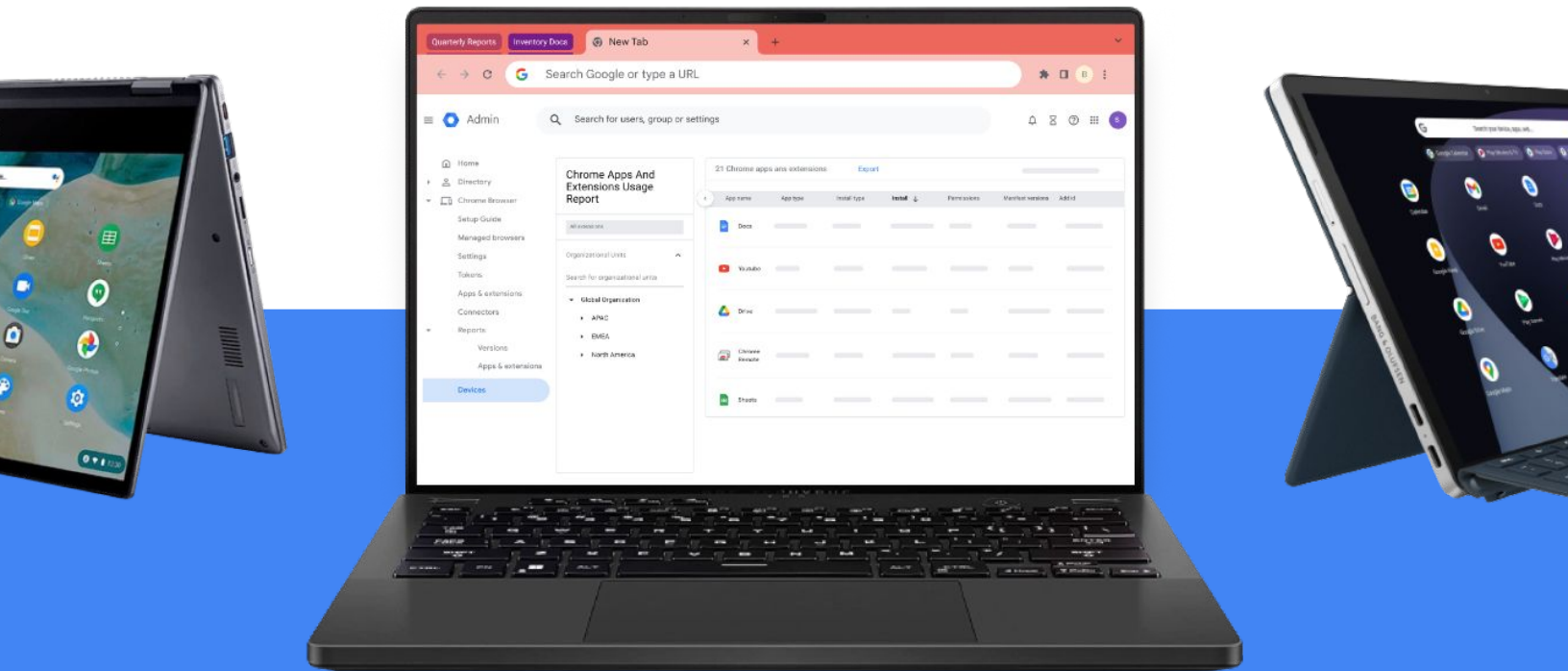




# Managing Extensions in Your Enterprise

Securely manage Chrome extensions at scale

Updated April 2024



# Table of Contents

## [Purpose of this Guide](#)

## [Introduction](#)

## [Considerations for Managing Chrome Extensions](#)

- What are Extension Permissions?
- How do Extensions Update?

## [Managing Extensions](#)

## [Overview of the Different Extension Management Policies](#)

### [Block Extensions Based on Their Permissions](#)

- Manage Extensions by their Permissions in Chrome's Cloud Management Tool
- Manage Extensions by Their Permissions in Group Policy
- Creating an Exception Process for Extensions that Require Risky Permissions

### [Managing Extensions by the ExtensionSettings Policy](#)

- Configure the ExtensionSettings Policy Using the Windows Registry
- Configure Using a JSON String in Windows Group Policy Console
- Prevent Extensions from Altering Webpages

### [Allow or Block Extensions in the Google Admin Console](#)

- Allow All Extensions Except Those You Want to Block
- Block all Extensions Except Those You Want to Allow
- Block or Allow a Single Extension
- Force Install an Extension

### [Allow Users to Request Extensions](#)

- Chrome App and Extensions Usage Report

### [Allow or Block Extensions in Group Policy](#)

- Allow All Extensions Except Those You Want to Block
- Block or Allow a Single Extension
- Force Install an Extension

## [Validating Your Policy](#)

### [Self-Hosting Your Extensions](#)

- [Alternatives to Self-Hosting Extensions](#)
- [Pin an Extension to a Specific Version in the Google Admin Console](#)
- [Requirements for Self-Hosting Extensions](#)
- [Packaging Your Extension](#)
- [Hosting Your Extension](#)
- [Publishing Updates to Your Extension](#)
- [Distributing Privately Hosted Extensions](#)

### [Manage Extensions Using Chrome Enterprise Core](#)

### [Additional Resources](#)

## Purpose of This Guide

There are many useful extensions built for Chrome browser, and users may already have some installed. This can make controlling and monitoring extensions difficult for IT admins.

This guide is for IT admins who are looking for the best ways to manage extensions. It provides steps for managing extensions using both [Chrome Enterprise Core](#) and Windows Group Policies.

This guide is organized by the ways you can manage extensions. You can:

1. Block extensions based on their permissions
2. Manage which websites extensions have access to
3. Allow or block extensions in Chrome's cloud management tool or by Windows Group Policy
4. Self-host your own extensions on-premise

<b>What's covered</b>	Instructions and recommendations for managing extensions with Chrome Enterprise
<b>Primary audience</b>	Microsoft® Windows® and Chrome Enterprise administrators (Windows, Mac, and Linux supported)
<b>Takeaways</b>	Best practices for managing extensions with Chrome browser

Last updated: April 2024

Published location: <https://support.google.com/chrome/a/answer/9296680>

Third-party products: This document describes how Google products work with the Microsoft Windows operating systems and the configurations that Google recommends. Google does not provide technical support for configuring third-party products. Google accepts no responsibility for third-party products. Please consult the product's website for the latest configuration and support information. You may also contact Google Solutions Providers for consulting services.

©2024 Google LLC All rights reserved. Google and the Google logo are registered trademarks of Google LLC. All other company and product names may be trademarks of the respective companies with which they are associated. [EXTENSIONS-en-1.0]

## Introduction

Companies want to protect their user data. They want to easily vet extensions to be safe and relevant for their users. IT administrators need to:

1. Prevent bad extensions from being installed
2. Keep extensions that users need
3. Provide limited access to user and company data

There are multiple ways to manage extensions. This guide shows you the options and helps you select the method that's right for your organization.

## Considerations for Managing Chrome Extensions

Your users need access to certain apps, sites, and extensions to do their jobs. As the IT admin, you need to protect user and company data. Developing an extension management strategy will help you navigate a path forward.

Key questions to ask:

- What regulations and compliance measures do I need to follow?
- What kind of device or website access could contravene my company's security policies?
- How much user or corporate data is stored on my users' machines?

As you make these decisions, Google provides policies that allow you to:

- Block or allow extensions based on your data protection policies
- Force install necessary extensions on your users' machines
- Manage extensions while providing them with the least amount of rights needed to work

For a long time, the only way to manage browser extensions was to manually evaluate each extension and then create allowlists and blocklists to dictate which extensions could and could not be installed on users' devices.

To make enterprise extension management more efficient, stable, and secure, Chrome Enterprise allows you to manage extensions according to permissions. Managing extensions by permissions makes it possible for IT teams to give their users the extensions they want, without putting corporate data at risk. It's the method Google's IT team uses and recommends for other enterprises.

## What are Extension Permissions?

Extensions typically require the ability to make changes on a machine or web page. These abilities are called permissions, and developers must list what permissions and access their extensions require. There are two main categories, but many extensions have both:

- Site permissions may access or modify the websites that your users visit
  - For example, modifying a webpage, accessing your cookies, modifying tabs
- Device permissions may access the machine where the browser is running and run specific actions
  - For example, access to a USB port/storage/viewing screen

There are multiple ways to manage extensions. This guide shows you the options and helps you select the method that's right for your organization.

## How do Extensions Update?

Extensions only update when Chrome is running. The update occurs within the first minutes of Chrome launching and again every 5 hours.

Extensions are updated via this process:

- A. Chrome sends a request that contains a list of installed extensions and versions to a Google server
- B. Our servers respond with a set of instructions about which extensions to update
- C. Chrome then requests the CRX files for each of the out-of-date extensions and applies the update locally

A resolution to out-of-date extensions is to either uninstall and reinstall an extension, or manually force an update of an extension through **chrome://extension > enable developer mode > click the update button.**

How extensions can become out of date:

- A. Due to a large update size, or if the user has too many extensions, the update may not complete during a short session
- B. Chrome not being launched
- C. Extension developers have chosen to limit the amount of clients that they deploy an update to
- D. If an enterprise is self-hosting an extension – this could be due to an access issue or configuration error
- E. Other issues that would be attributed to errors in the development of the extension

# Managing Extensions

Most organizations should manage extensions based on access and permissions as it's scalable, more secure, and easier to manage.

This method saves you time as you only need to set the policies once. The days of managing long allowlists and blocklists are gone. You can still include a small blocklist of extensions that should not be installed. And with the runtime hosts policy, your most important sites will be protected. Take this approach to managing your organization's extensions by following these steps:

1. Find out which extensions are installed on your users' computers
  - a. **Method 1 (Recommended):** Use [Chrome's cloud management tool](#), offered at no additional cost for your users
    - i. You will be able to see the following for each extension:
      - Installed version, install count, and if it was installed by a user or admin
      - Permissions required
      - Status (active or disabled)
    - ii. The steps to set up Chrome's cloud management tool are located [here](#)
    - iii. Once you have the console set up and your machines enrolled with cloud reporting enabled, you will be able to view all of the installed extensions under **Devices > Chrome > Apps and extensions usage report**
    - iv. Clicking on an extension will take you to the extension details page where you can see the permissions it requires, where it's installed, and other information
      - You can get more information about an extension from its page in the Chrome Web Store
      - For more information on managing extensions in Chrome's cloud management tool, check out this [YouTube video](#)
    - v. You can also use the Takeout API from Chrome's cloud management tool to export all extension data from enrolled browsers into a CSV file
      - For more information see: [Step by step guide](#) | [Blog entry](#) | [Demo video](#)
  - b. **Method 2:** Survey your coworkers and their managers, and create a list of regularly used extensions

2. Choose which sites you need to be secure:
  - a. Find out which sensitive websites or domains you need to block extensions from making changes or reading data
    - i. You will prevent access to these sites by blocking the API calls when the extension is run. These include blocking web requests, reading cookies, JavaScript injection, XHR, etc.
3. Identify which permissions could pose risks to your users:
  - a. Review the list of the extensions that you created in step one. Review the extensions that are installed and what permissions they require
    - i. **Helpful tip:** Permissions that extensions use can be vague. For your required extensions, reach out to the vendor to get more information. They should be able to detail the changes that the extension could make on machines and websites
  - b. Examine the [permissions](#) and decide which permissions you want to allow in your organization
    - i. For more information about the risks of specific extensions permissions, review this document on [permission risks](#)
4. Create a list from the data you collected, including:
  - a. **Required extensions:** This list could be broken down by department, office location, or other relevant information
  - b. **Allowlist:** Required extensions with permissions that would normally be blocked but need an exception, for example:
    - i. Extensions required by your users
    - ii. Extensions determined to not be a risk through conversations with the vendor
  - c. **Blocklist:** Extensions that will be blocked from installation and permissions that aren't allowed to run
    - i. List the websites and domains that need to be kept secure and will not have extension access
      - Compare this blocklist to others you have in place and you may find that you can relax your current blocklist policies
5. Present your list to your stakeholders and IT team for approvals



6. Test out the new policy in your lab or with a small pilot in your organization
7. Roll out these new sets of policies to employees in phases
8. Review feedback from your users
9. Repeat and fine-tune the process monthly, quarterly, or yearly

This will create a baseline of the permissions you allow and block others. Sensitive websites will be protected. Your browser security will improve with a better experience for users. Employees might be able to install extensions that they couldn't before. On your sensitive websites, they won't run unless you want them to. For steps on how to set up this method, check out these sections in the guide:

- [Block Extensions Based on Their Permissions](#)
- [Prevent Extensions from Altering Webpages](#)
- [Force Install an Extension](#)
- [Allow or Block Extensions in Group Policy](#)

For an overview on managing extensions within Chrome's cloud management tool, check out this [YouTube video that covers extension management in the Google Admin console](#).

## Overview of the Different Extension Management Policies

Many of these policies will be covered in detail in the other sections of the document but here is a review of some of the options that you currently have for managing extensions (some also apply to apps) via Windows Group Policy or via Plists on Macs:

- [ExtensionInstallAllowlist](#): These are the extensions that you have approved to be installed within your environment
- [ExtensionInstallBlocklist](#): These are the extensions that you will not allow to be installed
  - If they are installed already, they will be disabled
  - If a user tries to install them, it will be blocked
  - If a user visits the extension's page on the Chrome Web Store, the "add to Chrome" button will be red and the user will be advised that the extension is not allowed to be installed

- [ExtensionInstallForceList](#): This will silently install the extension on your user's machine
  - The user cannot disable or uninstall the extension
  - This setting will override the extension blocklist policy
- [BlockExternalExtensions](#): This setting will block extensions from external sources being installed
  - For example, if an installed application is adding an extension to Chrome via the registry, this setting will block that extension from loading
- [ExtensionAllowedTypes](#): Here you can create a list of what types of extensions and apps you will allow to be installed
  - Extensions, themes, user scripts, hosted applications, legacy packaged applications and platform applications are the values that are supported
  - You must include everything you want to allow on the list, as anything not on the list will not be installed
  - For more information on the different types, here is a link on [Extensions and Apps in the Chrome Web Store](#)
- [ExtensionInstallSources](#): This policy allows you to access older install functionality for URLs that you specify in this policy
  - Here is [a link to the URL match patterns](#) that can be used in this policy
  - Previously, users could click on a link to a .crx file and Chrome would offer to install the extension after a few warnings. This functionality was removed for security reasons after Chrome 21
- [ExtensionsSettings](#): This policy provides a varied amount of functionality, requires a JSON script to be created, and is required to be formatted in a single-line string
  - This setting can be complex and will be covered in depth in various sections of this document
    - Consider using Chrome's cloud management tool as almost all of the functionality is included without the need for writing JSON, including the ability to audit installed extensions

A note on Google's commitment to inclusive naming conventions. The following policies have been deprecated and will be removed in Chrome 97, so make sure that you switch over to the new policy by then.

- [ExtensionInstallWhitelist](#) replaced with [ExtensionInstallAllowlist](#)
- [ExtensionInstallBlacklist](#) replaced with [ExtensionInstallBlocklist](#)

# Block Extensions Based on Their Permissions

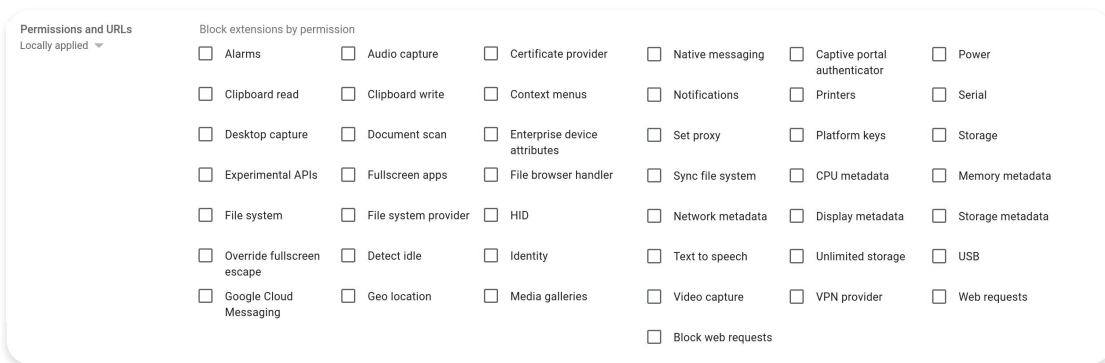
You can choose the extensions your users can install by managing permissions. Users are unable to install extensions with blocked permissions, and any installed extensions with blocked permissions will be disabled.

## Manage Extensions by Their Permissions in Chrome's Cloud Management Tool

(Windows, Mac, and Linux)

You can block extensions that require permissions that aren't allowed. For example, you could block extensions from connecting to USB devices or accessing cookies.

1. In your Google Admin console, go to **ADDITIONAL SETTINGS** under **Chrome browser > Apps and extensions > Users and browsers**
2. Select the Organizational Unit (OU) with the users you want to allow extensions for
3. Click on the additional settings gear
4. Check each permission to block or allow under the Permissions and URLs section

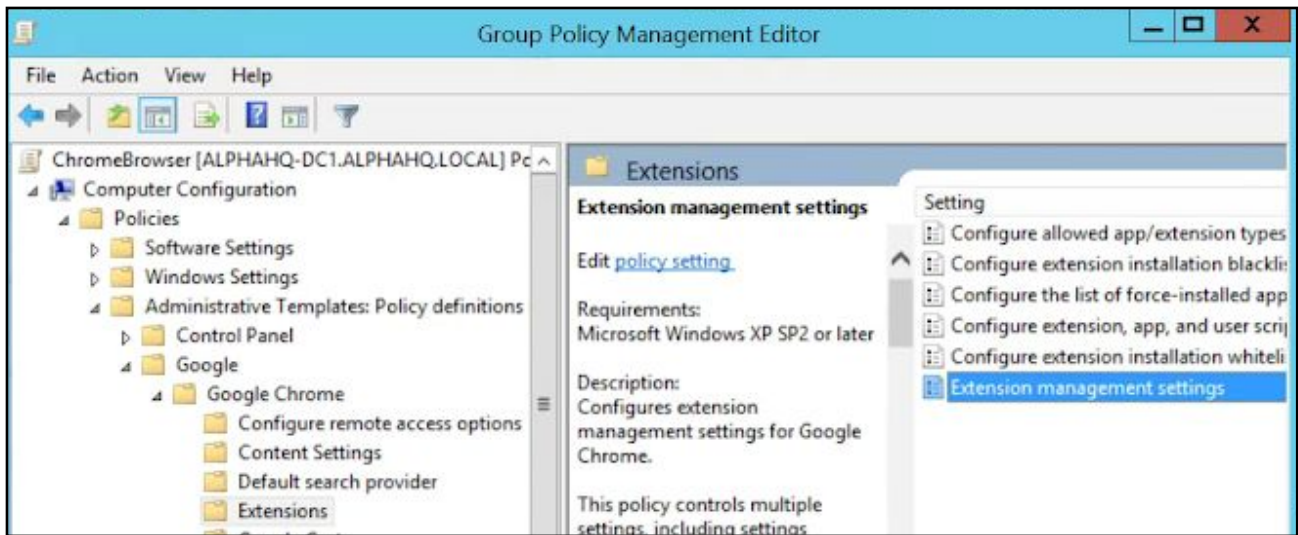


- a. You can also click on an individual extension in the Users and browsers tab and manage its permissions under **Permission and URL access > Customize permissions for this app/extension**
    - i. Note: This will override any global policy that is already applying to this extension
    - ii. For complete details on each permission, see this [list of permissions](#)
5. Click Save

## Manage Extensions by Their Permissions in Group Policy

(Windows only)

1. Browse to the Group Policy object in the Microsoft Management console
2. **Right-click > Click Edit**
3. In the Group Policy Management Console, browse to **Policies > Administrative templates > Google Chrome > Extensions > Extensions management settings**



Configure extension management settings path

4. Enable the policy, then enter the permissions that you want allowed or blocked, compressing it to a single JSON string

Format according to this example JSON data. (This example blocks any extension that needs use of USB.)

```
{
  "*" : {
    "blocked_permissions": ["usb"]
  }
}
```

Compact JSON data:

```
{"*":{"blocked_permissions":["usb"]}}
```

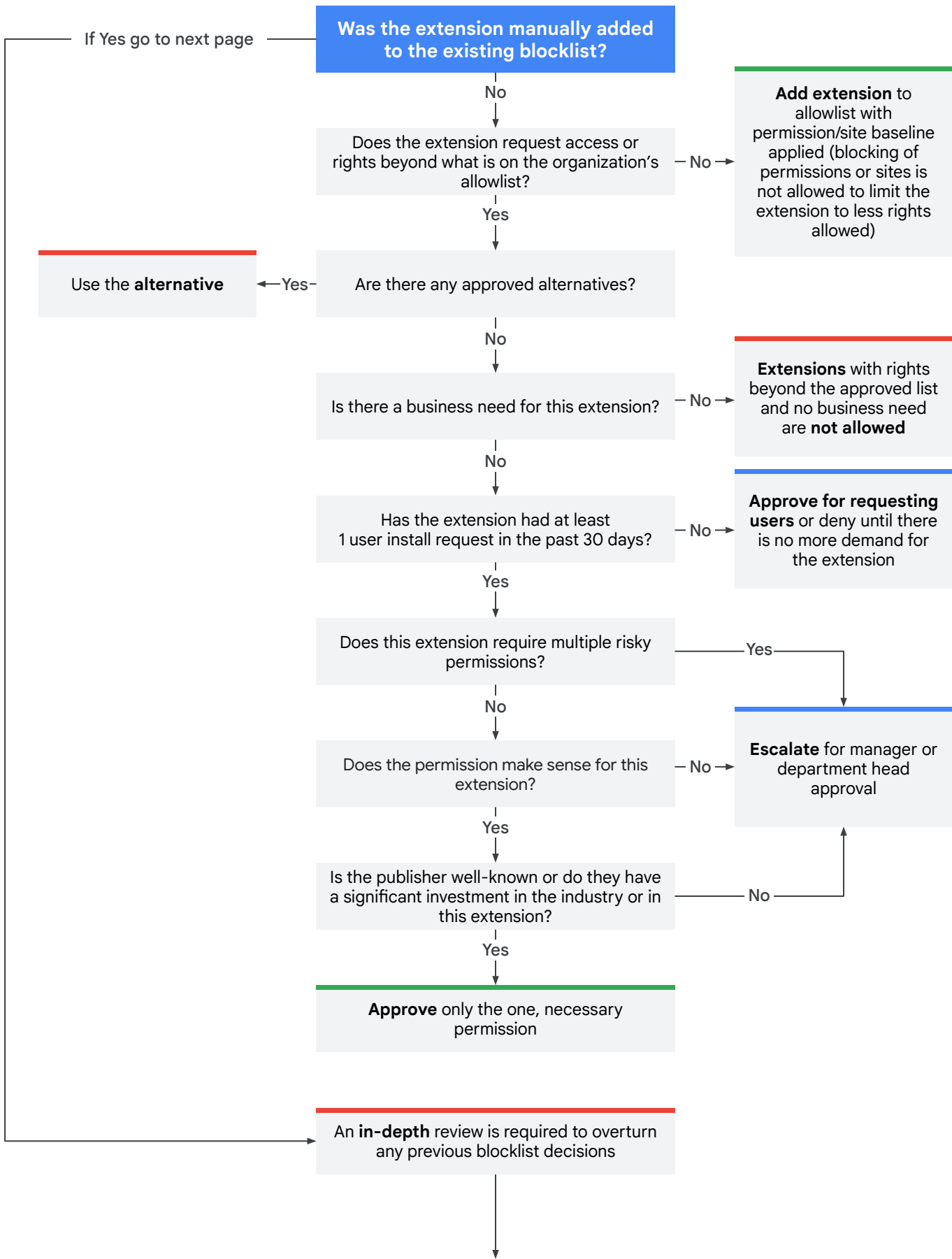
### Helpful tips:

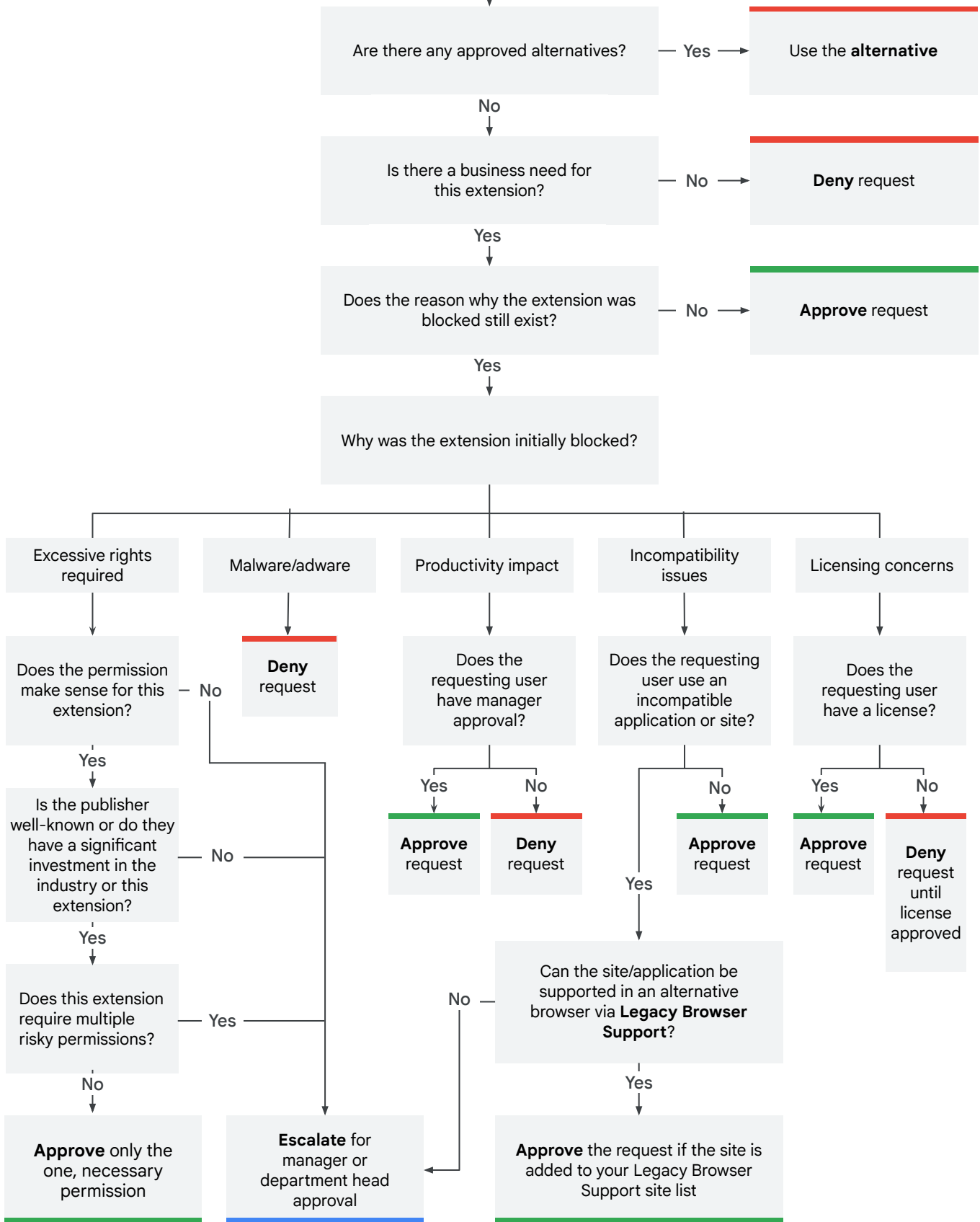
- To block all extensions that use that permission, use an asterisk (as shown above) for the extension ID
- If you want to block multiple permissions via JSON, here is an example that blocks power, printerProvider, serial, and USB for all extensions:
  - ```
{ "*": { "blocked_permissions": [ "power", "printerProvider", "serial", "usb" ] } }
```
- If you specify one extension ID, the policy will only apply to that extension
- In the example above, replace the \* with the extension ID
- You can block more than one, but they need to be separated into their own entries in the JSON string
  - Follow step 3 of [this article](#) to find the extension ID

## Creating an Exception Process for Extensions that Require Risky Permissions

There might be a business need for extensions that require permissions that you have deemed to be too risky to be run in your environment. To give you an idea of what an exception workflow might look like, here is an example workflow for a requested extension that requires a currently blocked extension.

## Start Here





N.B. this is an example workflow only – your organization may require following a different process.

# Managing Extensions by the ExtensionSettings Policy

Windows offers multiple ways to manage extensions. A common way is to set multiple policies with a JSON string or in the Windows Registry using the [ExtensionSettings policy](#).

**Helpful tip:** This policy is supported on [Mac](#), [ChromeOS](#), and [Linux](#). [The policy page](#) has example values for these other platforms.

This policy can control settings such as the update URL, where the extension will be downloaded from for initial installation, and blocked permissions, which are not allowed to run. For more info, read the [ExtensionSettings full description](#). There is also further information in these help articles: [Configure ExtensionSettings policy](#) and [App and extension policies](#).

You can decide if you want to set all extension management settings via this policy or through individual policies.

- The runtime allowed/blocked hosts setting (blocking extensions on specific websites) can be set via GPO within the ExtensionSettings policy
  - It can also be set via [Chrome's cloud management tool](#)
- Note that the ExtensionSettings policy can overwrite other policies that you have elsewhere in group policy, including:
  - [ExtensionAllowedTypes](#)
  - [ExtensionInstallAllowlist](#)
  - [ExtensionInstallForcelist](#)
  - [ExtensionInstallSources](#)
  - [ExtensionInstallBlocklist](#)

The ExtensionSettings policy is set by one of these 2 methods:

- [Windows Registry](#)
- [JSON string in Windows Group Policy Console](#)

## Helpful tips:

- Use a JSON checker to ensure proper formatting before implementing the policy
- If you struggle with getting the JSON formatted correctly, you can use the registry key method and Chrome will convert it to JSON within **chrome://policy** within the browser on the target machine
  - Just copy that JSON and you can apply it via GPO via the ExtensionSettings policy
  - You can also use this method through setting ExtensionSettings via Chrome's cloud management tool and copying the JSON output



## Configure the ExtensionSettings Policy Using the Windows Registry

The ExtensionSettings policy needs to be written to the registry under:

```
HKLM\Software\Policies\Google\Chrome\ExtensionSettings\
```

- It's possible to use HKCU instead of HKLM by configuring the equivalent path with GPO
- The keys can be created with your chosen method on your user's machine

For Chrome, all settings will start under this key:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Google\Chrome\ExtensionSettings\
```

The next key that you will create is for the scope of the policy. Name the key after the extension ID for applying to one extension. Name the key with an asterisk to apply to all extensions. For example, use the following location for settings that apply just to the Google Translate extension:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Google\Chrome\ExtensionSettings\aapbdbdom  
jkkjkaonfhkkikfgjllcleb
```

For settings that apply to all extensions, use this location:



```
HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Google\Chrome\ExtensionSettings\*
```

Different settings will require different formats, depending on whether they are a string or an array of strings. Array values require [ " value " ]. String values can be entered without the [ " " ]. The list of which settings are arrays or strings:

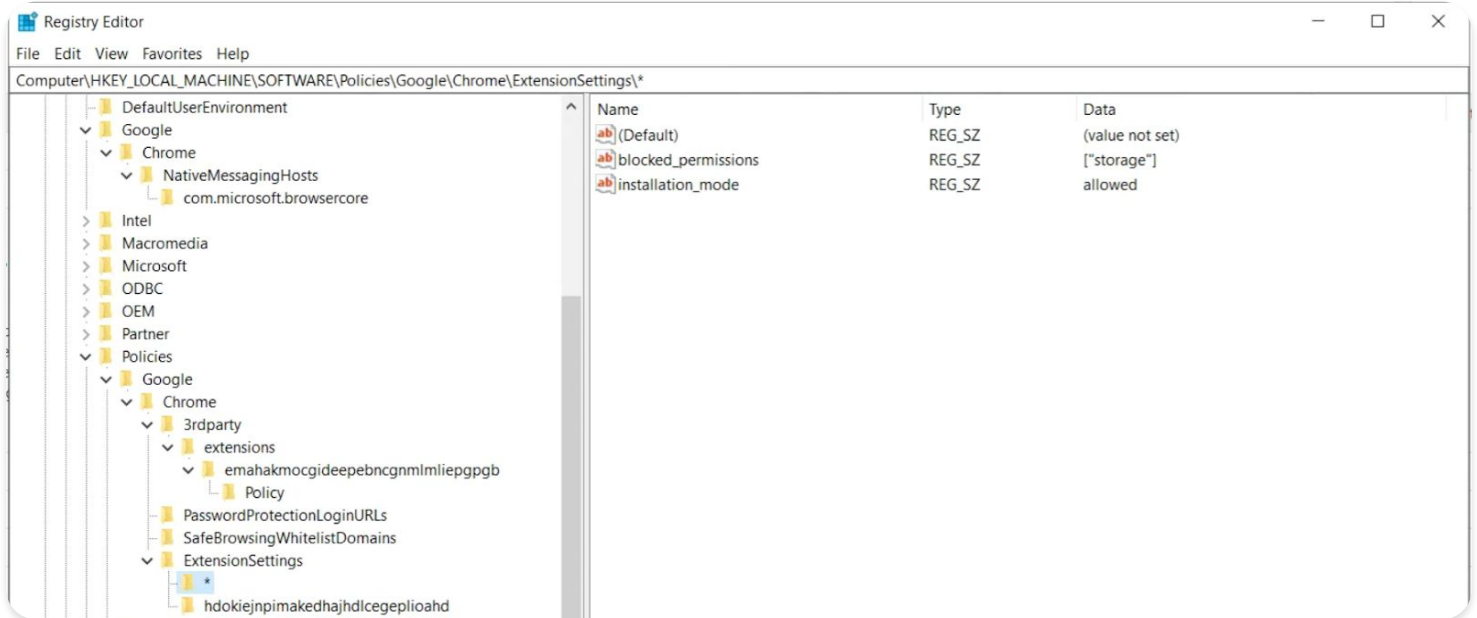
- Installation\_mode = String
- update\_url = String
- blocked\_permissions = Array of strings
- allowed\_permissions = Array of strings
- minimum\_version\_required = String
- runtime\_blocked\_hosts = Array of strings
- runtime\_allowed\_hosts = Array of strings
- blocked\_install\_message = String

If you want to set multiple values in a single string (like blocked permissions), here is an example of that syntax:

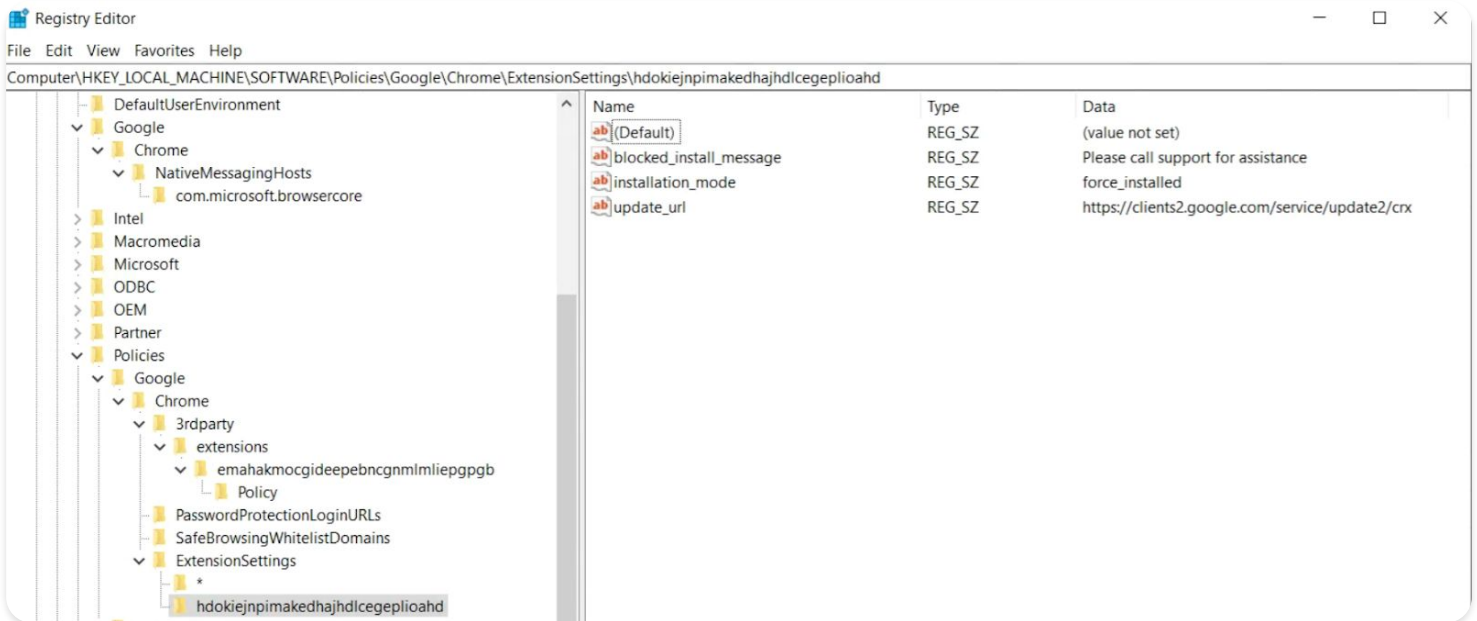
- [ "power", "printerProvider", "serial", "usb" ]

| Name                                                                                                   | Type   | Data                                            |
|--------------------------------------------------------------------------------------------------------|--------|-------------------------------------------------|
|  (Default)           | REG_SZ | (value not set)                                 |
|  blocked_permissions | REG_SZ | [ "power", "printerProvider", "serial", "usb" ] |

Examples of what the keys look like within the registry:



The default (\*) scope key and its values



The default (\*) scope key and its values

Here, the keys set in the registry are converted to JSON with the policy in **chrome://policy** within the browser:

Chrome policies

| Applies to | Level     | Source   | Policy name                                  |
|------------|-----------|----------|----------------------------------------------|
| Machine    | Mandatory | Platform | <a href="#">DefaultBrowserSettingEnabled</a> |
| Machine    | Mandatory | Platform | <a href="#">ExtensionSettings</a>            |

```
{
  "*": {
    "blocked_permissions": [ "storage" ],
    "installation_mode": "allowed"
  },
  "hdokiejnpimakedhajhdcegeplioahd": {
    "blocked_install_message": "Please call support for assistance",
    "installation_mode": "force_installed",
    "update_url": "https://clients2.google.com/service/update2/crx"
  }
}
```

## Configure Using a JSON String in Windows Group Policy Console

The steps to use the ExtensionSettings policy using GPO assume you have already imported the [ADM/ADMX for Chrome Policies](#).

For other OS platforms, check: [Mac](#) | [Linux](#) | [ChromeOS](#)

1. Within the GPO management editor, go to **Google Chrome > Extensions > Extensions management setting policy**
2. Enable the policy and enter its compact JavaScript Object Notation (JSON) data in the text box as a single line with no line breaks
3. You can use a third-party JSON compression tool to validate policies and compact them into a single line

### Properly Formatting JSON for the ExtensionSettings Policy:

To use this method, you need to understand the 2 parts to this policy – the **default** and the **individual** scope. The default scope applies to all extensions. The individual scope is applied to the specified extension only.

The default scope is identified by the asterisk (\*). This example defines a default scope and a single individual extension scope:

```
{
  "*": {},
  "nckgahadagoaajjgafhacjanaoiihapd": {}
}
```

An extension will only get its settings from one scope. If there's an individual scope for that extension, those settings will apply. If no individual extension scope exists, it will use the default scope.

Here is an example JSON that blocks any extension from running on .example.com and blocks any extension that requires the permission "USB":

```
{
  "*": {
    "runtime_blocked_hosts": ["*://*.example.com"],
    "blocked_permissions": ["usb"]
  }
}
```

Compact JSON data:

```
{"*":{"runtime_blocked_hosts":["*://*.example.com"],"blocked_permissions":["usb"]}}
```

## Reference Examples with Example Values for Installation Management of Extensions:

- “allowed” (default)  
Your user can install the extension from the Chrome Web Store  
Example JSON:  

```
{ "*" : { "installation_mode" : "allowed" } }
```
- “blocked”  
Your user can't install the extension from the Chrome Web Store  
Example JSON:  

```
{ "*" : { "installation_mode" : "blocked" } }
```
- “blocked\_install\_message”  
Here you can specify a custom message to display when installation is blocked  
Example JSON - blocked\_install\_message:  

```
{ "*" : { "blocked_install_message" : ["Call IT(408 - 555 - 1234) for an exception" ] } }
```
- “force\_installed”
  - The extension is automatically installed without your user's interaction
  - Your user can't disable or remove the extension.

```
{ "*" : { "installation_mode" : "force_installed" } }
```
- “normal\_installed”  
The extension is automatically installed without your user's interaction, but they can disable the extension  
<https://chromeenterprise.google/policies/#ExtensionSettings>
- “removed”  
(Chrome version 75 or later) Users can't install the extension. If users previously installed the extension, Chrome removes it.  

```
{ "*" : { "installation_mode" : "removed" } }
```
- “toolbar\_pin”  
Controls if the extension icon is pinned to the toolbar. You can set it to:  
Force\_pinned – The extension icon is pinned to the toolbar and visible at all times.  
The user can't hide it in the extension menu.  
Default\_unpinned – The extension starts hidden in the extension menu, and the user can pin it to the toolbar  
If you do not set this field, it defaults to the default\_unpinned behavior  

```
{ "*" : { "toolbar_pin" : "forced_pinned" } }
```

If an extension uses the `installation_mode` feature, then another field “`update_url`” must also be defined, pointing to where the extension can be installed from.

- If the extension you’re downloading is hosted on the Chrome Web Store, use `"https://clients2.google.com/service/update2/crx "`
- If you’re hosting the extension on your own server, put the URL where Chrome can download the packed extension (.crx file)  
Example JSON - `force_installed` extension with `update_url`:  

```
{"nckgahadagoaajjgafhacjanaoiihapd": {"installation_mode": "force_installed", "update_url": "https://clients2.google.com/service/update2/crx "}}
```
- Since Chrome 89, you can also use the `override_update_url` setting to specify that Chrome uses the URL in the `update_url` field or the update URL specified in the `ExtensionInstallForcelist` policy for subsequent extension updates
  - If this is not set or is set to `false`, Chrome uses the URL specified in the extension's manifest for updates instead

## Prevent Extensions from Altering Webpages

This setting prevents extensions from changing and reading data from your most sensitive websites.

This policy will block extensions from:

- Injecting scripts into your websites
- Reading the cookies
- Making web-request modifications

This setting doesn’t prevent users from installing or removing extensions. It only prevents extensions from altering websites that you specify.

There are two settings you can use for this feature

- **Runtime\_blocked\_hosts** – Extensions are blocked from interacting with these hosts
- **Runtime\_allowed\_hosts** – Extensions may interact with hosts on this list even if defined in `runtime_blocked_hosts`

**Helpful Tip:** Each instance of `runtime_blocked_hosts` and `runtime_allowed_hosts` can have at most 100 Host Patterns. If you define more than that, your policy will be invalid.

### Chrome's Cloud Management Tool

Blocking by runtime host is simpler within [Chrome's cloud management tool](#) than in GPO. It requires no JSON and is as simple as entering the URL that you want to block in the extension settings. To set this up, you need to enroll your browser devices into Chrome's cloud management tool. The feature is offered at no additional cost. The steps for enrollment are [located here](#).

1. In your Google Admin console, go to  
**Chrome browser > Apps and Extensions > Users and browsers**
2. Select the Organizational Unit (OU) with the users you want to allow extensions for
3. Click on the additional settings gear
4. Enter the URL of the sensitive websites that you do not want the extensions to run on in the "runtime blocked hosts" section. For syntax info, review [syntax for blocked or allowed URLs](#).
  - a. You can enter multiple URLs by clicking enter after each URL for a new entry
  - b. You can also click on an individual extension and set allowed and blocked hosts in the permissions and URL access section
    - i. Note: This will override any global policy that is already applying to this extension
    - ii. There is also an allowed\_hosts section for exceptions for URLs that are listed in the runtime block hosts section
5. Click **Save**

Runtime blocked hosts

**\*://\*.sensitivesite.com**

---

This is a list of patterns for matching against hostnames. URLs that match one of these patterns cannot be modified by apps and extensions. This includes injecting Javascript, altering and viewing webRequests / webNavigation, viewing and altering cookies, exceptions to the same-origin policy, etc.  
The format is similar to full URL patterns except no paths may be defined. e.g.  
\*://\*.example.com

Runtime allowed hosts

---

Hosts that an extension can interact with regardless of whether they are listed in "Runtime blocked hosts".  
This is the same format as "Runtime blocked hosts".

Runtime hosts section in

**Chrome browser > Apps and extensions > Users and browsers > Additional settings**

## GPO

These instructions are for managing this GPO on Windows machines.

For other platforms, check: [Mac](#) | [Linux](#)

Within the ExtensionSettings policy, you can set the following settings to block (or allow) alterations of websites or domains:

- `Runtime_blocked_hosts`  
This setting blocks extensions from making changes or reading data from your chosen websites
- `Runtime_allowed_hosts`  
This setting allows extensions to make changes or read data from your chosen websites

The format for specifying your site(s) in the JSON string in either policy is:

```
[http|https|ftp|*]://[subdomain|*].[hostname|*].[eTLD|*] [http|https|ftp|*],
```

Note: `[hostname|*]`, and `[eTLD|*]` sections are required, but `[subdomain|*]` section is optional.

Examples of valid host patterns and matching patterns:

| Valid host patterns             | Matches                                                                    | Doesn't match                                                                       |
|---------------------------------|----------------------------------------------------------------------------|-------------------------------------------------------------------------------------|
| <code>*://*.example.*</code>    | <code>http://example.com</code><br><code>https://test.example.co.uk</code> | <code>https://example.google.com</code><br><code>http://example.google.co.uk</code> |
| <code>http://example.*</code>   | <code>http://example.com</code><br><code>http://example.ly</code>          | <code>https://example.com</code><br><code>http://test.example.com</code>            |
| <code>http://example.com</code> | <code>http://example.com</code>                                            | <code>https://example.com</code><br><code>http://test.example.co.uk</code>          |
| <code>*://*</code>              | All URLs                                                                   |                                                                                     |

Here is a sample of a JSON string that blocks access for a single extension. This string prevents a single extension from augmenting a specific site:

```
{
  "aapbdbdomjkkjkaonfhkkikfgjllcleb": {
    "runtime_blocked_hosts": ["*://*.importantwebsite"]
  }
}
```



**Compact JSON data:**

```
{"aapbdbdomjkkjkaonfhkkikfgjllcleb":  
{"runtime_blocked_hosts":["*://*.importantwebsite"]}}
```

Here is a sample for blocking multiple sites for all extensions:

```
{  
  "*": {"runtime_blocked_hosts": [ "*://*.importantwebsite.com",  
"*://*.importantwebsite2.com" ]  
}
```

**Compact JSON data:**

```
{"*":{"runtime_blocked_hosts":["*://*.importantwebsite.com","*://*.importantwebsite2.com"]}}
```

For multiple extensions, separate each into its own entry for each app ID that you want to block. Here's an example of how to block two extensions from running on the same domain:

```
{  
  "aapbdbdomjkkjkaonfhkkikfgjllcleb": {  
    "runtime_blocked_hosts": ["*://*.importantwebsite"]  
  },  
  "bfbmjmiodbnnpllbbbfblcplfjjepjdn": {  
    "runtime_blocked_hosts": ["*://*.importantwebsite"]  
  }  
}
```

**Compact JSON data:**

```
{"aapbdbdomjkkjkaonfhkkikfgjllcleb": {"runtime_blocked_hosts":  
["*://*.importantwebsite"]}, "bfbmjmiodbnnpllbbbfblcplfjjepjdn":  
{"runtime_blocked_hosts": ["*://*.importantwebsite"]}}
```

# Allow or Block Extensions in the Google Admin Console

Admins can control which extensions your users can install by creating allowlists and blocklists. You can allow users to install any app or extension, or set policies that block or allow apps for all or some users.

The following steps assume you're familiar with changing settings in your Google Admin console.

## Allow All Extensions Except Those You Want to Block

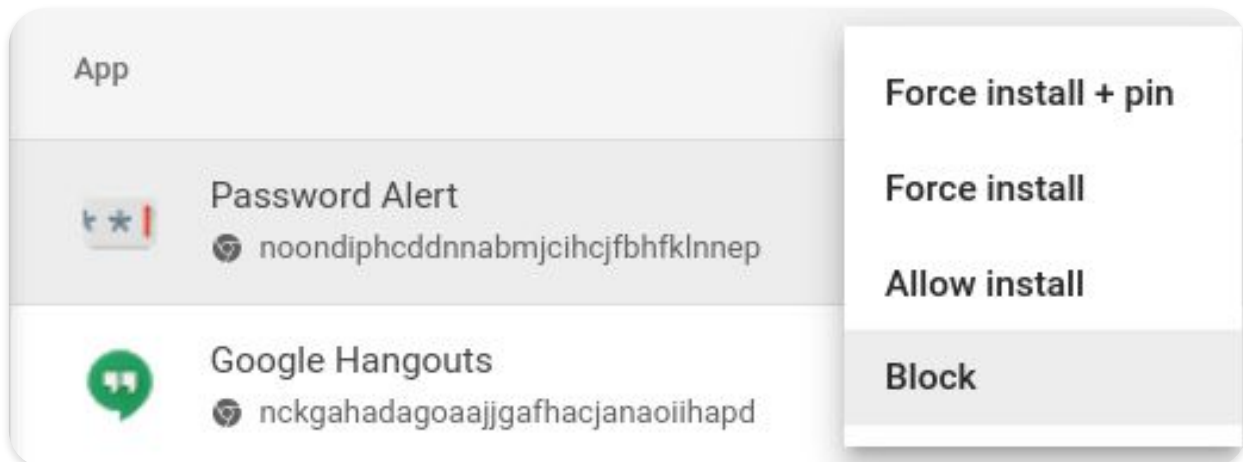
1. In your Google Admin console, go to  
**Chrome browser > Apps and extensions > Users and browsers > Additional settings**
2. Select the Organizational Unit (OU) that you want to allow extensions for, on the left
3. Scroll down to the Allow/block mode section under Chrome Web Store, click edit and select the option to **Allow all apps, admin manages blocklist**
4. Click **Save**
5. Click on the Users and browsers tab to return to the previous page
6. Add each extension you want to block by clicking on the yellow plus mark in the bottom right
7. Choose your method of adding it into the console (add from Chrome Web Store, add by extension ID, add by URL)
8. Select the dropdown by the extension and select **Block**
9. Click **Save**

## Block All Extensions Except Those You Want to Allow

1. In your Google Admin console, go to **Chrome browser > Apps and extensions > Users and browsers > Additional settings**
2. Select the Organizational Unit (OU) that you want to block extensions for, on the left
3. Scroll down to the Allow/block mode section under Chrome Web Store, select the option to **Block all apps, admin manages allowlist**
4. Click **Save**
5. Click on the Users and browsers tab to return to the previous page
6. Add each extension you want to allow by clicking on the yellow plus mark in the bottom right
7. Choose your method of adding it into the console (add from Chrome Web Store, add by extension ID, add by URL)
8. Select the drop-down by the extension and select **Allow install**
  - a. “You can also force install the extension to your user machines by selecting **Force install**
9. Click **Save**

## Block or Allow a Single Extension

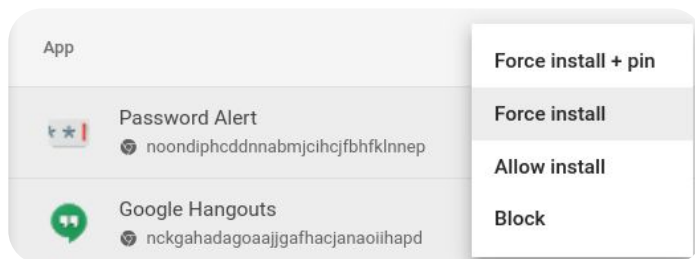
1. In your Google Admin console, go to  
**Chrome browser > Apps and extensions > Users and browsers**
2. Select the Organizational Unit (OU) that you want to allow or block the extension for
  - Note: The Organizational Unit (OU) will inherit the settings of the parent Organizational Unit (OU), but you can override per sub Organizational Unit (OU)
3. Select the extension you want to block or allow or add it in (see steps 6 & 7 of the previous section)
4. In the installation policy column, select **Block**, **Force install**, or **Allow install**
5. Click **Save**



## Force Install an Extension

If you know that a user requires an extension, you can install it for them. If you force install an extension, it will grant all the permissions it needs to run. The user will also not be able to remove it and it will be installed silently. If you remove an extension from the force install list, it will be removed from the user's machine.

1. In your Google Admin console, go to  
**Chrome browser > Apps and extensions > Users and browsers**
2. Select the Organizational Unit (OU) that you want to force install extensions to
3. Select the existing extensions you want to force install or add them in
  - a. To add the extensions you want to install, click on the yellow plus mark in the bottom right
  - b. Choose your method of adding it into the console (add from Chrome Web Store, add by extension ID, add by URL)
4. Select the extensions that you want to force install and in the installation policy column, select Force install from the drop-down menu



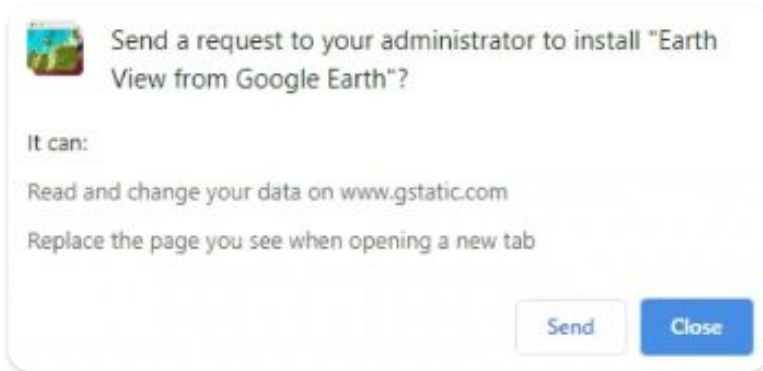
5. Click **Save**

**Helpful tip:** You can create a custom Chrome Web Store collection of admin selected extensions that will be displayed to your users. This setting requires your users to be signed in to Google identity using corporate credentials.

- This setting can be found within the Google Admin console under **Chrome browser > Apps and extensions > Users and browsers > Additional settings > Chrome Web Store homepage > Use the Chrome Web Store collection**
  - Then you can either have all of your extensions show up within this page or you can click on individual extensions under the Users and browsers section and select the **Include in Chrome Web Store collection** toggle

## Allow Users to Request Extensions

As an admin, you can use the Google Admin console to enable users to request the extensions they need in the Chrome Web Store. Then you can allow, block, or automatically install extensions that users request.



**Helpful tip:** This feature works as an allow/blocklist. When the feature is turned on, all extensions are blocked by default. To prevent any issues, it is recommended to follow this process:

1. Discover what extensions your users are currently using in the [extension takeout report](#) in Chrome's cloud management tool
  - For more information, check out this [YouTube video on setting up the Takeout API](#)
2. Create a list of essential extensions ([GPO](#) or [Google Admin console](#)) based on the data gathered in step 1
3. Turn on the extension workflow feature under **Chrome browser > Apps and extensions > Users and browsers > Additional settings > Allow/block mode and click the Edit button**
4. Under Chrome Web Store, select **"Block all apps, admin manages allowlist, users may request extensions"** from the drop-down menu
  - We recommend that you first apply settings to a small number of users and devices in a test Organizational Unit (OU) to prevent end user issues and gather feedback
  - Once you feel ready, you can apply it to your entire organization

5. Approval and denial requests are managed under **Chrome browser > Apps and extensions > Requests**
6. Click the row of the extension request you want to review
7. Here you can review details about the extension and select the installation policy from the drop-down menu:
  - a. Force install – installs the extension silently and it cannot be removed
  - b. Allow install – lets users install the extension
  - c. Block – prevents users from installing the extension and removes it from users who have it installed

For more information about this feature, please review [the help center article for extension workflows](#) or this [YouTube video](#).

### Edit Allow/block mode setting

Play Store  
**Allow all apps, admin manages blocklist** ▼

Chrome Web Store  
**Block all apps, admin manages allowlist, users may request extensions** ▼

**Warning:** Any existing extension that is not explicitly on the allowlist will be blocked.

CANCEL   INHERIT   **SAVE**

Enabling extension workflows in the Google Admin console

# Chrome App and Extensions Usage Report

As an admin, you can use your Google Admin console to see details about apps and extensions that are installed on users' enrolled Chrome browsers and ChromeOS devices. For more information about how to view these reports refer to this help center article [View app and extension usage details](#) that provides additional information.

The screenshot shows the Google Admin console interface. On the left is a navigation menu with 'Admin' at the top and 'Reports' expanded to show 'Apps & extensions usage'. The main content area is titled 'Chrome Apps And Extensions Usage Report' and shows a table of 7 Chrome apps and extensions. The table has columns for App name, App type, Install type, Chrome Web Store, Installs, Permissions, Manifest versions, and App id. The data rows are as follows:

| App name                        | App type         | Install type | Chrome Web Store | Installs | Permissions | Manifest versions | App id  |
|---------------------------------|------------------|--------------|------------------|----------|-------------|-------------------|---------|
| Google 翻訳                       | Chrome Extension | Multiple     | Published        | 3        | 3           | 2                 | aapbdt  |
| Google Docs Offline             | Chrome Extension | Multiple     | Published        | 2        | 5           | 2,3               | ghbmn   |
| Google Corporate Extension Repc | Chrome Extension | Admin        | Published        | 1        | 11          | 2                 | abjoigi |
| Chrome RDP for Google Cloud Plk | Chrome App       | Normal       | Published        | 1        | 7           | 2                 | mpbbn   |
| Google OpenVPN NG (User Interf) | Chrome Extension | Normal       | Published        | 1        | 13          | 2                 | dkjleh  |
| Google OpenVPN NG (Managem)     | Chrome App       | Normal       | Published        | 1        | 2           | 2                 | dfobjhf |
| Google BeyondCorp Extension (pr | Chrome Extension | Admin        | Published        | 1        | 8           | 3                 | aihpiht |

You can also click on a extension in this report and view third-party data about the app or extension risk score. Google makes no guarantee about the data provided by third-party companies. Google does not host, control, or modify risk assessment data.

For more information about how to view these reports refer to this help center article [View additional details of an app or extension>Risk assessment](#) that provides additional information.

## Risk assessment

The risk assessment scores are provided by the 3rd parties below. Google makes no guarantee about the data provided by 3rd party companies. Google does not host this data. [Learn more](#)

| Version         | Installs | CRXcavator            | Spin.AI                    |
|-----------------|----------|-----------------------|----------------------------|
| 1.75.4 (latest) | 2        | ● <a href="#">391</a> | ● <a href="#">82</a> / 100 |
| 1.65.0          | 2        | ● <a href="#">392</a> | ● <a href="#">75</a> / 100 |
| 1.63.3          | 1        |                       | ● <a href="#">74</a> / 100 |
| 1.46.0          | 1        | ● <a href="#">394</a> | ● <a href="#">74</a> / 100 |
| 1.44.2          | 1        | ● <a href="#">394</a> | ● <a href="#">82</a> / 100 |
| 1.33.0          | 1        | ● <a href="#">395</a> |                            |
| 1.31.0          | 1        | ● <a href="#">395</a> |                            |
| 1.7             | 249      | ● <a href="#">427</a> |                            |



# Allow or Block Extensions in Group Policy

**Before you begin:** The following steps assume that you're already using Chrome's cloud management tool. For more on how to deploy Chrome on Windows, refer to the [Chrome Browser Deployment Guide \(Windows\)](#). For Mac deployment and policy management, [follow these steps](#).

For Windows, there are two types of policy templates: an ADM and ADMX template. Ensure that you verify which type you can use on your network. The templates show which registry keys you can set to configure Chrome and what the acceptable values are. Chrome looks at the values set in these registry keys to determine how to act.

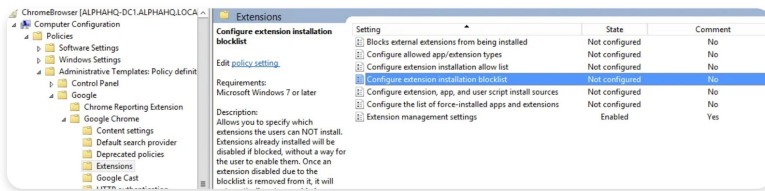
1. Download Chrome policy templates
  - a. Windows templates, as well as common policy documentation for all operating systems, can be found [via this link](#)
2. Open the ADM or ADMX template you downloaded:
  - a. Go to **Start > Run: gpedit.msc**
  - b. Go to **Local computer policy > Computer configuration > Administrative templates**
  - c. Right-click **Administrative templates** and select **Add/remove Templates**
  - d. Add the chrome.adm template through the dialog

Afterward, if it's not already there, a Google or Google Chrome folder will appear under Administrative Templates.

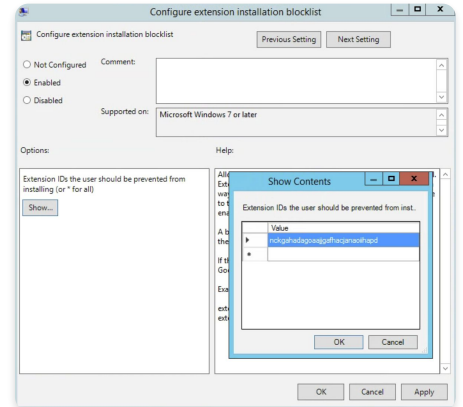
- If you add the ADM template on Windows 7 or 10, it will appear under Classic Administrative Templates/Google/Google Chrome

## Allow All Extensions Except Those You Want to Block

1. In the Group Policy Console, open the template you just added
2. Browse to **Google > Google Chrome > Extensions > Configure extension installation blacklist**
3. In the setting, select **Enabled**
4. Click **Show**
5. Enter the app ID of the extensions that you want to block



Path to extension management policies



Configure extension installation blacklist

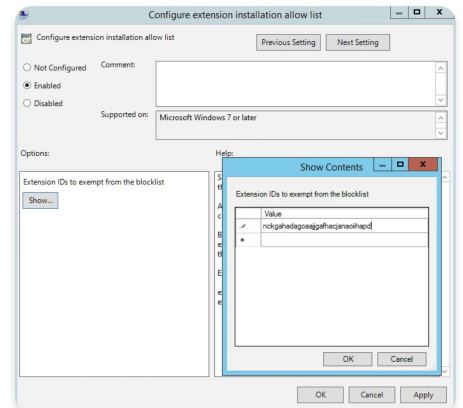
### Helpful tips:

- If you can't find the app ID of an extension, view it in the Chrome Web Store
  - The specific extension and the app ID will be at the end of the URL in the Chrome omnibox



App ID example located after Google Translate

- Enter \* into the policy to prevent any extensions from being installed
  - You can use this with the Configure extension installation allow list policy
  - This allows certain extensions to be installed by your users and blocks the rest
- You can add an extension to the blacklist that is already installed on a user's machine
  - It will disable the extension and prevent the user from re-enabling it
  - It will be disabled but not uninstalled



Configure extension installation allow list

## Block or Allow a Single Extension

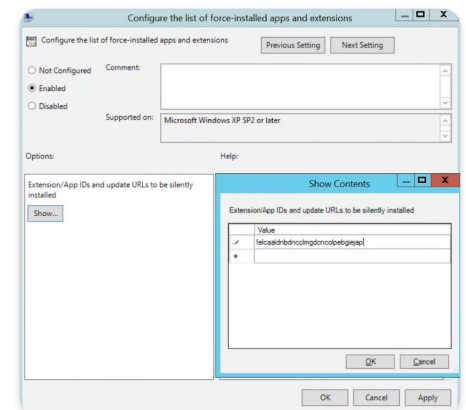
To block an extension, add the app ID of the extension you want blocked to the Configure extension installation blacklist policy. All of your other extensions will be allowed to be installed.

To allow only one extension:

1. In the content section in the Configure extension installation block list policy, enter \*
  - a. This will block all extensions on the list from being installed
2. Add the app ID of the allowed extension to the Configure extension installation allow list policy

## Force Install an Extension

1. In the Group Policy Console, browse to **Google > Google Chrome > Extensions > Configure the list of force-installed apps and extensions**
2. Select **Enabled**
3. Click **Show**
4. Enter the app ID or IDs of the extension or extensions you want to force install



Configure the list of force-installed apps and extensions

The extension will be installed silently with no need for a user to take action. The user won't be able to uninstall or disable the extension. This setting will override any blacklist policy that you might have enabled.

## Validating Your Policy

To make sure your policy is valid and works as expected, apply it to a test machine. From the test machine, follow these steps:

1. Browse to **chrome://policy**
2. Click the “Reload policies” button
3. In the top right-hand corner of the page is the policy filter; type in “ExtensionSettings” to only show this policy
4. Check the “Show policies with no value set” checkbox
5. Make sure the “Status” for your policy shows “OK”
6. Expand the policy by clicking “Show value” and make sure it isn’t empty – this confirms you have a valid policy

## Self-Hosting Your Extensions

The [Chrome Web Store](#) hosts extensions and provides many security features

- Automated and manual code scans – this prevents malicious code from being sent to your users

However, there’s also an option to host your extensions on your own server separate from the Chrome Web Store. Here are some of the pros and cons of this method:

### Pros:

- You’re outside the rules and requirements of the Chrome Web Store, so you’ll face less scrutiny and risk of the extension being removed for violating the terms of service

### Cons:

- The self-hosting method requires more setup and requires you to host your own file server for extension files
- Validating the security of extensions and keeping them updated can be tough, whereas the Chrome Web Store does this automatically

If you choose to self-host your extensions, this section explains how to package an extension and host it without using the Chrome Web Store. It also includes instructions on how to deploy these extensions to your devices and users.

## Alternatives to Self-Hosting Extensions

### Extension publishing options

Developers can publish extensions in the Chrome Web Store as public, private, or unlisted. As an alternative to self-hosting, consider publishing internal extensions as private. Here is a chart with more information about the pros and cons of each option:

|                 | Present in Chrome Web Store search | Requires sign-in                | Supported in Chrome Enterprise Core |
|-----------------|------------------------------------|---------------------------------|-------------------------------------|
| <b>Public</b>   | Yes                                | No                              | Yes                                 |
| <b>Private</b>  | No                                 | Yes                             | Yes                                 |
| <b>Unlisted</b> | No                                 | No - users need link to install | Yes                                 |

For more information, please read [this blog](#) on how to publish your extensions out of the view of the public, without the need to self-host your extensions.

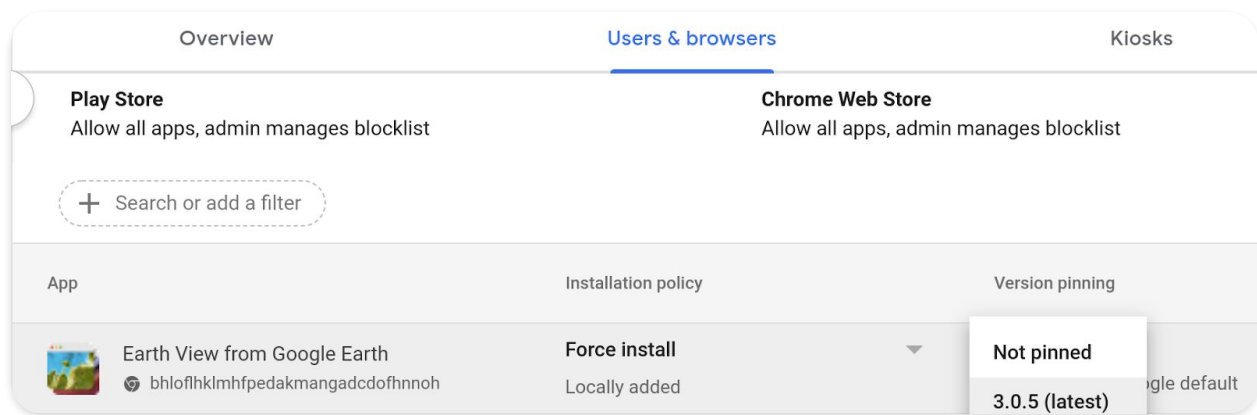
- If you are managing your extensions via the Google Admin console, you need to configure the Chrome Web Store permissions setting to enable private extensions to show up for your users
  - This is found in the Google Admin console under **Devices > Chrome > Apps and extensions > Additional settings > Chrome Web Store permissions > set to Allow** to enable users to publish private apps that are restricted to your domain on the Chrome Web Store

## Pin an Extension to a Specific Version in the Google Admin Console

The Google Admin console now offers other options for extension management, including the ability to pin to a version of an extension directly in the console. This provides more stability for enterprises that need to stick to a certain version of an extension. It is not recommended to pin to older versions as a best practice. If you do pin to an older version, make it a temporary measure to ensure that you have the latest functionality and security updates. This feature is only available for force-installed extensions.

For more information, please read [this article from the Chrome Enterprise and Education Help Center](#).

1. In your Google Admin console, go to **Chrome browser > Apps and extensions > Users and browsers**
2. Select the Organizational Unit (OU) that contains the extension that you want to pin
3. Select the existing extensions (or add a new one) you want to manage by version and under the version pinning column, select the version that you want to pin to from the drop-down menu and click **Save**
  - a. By pinning an app or extension, it will no longer get updates, including security and compatibility updates
  - b. You can also only pin to the current version of the extension that is present on the Chrome Web Store at the time of setup
  - c. You can also pin self-hosted apps and extensions and update the URL within the Google Admin console
    - i. [Read more about pinning self-hosted apps in this help center entry](#)



Version pinning in the Google Admin console

## Requirements for Self-Hosting Extensions

To host your extension, you will need your own web hosting services for the extension and its manifest file. This hosting location shouldn't require authentication. It needs to be accessible by devices wherever they are used. Keep this in mind if you want to host the file on your internal repository.

The following steps assume that you've already created your extension, have experience with XML files, and that you have knowledge about Group Policy and using the Windows registry. These steps do not apply to a third-party extension that you did not develop. If you want to self-host a third-party extension, you should discuss this with the extension vendor directly.

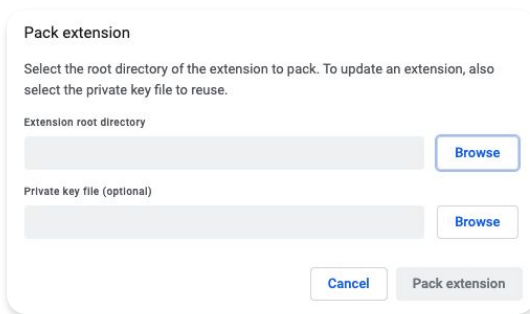
## Packaging Your Extension

Extensions first need to be packed into a CRX file. If the extension isn't packed as a CRX file, here are the steps:

1. Go to **chrome://extensions** in the Chrome address bar and check the box for **Developer mode**



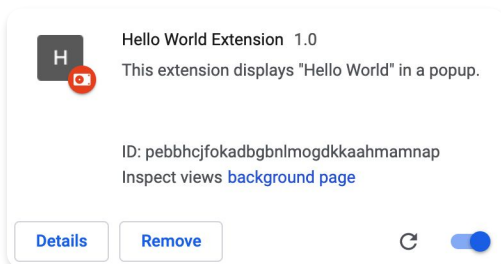
2. Once you're in Developer mode, create the CRX file by clicking **Pack extension**



3. Select the directory where your source is to create your CRX file and PEM file

**Helpful tip:** Keep the PEM file securely stored, because this is the key to your extension. You'll need it for future updates.

4. Drag the CRX into your extensions window and make sure that it loads
  - a. On Windows and Mac, the extension will be disabled by default, but on Linux it will not
5. Test the extension and take note of the ID field and version number, which you'll need later on



6. Place the CRX file in the host location where your users or devices will download it from
  - a. Note the URL of where the file is uploaded, which is important for the manifest XML file

7. To create a manifest XML file with the app/extension ID, download URL, and version, define these 3 fields:
  - **appid** (the extension ID from step 5)
  - **codebase** (the download location for the CRX file from step 3)
  - **version** (the version of the app/extension, which should match step 5)

Example XML manifest file:

```
<?xml version='1.0' encoding='UTF-8'?>
<gupdate xmlns='http://www.google.com/update2/response' protocol='2.0'>
  <app appid='abcdefghijklmnopqrstuvwxy 123456
  '>
    <updatecheck codebase='https://example.com/chrome/helloworld.crx'
version='1.0' />
  </app>
</gupdate>
```

8. Upload the completed XML file to a location from where your users or devices can download it, while noting the URL

## Hosting Your Extension

The server that hosts your extension's .crx files must use appropriate HTTP headers to allow users to install the extension by clicking a link. Chrome considers a file to be installable if either of the following is true:

- The file has the content type application/x-chrome-extension
- The file suffix is .crx and both of the following are true:
  - The file is not served with the HTTP header X-Content-Type-Options: nosniff
  - The file is served with one of the following content types:
    - empty string
    - "text/plain"
    - "application/octet-stream"
    - "unknown/unknown"
    - "application/unknown"
    - "\*/\*"

The most common reason for failing to recognize an installable file is that the server sends the header X-Content-Type-Options: nosniff. The second most common reason is that the server sends an unknown content type – one that isn't in the previous list. To fix an HTTP header issue, either change the configuration of the server or try hosting the .crx file at another server.



## Publishing Updates to Your Extension

Make sure that you've made the required changes to your extension and tested it. To publish updates:

1. Change the version number in your extension's manifest JSON file to a higher number, for example:

```
"version": "versionString"
```

If the version is 1.0, then you can update it to 1.1 or any number higher than 1.0.

2. Update the "version" of <updatecheck> in the XML file to match the number that you put in the manifest file in the last step, for example:

```
<updatecheck  
codebase='https://app.somecompany.com/chrome/helloworld.crx' version='1.1'  
>
```

3. Recreate a CRX file that includes the new changes:
  - Go to **chrome://extensions** in the Chrome address bar
  - Check the box for **Developer mode**
4. Create the CRX file by clicking **Pack extension** and selecting the directory where your source is
  - a. Note: For the PEM file, use the same file that was generated and saved during the first time the CRX file was created
5. Drag the CRX into your extensions window and make sure that it loads
6. Test the extension
7. Replace the old CRX file and XML file with the new file
  - a. This needs to be at the same host location where the users or devices previously downloaded the files

The changes will be picked up during the next policy sync cycle.

Reference URLs:

- [Auto-updating](#)
- [Update URL](#)
- [Update manifest](#)

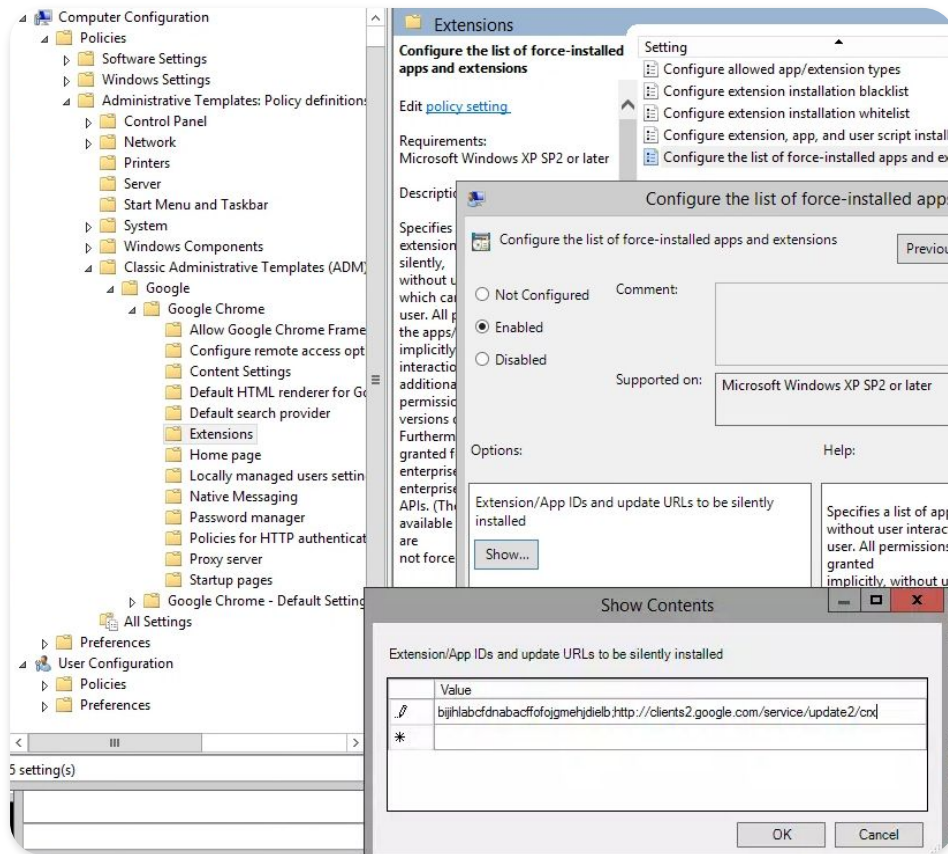
## Distributing Privately Hosted Extensions

In Group Policy: If you aren't using the Google Admin console, you can use the policy called Configure the list of force-installed apps and extensions to force install an extension on your user's device.

For privately hosted apps (not in the Chrome Web Store), use a string such as:

```
pckdojakecnhhplcgfflhndiffaohfah;https://sites.google.com/site/pushcrx/privatewebstore/extension_info.xml
```

The URL is specified to the **internal app's update.xml**, rather than the public-facing URL, `clients2.google.com`.



GPO Policy **Configure the list of force-installed apps/extensions** (Show Contents)

The policies can then be applied to your chosen users, machines, or both. It can take some time for the policy to take effect. Speed things up by running "gpupdate" on your user's machine.

# Manage Extensions with Chrome Enterprise Core

Manage Chrome browser for your Windows, Mac, and Linux machines all in one place, and get an in-depth view of the state of Chrome browser in your environment. Chrome's cloud management tool is a great method for managing and securing your browser deployments, and access is available at no additional cost with Chrome Enterprise Core. All sections within this document that reference the Google Admin console can be accessed with this tier of Chrome Enterprise. With the console, you can quickly get insights into the:

- Current Chrome browser versions deployed across your fleet
- Extensions installed on each browser
- Policies being applied to each browser

For more information about managing extensions within Chrome's cloud management tool, [please watch this video](#).

## Additional Resources



Here are more resources to help you with securing and managing your organization's Chrome browser fleet:

- [Chrome Enterprise Core](#)
- [Download Chrome for your enterprise](#)
- [Chrome Enterprise policy list](#)
- [Chrome Enterprise and Education release notes](#)
- [Chrome update management strategies](#)
- [Chrome Enterprise and Education Help Center](#)
- [Make Chrome your default browser \(Windows 10 and above\)](#)
- [Chrome Insider](#)
- [The transition of Chrome extensions to Manifest V3](#)