERVICE'
users:
- system:serviceaccount:cbcontainers-dataplane:cbcontainers-image-scanning
---
kind: SecurityContextConstraints
apiVersion: security.openshift.io/v1
metadata:
  name: scc-node-agent
runAsUser:
  type: RunAsAny
allowHostPID: true
allowHostPorts: false
allowHostNetwork: true
allowHostDirVolumePlugin: true

```

```

allowHostIPC: false
allowPrivilegedContainer: true
readOnlyRootFilesystem: false
seLinuxContext:
  type: RunAsAny
fsGroup:
  type: RunAsAny
supplementalGroups:
  type: RunAsAny
volumes:
- configMap
- downwardAPI
- emptyDir
- hostPath
- persistentVolumeClaim
- projected
- secret
users:
- system:serviceaccount:cbcontainers-dataplane:cbcontainers-agent-node

```

## Uninstalling the Operator on OpenShift

Add this `SecurityContextConstraints` before running the operator uninstall command:

```

kind: SecurityContextConstraints
apiVersion: security.openshift.io/v1
metadata:
  name: scc-edr-cleaner
runAsUser:
  type: RunAsAny
allowHostPID: true
allowHostPorts: false
allowHostNetwork: true
allowHostDirVolumePlugin: true
allowHostIPC: false
allowPrivilegedContainer: true
readOnlyRootFilesystem: false
seLinuxContext:
  type: RunAsAny
fsGroup:
  type: RunAsAny
supplementalGroups:
  type: RunAsAny
volumes:
- configMap
- downwardAPI
- emptyDir
- hostPath
- persistentVolumeClaim
- projected
- secret
users:
- system:serviceaccount:cbcontainers-edr-sensor-cleaners:cbcontainers-edr-sensor-cleaner

```

## Reading Metrics by using Prometheus

Operator metrics are protected by `kube-auth-proxy`. You must grant permissions to a Prometheus server before it can scrape the protected metrics.

You can create a `ClusterRole` and bind it with `ClusterRoleBinding` to the service account that your Prometheus server uses.

If you have not configured this cluster role and cluster role binding, you can use the following configuration:

### Cluster Role

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: cbcontainers-metrics-reader
rules:
  - nonResourceURLs:
    - /metrics
    verbs:
    - get
```

### Cluster Role Binding

```
kubectl create clusterrolebinding metrics --clusterrole=cbcontainers-metrics-reader --
serviceaccount=<prometheus-namespace>:<prometheus-service-account-name>
```

Use the following `ServiceMonitor` to scrape metrics from the Carbon Black Container Operator. Your Prometheus custom resource service monitor selectors must match this configuration.

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  labels:
    control-plane: operator
  name: cbcontainers-operator-metrics-monitor
  namespace: cbcontainers-dataplane
spec:
  endpoints:
  - bearerTokenFile: /var/run/secrets/kubernetes.io/serviceaccount/token
    path: /metrics
    port: https
    scheme: https
    tlsConfig:
      insecureSkipVerify: true
  selector:
    matchLabels:
      control-plane: operator
```

## Custom Resources Definitions

The Carbon Black Container Operator implements controllers for Carbon Black Container custom resources definitions (CRDs).

### Carbon Black Container Agent Custom Resource

Deploy `cbcontainersagents.operator.containers.carbonblack.io` to prompt the Operator to deploy the dataplane components.

**Table 11-4. Required Parameters**

Parameter	Description
<code>spec.account</code>	Carbon Black Container org key
<code>spec.clusterName</code>	Carbon Black Container cluster name ( <code>&lt;cluster_group:cluster_name&gt;</code> )
<code>spec.version</code>	Carbon Black Container Agent version
<code>spec.gateways.apiGateway.host</code>	Carbon Black Container API host
<code>spec.gateways.coreEventsGateway.host</code>	Carbon Black Container core events host (for example, health checks)
<code>spec.gateways.hardeningEventsGateway.host</code>	Carbon Black Container hardening events host (for example, deleted, validated, and blocked resources)
<code>spec.gateways.runtimeEventsGateway.host</code>	Carbon Black Container runtime events host (for example, traffic events)

**Table 11-5. Optional Parameters**

Parameter	Description	Default Value
<code>spec.apiGateway.port</code>	Carbon Black Container API port	443
<code>spec.accessTokenSecretName</code>	Carbon Black Container API access token secret name	<code>cbcontainers-access-token</code>
<code>spec.gateways.coreEventsGateway.port</code>	Carbon Black Container core events port	443
<code>spec.gateways.hardeningEventsGateway.port</code>	Carbon Black Container hardening events port	443
<code>spec.gateways.runtimeEventsGateway.port</code>	Carbon Black Container runtime events port	443

**Table 11-6. Basic Components Optional Parameters**

Parameter	Description	Default Value
<code>spec.components.basic.enforcer.replicasCount</code>	Carbon Black Container Hardening Enforcer number of replicas	1
<code>spec.components.basic.monitor.image.repository</code>	Carbon Black Container Monitor image repository	<code>cbartifactory/monitor</code>

Table 11-6. Basic Components Optional Parameters (continued)

Parameter	Description	Default Value
<code>spec.components.basic.enforcer.image.repository</code>	Carbon Black Container Hardening Enforcer image repository	<code>cbartifactory/guardrails-enforcer</code>
<code>spec.components.basic.stateReporter.image.repository</code>	Carbon Black Container Hardening State Reporter image repository	<code>cbartifactory/guardrails-state-reporter</code>
<code>spec.components.basic.monitor.resources</code>	Carbon Black Container Monitor resources	<code>{requests: {memory: "64Mi", cpu: "30m"}, limits: {memory: "256Mi", cpu: "200m"}}</code>
<code>spec.components.basic.enforcer.resources</code>	Carbon Black Container Hardening Enforcer resources	<code>{requests: {memory: "64Mi", cpu: "30m"}, limits: {memory: "256Mi", cpu: "200m"}}</code>
<code>spec.components.basic.stateReporter.resources</code>	Carbon Black Container Hardening State Reporter resources	<code>{requests: {memory: "64Mi", cpu: "30m"}, limits: {memory: "256Mi", cpu: "200m"}}</code>

Table 11-7. Runtime Components Optional Parameters

Parameter	Description	Default Value
<code>spec.components.runtimeProtection.enabled</code>	Carbon Black Container flag to control Runtime components deployment	True
<code>spec.components.runtimeProtection.resolver.image.repository</code>	Carbon Black Container Runtime Resolver image repository	<code>cbartifactory/runtime-kubernetes-resolver</code>
<code>spec.components.runtimeProtection.sensor.image.repository</code>	Carbon Black Container Runtime Sensor image repository	<code>cbartifactory/runtime-kubernetes-sensor</code>
<code>spec.components.runtimeProtection.internalGrpcPort</code>	Carbon Black Container Runtime gRPC port that the resolver exposes for the sensor	443
<code>spec.components.runtimeProtection.resolver.logLevel</code>	Carbon Black Container Runtime Resolver log level	<code>"panic", "fatal", "error", "warn", "info", "debug", "trace" (default info)</code>
<code>spec.components.runtimeProtection.resolver.resources</code>	Carbon Black Container Runtime Resolver resources	<code>{requests: {memory: "64Mi", cpu: "200m"}, limits: {memory: "1024Mi", cpu: "900m"}}</code>
<code>spec.components.runtimeProtection.sensor.logLevel</code>	Carbon Black Container Runtime Sensor log level	<code>"panic", "fatal", "error", "warn", "info", "debug", "trace" (default info)</code>
<code>spec.components.runtimeProtection.sensor.resources</code>	Carbon Black Container Runtime Sensor resources	<code>{requests: {memory: "64Mi", cpu: "30m"}, limits: {memory: "1024Mi", cpu: "500m"}}</code>

Table 11-8. Cluster Scanning Components Optional Parameters

Parameter	Description	Default Value
<code>spec.components.clusterScanning.enabled</code>	Carbon Black Container flag to control Cluster Scanning components deployment	True
<code>spec.components.clusterScanning.imageScanningReporter.image.repository</code>	Carbon Black Container Image Scanning Reporter image repository	<code>cbartifactory/image-scanning-reporter</code>
<code>spec.components.clusterScanning.clusterScanner.image.repository</code>	Carbon Black Container Scanner Agent image repository	<code>cbartifactory/cluster-scanner</code>
<code>spec.components.clusterScanning.imageScanningReporter.resources</code>	Carbon Black Container Image Scanning Reporter resources	<code>{requests: {memory: "64Mi", cpu: "200m"}, limits: {memory: "1024Mi", cpu: "900m"}}</code>
<code>spec.components.clusterScanning.clusterScanner.resources</code>	Carbon Black Container Cluster Scanner resources	<code>{requests: {memory: "64Mi", cpu: "30m"}, limits: {memory: "1024Mi", cpu: "500m"}}</code>
<code>spec.components.clusterScanning.clusterScanner.k8sContainerEngine.engineType</code>	Carbon Black Container Cluster Scanner Kubernetes container engine type. One of these options: <code>containerd / docker-daemon / cri-o</code>	N/A
<code>spec.components.clusterScanning.clusterScanner.k8sContainerEngine.endpoint</code>	Carbon Black Container Cluster Scanner Kubernetes container engine endpoint path	N/A
<code>spec.components.clusterScanning.clusterScanner.k8sContainerEngine.CRIO.storagePath</code>	Carbon Black Container Cluster Scanner override default image storage path (CRI-O only)	N/A
<code>spec.components.clusterScanning.clusterScanner.k8sContainerEngine.CRIO.storageConfigPath</code>	Carbon Black Container Cluster Scanner override default image storage config path (CRI-O only)	N/A
<code>spec.components.clusterScanning.clusterScanner.k8sContainerEngine.CRIO.configPath</code>	Carbon Black Container Cluster Scanner override default CRI-O config path (CRI-O only)	N/A
<code>spec.components.clusterScanning.clusterScanner.cliFlags.enableSecretDetection</code>	Carbon Black Container Cluster Scanner flag of whether the scan should scan for secrets	False
<code>spec.components.clusterScanning.clusterScanner.cliFlags.skipDirsOrFiles</code>	Carbon Black Container Cluster Scanner flag of files or directories to not scan for secrets	N/A



Table 11-8. Cluster Scanning Components Optional Parameters (continued)

Parameter	Description	Default Value
<code>spec.components.clusterScanning.clusterScanner.cliFlags.scanBaseLayers</code>	Carbon Black Container Cluster Scanner flag of whether the scan should include the base layers scan for secrets	False
<code>spec.components.clusterScanning.clusterScanner.cliFlags.ignoreBuildInRegex</code>	Carbon Black Container Cluster Scanner flag of whether the scan should ignore the built-in regexes of files to skip secret detection	False

Table 11-9. Components Common Optional Parameters

Parameter	Description	Default Value
<code>labels</code>	Carbon Black Container component deployment and pod labels	Empty map
<code>deploymentAnnotations</code>	Carbon Black Container component deployment annotations	Empty map
<code>podTemplateAnnotations</code>	Carbon Black Container component pod annotations	{}
<code>env</code>	Carbon Black Container component pod environment variables	Empty map
<code>image.tag</code>	Carbon Black Container component image tag	Agent version
<code>image.pullPolicy</code>	Carbon Black Container component pull policy	IfNotPresent
<code>probes.port</code>	Carbon Black Container component probes port	8181
<code>probes.scheme</code>	Carbon Black Container component probes scheme	HTTP
<code>probes.initialDelaySeconds</code>	Carbon Black Container component probes initial delay seconds	3
<code>probes.timeoutSeconds</code>	Carbon Black Container component probes timeout seconds	1
<code>probes.periodSeconds</code>	Carbon Black Container component probes period seconds	30
<code>probes.successThreshold</code>	Carbon Black Container component probes success threshold	1
<code>probes.failureThreshold</code>	Carbon Black Container component probes failure threshold	3
<code>prometheus.enabled</code>	Carbon Black Container component enable Prometheus scraping	False
<code>prometheus.port</code>	Carbon Black Container component Prometheus server port	7071

Table 11-9. Components Common Optional Parameters (continued)

Parameter	Description	Default Value
<code>nodeSelector</code>	Carbon Black Container component node selector	<code>{}</code>
<code>affinity</code>	Carbon Black Container component affinity	<code>{}</code>

Table 11-10. Centralized Proxy Parameters

Parameter	Description	Default Value
<code>spec.components.settings.proxy.enabled</code>	Enables applying the centralized proxy settings to all components	False
<code>spec.components.settings.proxy.httpProxy</code>	HTTP proxy server address to use	Empty string
<code>spec.components.settings.proxy.httpsProxy</code>	HTTPS proxy server address to use	Empty string
<code>spec.components.settings.proxy.noProxy</code>	A comma-separated list of hosts to which to connect without using a proxy	Empty string
<code>spec.components.settings.proxy.noProxySuffix</code>	A comma-separated list of hosts to which to append the <code>noProxy</code> list of values	The API server IP addresses followed by <code>cbcontainers-dataplane.svc.cluster.local</code>

Table 11-11. Other Components Optional Parameters

Parameter	Description	Default Value
<code>spec.components.settings.daemonSetsTolerations</code>	Carbon Black DaemonSet component tolerances	Empty array

## Changing Components Resources

Needs description/intro.

```
spec:
  components:
    basic:
      monitor:
        resources:
          limits:
            cpu: 200m
            memory: 256Mi
          requests:
            cpu: 30m
            memory: 64Mi
      enforcer:
        resources:
          ##### DESIRED RESOURCES SPEC - for hardening enforcer container
      stateReporter:
```

```

resources:
  ##### DESIRED RESOURCES SPEC - for hardening state reporter container
runtimeProtection:
resolver:
  resources:
    ##### DESIRED RESOURCES SPEC - for runtime resolver container
sensor:
  resources:
    ##### DESIRED RESOURCES SPEC - for node-agent runtime container
clusterScanning:
imageScanningReporter:
  resources:
    ##### DESIRED RESOURCES SPEC - for image scanning reporter pod
clusterScanner:
  resources:
    ##### DESIRED RESOURCES SPEC - for node-agent cluster-scanner container

```

## Cluster Scanner Component Memory

By default, the `clusterScanning.clusterScanner` component attempts to scan images of sizes up to 1GB. Its recommended resources are:

```

resources:
  requests:
    cpu: 100m
    memory: 1Gi
  limits:
    cpu: 2000m
    memory: 6Gi

```

To scan images larger than 1GB, allocate higher memory resources in the component's `requests.memory` and `limits.memory`, and add an environment variable `MAX_COMPRESSED_IMAGE_SIZE_MB` to override the maximum images size in MB that the scanner tries to scan.

For example, to set the cluster scanner to scan images up to 1.5 GB. the configuration is:

```

spec:
  components:
    clusterScanning:
      clusterScanner:
        env:
          MAX_COMPRESSED_IMAGE_SIZE_MB: "1536" // 1536 MB == 1.5 GB
        resources:
          requests:
            cpu: 100m
            memory: 2Gi
          limits:
            cpu: 2000m
            memory: 5Gi

```

If your nodes have low memory and you want the cluster scanner to consume less memory, you must reduce the component's `requests.memory` and `limits.memory`, and override the `MAX_COMPRESSED_IMAGE_SIZE_MB` parameter to be less than 1GB (1024MB).

For example, assign lower memory resources and set the cluster-scanner to scan images up to 250MB:

```
spec:
  components:
    clusterScanning:
      clusterScanner:
        env:
          MAX_COMPRESSED_IMAGE_SIZE_MB: "250" // 250 MB
        resources:
          requests:
            cpu: 100m
            memory: 250Mi
          limits:
            cpu: 2000m
            memory: 1Gi
```

## Configuring Container Services to use HTTP Proxy

You can configure the Carbon Black Container to use an HTTP proxy by enabling the centralized proxy settings or by manually setting `HTTP_PROXY`, `HTTPS_PROXY`, and `NO_PROXY` environment variables.

The centralized proxy settings apply an HTTP proxy configuration for all components. The manual setting of environment variables allows you to set the configuration parameters on a per component basis. If both HTTP proxy environment variables and centralized proxy settings are provided, the environment variables take precedence. The Operator does not use the centralized proxy settings, so you must use the environment variables for it instead.

### Configure Centralized Proxy Settings

To configure the proxy environment variables in the Operator, use the following command to patch the Operator deployment:

```
kubectl set env -n cbcontainers-dataplane deployment cbcontainers-operator HTTP_PROXY="<proxy-url>" HTTPS_PROXY="<proxy-url>" NO_PROXY="<kubernetes-api-server-ip>/<range>"
```

Update the `CBContainersAgent` CR to use the centralized proxy settings (`kubectl edit cbcontainersagents.operator.containers.carbonblack.io cbcontainers-agent`):

```
spec:
  components:
    settings:
      proxy:
```

```

enabled: true
httpProxy: "<proxy-url>"
httpsProxy: "<proxy-url>"
noProxy: "<exclusion1>,<exclusion2>"

```

You can disable the centralized proxy settings without deleting them by setting the `enabled` key to `false`.

By default, the centralized proxy settings determine the API server IP address(es) and the necessary proxy exclusions for the `cbcontainers-dataplane` namespace. These determined values are automatically appended to the `noProxy` values or the specified `NO_PROXY` environment variable for a particular component. To change those pre-determined values, you can specify the `noProxySuffix` key at the same level as the `noProxy` key. It has the same format as the `noProxy` key and its values are treated as if they were pre-determined. You can also force nothing to be appended to `noProxy` or `NO_PROXY` by setting `noProxySuffix` to an empty string.

## Configure HTTP Proxy Per-Component Environment Variables

To configure environment variables for the `basic`, `Runtime`, and `Image Scanning` components, update the `CBContainersAgent` CR using the proxy environment variables (`kubectl edit cbcontainersagents.operator.containers.carbonblack.io cbcontainers-agent`):

```

spec:
  components:
    basic:
      enforcer:
        env:
          HTTP_PROXY: "<proxy-url>"
          HTTPS_PROXY: "<proxy-url>"
          NO_PROXY: "<kubernetes-api-server-ip>/<range>"
      stateReporter:
        env:
          HTTP_PROXY: "<proxy-url>"
          HTTPS_PROXY: "<proxy-url>"
          NO_PROXY: "<kubernetes-api-server-ip>/<range>"
      runtimeProtection:
        resolver:
          env:
            HTTP_PROXY: "<proxy-url>"
            HTTPS_PROXY: "<proxy-url>"
            NO_PROXY: "<kubernetes-api-server-ip>/<range>"
      sensor:
        env:
          HTTP_PROXY: "<proxy-url>"
          HTTPS_PROXY: "<proxy-url>"
          NO_PROXY: "<kubernetes-api-server-ip>/<range>,cbcontainers-runtime-resolver.cbcontainers-dataplane.svc.cluster.local"
      clusterScanning:
        clusterScanner:
          env:
            HTTP_PROXY: "<proxy-url>"
            HTTPS_PROXY: "<proxy-url>"

```

```

    NO_PROXY: "<kubernetes-api-server-ip>/<range>,cbcontainers-image-scanning-
reporter.cbcontainers-dataplane.svc.cluster.local"
  imageScanningReporter:
    env:
      HTTP_PROXY: "<proxy-url>"
      HTTPS_PROXY: "<proxy-url>"
      NO_PROXY: "<kubernetes-api-server-ip>/<range>"

```

**Important** You must configure the `NO_PROXY` environment variable to use the value of the Kubernetes API server IP address. To find the API-server IP address, run the following command:

```
kubectl -n default get service kubernetes -o=jsonpath='{..clusterIP}'
```

## Additional Proxy Considerations

When using a non-transparent HTTPS proxy, you must configure the agent to use the proxy certificate authority:

```

spec:
  gateways:
    gatewayTLS:
      rootCAsBundle: <Base64 encoded proxy CA>

```

Alternatively, you can allow the agent to communicate without verifying the certificate. We do not recommend this option because it exposes the agent to an MITM attack.

```

spec:
  gateways:
    gatewayTLS:
      insecureSkipVerify: true

```

## Changing the Image Source

By default, all images for the Operator and Agent deployments are pulled from Docker Hub. If you prefer to mirror the images in your internal repositories, you can specify the image by modifying the `CBContainersAgent` resource that you apply to your cluster.

Modify the following properties to specify the image for each service:

- `monitor` - `spec.components.basic.monitor.image`
- `enforcer` - `spec.components.basic.enforcer.image`
- `state-reporter` - `spec.components.basic.stateReporter.image`
- `runtime-resolver` - `spec.components.runtimeProtection.resolver.image`
- `runtime-sensor` - `spec.components.runtimeProtection.sensor.image`
- `image-scanning-reporter` - `spec.components.clusterScanning.imageScanningReporter.image`
- `cluster-scanner` - `spec.components.clusterScanning.clusterScanner.image`

The image object consists of four properties:

- `repository` - the repository of the image; for example, `docker.io/my-org/monitor`
- `tag` - the version tag of the image; for example, `1.0.0`, `latest`, and so forth.
- `pullPolicy` - the pull policy for that image; for example, `IfNotPresent`, `Always`, or `Never`. See [Image pull policy](#) (external link).
- `pullSecrets` - the image pull secrets that are going to be used to pull the container images. The secrets must already exist in the cluster. See [Pull an Image from a Private Registry](#) (external link).

### Sample configuration:

```
spec:
  monitor:
    image:
      repository: docker.io/my-org/monitor
      tag: 1.0.0
      pullPolicy: Always
      pullSecrets:
        - my-pull-secret
```

In this case, the operator attempts to run the monitor service from the `docker.io/my-org/monitor:1.0.0` container image and the kubelet is instructed to always pull the image by using the `my-pull-secret` secret.

## Using a Shared Secret for all Images

To use just one pull secret to pull all the custom images, specify it under `spec.settings.imagePullSecrets`.

The secret is added to the `imagePullSecrets` list of all Agent workloads.

## Operator Role-based Access Control

This section describes how to configure and use Carbon Black Container Operator Role-based Access Control (RBAC).

### RBAC Definition and Design

Following the principle of least-privilege, any permission given to the Operator should have good reason and be scoped as tightly as possible.

In practice, this means:

- If the resource is namespaced and part of the agent, use a `Role` to give permissions in the agent's namespace only.
- If the resource is namespaced and not part of the agent:
  - To read it, use a `ClusterRole` unless you are sure what the namespace will be.

- To modify it, examine whether this is absolutely necessary.
- If the resource is non-namespaced, use a `ClusterRole` and restrict `delete`, `get`, `update`, and `patch` through `resourceNames`. `Create`, `list`, and `watch` either do not support this restriction or require extra care.

## Changing the Operator Access Levels

Operator access level permissions are generated by `controller-gen` and controlled by using `+kubebuilder` directives. See [controller definitions](#) (external link). Any change to those directives requires running `make manifests` to update the respective `role.yaml` file. You must also propagate changes to the helm charts.

## Changing the Agent Component Access Levels

Agent component access levels, service accounts, and role bindings are manually maintained in `dataplane_roles.yaml` and the helm equivalent. You must apply changes in both locations.

The roles should follow the least-privilege principle. Agent components often need more permissions than the Operator to work as expected.

## Container Operator Developer Instructions

This topic describes instructions using the SDK version 1.29.0 for the Operator.

### Deploy the Operator without using an Image

To install dependencies to verify the `kubeconfig` context:

```
make deploy OPERATOR_REPLICAS=0
```

To run the Operator from the terminal to verify the `kubeconfig` context:

```
make run
```

From your editor, run and debug `main.go` to verify the `KUBECONFIG` environment variable.

### Install the Dataplane on your own Control Plane

Under the Carbon Black Container Cluster CR:

```
spec:
  apiGatewaySpec:
    adapter: {MY-ADAPTER-NAME}
```

where `{MY-ADAPTER-NAME}` is your control plane adapter name. The default value is `containers`.



## Uninstall the Container Operator

From a terminal, run the following command:

```
make undeploy
```

---

**Note** This command does not clean up the Carbon Black directory on the dataplane nodes.

---

## Changing Security Context Settings

### Hardening enforcer/state\_reporter security context settings:

You can change the values under `cbcontainers/state/hardening/objects` for `enforcer_deployment.go` Or `state_reporter_deployment.go`.

### Using defaults:

Defaults in the `OpenAPISchema` is a feature in `apiextensions/v1` version of `CustomResourceDefinitions`. These default values are supported by `kubebuilder` by using tags; for example, `kubebuilder:default=something`. For backwards compatibility, all defaults should also be implemented and set in the controllers to make sure that they work on clusters v1.15 and below.

---

**Note** `kubebuilder` does not support an empty object as a default value. See [related issue](#) (external link). The root issue is in regard to maps, but the same code causes issues with objects.

---

Therefore, the following specification will not apply the default for `test` unless the user specifies `bar`.

```
spec:
  properties:
    bar:
      properties:
        test:
          default: 10
          type: integer
```

Applying this YAML will save an empty object for `bar: spec: {}`.

Instead, applying `spec: { bar: {} }` works as expected and saves the following object:

```
spec: { bar: { test: 10 }}
```

For example:

```
spec:
  properties:
    bar:
      default: {}
```

```
properties:
  test:
    default: 10
    type: integer
```

kubebuilder cannot currently produce that output. Therefore, replacing all instance of `<>` with `{ }` so that using `kubebuilder:default=<>` produces the correct output.

Defaulting is not supported by v1beta1 versions of CRD.

## Local Debugging

To debug locally, run `make run-delve`. This command builds and starts a delve debugger in headless mode. Then use an editor to start a remote session and connect to the delve instance.

For `goland`, the built-in go remote configuration works.

## Custom Namespace

If the Operator is not deployed in the default namespace (`cbcontainers-dataplane`), you must set the `OPERATOR_NAMESPACE` environment variable when using `make run` or `make run-delve`.

## Helm Charts

This topic describes the official Helm charts for installing the Carbon Black Container Agent (Operator, CRD, and Agent components).

### cbcontainers-operator

The [cbcontainer-operator](#) chart (external link) is the official Helm chart for installing the Carbon Black Container Operator and CRD. Helm 3 is supported.

You can install the chart without any customizations or modifications, and you can create the Helm release in any namespace. You can customize the namespace in which the Operator is installed.

To install the Helm chart from the source:

```
cd charts/cbcontainers-operator
helm install cbcontainers-operator ./cbcontainers-operator-chart
```

**Table 11-12. Customization**

Parameter	Description	Default Value
<code>spec.operator.image.repository</code>	Repository of the Operator image	<code>cbartifactory/octarine-operator</code>
<code>spec.operator.image.version</code>	Version of the Operator image	The latest version of the Operator image
<code>spec.operator.resources</code>	Carbon Black Container Operator resources	<code>{requests: {memory: "64Mi", cpu: "30m"}, limits: {memory: "256Mi", cpu: "200m"}}</code>

Table 11-12. Customization (continued)

Parameter	Description	Default Value
<code>spec.rbacProxy.resources</code>	Kube RBAC proxy resources	{requests: {memory: "64Mi", cpu: "30m"}, limits: {memory: "256Mi", cpu: "200m"}}
<code>spec.operator.environment</code>	Environment variables to be set to the Operator pod	[]

## Namespace

By default, the Carbon Black Container Operator is installed in the `cbcontainers-dataplane` namespace.

To change the namespace, set the `operatorNamespace` field in your `values.yaml` file.

The chart automatically creates the namespace. If you do not want to do that (because you have already created the namespace), set the `createOperatorNamespace` field in your `values.yaml` file to `false`.

If the namespace is pre-created, then it must also be labeled properly or the Operator and Agent might not reconcile successfully. The following commands show an example of creating a custom namespace and labeling and installing the operator inside.

```
NAMESPACE=<your_value>
kubectl create namespace $NAMESPACE
kubectl label namespace $NAMESPACE control-plane=operator octarine=ignore
helm install cbcontainers-operator ./cbcontainers-operator-chart --set
createOperatorNamespace=false,operatorNamespace=$NAMESPACE
```

## CRD Installation

By default, installing the chart will also create the `CBContainersAgent` CRD.

To manage the CRD in a different way and not install it together with the chart, set the `installCRD` field in your `values.yaml` file to `false`.

## HTTP Proxy

To use an HTTP proxy for the communication with the Carbon Black Cloud backend, you must set 3 environment variables. These variables are exposed through the `Values.operator.proxy` parameters in the `values.yaml` file:

- `Values.operator.proxy.http`
- `Values.operator.proxy.https`
- `Values.operator.proxy.noProxy`

See also [Configuring Container Services to use HTTP Proxy](#) .

## Templates

The `cbcontainers-operator` chart consists of four [templates](#) (external link).

The [operator.yaml](#) file (external link) contains all resources except for the Operator deployment. It is generated by kustomize. For more info see [config/default\\_chart](#) (external link).

The `deployment.yaml` file contains the Operator `Deployment` resource. It is derived from [this Kustomize configuration](#). Because it must be configurable through Helm, it is heavily templated. Therefore, it cannot be generated automatically, so it must be maintained by hand. If any changes are made to the [Kustomize configuration](#), they must also be reflected in the `deployment.yaml` file.

The `dataplane_rbac.yaml` and `dataplane_service_accounts` files contain necessary RBAC objects for the Agent to work as expected.

## cbcontainers-agent

The [cbcontainer-agent](#) chart (external link) is the official Helm chart for installing the Carbon Black Container Agent components. Helm 3 is supported.

---

**Note** Before installing the Agent components, you must install the Operator and the CRD.

---

### Installation

Before you can install the chart, you must configure it. You must provide the following eight required fields:

Parameter	Description
<code>spec.orgKey</code>	Org key of the organization using Carbon Black Cloud
<code>spec.clusterName</code>	Name of the cluster that will be added to Carbon Black Cloud
<code>spec.clusterGroup</code>	The group that the cluster belongs to in Carbon Black Cloud
<code>spec.version</code>	Version of the Agent images
<code>spec.gateways.apiGatewayHost</code>	URL of the Carbon Black Cloud API gateway
<code>spec.gateways.coreEventsGatewayHost</code>	URL of the Carbon Black Cloud core events gateway
<code>spec.gateways.hardeningEventsGatewayHost</code>	URL of the Carbon Black Cloud hardening events gateway
<code>spec.gateways.runtimeEventsGatewayHost</code>	URL of the Carbon Black Cloud runtime events gateway

After setting these required fields in a `values.yaml` file, you can install the chart from source:

```
cd charts/cbcontainers-agent
helm install cbcontainers-agent ./cbcontainers-agent-chart -n cbcontainers-dataplane
```

### Customization

The way in which the Carbon Black Container components are installed is highly customizable.

You can set different properties for the components or enable and disable components by using the `spec.components` section of your `values.yaml` file.

For a list of all possible values, see [Custom Resources Definitions](#).

## Namespace

The Carbon Black Cloud Containers Agent will run in the same namespace as the deployed Operator. This is by design because only one running agent per cluster is supported. To customize that namespace, see [operator-chart](#) (external link).

The actual namespace where Helm tracks the release (see `--namespace flag`, external link) is not important to the Agent chart, but the recommended approach is to also use the same namespace as the Operator chart.

The `agentNamespace` value is only required if the Agent chart is responsible for deploying the Agent's secret as well. If the secret is pre-created before deploying the agent, then `agentNamespace` has no effect.

## Secret Creation

### *Carbon Black API Key*

For the Agent components to function correctly and communicate with the Carbon Black Cloud backend, an access token is required. This token is located in a *secret*. By default, the secret is named `cbcontainers-access-token`, but that name is configurable through the `accessTokenSecretName` property. If that secret does not exist, the Operator will not start any of the Agent components.

To create the secret as part of the chart installation, provide the `accessToken` value to the chart.

Inject this value as part of your pipeline in a secure way: store the secret as plain text in your `values.yaml` file.

To create the secret in an alternative and more secure way, do not set the `accessToken` value: the chart will not create the secret objects.

---

**Important** Do not store the token in your source code.

---

### *Carbon Black Company Codes*

For the agent CNDR component to function correctly and communicate with the Carbon Black Cloud backend, a company code is required. This code is located in a secret. By default, the secret is named `cbcontainers-company-code`, but that name is configurable through the `components.cndr.companyCodeSecretName` property.

If that secret does not exist, the CNDR component will fail.

If you want to create the secret as part of the chart installation, provide the `companyCode` value to the chart.

Inject this value as part of your pipeline in a secure way: store the secret as plain text in your `values.yaml` file.

To create the secret in an alternative and more secure way, do not set the `companyCode` value: the chart will not create the secret objects.

---

**Important** Do not store the code in your source code.

---

# Signature Mirror Instructions

# 12

---

**Attention** Carbon Black Cloud supports the use of a signature mirror repository; however, Carbon Black does not assist in the configuration of a signature mirror repository.

Customers who require a local mirror repository can use, at their own risk, the following procedure which is authored and maintained by the Carbon Black Community on UEX: [Endpoint Standard: Local Mirror Server for Signature Updates](#).

**IMPORTANT:** The procedure has not been qualified by the Carbon Black Cloud team.

---

# Configuring Carbon Black Cloud Communications

# 13

Configure your network infrastructure and endpoints to ensure proper communication between sensors and the backend.

The current implementation of the Carbon Black Cloud service uses dynamically managed load balancers to provide the best possible levels of scalability, reliability, and performance. The Carbon Black Cloud services hostname resolves several possible IP addresses that can change dynamically.

There is no static IP, range of IP addresses, or subnet to allow or exclude in firewall or proxy settings.

Network proxies and firewalls can interfere with communication between the Carbon Black Cloud sensor and the Carbon Black Cloud backend if they are improperly configured.

Read the following topics next:

- [Configure a Firewall](#)
- [Configure a Proxy](#)
- [Cryptographic Protocol Requirements](#)

## Configure a Firewall

A sensor can connect to the backend in a firewall-protected network in several ways.

URLs are used for the following purposes:

- Console/API — Console access and API requests
- Sensor — Communication between the sensor and the console/backend
- UBS download — Downloading Unified Binary Store (UBS) binaries and metadata
- Content management — Allowing the Carbon Black sensor to receive instructions (manifests) that configure a wide variety of the Carbon Black Cloud features and their underlying rules. Without the first manifest update, some features, including the following, might not be available.
  - Carbon Black Cloud Enterprise EDR event collection
  - Carbon Black XDR event collection



- Device control
- Host-based firewall
- Unified Binary Store (UBS)
- A large percent of Carbon Black Cloud Endpoint Standard blocking capabilities

When the initial manifest download completes, an access to [content.carbonblack.io](https://content.carbonblack.io) is required to receive configuration changes done by using the Carbon Black Cloud console (in the **Enforce > Policies** page) and to receive the most up-to-date rule sets.

- Signature — Updating signature packs
- Third-party certificate validation — Verifying sensor comm certificates
- Live Response Uploads - Used when performing the "get" command from Live Response

Configure the firewall to allow incoming and outgoing TCP/443 (default) and TCP/54443 (backup) connections to the following environment specific URLs:

**Table 13-1. Environment-specific URLs**

Environment/AWS Region	Console/API	Sensor	UBS download	Live Response Uploads
Prod01 (US-East-1)	<a href="https://dashboard.confer.net">https://dashboard.confer.net</a>	<a href="https://devices.confer.net">https://devices.confer.net</a>	<a href="https://cdc-file-storage-production-us-east-1.s3.amazonaws.com">https://cdc-file-storage-production-us-east-1.s3.amazonaws.com</a>	<a href="https://defense-cblr-file-uploads-us-east-1.s3.amazonaws.com">https://defense-cblr-file-uploads-us-east-1.s3.amazonaws.com</a>
Prod02 (US-East-1)	<a href="https://defense.conferdeploy.net">https://defense.conferdeploy.net</a>	<a href="https://dev5.conferdeploy.net">https://dev5.conferdeploy.net</a>	<a href="https://cdc-file-storage-production-us-east-1.s3.amazonaws.com">https://cdc-file-storage-production-us-east-1.s3.amazonaws.com</a>	<a href="https://defense-cblr-file-uploads-us-east-1.s3.amazonaws.com">https://defense-cblr-file-uploads-us-east-1.s3.amazonaws.com</a>
Prod05 (US-East-1)	<a href="https://defense-prod05.conferdeploy.net">https://defense-prod05.conferdeploy.net</a>	<a href="https://dev-prod05.conferdeploy.net">https://dev-prod05.conferdeploy.net</a>	<a href="https://cdc-file-storage-production-us-east-1.s3.amazonaws.com">https://cdc-file-storage-production-us-east-1.s3.amazonaws.com</a>	<a href="https://defense-cblr-file-uploads-us-east-1.s3.amazonaws.com">https://defense-cblr-file-uploads-us-east-1.s3.amazonaws.com</a>
Prod06 (EU-Central-1)	<a href="https://defense-eu.conferdeploy.net">https://defense-eu.conferdeploy.net</a>	<a href="https://dev-prod06.conferdeploy.net">https://dev-prod06.conferdeploy.net</a>	<a href="https://cdc-file-storage-production-eu-central-1.s3.amazonaws.com">https://cdc-file-storage-production-eu-central-1.s3.amazonaws.com</a>	<a href="https://defense-cblr-file-uploads-eu-central-1.s3.amazonaws.com">https://defense-cblr-file-uploads-eu-central-1.s3.amazonaws.com</a>

Table 13-1. Environment-specific URLs (continued)

Environment/AWS Region	Console/API	Sensor	UBS download	Live Response Uploads
ProdNRT (AP-Northeast-1)	https://defense-prodnrt.conferdeploy.net	https://dev-prodnrt.conferdeploy.net	https://cdc-file-storage-production-ap-northeast-1.s3.amazonaws.com>	https://defense-cblr-file-uploads-ap-northeast-1.s3.amazonaws.com
ProdSYD (AP-Southwest-2)	https://defense-prodsyd.conferdeploy.net/	https://dev-prodsyd.conferdeploy.net/	https://cdc-file-storage-production-ap-southeast-2.s3.amazonaws.com	https://defense-cblr-file-uploads-ap-southeast-2.s3.amazonaws.com

Additionally, all environments use the following URLs:

Table 13-2. All environments

Category	URL	Protocol/Port	Notes
Content Management URL	https://content.carbonblack.io	TCP/443	
Signature URL	http://updates2.cdc.carbonblack.io/update2	TCP/80	Windows sensor versions prior to 3.3
Signature URL	https://updates2.cdc.carbonblack.io/update2	TCP/443	Windows sensor versions 3.3+
Third-party certificate validation URL	http://ocsp.godaddy.com	TCP/80	Online Certificate Status Protocol (OCSP). Sensor version 3.3+: required unless <code>CURL_CRL_CHECK</code> is disabled.
Third-party certificate validation URL	http://crl.godaddy.com http://crl3.digicert.com http://crl4.digicert.com	TCP/80	Certificate Revocation List (CRL). Sensor version 3.3+: required unless <code>CURL_CRL_CHECK</code> is disabled.

If you do not make specific network firewall changes to access the Carbon Black Cloud backend applications, the sensors try to connect through existing proxies. See [Configure a Proxy](#).

**Note** Operational environments that implement a man-in-the-middle proxy should note that additional third-party certificate validation URLs can be needed depending on the server certificates that the proxy uses. Additional URLs include anything specified under the "CRL Distribution Points" and "Authority Information Access" extensions of the proxy server SSL certificate. Failing to allow communication to third-party certificate validation URLs on TCP port 80 can lead to communication failures between the sensor and the backend. The Windows 3.3 and higher sensor relies on Windows to execute a CRL check. This sensor communication certificate verification is recommended but not required. If the sensor fails to validate its own communication certificate, installation will fail unless you set `CURL_CRL_CHECK=0` (see [Disable CURL CRL CHECK](#)).

Alternatively, you can set `CURL_CRL_REVOKE_BEST_EFFORT=1` where the sensor will do a best effort attempt to verify the SSL certificate but will not reject the connection if revocation information cannot be obtained due to firewall or other network restrictions.

If installation fails for this reason and you do not want to disable the CRL check, you can implement one of the following options:

- Configure the Winhttp service to use the proxy for Windows CRL checks
- Configure the proxy or firewall to allow CRL traffic
- Allow port 80 traffic to `crl.godaddy.com` and `ocsp.godaddy.com` through the proxy or firewall

## Carbon Black Cloud Workload Appliance

Carbon Black Service URL / Hostname	IP Address	Protocol/Port	Description
<code>prod.cwp.carbonblack.io</code>	Dynamic	TCP/443	Appliance logging and updates.
vCenter Server Host	User defined	TCP/443	Communication with the vCenter Server .
Carbon Black Cloud console URL (refer to Console/API URL) For example, <code>https://defense-prod05.conferdeploy.net</code> if you are a Prod05 user	Dynamic	TCP/443	Communication with the Carbon Black Cloud.

## Disable CURL CRL CHECK

The `crl.godaddy.com` and `ocsp.godaddy.com` domains use OCSP (Online Certificate Status Protocol) and Certificate Revocation List (CRL) checks to validate a sensor's install certificate. You can disable this check.

---

**Caution** You can disable CRL checks either during or after a sensor installation. However, disabling CRL can potentially open devices up to *man in the middle* attacks if Carbon Black Cloud revokes the certificate (this has never happened), and if an attacker then leverages the revoked certificate for such an attack.

---

For more information about RepCLI, see *Managing Sensors by using RepCLI* in the User Guide.

### To disable CRL check during an initial sensor install:

Using the command line install method, add the `CURL_CRL_CHECK=0` option to the install command. For example:

```
msiexec.exe /q /i CBDefense-setup.msi /L*vx log.txt CURL_CRL_CHECK=0
```

### To disable CRL checks after the sensor is installed:

#### Procedure

- 1 In the Carbon Black Cloud console, click **Inventory** and then click **Endpoints**.
- 2 Select the endpoint, click **Take Action**, and then click **Enable bypass**. Confirm the action.
- 3 To confirm that the endpoint is in bypass mode, run the following RepCLI command: `repcli status`
- 4 As a best practice, create a backup of the `cfg.ini` file into another directory. For Windows sensor versions 3.6 and earlier, `cfg.ini` is located at `C:\Program Files\Confer\cfg.ini`. For Windows sensors 3.7 and later, `cfg.ini` is located at `C:\ProgramData\CarbonBlack\DataFiles\cfg.ini`. After you successfully complete the procedure, delete the backup file.
- 5 Edit `cfg.ini`. Add the following parameter to the end of the file: `CurlCrlCheck=false`.
- 6 Run the following RepCLI command: `RepCLI updateconfig`.
- 7 In the Carbon Black Cloud console, click **Inventory** and then click **Endpoints**.
- 8 Select the endpoint, click **Take Action**, and then click **Disable bypass**. Confirm the action.  
See also [Configure a Firewall](#).

## Configure a Proxy

The Carbon Black Cloud sensor uses a variety of mechanisms to determine whether a network proxy is present.

If a proxy is detected (or if one is specified at install time), the sensor attempts to use that proxy. If no proxy is detected, the sensor will attempt a direct connection through port 443 or 54443.

The sensor attempts to contact the Carbon Black Cloud backend by using the following methods:

- A static configured proxy that is configured during sensor installation.
- A direct connection over TCP/443.
- Auto-detection of a proxy and proxy credentials (when applicable) from the local computer's operating system settings.

If you cannot establish connectivity over the standard SSL port, the sensor can fail over to the alternate port, which is TCP/54443.

---

**Note** Carbon Black Cloud sensors automatically try to detect proxy settings during initial installation. This should be tested. If the automatic proxy detection doesn't succeed, you must define the parameters to include the Proxy IP and Port in the MSI command line during a command line installation.

---

If user authentication is required, the user might be prompted for credentials. This typically does not occur in environments that require proxy credentials because the sensor uses an existing configuration that avoids requiring end users to enter credentials.

---

**Note** Windows sensor 3.3 and later versions performs a CRL check. OCSP and CRL traffic is not handled directly by the sensor or the installer, and does not use the proxy parameters that are specified at install. This traffic requires having WinHTTP set to the proxy.

You must either disable the CRL check (see [Disable CURL CRL CHECK](#)), or configure WinHTTP to use an existing `proxy server:port`. You can perform the latter option in the following ways:

- Set WinHTTP proxy information through proxy-side configuration.
- Manually set WinHTTP proxy through a command line interface on specific machines:

```
netsh winhttp set proxy <proxy>:<port>
```

- Set WinHTTP on multiple machines by using Group Policy.
- 

To avoid going through a network proxy (and/or to avoid being blocked by a firewall), you might need to configure a bypass on your proxy server/firewall to allow outgoing connections from the sensor to the backend. Options for bypass configuration include the following:

- Configure a bypass on your firewall or proxy to allow outgoing connections to your Carbon Black Cloud domain over TCP/443.

- Configure a bypass in your firewall or proxy to allow outgoing connections to the Carbon Black Cloud alternate port TCP/54443.

---

**Important** The host domain name for the Carbon Black Cloud backend server is included in the server's certificate. Some network proxies and gateways might try to validate the certificate and deny the Carbon Black Cloud backend application connection because of a name mismatch between the certificate and real host name of the system that is running in AWS. If this occurs, you must configure the proxy or gateway so that it does not validate the backend server certificate. Note that you cannot access the certificate or hostname in the server's certificate.

---

## Connection Mechanism Precedence

If a sensor fails to connect to the backend, it tries the last known working settings, starting with the most recent ones.

These include the following:

- Proxy
- No proxy
- Credentials
- No credentials
- Proxy used at install time
- Direct connection
- Alternate 54443 port

If the sensor cannot connect using its last valid settings, it reattempts the connection in the following sequence:

- 1 The proxy server that was provided during sensor installation (if applicable).
- 2 Variants of the proxy that was set during sensor installation (if applicable). These variants are with or without credentials using default port (443) and the alternate port (54443).
- 3 A direct connection to the backend with no proxy and default port (443).
- 4 A direct connection to the backend with no proxy using the alternate port 54443.
- 5 Dynamically set proxies such as:
  - Proxies configured within `inetcp1.cpl` (Internet Options) – For each server, also try default (443) and alternate (54443) ports.
  - `.pac` files configured within `inetcp1.cpl` – For each server, also try default (443) and alternate (54443) ports.

---

**Note** Sequence numbers 1 and 2 can be switched by using `PreferStaticProxyOverLastUsed=true` as described in [Configure a Proxy for Windows after Sensor Installation](#).

---

## Configure a Proxy for Windows after Sensor Installation

This article describes how to configure a proxy for Windows after the sensor has been installed.

### Prerequisites

This procedure requires that you have RepCLI authentication. For more information about RepCLI, see *Managing Sensors by using RepCLI* in the User Guide.

### Procedure

- 1 Place the sensor into bypass mode:

```
repcli bypass 1
```

- 2 Confirm that the sensor is in bypass mode:

```
repcli status
```

- 3 Shut down the sensor service:

```
repcli stopCbServices
```

- 4 As a best practice, create a backup of the `cfg.ini` file into another directory. For Windows sensor versions 3.6 and earlier, `cfg.ini` is located at `C:\Program Files\Confer\cfg.ini`. For Windows sensors 3.7 and later, `cfg.ini` is located at `C:\ProgramData\CarbonBlack\DataFiles\cfg.ini`. After you successfully complete the procedure, delete the backup file.

- 5 Edit `cfg.ini` in a plain text editor.

- a If the following parameters exists in `cfg.ini`, remove them:

```
ProxyServer=
ProxyServerCredentials=
```

- b Add the following parameters:

```
ProxyServer=[PROXY_IP_OR_DOMAIN]:[PROXY_PORT]
ProxyServerCredentials=[USERNAME]:[PASSWORD] (Optional- if proxy requires
authentication)
```

```
---Example---
```

```
ProxyServer=TestProxy.net:8080
ProxyServerCredentials=TestUsername:TestPassword
```

- c If the original proxy is still functioning, add the following value to override the previously used value. This option is only available in Windows sensors 3.6+.

```
PreferStaticProxyOverLastUsed=true
```

- 6 Save `cfg.ini`.

**7** Restart the sensor service:

```
sc start CbDefense
```

**8** Take the sensor out of bypass mode:

```
repcli bypass 0
```

**9** To force an immediate check-in (optional):

```
repcli cloud hello
```

## Configure a Proxy for Linux (all Sensor Versions)

Use this procedure to configure a proxy through the `cfg.ini` file for all distributions.

### Procedure

- 1 Extract the contents of the installer package into a temporary directory.
- 2 Use the `install.sh` script to install the agent, but do not provide a company code:

```
sudo cb-psc-install/install.sh
```

- 3 Update the `cfg.ini` file with the v3.x+ company code:

```
sudo /opt/carbonblack/psc/bin/cbagentd -d '<COMPANY_CODE>'
```

- 4 Append the following entry in the `/var/opt/carbonblack/psc/cfg.ini` file. You can use the IP address instead of the hostname.

```
ProxyServer=<hostname>:<port number>
```

**Note** The Linux sensor only supports a HTTP non-authenticated proxy server through `cfg.ini`.

**Table 13-3.**

Proxy Type	IP Format	FDQN Format
HTTP	ip:port	fdqn:port
HTTP	http://ip:port	http://fdqn:port

Example `Cfg.ini` settings:

```
[customer]
ProxyServer=proxy.example.com:3128
```

ProxyServer=<hostname>:<port number>



## 5 Start the agent:

- Centos/Rhel 6:
  - `$ service cbagentd start`
- All other distributions:
  - `$ systemctl start cbagentd`

## Configure a Proxy for Linux (Sensor Versions 2.11.1+)

For Linux sensor version 2.11.1 onwards, use this procedure to configure a proxy through the `install.sh` script for all distributions.

### Procedure

- 1 Extract the contents of the installer package into a temporary directory.
- 2 Use the `install.sh` script to install the agent together with proxy server details and the company code.

You can use the IP address or hostname as part of `ProxyHost`.

```
sudo cb-psc-install/install.sh -p 'ProxyHost:ProxyPort' '<COMPANY_CODE>'
```

## macOS Proxy Server Information

The macOS 3.7.2 sensor uses macOS Keychain APIs to improve proxy server information storage. If you downgrade the macOS sensor from 3.7.2+ to an earlier sensor version, proxy settings must be repopulated.

- If you used macOS System Preferences for the proxy configuration, the sensor attempts to repopulate the proxy information after the sensor is downgraded.
- If the proxy configuration cannot be retrieved from the macOS System Preferences, you must use the sensor unattended installer options `-p PROXY_SERVER:PORT` and `-x PROXY_USER:PASSWORD` to repopulate the proxy settings during the sensor downgrade.

## Cryptographic Protocol Requirements

The following cryptographic protocols are required for proper communication with Carbon Black Cloud.

## Supported SSL Cypher Suites

### Environment:

- Carbon Black Cloud Console: All Versions
- Carbon Black Cloud Sensor: All Versions

- Apple macOS: All Supported Versions
- Linux: All Supported Versions
- Microsoft Windows: All Supported Versions

The following SSL cipher suites are supported by Carbon Black Cloud:

**Table 13-4. Supported Cipher Suites**

Site	Cipher Suite	Strength	TLS 1.2	TLS 1.3
Environment-specific URLs	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	STRONG	X	
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	WEAK	X	
	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	STRONG	X	
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	WEAK	X	
	TLS_RSA_WITH_AES_128_GCM_SHA256	WEAK	X	
	TLS_RSA_WITH_AES_128_CBC_SHA256	WEAK	X	
	TLS_RSA_WITH_AES_256_GCM_SHA384	WEAK	X	
	TLS_RSA_WITH_AES_256_CBC_SHA256	STRONG	X	
content.carbonblack.io	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	STRONG	X	
	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	WEAK	X	
	TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256	STRONG	X	
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	WEAK	X	
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	WEAK	X	
https://updates2.cdc.carbonblack.io	TLS_AES_256_GCM_SHA384	STRONG		X
	TLS_CHACHA20_POLY1305_SHA256	STRONG		X
	TLS_AES_128_GCM_SHA256	STRONG		X
	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	STRONG	X	
	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	STRONG	X	
	TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	STRONG	X	
	TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	STRONG	X	
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	WEAK	X	
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	WEAK	X	
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	WEAK	X	

Table 13-4. Supported Cipher Suites (continued)

Site	Cipher Suite	Strength	TLS 1.2	TLS 1.3
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	WEAK	X	
	TLS_DHE_RSA_WITH_AES_128_CBC_SHA256	WEAK	X	
	TLS_DHE_RSA_WITH_AES_128_CBC_SHA	WEAK	X	
	TLS_DHE_RSA_WITH_AES_256_CBC_SHA256	WEAK	X	
	TLS_DHE_RSA_WITH_AES_256_CBC_SHA	WEAK	X	
	TLS_RSA_WITH_AES_128_GCM_SHA256	WEAK	X	
	TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA	WEAK	X	
	TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA	WEAK	X	
	TLS_RSA_WITH_AES_256_CBC_SHA	WEAK	X	
	TLS_RSA_WITH_CAMELLIA_256_CBC_SHA	WEAK	X	
	TLS_RSA_WITH_AES_128_CBC_SHA	WEAK	X	
	TLS_RSA_WITH_CAMELLIA_128_CBC_SHA	WEAK	X	
	TLS_DHE_RSA_WITH_SEED_CBC_SHA	WEAK	X	
	TLS_RSA_WITH_SEED_CBC_SHA	WEAK	X	

**Important**

- As of 26 September 2022, Carbon Black Cloud signature update servers no longer accept TLS v1.0 or v1.1 for secure connections. As a result, some older operating systems, such as Windows 2012 and earlier, might need to be updated.

## Advanced Encryption Standard (AES)

Carbon Black Cloud supports:

- AES 128

## Hash Support

Carbon Black Cloud supports:

- SHA256

## Key Exchange Algorithm

Carbon Black Cloud supports:

- Elliptic-curve Diffie–Hellman (ECDH)