chrome enterprise

Managing Extensions in Your Enterprise

Securely manage Chrome extensions at scale

Enterprise & Education

Table of contents

Purpose of this guide

There are many useful extensions built for Chrome browser. There may be many running on your user's computers. 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 the Chrome Enterprise Core or by Windows Group Policy
4. Self host your own extensions on-premise

What's covered	Instructions and recommendations for managing extensions for Chrome browser in an your enterprise
Primary audience	Microsoft® Windows® and Chrome Enterprise Core administrators (Windows, Mac and Linux also supported)
Takeaways	Best practices for managing extensions with Chrome browser

Last updated: *October, 2025*

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

Introduction

Protecting user and company data requires careful management of browser extensions. IT administrators face the challenge of:

- Blocking malicious or unsafe extensions.
- Enabling extensions that users need for their work.
- Limiting extension access to sensitive user and company data.

This guide simplifies extension management. We will explore the available methods and help you select the right approach for your organization..

Considerations for managing Chrome extensions

Balancing user productivity with data security is key to your extension strategy. Users need tools to do their jobs, while IT must protect company data.

Start by asking key questions:

- What compliance or regulatory requirements must we meet?
- What types of device or website access violate our security policies?
- How much sensitive data is stored on user devices?

Google provides policies to help you:

- Allow or block extensions based on your security policies.
- Force-install essential extensions for your users.
- Manage extensions using the principle of least privilege.

While traditional management involves manually allowing or blocking specific extensions, a more efficient method is to manage them by their required permissions or preventing them from running on websites that have sensitive information. By defining which permissions and website access are acceptable, you can create broad policies that automatically secure your environment.

What are extension permissions?

Extensions require specific rights, or "permissions," to function. Developers must declare what their extension needs to access. These permissions fall into two main categories, though many extensions use both:

- Site Permissions: Allow access to the websites a user visits.
    - *Examples: Modifying a webpage, reading cookies, managing tabs.*
- Device Permissions: Allow access to the user's computer hardware.
    - *Examples: Accessing USB ports, storage, or screen contents.*

How do extensions update?

Extensions are designed to update automatically. An update check runs shortly after Chrome launches and then again every few hours while the browser is open.

The process works as follows:

1. Chrome sends a list of its installed extensions and their versions to a Google server.
2. The server responds by identifying which extensions have updates.
3. Chrome downloads the new extension files (CRX files) and installs them.

Updates can fail or be delayed for several reasons:

- The user's session ends before a large update finishes downloading.
- Chrome is not relaunched for a long period.
- The developer is staggering the update's release.
- A self-hosted extension has a server access or configuration error.
- The extension itself contains development errors.

To manually fix an out-of-date extension: You can uninstall and reinstall it. Alternatively, go to `chrome://extensions`, enable "Developer mode," and click the "Update" button.

Managing extensions

We recommend managing extensions based on two factors: the **permissions** they require and the **websites** they can access. This method is more secure, scalable, and easier to maintain than managing individual extensions.

For a simpler, "lite touch" approach, you can start by only managing website access. This strategy is highly effective at protecting your most sensitive data (e.g., on internal portals or payroll sites) with minimal configuration. It gives users the freedom to install the extensions they want, while ensuring those extensions cannot read or modify data on the pages you've secured.

Both methods save time by replacing long, manual allow/block lists with broad policies. You can still block specific extensions and use policies to protect your most sensitive websites from *all* extension activity.

Recommended Implementation Steps

1. **Audit Your Current Extensions** First, identify which extensions are currently used in your organization.
    - **Method 1 (Recommended): Use Chrome Enterprise Core.** This free tool provides a detailed report of all installed extensions, including their required permissions, install count, and risk scores. You can see this under `Chrome browser > Reports > Apps and extensions usage`. Clicking an extension shows its details, permissions, and third-party risk scores.
    - **Method 2: Survey.** Ask department managers what extensions are essential for their

teams.

2. **Identify Sensitive Websites** List the internal and external websites (e.g., `payroll.example.com`) that contain sensitive data. You may already have this list as part of your existing security policies. You can create policies to prevent extensions from reading or modifying these sites.
3. **Define "Risky" Permissions** Review the permissions required by your users' extensions (from Step 1). Decide which permissions pose an unacceptable risk. For example, you may want to block extensions that can read all website data or access hardware like USBs.
    - **Tip:** If a required extension has vague permissions, contact the vendor for a clear explanation of why they are needed.
4. **Create Your Policies** Based on your findings, build a set of policies. Start with a baseline and create exceptions as needed.
    - **Permission Policy:** Block all extensions that require one of your defined "risky" permissions.
    - **Site Access Policy:** Block all extensions from running on your list of sensitive websites (from Step 2).
    - **Allow/Block Lists:** Create small, manageable lists for exceptions:
        - **Allow List:** Add essential extensions (after vetting) that might otherwise be blocked by your permission policy.
        - **Block List:** Add specific extensions you want to ban, regardless of their permissions.
5. **Test, Deploy, and Review**
    - Test your new policies in a pilot group or lab environment.
    - Roll out the policies to your organization in phases.
    - Review user feedback and fine-tune your policies on a regular basis (e.g., quarterly or yearly).

This process establishes a strong security baseline and protects data.

Apps and Extensions Usage Report in Chrome Enterprise Core

The **Apps and Extensions Usage Report** is a central hub within the Chrome Enterprise Core for monitoring and managing the software installed across your fleet of enrolled Chrome browsers and ChromeOS devices. By leveraging this report, administrators can gain visibility into extension risks, permission requirements, and installation patterns. This will help with your implementation steps outlined in the previous section.

Prerequisites and Setup

Before data appears in the report, ensure the following configurations are in place:

- **Enable Reporting:** You must enable Chrome browser reporting in the Admin console.
- **Permissions:** You require the **Chrome administrator privilege**.
- **Propagation Time:** It can take up to **24 hours** for data to populate after reporting is first enabled.

How to Access:

Navigate to Devices > Chrome > Reports > Apps & extensions usage.

Key Metrics for Extension Auditing

The report provides a high-level overview of all installed apps and extensions. Key columns include:

Metric	Description
Install Type	Identifies if an extension was `admin` (force-installed), `normal` (user-installed), or `development` (unpacked).
Chrome Web Store Status	Shows if an extension is `Published`, `Taken down by Google` (due to policy/malware), or has `No record`.
Manifest Version	Crucial for tracking the transition from **Manifest V2** to **V3**. A warning icon appears for extensions still using the deprecated V2.
Installs	The total count of unique browsers or devices where the extension is present.
Permission Count	The number of distinct permissions requested by the extension.

App Details & Risk Assessment

Clicking on any specific extension in the report opens the **App Details** page, providing a granular security view.

Risk Assessment Scores

Chrome Enterprise Core integrates third-party risk data to help you evaluate the safety of publicly available extensions. Clicking on the score in the app details page will take you to that vendor's specific assessment of that extension. Details on the specific risk levels displayed in the admin console are listed below:

Provider	Low Risk	Medium Risk	High Risk
LayerX	0–3.9	4–6.9	7–10
Spin.AI	100–66	65–36	35–1

Permissions & Website Access

- **Requested Permissions:** View exactly what the extension can do (e.g., "Access to user data," "Management," or "Identity").
- **Requested Website Access:** See a list of host match patterns showing which websites the extension can read or change data on. This is vital for identifying extensions with overly broad "All Sites" access.

Administrative Actions

The usage report isn't just for viewing—you can take direct action to secure your environment:

- **Export Data:** Use the **Export** button to download a CSV for offline analysis or custom reporting.
- **Quick Management:** Hover over any extension row to see the **Action menu** (three dots). From here, you can instantly:
  - **Block:** Remove the app from all browsers in the selected organizational unit and prevent re-installation.
  - **Force Install:** Ensure the extension is automatically deployed and cannot be disabled by users.
- **Filter by Last Activity:** Use filters to identify stale or "shadow IT" extensions that haven't been seen by a reporting browser in up to 1.5 years.

Known Limitations & Considerations

- **Data Latency:** Data may not reflect usage from the most recent 6 hours.
- **Installation Counting:**
  - **ChromeOS:** Counted by user organizational unit. (Multiple users on one device = multiple installs).
  - **Chrome Browser:** Counted by device organizational unit. (Multiple users on one enrolled device = 1 install).
- **Permission Variance:** In some cases, the permission count in the main report may vary slightly

from the deep-dive details page; always refer to the **App Details** page for the most list.

Overview of the different extension management policies

You can manage extensions using several policies, most of which are configured via Windows Group Policy (GPO) or Plists on macOS. A complete list is located in the extension policy group but here are the commonly used policies:

- ExtensionInstallAllowList Specifies which extensions are approved for installation.
- ExtensionInstallBlockList Specifies which extensions are blocked. Already-installed extensions will be disabled, and new installs will be prevented. The Chrome Web Store will also show a notice that the extension is blocked by the administrator.
- ExtensionInstallForceList Silently installs extensions on user machines. Users cannot disable or uninstall them. This policy overrides the `ExtensionInstallBlockList`.
- BlockExternalExtensions Blocks extensions from "external" sources (e.g., an application installing an extension via the registry).
- ExtensionAllowedTypes Defines which types of items can be installed (e.g., `extension`, `theme`, `user_script`). Any type not explicitly included in this list will be blocked.
- ExtensionInstallSources Allows installation from specific URLs that you designate. This re-enables a behavior that was removed for security reasons (post-Chrome 21) where users could be prompted to install from a direct `.crx` file link.
- ExtensionSettings A powerful and complex policy that controls multiple settings at once using a single-line JSON string. It can manage permissions, site access, and more.
    - **Recommendation:** Because this policy is complex, we recommend using **Chrome Enterprise Core**. It provides nearly all the same functionality (plus extension auditing) without the need to write and maintain JSON.

A Note on Policy Naming

Google has adopted more inclusive naming conventions. The following old policy names are deprecated and were removed in Chrome 97. Please ensure you are using the new policy names.

- `ExtensionInstallWhitelist` is now **`ExtensionInstallAllowlist`**
- `ExtensionInstallBlacklist` is now **`ExtensionInstallBlocklist`**

Block extensions based on their permissions

You can control extensions your users can install by their permissions. An installed extension with blocked permissions will be disabled. If a user tries to install one with a blocked permission, it won't install.

Manage extensions by their permissions in the Chrome Enterprise Core

(Windows, Mac and Linux)

You can block extensions that need permissions which aren't allowed. For example, you could block extensions from connecting to USB devices or prevent access to cookies.

1. In your Admin console, go to **Chrome browser >Apps and extensions**
2. Select the organizational unit that you want to configure the setting(s) for.
3. Click on the settings tab at the top of the page
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 via permissions under Permission and URL Access> Customize Permissions for this app/extension.
        i. Note: that 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 editor, 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:

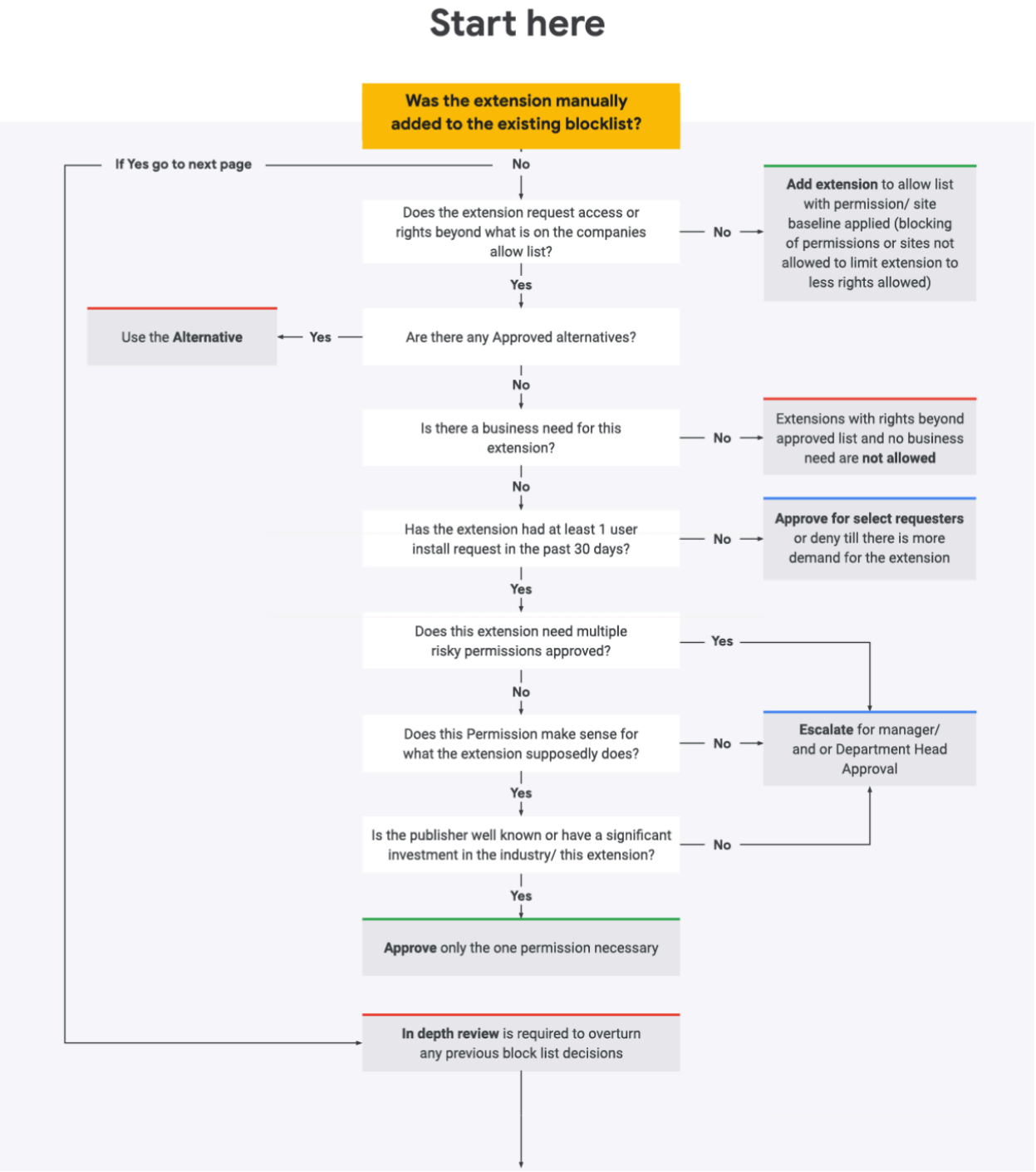
```
{"*":{"blocked_permissions":["usb"]}}
```

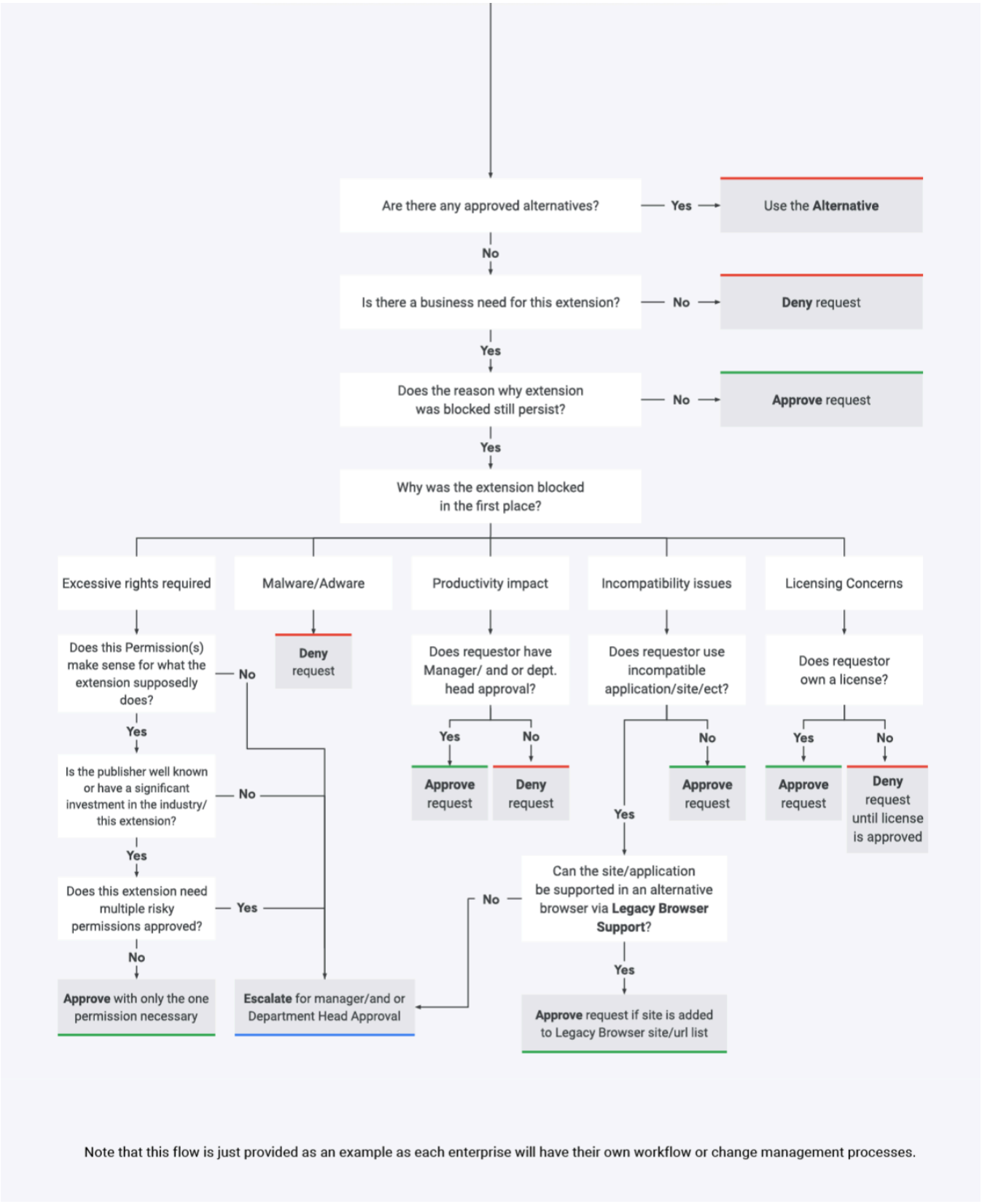
Top Tip:

- 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.
    - For steps for finding the extension ID , see step 3 of this help article.

Creating an exception process for extensions that require risky permissions

There might be a business need for extensions that might 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.





Note that this flow is just provided as an example as each enterprise will have their own workflow or change management processes.

Managing extensions by the extensions setting policy

For granular, cross-platform control (Windows, macOS, Linux, and ChromeOS), you can use the powerful `ExtensionSettings` policy. This single policy can configure multiple settings at once.

This policy can control:

- Initial installation source and `update_url`.
- A list of `blocked_permissions`.
- Whether to block extensions from accessing specific websites (`runtime_allowed_hosts` / `runtime_blocked_hosts`).

You can use this policy to set a default configuration for all extensions and then override it for specific ones. For more details, read the Configure ExtensionSettings policy and App and extension policies help articles.

> Important: Policy Overrides
>
> Be aware that the `ExtensionSettings` policy will **overwrite** any settings configured in the following individual policies:
>
> - `ExtensionAllowedTypes`
> - `ExtensionInstallAllowlist`
> - `ExtensionInstallForcelist`
> - `ExtensionInstallSources`
> - `ExtensionInstallBlocklist`
>
> Note on Website Access (Runtime Hosts)
>
> In a GPO environment, the `ExtensionSettings` policy is the **only** way to configure `runtime_allowed_hosts` and `runtime_blocked_hosts` (blocking extensions on specific websites).
>
> This functionality is also available in the **Chrome Enterprise Core** via the Google Admin Console, which provides a much simpler UI and does not require writing JSON.

Configuration Methods

You can set this policy in one of two ways:

1. **JSON String in Windows Group Policy Editor:** Enter a single-line, compact JSON string into the GPO.
2. **Windows Registry:** Manually create the registry keys.

Top Tips for Configuration

- **Validate Your JSON:** Correctly formatting a single-line JSON string is difficult. Always use a JSON checker or validation tool before deploying the policy.
- **Generate JSON Automatically:** If you struggle with the formatting, use one of these methods to generate the correct JSON for you:
    1. **Registry Method:** Configure the settings using the Windows Registry method on a test machine.
    2. **Chrome Enterprise Core:** Configure your desired settings in the admin console.
    3. **Copy the JSON:** Go to `chrome://policy` on the test machine, find the `ExtensionSettings` policy, and copy the auto-generated JSON string. You can now paste this validated string into your GPO.

Configure the extension policy using the Windows Registry

You can set the `ExtensionSettings` policy by creating keys directly in the Windows Registry.

1. Choose Registry Hive

First, navigate to the base path for the policy. You can use either the `HKEY_LOCAL_MACHINE` (HKLM) or `HKEY_CURRENT_USER` (HKCU) hive. The GPO equivalent will configure this path for you.

- HKLM Path: `Software\Policies\Google\Chrome\ExtensionSettings\`
- HKCU Path: `Software\Policies\Google\Chrome\ExtensionSettings\`

2. Define the Scope (Create Subkey)

Next, create a new subkey under `ExtensionSettings` to define the policy's scope (i.e., which extensions it applies to).

- **For all extensions (default):** Use an asterisk (`*`) as the subkey name.
    - *Example:* `...\ExtensionSettings\*`
- **For a specific extension:** Use the extension's 32-character ID as the subkey name.
    - *Example:* `...\ExtensionSettings\nckgahadagoaajjgafhacjanaoiihapd` (for Google Hangouts)

3. Add Values (Settings)

Inside the scope subkey you just created (e.g., `*` or the extension ID), create new `REG_SZ` (String Value) entries for each setting you want to configure.

The format for the value depends on the setting type:

- **String:** Enter the value as a simple string (e.g., `force_installed`).
- **Array of Strings:** The value must be entered as a JSON-formatted string array (e.g., `["value1", "value2"]`).

Registry Value Types (Reference)

Here is a list of common settings and their required format:

String (REG_SZ)

- `installation_mode`
- `update_url`
- `minimum_version_required`
- `blocked_install_message`

Array of Strings (REG_SZ, JSON format)

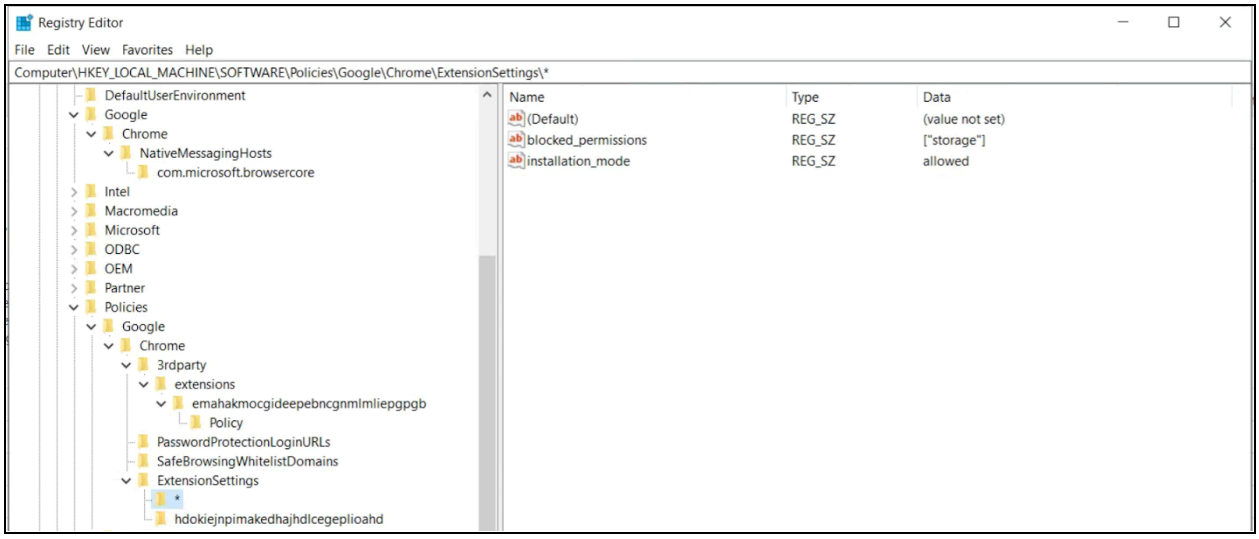
- `blocked_permissions`
- `allowed_permissions`
- `runtime_blocked_hosts`
- `runtime_allowed_hosts`

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

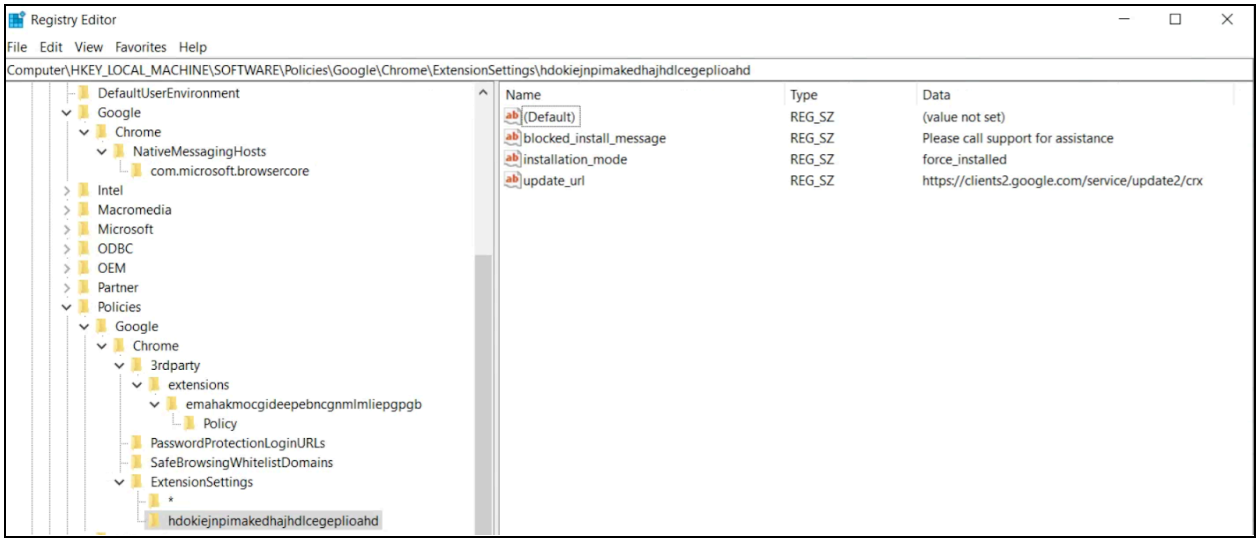
- `[ “power”,”printerProvider”,”serial”,”usb”]`

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



An individual extension scope 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	DefaultBrowserSettingEnabled
Machine	Mandatory	Platform	ExtensionSettings


```
{
  "*": {
    "blocked_permissions": [ "storage" ],
    "installation_mode": "allowed"
  },
  "hdokiejnpimakedhajhdlcegeplioahd": {
    "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 Editor

The steps to use the extension settings policy using GPO assume you have already imported the ADM/ADMX for Chrome Policies.

For other OS platforms, check: Mac | Linux | Chrome OS

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. It is recommended to use a JSON compression tool to help with this.

Formatting the ExtensionSettings JSON Policy

This policy is configured using a compact, single-line JSON string. Understanding its structure is key to using it correctly.

Policy Structure: Default vs. Individual Scope

The JSON structure has two levels of scope:

1. **Default Scope (`*`)**: This is a "catch-all" scope identified by an asterisk (`"*"`). These settings apply to **all** extensions *unless* they are overridden by an individual scope.
2. **Individual Scope (`<extension_id>`)**: This scope applies settings **only** to a specific extension, identified by its 32-character ID (e.g., `"nckgahadagoaajjgafhacjanaoiihapd"`).

How Settings are Applied: An extension gets its settings from only one scope.

- If an individual scope exists for that extension, **only those settings will apply**.
- If no individual scope exists, the extension will use the settings from the **default (`*`) scope**.

Example Structure: This JSON defines a default scope and one individual scope.

```
None
{
  "*": {
    "installation_mode": "allowed"
  },
  "nckgahadagoaajjgafhacjanaoiihapd": {
    "installation_mode": "force_installed",
    "update_url":
"[https://clients2.google.com/service/update2/crx](https://client
s2.google.com/service/update2/crx)"
  }
}
```

Example JSON (Functional): This example blocks all extensions from running on `*.example.com` and blocks any extension that requires the "usb" permission.

```
None
{
  "*": {
    "runtime_blocked_hosts": ["*://*.example.com"],
    "blocked_permissions": ["usb"]
  }
}
```

Compact (Single-Line) Version: This is the format you must use in the GPO.
`{"*":{"runtime_blocked_hosts":["*://*.example.com"],"blocked_permissions":["usb"]}}`

Reference: Installation Mode Settings

Here are common examples for controlling extension installation.

- **`"allowed"` (Default)**
  - Allows users to install the extension from the Chrome Web Store.
  - *Example JSON:* `{"*":{"installation_mode":"allowed"}}`
- **`"blocked"`**
  - Prevents users from installing the extension.
  - *Example JSON:* `{"*":{"installation_mode":"blocked"}}`
- **`"force_installed"`**
  - Automatically installs the extension without user interaction. Users cannot disable or remove it.
  - **Requires `update_url`:** You MUST also define the `update_url` field.
  - *Example JSON (for a specific extension):* `{"nckgahadagoaajjgafhacjanaoiihapd":{"installation_mode":"force_installed","update_url":"https://clients2.google.com/service/update2/crx"}}`
- **`"normal_installed"`**
  - Automatically installs the extension, but users can disable it.
  - **Requires `update_url`:** You MUST also define the `update_url` field.
  - *Example JSON:* `{"*":{"installation_mode":"normal_installed","update_url":"..."}}`
- **`"removed"`**
  - (Chrome 75+) Prevents users from installing the extension. If it was previously installed, Chrome removes it.
  - *Example JSON:* `{"*":{"installation_mode":"removed"}}`

Reference: Other Common Settings

- `"blocked_install_message"`
  - Displays a custom message when an installation is blocked.
  - *Example JSON:* `{"*":{"blocked_install_message":"Call IT (408-555-1234) for an exception."}}`
- `"toolbar_pin"`
  - Controls the extension's icon in the toolbar.
  - `"force_pinned"`: Icon is pinned and cannot be hidden by the user.
  - `"default_unpinned"`: (Default) Icon starts in the extension menu; the user can pin it.
  - *Example JSON:* `{"*":{"toolbar_pin":"force_pinned"}}`

Important Note on `update_url`

When using `"force_installed"` or `"normal_installed"`, you must provide an `update_url`:

- **Chrome Web Store:** Use `https://clients2.google.com/service/update2/crx`
- **Self-Hosted:** Use the URL pointing to your packed extension (`.crx`) file.

`override_update_url` (Chrome 89+)

- By default (or if set to `false`), Chrome uses the update URL specified in the *extension's manifest* for updates after the initial install.
- If set to `true`, Chrome will use the `update_url` you provided in this policy (or in the `ExtensionInstallForcelist` policy) for all subsequent updates.

Validation & Common Pitfalls

- **Pitfall: Scope Does Not Merge**
    1. An individual scope **does not** inherit settings from the default (`*`) scope. It completely overrides it.
    2. **Example:** If your default scope blocks "usb" but you add an individual scope for an extension, that extension will *no longer* block "usb" unless you explicitly add `"blocked_permissions": ["usb"]` to its *individual* scope.
- **Pitfall: Invalid JSON**
    1. The policy requires a **single-line, compact JSON string** with no line breaks or comments.
    2. Always use a JSON validator and minifier/compressor tool before pasting the string into the Group Policy Editor.
- **How to Validate Your Policy**
    1. On a target client machine, open Chrome.
    2. Go to the `chrome://policy` page.
    3. Click **"Reload policies"**.
    4. Find the `ExtensionSettings` policy.
    5. Check its **Status**. If it says **"OK"**, the policy was applied. If it shows an error, your JSON is likely malformed.
    6. Click **"Show value"** to see the exact JSON that Chrome applied.

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. They will still continue to function on other websites.

There are two settings you can use for this feature

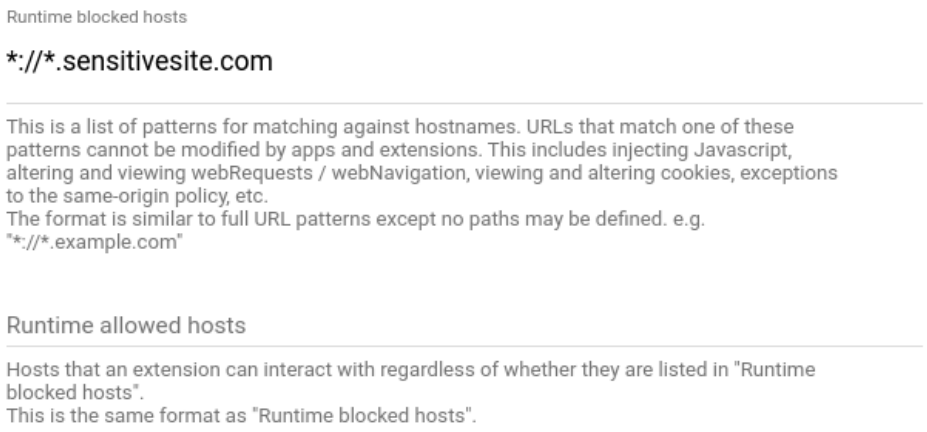
- **runtime_blocked_hosts** - Extensions are blocked from interacting with these hosts
- **runtime_allowed_hosts** - Extensions may interact with hosts on this list even if defined in runtime_blocked_hosts.

Top 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 Enterprise Core

Blocking by runtime host is simpler within Chrome Enterprise Core 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 Enterprise Core. The feature is offered at no additional cost. The steps for enrollment are located here.

1. In your Admin console, go to **Devices > Chrome >Apps and Extensions>Users and browsers**
2. Select the organizational unit with the users you want to allow extensions for.
3. Click on the settings tab at the top of the page
4. Go to the Permissions and URLs section and scroll down to the Runtime blocked hosts section
5. 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 hitting 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: that 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.
6. Click **Save**.



Runtime hosts section in **Chrome browser > Apps and Extensions>Users and browsers>Settings>Permissions and URLs**

GPO

These instructions are for managing this GPO on Windows machines. For other platforms, check: Mac | Linux

Within the Extension Settings 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
://.example.*	http://example.com https://test.example.co.uk	https://example.google.com http://example.google.co.uk
http://example.*	http://example.com http://example.ly	https://example.com http://test.example.com
http://example.com	http://example.com	https://example.com http://test.example.co.uk
://	All Urls	

Here is a sample of a JSON string that blocks access for a single extension. This string prevents an 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","*://*.importantweb
site2.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"]}}
```

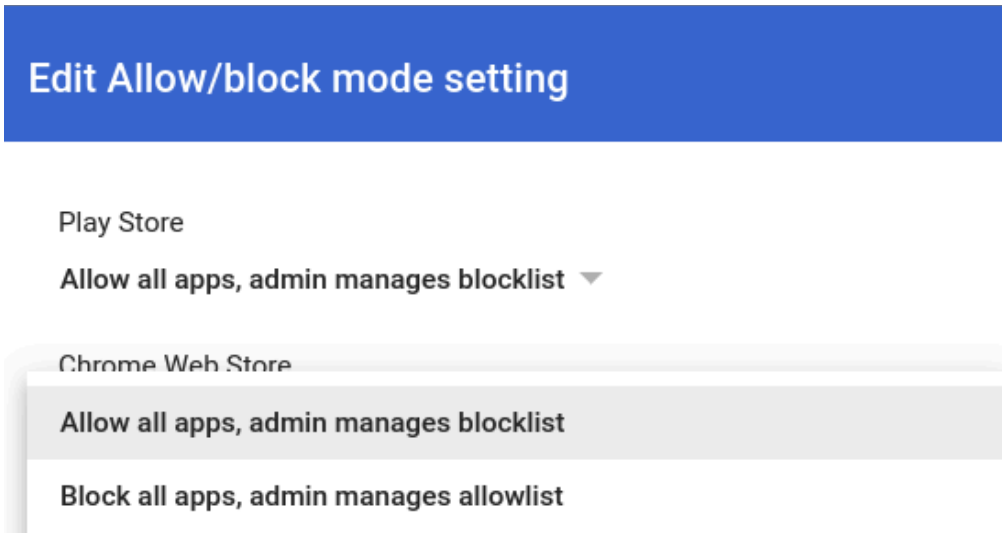
Allow or block extensions in the Google Admin console

Admins can control which extensions your users can install by creating allow and block lists. You can allow users to install any app or extension. You can set policies to block or allow apps for all users or certain employees.

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

Allow all extensions except those you want to block

1. In your Admin console, go to **Chrome browser> Apps and extensions>Users and browsers> Settings**.
2. Select the organizational unit that you want to allow extensions for, on the left.
3. Scroll down to the Allow/block mode section, click edit and under Chrome Webstore select the option to **Allow all apps, admin manages blocklist**.

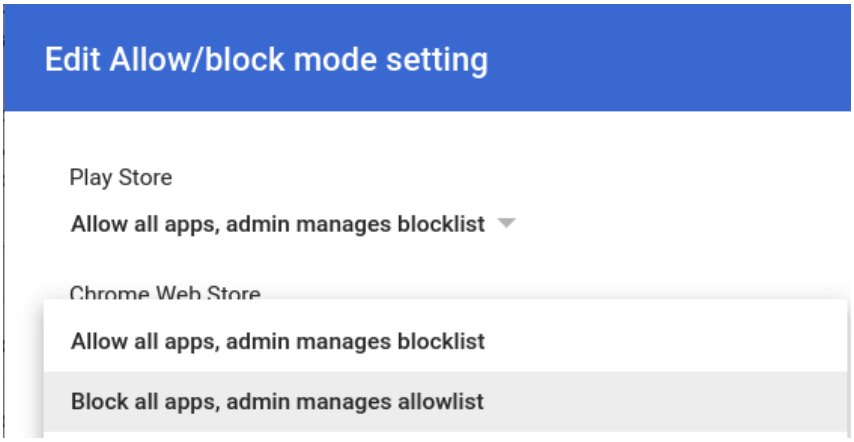


Allow/block mode settings

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

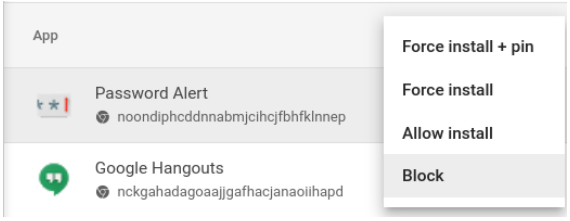
1. In your Admin console, go to **Chrome browser> Apps and extensions>Users and browsers> Settings**.
2. Select the organizational unit that you want to block extensions for, on the left.
3. Scroll down to the Allow/block mode section, click edit and under Chrome Webstore select the option to **Block 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 dropdown 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 one extension

1. In your Admin console, go to **Chrome browser> Apps and extensions>Users and browsers**
2. Select the organizational unit that you want to allow or block the extension for.
   - Note: that the organizational unit will inherit the settings of the parent Organizational unit, but you can override per sub organizational unit.
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.

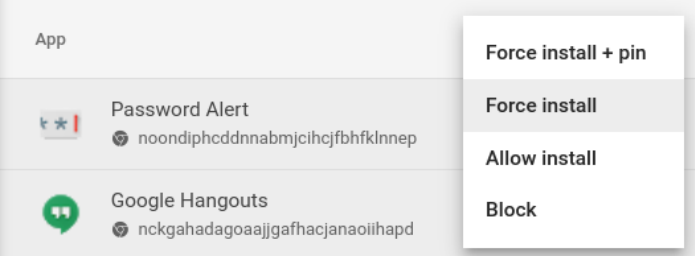


5. Click **Save**.

Force-installing extensions

If you know that the 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 also will 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 Admin console, go to **Chrome browser> Apps and extensions>Users and browsers**
2. Select the organizational unit that you want to force install extensions to.
3. Select the existing extension(s) you want to force install or add them in.
   a. To add the extension(s) 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 extension(s) that you want to force install and in the installation policy column, select **Force install** from the dropdown menu.



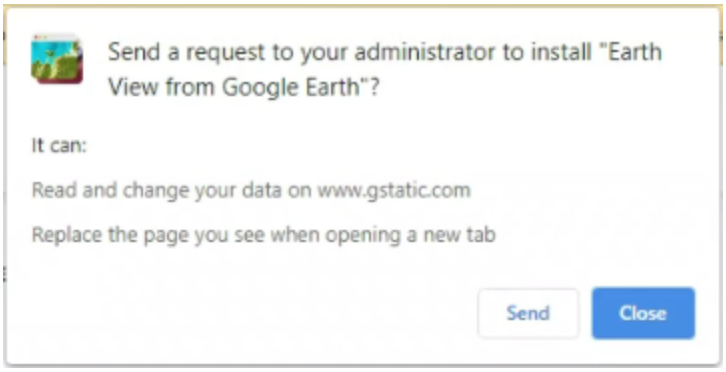
5. Click **Save**.

Note that you can create a custom Chrome Web Store for your users signed into the browser as well as devices that are enrolled in Chrome Enterprise Core. You can customize things like showing your company's name and logo, displaying selected extensions, hiding or showing specific categories and more.

- These settings can be found within the admin console under **Chrome browser>Apps and extensions>Users and browsers>Settings>**Chrome Web Store Settings

Let users request Extensions: Extension Workflows

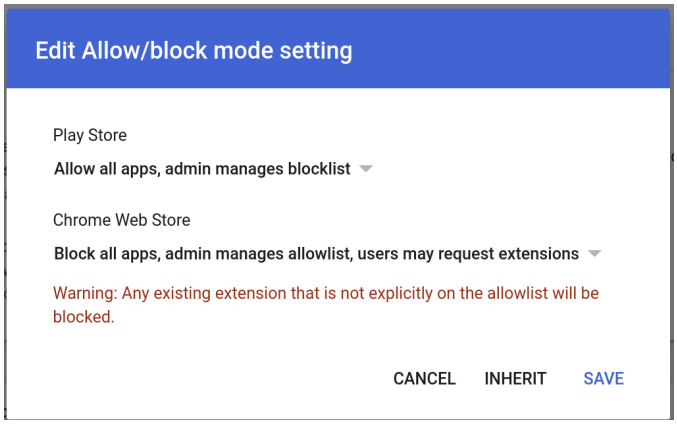
As an admin, you can use the Google Admin console to let users request the extensions that they need in the Chrome Web Store. Then, you can allow, block, or automatically install extensions that users request.



Example of request dialog from Chrome Web store

Note that this feature works as an allow/block list. 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, through using the Apps and extension usage report in Chrome Enterprise Core.
2. Create a list of essential extensions (GPO or admin console) based on the data gathered in step one.
    - It is recommended to focus on extensions that have the largest install count as they are most likely to be business critical. You can bulk add extensions via the API.
3. Turn on the extension workflow feature under **Chrome browser > Apps and extensions>Settings**>Allow/block mode and hit the edit button
4. Under Chrome Web store, select **"Block all apps,admin manages allowlist, users may request extensions"** from the drop down menu.



Enabling Extension Workflows in the admin console

    - We recommend that first you apply settings to a small number of users and devices in a test organizational unit to prevent end user issues and gather feedback. Once you feel ready, then you can apply it to your entire organization.
5. Approval and denial requests are managed under **Chrome browser>Apps & Extensions>Requests**
6. Click the row of the extension request that you want to review..
7. Here you can review details about the extension and select the installation policy from the dropdown menu:
    - Force install- installs the extension silently and it can not be removed
    - Allow install — Lets users install the extension
    - Block — Prevents users from installing the extension. Removes the extension from users that have it installed

For more information about this feature, please review the help center article for Extension Workflows or this Youtube video on extension workflow.

Allow or block extensions in Group Policy

Before you begin: The following steps assume that you already have Chrome managed for your users. For more on how to deploy Chrome on Windows, refer to Chrome browser Deployment Guide (Windows). For Mac® deployment and policy management, go to Set up Chrome browser on Mac.

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.

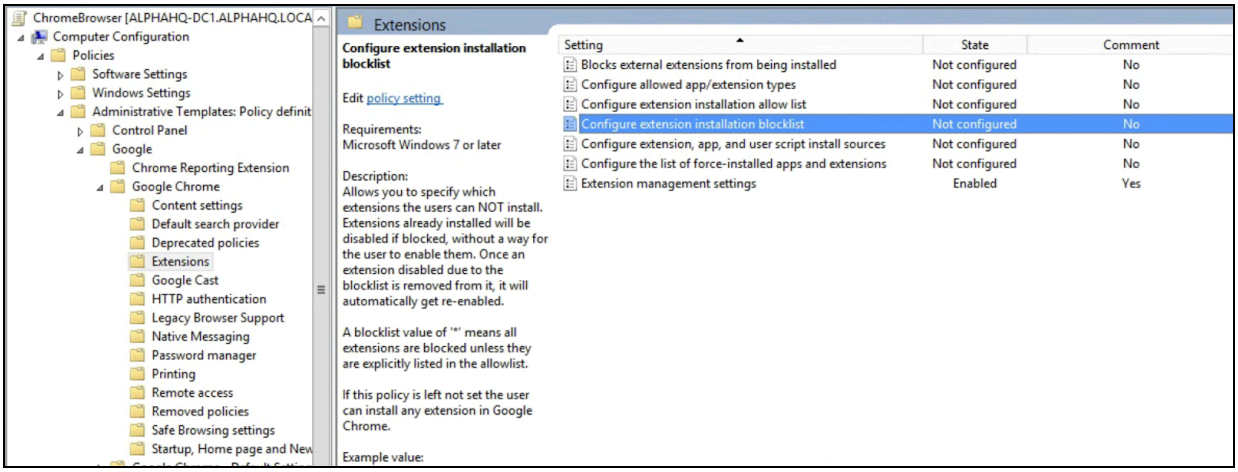
1. Download Chrome policy templates.
   Windows templates, as well as common policy documentation for all operating systems, can be found in this via this link
2. Open the ADM or ADMX template you downloaded:
    a. Go to **Start > Run: gpedit.msc**.
    b. Go to **Local Computer Policy > Computer Configuration > 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.

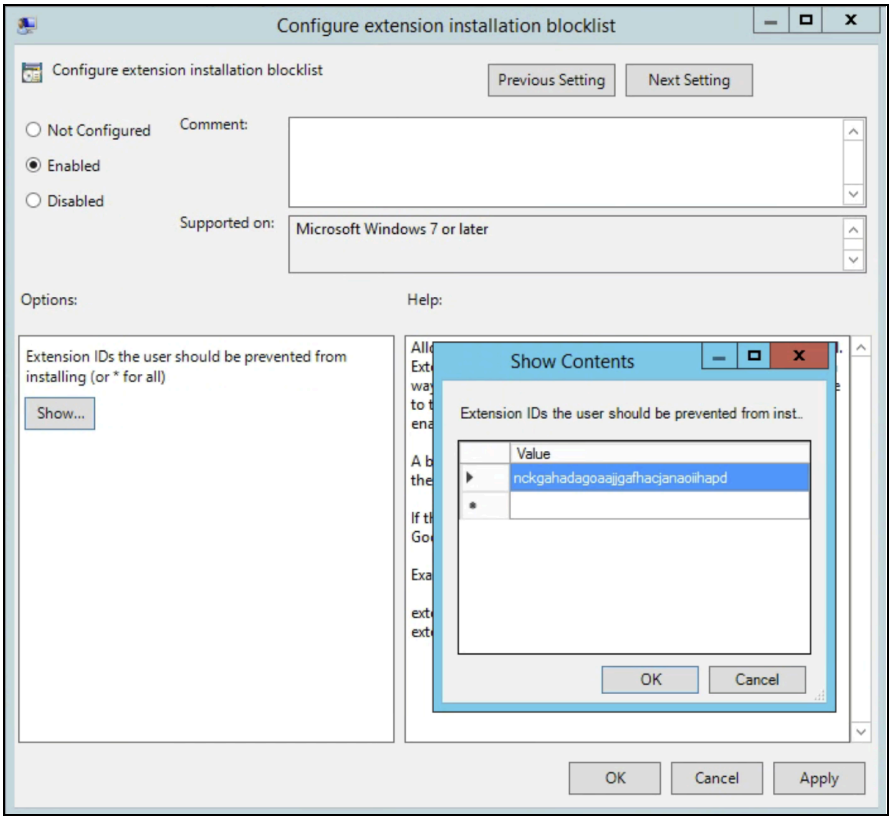
Allow all extensions except those you want to block

1. In the Group Policy Editor, open the template you just added.
2. Browse to **Google > Google Chrome > Extensions > Configure Extension installation blocklist**.



Path to Extension management policies

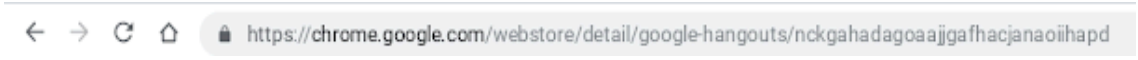
2. In the setting, select **Enabled**.
3. Click **Show**.
4. Enter the app ID of the extensions that you want to block.



Configure extension installation blocklist

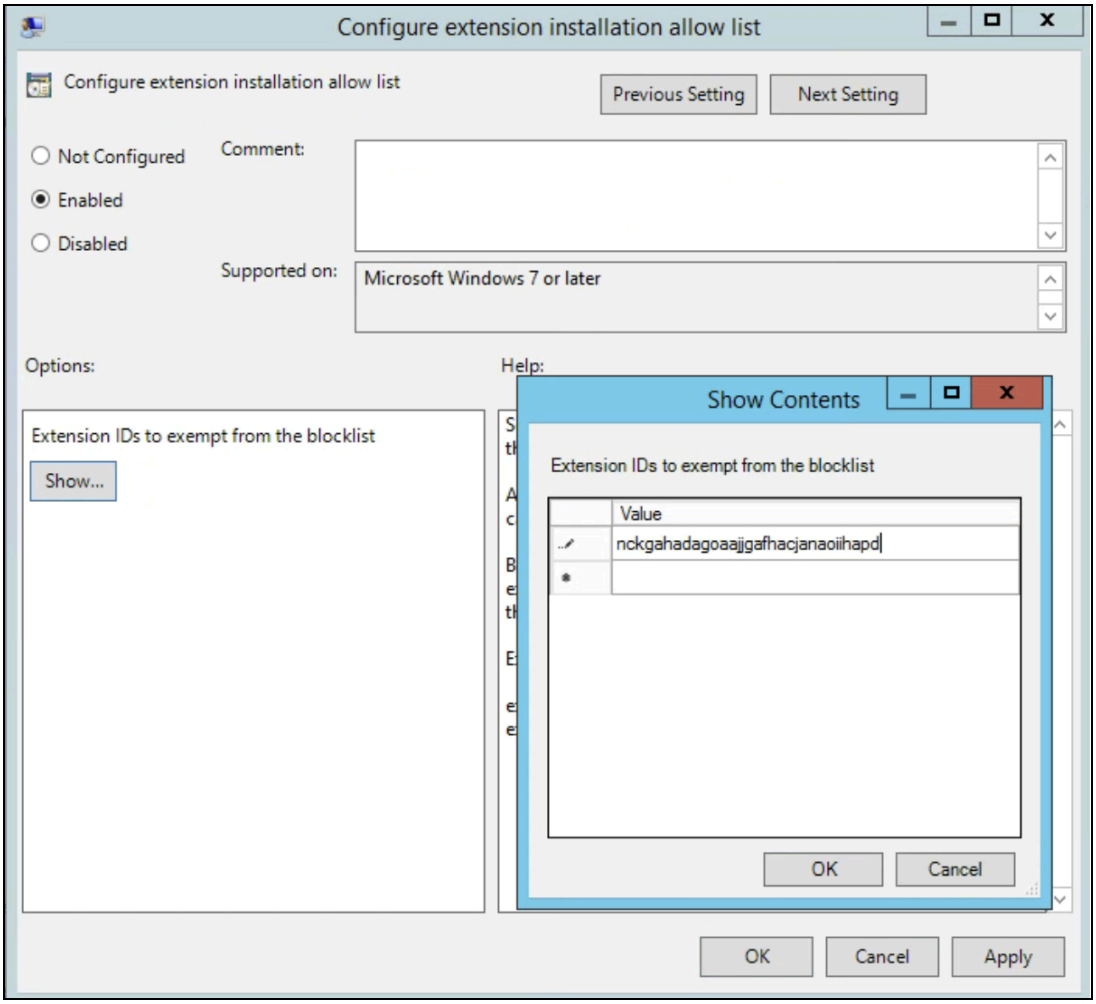
Notes:

- If you can't find the app ID of an extension, view it in the Chrome Web Store. Find 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-hangouts/

- Enter * into the policy to prevent any extensions from being installed. You can use this with the Configure extension installation allowlist policy. This way you only allow certain extensions to be installed by your users and block the rest.
- You can add an extension to the block list that is already installed on a user's machine. It will disable the extension and prevent the user from re-enabling it. It will not be uninstalled, just disabled.



Configure extension installation allowlist

Block or allow one extension

To block a single extension, add the app ID of the extension you want blocked to the configure extension installation blocklist policy. All of your other extensions will be allowed to be installed.

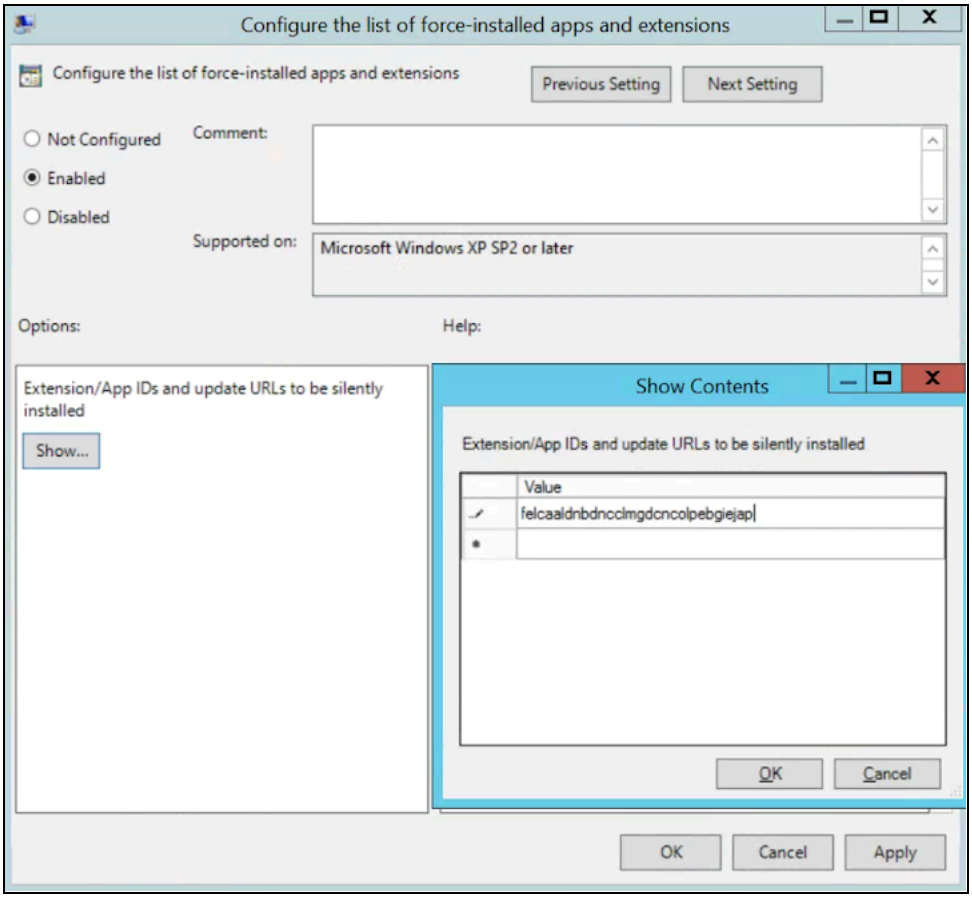
To allow only one extension:

1. In the content section in the Configure extension installation blocklist policy, enter *****.
   This will block all extensions from being installed that are on the list.
2. Add the app ID of the allowed extension to the Configure extension installation allowlist policy.

Force-install an extension

1. In the Group Policy Editor, 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.

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



Configure the list of force-installed apps and extensions

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 “Extension” to only show extension related policies.
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
7. Congrats! You’ve got a valid policy

Self hosting your own extensions

The Chrome Web Store hosts extensions and provides many security features.

- Features such as automated and manual code scans.
    - This prevents malicious code from being sent to your users.

However there’s 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:

- Hosting your own extensions means that you are outside of the rules and requirements of the Chrome Web Store.
    - So there is less scrutiny on the extension and limits the risk that the extension could be 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, where the Chrome Web Store does this automatically.

If you choose to self-host your extensions, this section tells you how. It covers 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

As an alternative to self hosting, consider marking internal extensions on the Chrome Web Store as private. There are three options to publish extensions, public, private and unlisted. Here is a chart with more information about the pro and cons of each:

	Present in Chrome Web Store Search?	Require Sign-in?	Supported in Chrome Enterprise Core
Public	Yes	No	Yes
Private	No	Yes	Yes
Unlisted	No	No - users need link to install	Yes

For more information, please review this developer article on enterprise publishing options on how to publish your extensions out of the view of the public, without the need to self host your extensions.

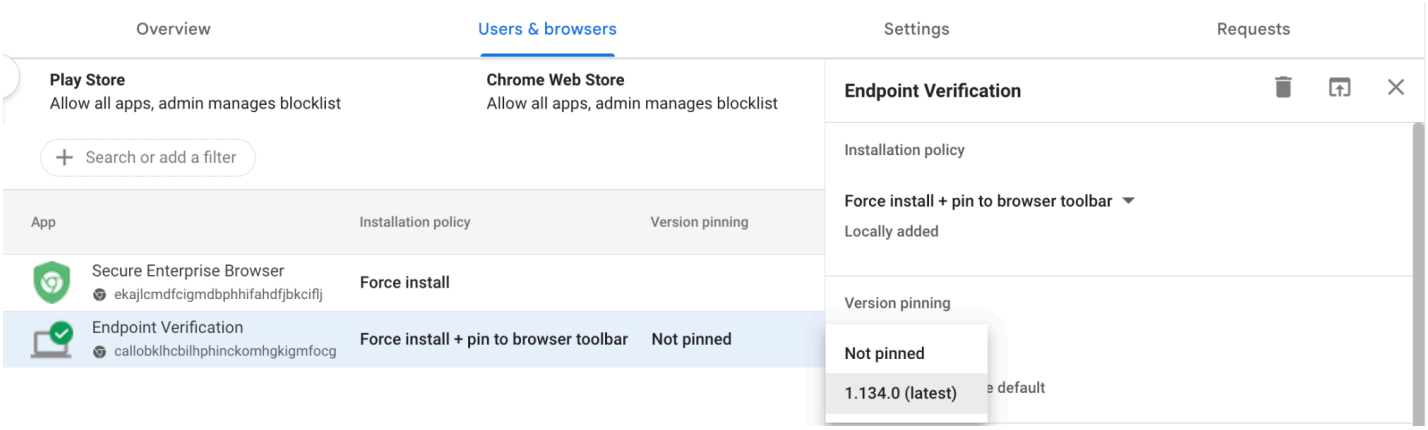
- Note that if you are managing your extensions via the admin console you need to configure the Chrome Web store permissions setting to enable private extensions to show up for your users.
    - This is found in the admin console under **Chrome browser>Apps and Extensions>Settings>Chrome Web Store Permissions>**and set to Allow 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 admin console

The Google admin console now offers a few new options for extension management. First is the ability to pin to a version of an extension directly in the admin 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 of extensions 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 review this help center entry.

1. In your Admin console, go to **Chrome browser> Apps and extensions>Users & browsers**
2. Select the organizational unit that contains the extension that you want to pin.
3. Select the existing extension(s) (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 hit save.
    a. Note that by pinning an app or extension, it will no longer get updates, including security and compatibility updates.

b. You can 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 admin console. See the section Pin self-hosted apps in this help center entry.



Version pinning in the admin console

Requirements for self hosting extensions

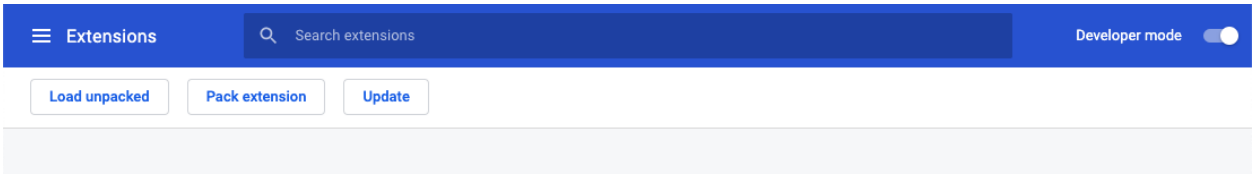
To host your extension, you will need your own web hosting services for the extension and its manifest file. This hosting location shouldn't require authentication. It needs to be accessible by devices wherever they are used. Keep this in mind if you want to host the file on your internal repository.

The steps assume that you've already created your extension, and have experience with XML files. Also that you have knowledge about group policy and using the windows registry. These steps do not apply to a 3rd party extension that you did not develop. If you want to self-host a 3rd party extension, you should discuss this with the extension vendor directly. For more information on self hosting check out this developer documentation.

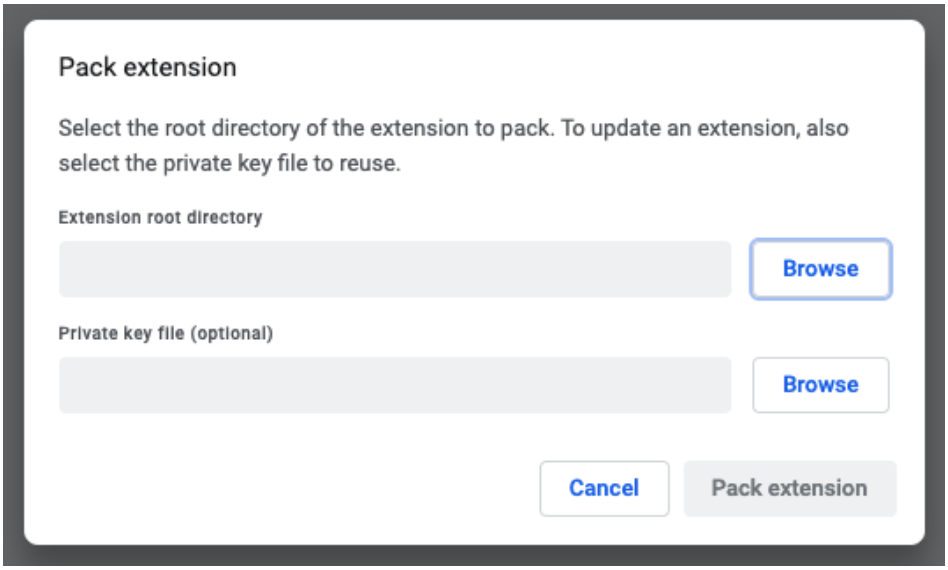
Packaging your extension

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**.



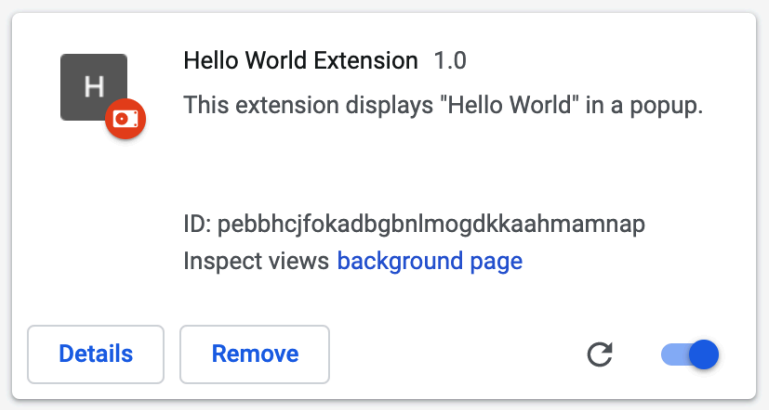
2. After you're in developer mode, create the CRX file by clicking **Pack Extension**.



3. Select the directory where your source is. This will create your CRX file, along with a PEM file.

Top 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. Note 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.
   These will be important later on.



5. Place the CRX file in the host location where your users or devices will download it from.
    - Note the URL of where the file is uploaded.
    - This will be important for the manifest XML file.
6. 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='abcdefghijklmnopqrstuvwxyz123456
'>
    <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.
Google 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.
   Example:
   `"version": "versionString"`
   If the `"version":"1.0"`, then you can update to `"version":"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.
   Another example:
   ```
   <updatecheck codebase='https://app.somecompany.com/chrome/helloworld.crx'
   version='1.1' />
   ```

3. Recreate a CRX file which includes the new changes:
   a. Go to **chrome://extensions** in the Chrome address bar.
   b. Check the box for **Developer mode**.

4. Create the CRX file by clicking **Pack Extension** and selecting the directory where your source is at.
   Note: For the PEM file, use the same file which 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 from where the users or devices downloaded the files before.

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

Reference URLs:

- Autoupdating
- Hosting
- Update URL
- Update manifest

Distributing privately hosted extensions

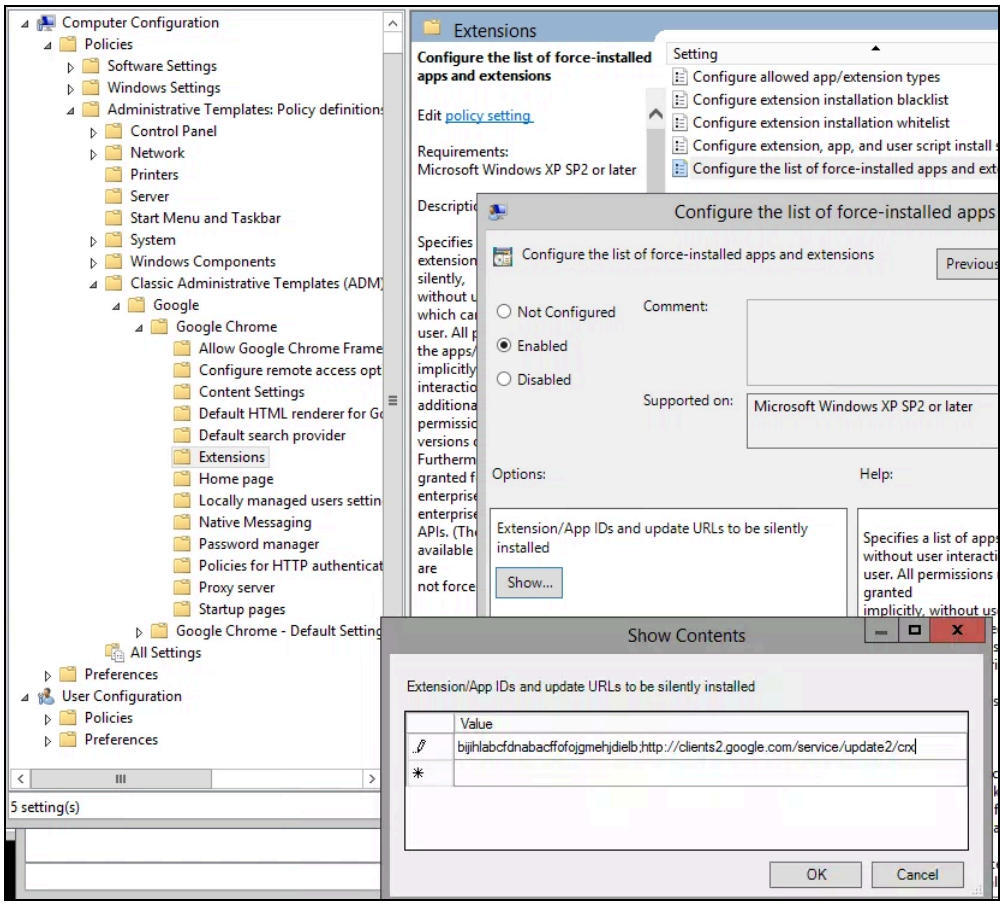
In Chrome Enterprise Core:

1. In your Admin console, go to **Chrome browser> Apps and extensions>Users and browsers**
2. Select the organizational unit that you want to force install the private extension to.
3. To add the private extension(s) you want to install, click on the yellow plus mark in the bottom right and select Add Chrome app or Extension by ID
4. Enter in your private extension's ID and select From a custom URL from the drop-down menu.
5. Enter in the URL your hosting server. Note that this needs to be accessible without any authentication in order for the extension to install.
6. Select the extension(s) that you want to force install and in the installation policy column, select **Force install** from the dropdown menu.

In Group Policy: If you aren't using the 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 `clients2.google.com` URL.



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 using 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 Enterprise Core is a great method for managing Chrome browser settings. Access to this console has no additional cost. All sections within this document that reference the Google admin console can be accessed with this feature of Chrome. With the console, you can quickly get insights on the:

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

Chrome Enterprise Premium

Additional security needed? Chrome Enterprise Premium provides advanced Data Loss Prevention (DLP) that includes granular controls for data masking, watermarking, and screenshot protection to prevent unauthorized data sharing and leaks. It also enables agentless Context-Aware Access, ensuring secure, zero-trust connectivity to SaaS and private web applications in any cloud environment based on user identity, device posture, and location.

Additional resources

Here are more resources to help you with managing the Chrome browser in your organization:

- Chrome Enterprise Core landing page
- Chrome Browser Enterprise bundle
- Chrome Policy list
- Chrome Enterprise release notes
- Chrome browser update management strategies
- Chrome Enterprise Help Center
- Make Chrome default browser (Windows 10 and above)
- The transition of Chrome extensions to Manifest V3
- Enhanced browser security with Chrome Enterprise Premium