



# Ivanti Security Controls

## Best Practice Implementation Guide – Application Control

# Contents

<b>Introduction .....</b>	<b>6</b>
Purpose of this document .....	7
Best Practices Workflow .....	7
<b>Before you get started – Application Control strategy .....</b>	<b>8</b>
Identify Decision Makers .....	8
Communicate with Users .....	8
Decide Where to Start – which features are you going to use / start with? .....	8
What is the primary security problem you are trying to address? .....	9
Rule Creation .....	10
Rule Collections.....	10
Rule Sets .....	11
To Use Trusted Ownership or Not? .....	11
Software Sources – how do users obtain approved software? .....	12
Change Control – process for handling exceptions .....	12
Get Ready for Application Control – patch and clean.....	12
<b>Audit Mode (Passive Monitoring) .....</b>	<b>13</b>
Population Selection .....	13
Communicating with Users .....	13
Agent Rollout.....	13
<b>Review Logs and Update Configuration .....</b>	<b>14</b>
Create Robust Rather Than Brittle Rules .....	15
Least Restrictive Rule Applies .....	15
Test Application Updates .....	15
Where to Start – Privilege Management or Execution Control? .....	16
Privilege Management.....	16
Execution Control - Use Trusted Ownership.....	16
Allow file to run even if it is not owned by a trusted owner.....	16
Use Vendor Certificates where possible .....	17
What if an Application has Unsigned or Untrusted dlls? .....	18
Configuration Snippets.....	20

What about Installers? .....	20
Are you ready for Restricted Mode? .....	21
<b>Restricted Mode .....</b>	<b>22</b>
Communicating with Users .....	22
Silent Deny .....	22
Exception Process Workflow .....	22
Self-Authorizing Mode as an Interim Step for Application Control .....	23
Self-Elevation Mode as an Interim Step for Privilege Management .....	24
Potential Pitfalls .....	25
<b>Monitor and Refine .....</b>	<b>26</b>
Ongoing Configuration Refinement and Deployment .....	26
<b>Appendix 1 - Features &amp; Benefits .....</b>	<b>27</b>
Features & Benefits Table .....	27
<b>Appendix 2 – Configuration Settings .....</b>	<b>30</b>
Features .....	30
Enable Executable Control .....	30
Enable Privilege Management .....	30
Enable Browser Control .....	30
Hash Algorithm .....	31
Advanced Settings .....	31
Executable Control Configuration Settings .....	32
Trusted Owners .....	32
Options .....	32
Validation .....	34
Application Termination .....	35
Access Times .....	36
Privilege Management Configuration Settings .....	37
Self-Elevation .....	37
Events Configuration Settings .....	37
Message Settings Configuration Settings .....	38
<b>Appendix 3 – How does it work? .....</b>	<b>39</b>
How does Application Control Work? .....	39

How does Privilege Management Work? .....	39
How does Browser Control Work? .....	39

This document is provided strictly as a guide. No guarantees can be provided or expected. This document contains the confidential information and/or proprietary property of Ivanti, Inc. and its affiliates (referred to collectively as "Ivanti"), and may not be disclosed or copied without prior written consent of Ivanti.

Ivanti retains the right to make changes to this document or related product specifications and descriptions, at any time, without notice. Ivanti makes no warranty for the use of this document and assumes no responsibility for any errors that can appear in the document nor does it make a commitment to update the information contained herein. For the most current product information, please visit [www.Ivanti.com](http://www.Ivanti.com).

© 2019, Ivanti. All rights reserved.

# Introduction

Application Control prevents unauthorized code execution and enforces software licensing through the “trusted ownership” model which provides less effort to manage over traditional whitelisting and blacklisting. It also manages user privileges and policy at a granular level whilst allowing for optional self-elevation when exceptions occur. Application Control keeps IT security requirements in balance with user productivity needs, delivering endpoint security through privilege and application control, increased corporate compliance, improved platform stability and consistency, and significant reductions in both IT support and software licensing costs.

The Application Control module in Ivanti Security Controls contains the following main features:

- Executable Control

By defining rules to allow or block executable files from running in their environment, administrators can control which applications are allowed to execute. Executable control or application whitelisting is a key component in a layered defense-in-depth strategy providing very effective protection against zero-day threats, ransomware and target attacks. By default, executable control automatically authorizes applications installed by your software imaging process or deployed by your endpoint management solution and blocks any software installed locally by the user or downloaded from the Internet making it very easy to get up and running and very effective at blocking untrusted software including malware.

- Windows Privilege Management

IT & System administrators prefer their users to not have full administrative rights, as this can open up endpoints to security vulnerabilities. However, users often require administrative privileges to perform their role. By removing users’ full admin rights and providing them with elevated privileges for just the applications or tasks they need, you can simplify endpoint security, reduce support calls, and lower total cost of ownership which helps to reduce your organization’s attack surface and impede the spread of malware.

- Browser Control

This feature provides administrators with the ability to control access to the Internet using either a blacklisting or whitelisting approach which helps to eliminate distractions and enhance end-user productivity as end users are prevented from accessing certain websites, such as social media websites, for example.

Each of these main features can be enabled or disabled independently per configuration, depending on whether they are required or not.

For information on the additional sub-features within Ivanti Application Control, refer to Appendix 1 - Features and Benefits.

Application Control complements the patch management module within Ivanti Security Controls to significantly reduce security risk with minimal impact to endpoint performance and end user productivity.

## Purpose of this document

This document aims to help customers be successful with their Ivanti Security Controls Application Control implementation by outlining a best practices implementation workflow that has been developed based on the experience of the many customers who have already implemented either Ivanti Security Controls Application Control or Ivanti Application Control on which this solution is based. If you have any feedback on this document or would like to discuss either the document itself or the Ivanti Security Controls Application Control module, please send an email to [productmanagementsecurity@ivanti.com](mailto:productmanagementsecurity@ivanti.com) to arrange a meeting. We would love to get your feedback.

## Best Practices Workflow

To be successful, you should follow a methodical approach to your Application Control rollout. There are a number of phases in the rollout, as follows:

1. [Before you get started – Application Control strategy](#)
2. [Audit mode \(Passive Monitoring\)](#)
3. [Review logs and update configuration](#)
4. [Restricted mode \(Enforcement\)](#)
5. [Monitor and update](#)

These steps are outlined in more detail in the following chapters.

# Before you get started – Application Control strategy

Preparation is the key to a successful Application Control implementation. Spending a bit of time up front planning out your implementation will really help reduce your ongoing maintenance once you have deployed Application Control to your users. This section helps you understand the questions you need to ask and the decisions you need to make as you roll out the solution.

## Identify Decision Makers

As with any IT project, it is important to clarify who is going to be responsible for each aspect of the implementation. In particular, you need to clarify who will have the following roles:

- Tech Lead - This person will be responsible for rule creation when the configuration needs updating to allow a blocked file to execute, or to prevent specific applications from being used.
- Application Authorization - This person will be responsible for deciding whether applications, particularly blocked applications, should be authorized for a given user or group. In summary, applications deployed by IT are automatically allowed. Everything else requires authorization, which primarily requires someone to make a decision as to whether it should, or should not, be allowed to execute.

## Communicate with Users

Notify your users that you plan to roll out Application Control and provide an overview of what they can expect. Continue communicating with users during each phase.

## Decide Where to Start – which features are you going to use / start with?

Ivanti Security Controls Application Control is not a single feature but a collection of features. As discussed earlier, the main features are:

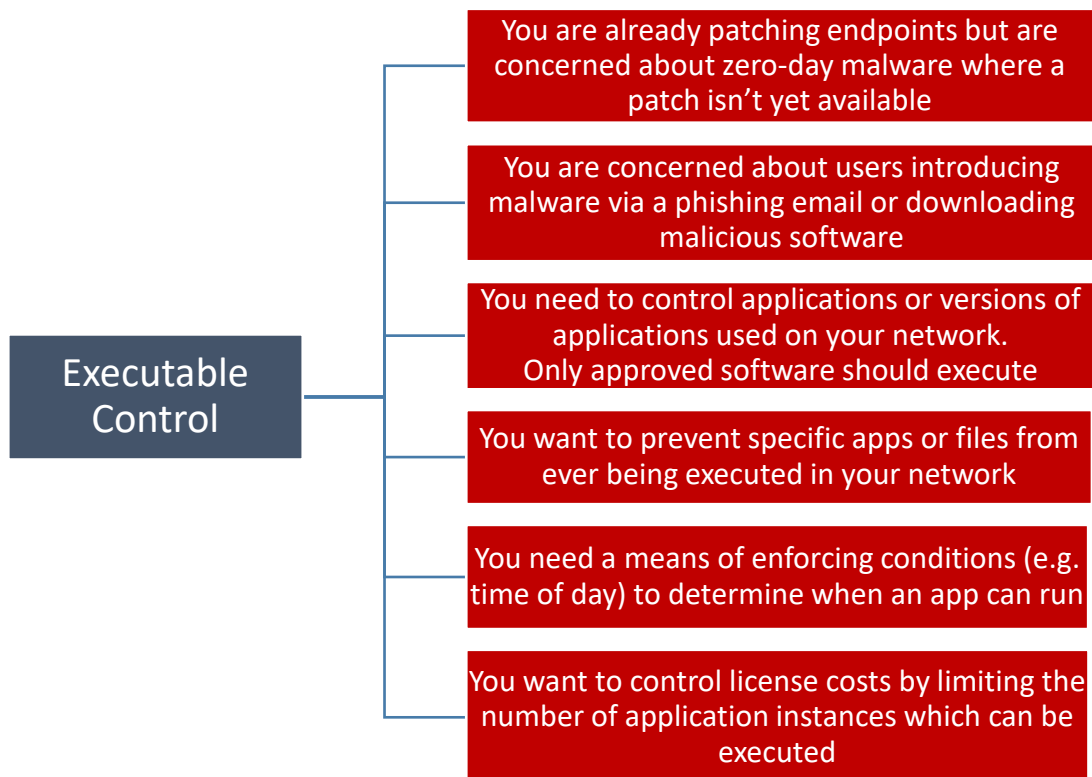
- Execution Control
- Windows Privilege Management
- Browser Control

Refer to [Appendix 1 – Features & Benefits](#) for more information on these and the associated sub-features of each. Additional information is also available at [help.ivanti.com](https://help.ivanti.com).

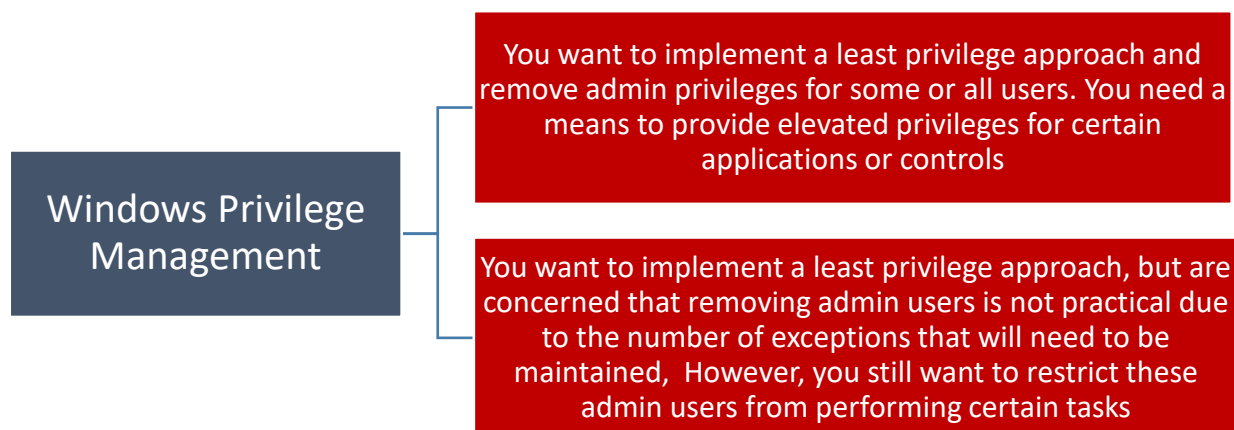


## What is the primary security problem you are trying to address?

If your primary concern is one of the following, then you will probably get started with Executable Control:



If your primary concern is one of the following, then you will probably get started with Windows Privilege Management:



If your primary concern is one of the following, then you will probably get started with Browser Control:



Having reviewed these features and their benefits, decide where you are going to start and follow the relevant steps in this guide to implement these features.

## Rule Creation

Rules are used to define whether applications should be allowed to execute or should be blocked. These rules are defined in the Ivanti Security Controls console.

### Rule Collections

Don't just rush in and start to create rules for individual executables. Think instead about strategies to group application rules so that they can be managed collectively.

Use Rule Collections to group rules together for inclusion within configuration rules. Investing time up front in deciding on the right strategy for your organization will pay dividends by ensuring that you have an easy-to-manage configuration once Application Control has been fully rolled out. Your goal when creating rule collections is to find the right balance for your organization between granularity and flexibility. Once you have defined your rule collections, you will assign these to; Groups, Users and/or Devices. The more granular you make your rule collections, the greater flexibility you have in terms of assigning them. However, the question you need to ask is whether you really need that level of granularity. The best practice is to have as few collections as possible, yet which give you enough flexibility to be able to provide all your users with access to the applications that they need to do their job, while not providing them with access to applications which expose the organization to additional risk or costs.

The following are examples of strategies for grouping applications into collections:

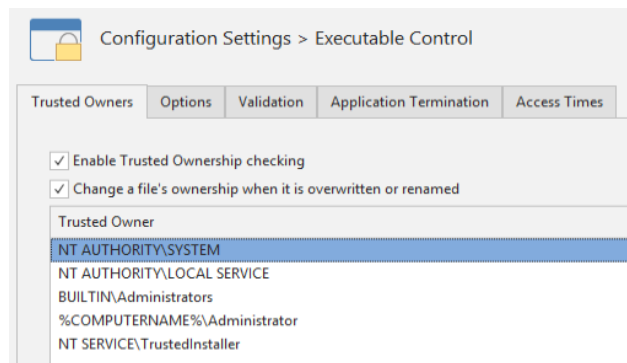
1. Application or application type collection.  
Your rule collections might contain a list of individual applications such as Visual Studio, Webex, Google Chrome, Adobe Acrobat, etc.
2. Roll up by application vendor, application suite or application type.  
In this case, your rule collections might contain entries such as Adobe, Microsoft Office, Cisco, Remote Connectivity, etc.
3. Risk based strategy.  
Your rule collections might contain a single rule collection for all lower risk applications that you expect any user in the organization to be allowed to use (e.g. Microsoft Office) and have other collections for higher risk applications that you only want to be available to certain groups, such as the IT department.

## Rule Sets

Once you have defined your strategy for rule collections, the next step is to assign these rule collections to rule sets. Typically, you will assign these to your AD groups, but they may also be assigned to individual users or to specific devices.

## To Use Trusted Ownership or Not?

Trusted Ownership is a concept whereby a file is allowed or denied based on whether the NTFS owner of the file matches a predefined list of trusted owners. Trusted Ownership checking is enabled by default and is the best practice configuration to achieve the best balance between security and ease of maintenance. With Trusted Ownership any software installed by a Trusted Owner (see default list below) is automatically trusted, which greatly reduces the day-to-day administrative burden of managing a whitelist of applications that are allowed to run. By default, files that are downloaded by a user will not be owned by a trusted owner and will be blocked from executing.



A default set of Trusted Owners is provided for new configurations. It is possible to edit the list of Trusted Owners. However, changes and additions should be avoided to prevent security holes from being introduced. Individual users, in particular, should not be added to the Trusted Owners list.

There may be some situations where customers opt to disable Trusted Ownership. For example, some customers may opt for a blacklisting-only approach, whereby their primary use case is to block certain applications from being used in their environment. The default product behavior is that local drives are allowed by default, so that any executable on a local drive is allowed to run. When Trusted Ownership is enabled, rather than simply being allowed to run, the ownership properties of the executables are checked against the list of Trusted Owners to determine whether they should be allowed to run. When Trusted Ownership checking is disabled, these checks no longer take place, so any executable on a local drive is allowed to execute. Customers can then create rules for applications that they want to prevent from executing so that these applications are blocked.

Determining the correct strategy for your organization depends on your overall security posture and the amount of day-to-day administration effort you are willing to accept. Trusted Ownership is enabled by default and is the recommended configuration.

## Software Sources – how do users obtain approved software?

When introducing Application Control, your goal is to block any unauthorized software from executing. However, it is important that you do this in a way that minimizes the administrative burden associated with users installing authorized software.

You may want users to be able to self-serve and add approved applications that they need to do their job. These applications will typically be located on a network fileshare with restricted write access.

Alternatively, you may want users to submit an IT ticket to request the new software and have it installed via IT. In either case, you will need to ensure that you have a mechanism that is supported with Application Control and conduct testing to confirm that software installation proceeds successfully once Application Control is introduced.

## Change Control – process for handling exceptions

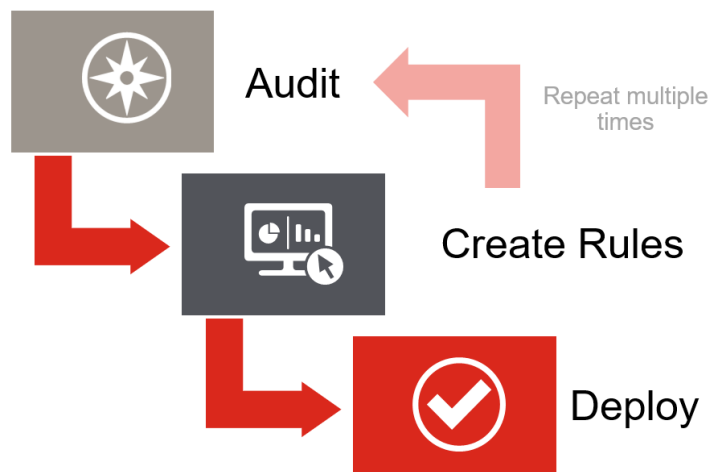
If you follow this best practice guide, by the time you move endpoints into restricted mode, it should result in a steady-state situation whereby applications already on the endpoints are allowed to run and any new, untrusted software, is blocked. However, there will be a need to support ongoing changes including new applications and new versions of existing applications. Users may experience situations whereby applications they need to do their job are being blocked. You need to plan for this ahead of time and ensure that you have processes in place whereby users can report that these applications have been blocked and request that they are authorized. To avoid user frustration and productivity loss, you need to also plan for a quick turnaround on such requests.

## Get Ready for Application Control – patch and clean

Application Control works as part of a layered defense-in-depth. Rather than relying on a single technology to protect your endpoints, multiple defensive layers working together provide a more effective security solution.

As you roll out Application Control, ensure that you continue to keep your endpoints patched up to date to prevent known vulnerabilities from being exploited and that you are also conducting regular malware scans to remove any dormant malware. In addition to creating that layered defense, it will also ensure that these processes are also compatible with Application Control and enable you to resolve any compatibility issues early on.

# Audit Mode (Passive Monitoring)



Once you have decided on which Application Control features you plan to implement and how you are going to group your applications into rule collections, the next step is to enable Application Control on a subset of your endpoints in audit mode.

Security Level: ☐ Unrestricted ☒ Audit Only ☐ Self Authorizing ☐ Restricted

## Population Selection

You should start with a relatively small number of endpoints (50-100) initially to help you understand the rules that you will need to create in your network without running the risk of being overloaded with events from a larger number of endpoints. The initial endpoints selected for audit mode should be from a representative cross-section of the population to ensure the maximum diversity of applications.

This initial pilot group will help validate the rule collection structure that you developed earlier. As you discover new scenarios that you hadn't considered, you may need to make updates to this structure.

In addition, this group can be used to develop and test out your processes including:

- Support workflow
- Authorization decision making process
- Rule creation process

The pilot group will serve as real-world proof of viability for a wider rollout.

## Communicating with Users

As you introduce Application Control, it is important to communicate with users so that they are aware of the changes being introduced but also so that they can report any issues or anomalies that they experience during the roll out process. For example, a conflict with another security application could cause a performance issue which will need to be addressed.

## Agent Rollout

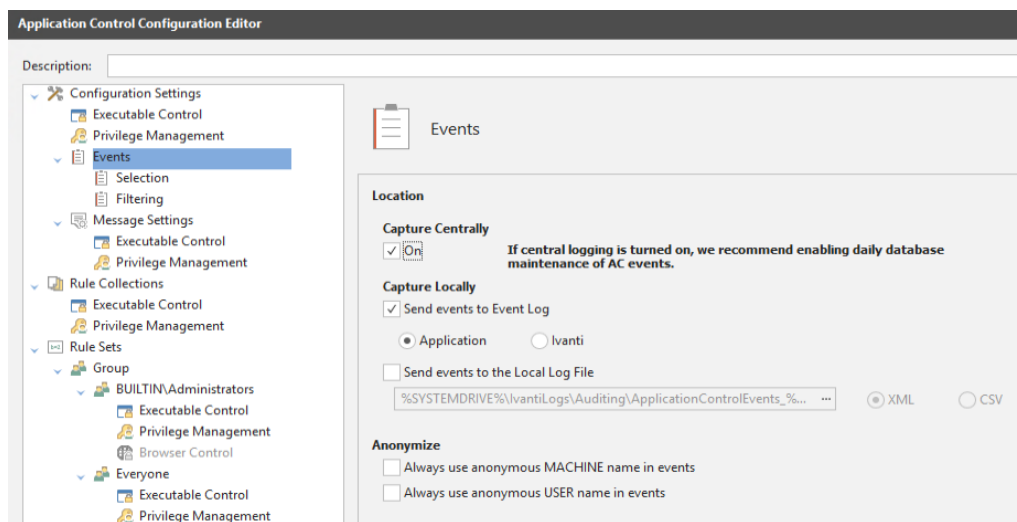
If you are using Ivanti Security Controls to patch your endpoints, you may already have the Ivanti Security Controls agent installed. If not, you will need to install the agent on your test endpoints and enable Application Control. Refer to [help.ivanti.com](http://help.ivanti.com) for instructions and a video demonstration of how to do this.

# Review Logs and Update Configuration

In audit mode, log events will be generated on the Application Control endpoints and these log events can be used to define the configuration rules that will be used once the endpoints are moved into restricted mode. For example, these log events will identify applications that would be blocked or when applications require elevated privileges.

These log events can be viewed from the Ivanti Security Controls console using the Event Viewer. Refer to the [Application Control Events Video](#) on the [Ivanti help youtube channel](#) for a brief overview of how to view log events and update the configuration.

Note that in order to view log events from the console, you will need to update the Application Control configuration to capture the log events centrally as shown below.



While in audit mode, end users will not be made aware of these log events (i.e. there will be no endpoint dialogs presented to the end users) so you can't rely on users contacting you regarding applications that may be blocked, and you will need to review the logs to see these events. You should plan to review log events at least once per day during the initial rollout. This will help you to keep on top of things, and also to understand how changes are occurring on endpoints and the types of changes that are occurring. If you ignore the logs for an extended period, it can be a bit of a daunting task to try to get caught up, depending on the amount of change that has taken place on the endpoints during the intervening period. This is also the reason to keep the number of endpoints small at the initial stage.

As you update the rules to cater for the log events that have occurred, these rules will reduce the number of log events in the next cycle. Over time as you build up the rules, your logs should progressively become quieter. At that point, you can start to think about moving these endpoints into restricted mode and also extending the population to a broader number of users.

Using the logs to update the configuration rules will really help develop your understanding of how the rules work and how to make effective rules. At the initial stage it will be a bit of an iterative process – you may add rules which don't work fully (i.e. additional log events continue to occur for that application) or you may add rules which are too generic and you may need to update them over time to make them more restrictive or secure. Best practice is to keep it simple and clear. Make sure you document as you go. Someone else may have to maintain the configuration in addition to you or in future so ensure that you add good descriptions for any rules you create or changes you make.

You should initially focus on applications that are being blocked across multiple users. Addressing these will have the biggest impact. The number of users is generally more important than just the number of times a given executable has been blocked.

## Create Robust Rather Than Brittle Rules

When creating rules, you are trying to find the best balance between security and administration overhead for your organization. This does not mean that you have to trade off one for the other. It is possible to create very secure rules which will persist over time and don't require ongoing maintenance. It is also possible to create very secure rules which are brittle and may need to be updated time and again as the application vendor releases new versions. This could make sense if you really want to control and only allow specific instances of application files to run but, if you adopt this approach, you need to have the necessary resources to maintain these rules on an ongoing basis. Best practice is to keep the configuration as simple as possible. So, unless you have specific reasons for doing otherwise, you should aim to create secure rules which require little or no ongoing maintenance. Wildcards and regex conditions can be used to help with this.

To allow files that are denied due to trusted ownership failure, one of the following actions should be taken:

- Use vendor certificates. Add an allowed item rule using vendor certificate metadata enabling any signed file from that vendor to be allowed.
- Add the specific file(s) to the configuration as an allowed item ensuring metadata for the file(s) is specified.

Both of these concepts will be explained in more detail later in this document.

## Least Restrictive Rule Applies

One important consideration to note when creating rules is that the least restrictive rule applies at the same level (file, folder, file hash) so that, if an application (e.g. Adobe Reader) is added to a denied file rule but there is another allowed file rule that allows it (e.g. any file signed by Adobe), it will be allowed to run. In other words, an allowed file rule beats a denied file rule.

However, note that there is a rule hierarchy as follows:

- 1) File Hash
- 2) File
- 3) Folder

So, a file hash rule beats a file rule which, in turn, beats a folder rule. For example, an allow folder rule is beaten by a deny file rule.

## Test Application Updates

There is no standard approach for how applications perform updates and different vendors will perform updates in different ways. Because of this, rules that work for one vendor may not work for another. As such, it is important to test that your rules will continue to work through application updates. You can test this by installing older versions of applications and allowing them to update to the latest version to determine whether any log events are generated. However, it is also good practice to ensure that the test endpoints go through at least one patch cycle (Patch Tuesday) to ensure that the rules are robust enough to accommodate the typical monthly updates across both Microsoft and 3<sup>rd</sup> party applications.

## Where to Start – Privilege Management or Execution Control?

The privilege management and execution control features complement each other very well and should both be implemented as part of a layered defense to provide the best protection. But, which feature should you implement first? In general, privilege management is an easier and faster implementation than execution control so, if you plan to implement both features, start with privilege management.

### Privilege Management

Security best practice dictates that, where possible, you should remove local admin privileges. In addition, untrained users with admin privileges can break their own machine, other users' machines, and potentially servers if they modify settings they don't understand. The Privilege Management feature set provides you with the ability to elevate or restrict admin privilege on applications and allow or restrict access to Windows OS components and OS system controls. Where it is deemed necessary to continue to use local admins, you can reduce the privileges for these users. Refer to the [9 ways privileged users create security risks](#) infographic and download the associated [whitepaper](#).

However, in most cases, you want to remove local admin privileges and configure your users as standard users, and for these users, you may need to provide them with additional privileges to perform their job. To understand what these required privileges are, you can view "Application user privileges changed" events via the Event Viewer. In a scenario where you have existing local admins that you are changing to standard admin accounts, you would gather event data for a period of time (e.g. 2 weeks) to identify any applications that are executed with admin privileges. Where these privileges are necessary for the users to perform their jobs, you would then add privilege management rules to provide the necessary elevated privileges. Once the users are switched to standard users, they will then be able to use these applications seamlessly as if they were local admins.

### Execution Control - Use Trusted Ownership

As described earlier, using Trusted Ownership will greatly ease the administrative burden associated with Execution Control. Any applications which are introduced by a trusted process (system management tools, administrators, etc) are allowed to run. Any applications which are not owned by a trusted owner are blocked by default. This is really simple and means, for example, that, by default, any applications installed on new systems which have been created from your gold image will automatically be allowed to run.

Depending on how you have introduced change in your environment and how long these systems have been in the field, you may have experienced some drift from this "pure" state but, Trusted Ownership provides you with a great starting point for Execution Control.

### Allow file to run even if it is not owned by a trusted owner

The default behavior for the rules engine is to block executable files if they are not owned by a trusted owner. However, when Trusted Ownership is enabled, which is the recommended operating mode, if a file gets blocked because it is not owned by a trusted owner and we are creating a rule to overcome that, we want the file to be allowed to run. As such, it is important to remember to check the option as shown below, when creating rules as, otherwise, the rule will have no effect.

Options

☒ Allow file to run even if it is not owned by a trusted owner

☐ Ignore event filtering



## Use Vendor Certificates where possible

Many applications are fully signed by the application vendor and this provides a simple, yet secure, method to allow these applications to run, if they are not already allowed by Trusted Ownership.

From the Event Viewer, you can easily create an allowed rule via drag-n-drop or by copying the event into the configuration pane. Doing so also copies all of the file metadata into the rule and you can use this metadata to create a secure, yet flexible rule. In the screenshot below, you can see that putty.exe has been blocked a number of times

Total	User Count	Event Id	Company Name	Path	File Description	File Hash
4	1	9000	Simon Tatham	%userprofile%\desktop\putty.exe	SSH, Telnet and Rlogin client	5EF9515E8FD9
2	1	9000		%userprofile%\desktop\message.ps1		ED8873E8471B
2	1	9000	Microsoft Corporation	%systemroot%\regedit.exe	Registry Editor	ASB2AC70A6E
1	1	9000	Microsoft Corporation	%systemroot%\system32\windowspowershell\v1.0\powershell_ica.exe	Windows PowerShell ISE	82F1D9C4D95F

In this case, you could create a filename rule for putty.exe by copying the event into the configuration pane and you can secure this rule with some additional metadata. In this particular case, this may be sufficient as putty is a relatively simple application with a single executable file. However, I could make this a more generic, yet secure rule, by changing the filename to \*.exe or even \*.\* if there were multiple executables from the same vendor that are being blocked.

While, at first glance, this might appear to be a very insecure rule, you can secure this rule using the file metadata and, in particular, the vendor certificate information. In this case, you see that the vendor is listed as “Simon Tatham” so you select the Vendor checkbox along with the “Verify certificate at runtime” checkbox.

So, when putty.exe runs, because it fails the Trusted Ownership check, the rules engine will use the above rule to check that the file is signed by Simon Tatham and will verify the certificate. As putty.exe meets these criteria it will be allowed to run. Other executable files which don't meet these criteria will be blocked.

Using vendor certificates in this way makes it really easy to trust signed applications which would otherwise be blocked. You can further restrict the rule, if you feel this is necessary, using the other metadata fields as part of the rule (e.g. Product Name, Product Version) so that you are not simply trusting any file from this vendor.

You should also update your processes to ensure that applications that are developed inhouse are signed with a corporate certificate. This will allow for a much smoother rollout experience both for Application Control and for updated versions of these applications.

## What if an Application has Unsigned or Untrusted dlls?

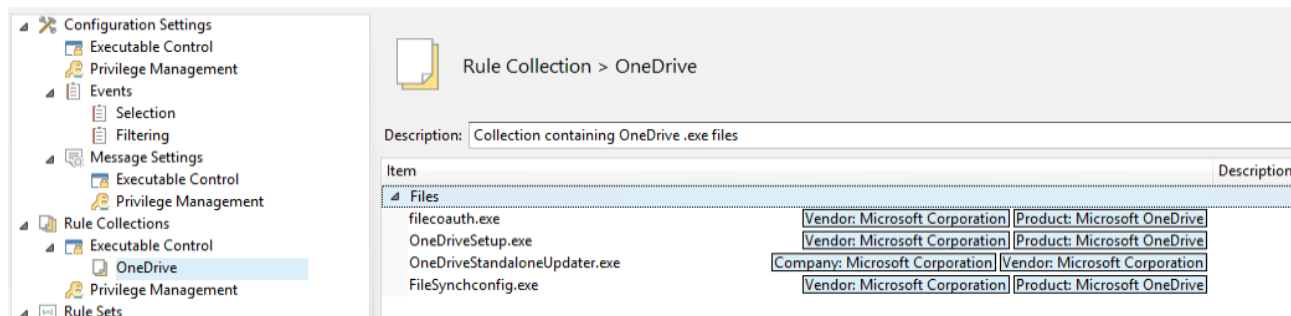
Between Trusted Ownership checking and Allowed Rules based on vendor certificates, you should be able to cover the vast majority of applications in your environment. However, there may still be some exceptions. One of these is where the primary executable for an application is signed but the application has secondary executables or dlls which are not owned by a Trusted Owner and are also unsigned. This can occur when an application is installed or updated in the user profile to overcome security restrictions. If the app is installed in the user profile, administrator rights are not required.

Applications such as OneDrive are updated in this way. OneDrive is an application that is updated regularly and to prevent these updates from being blocked due to the lack of admin credentials, they are updated in the user profile. As the resulting file owner is the logged-in user and not a Trusted Owner, these updated files will be blocked. If these files are signed, you can still use the vendor certificate to create a rule to allow these to run. But, if they are not or, maybe, as in the case of OneDrive which is Microsoft-signed, you don't want to simply trust any executable which is signed by that vendor, you need a different approach.

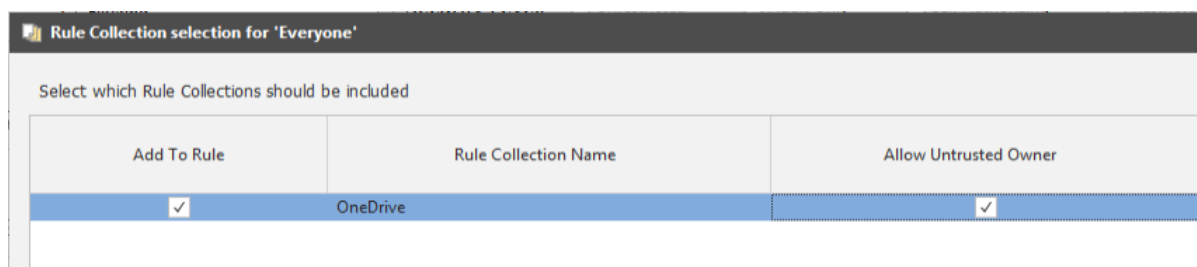
The best approach for this scenario is to use a combination of an Allowed Rule (e.g. Group Rule) and a Process Rule.

Process rules are good for securing executables and dll's we need to run but only in certain circumstances. With process rules, you specify a parent application. When the parent .exe (i.e. the process) launches a child executable or loads the dll they are allowed to execute, but only in that scenario.

Using the OneDrive example, you create a OneDrive rule collection and add the primary executable files to this collection.

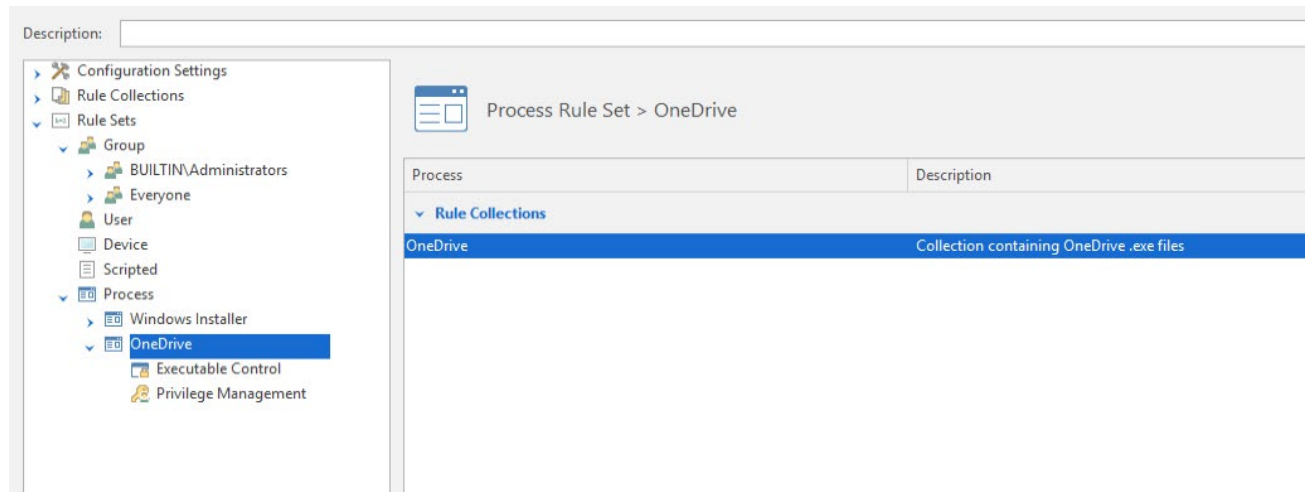


You would then assign this rule collection to the appropriate group (e.g. Everyone) ensuring to check the “Allow Untrusted Owner” checkbox:

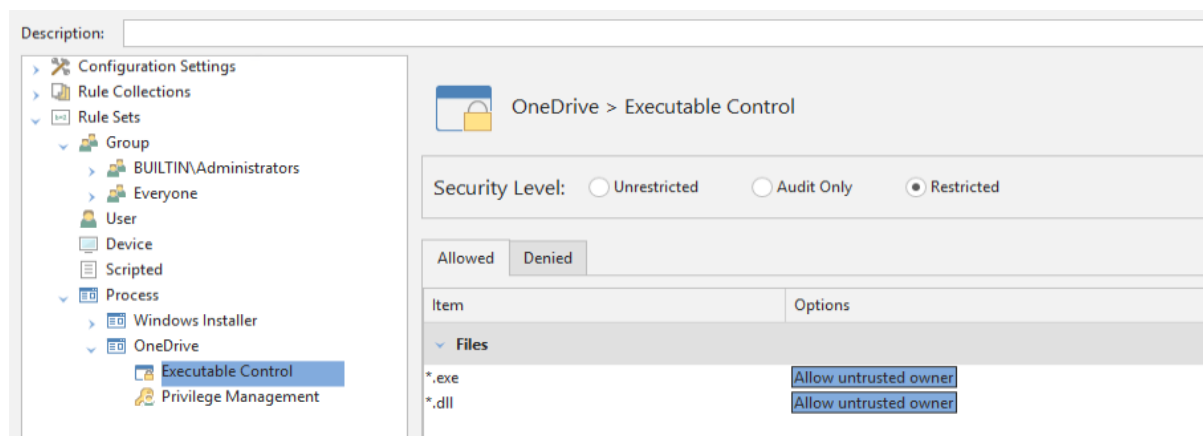


At this point, each of the parent OneDrive executables will be allowed to run, but any unsigned or untrusted dlls that they call will be blocked. This is where the process rule comes in.

Create a OneDrive process rule set and add the same OneDrive rule collection to the parent node.



Next, add \*.exe and \*.dll (or even \*.\* ) as allowed file items under Executable Control under the OneDrive process rule. Don't forget to also check the "Allow file to run even if it is not owned by a trusted owner" checkbox in the file rule. The resulting rule is that whenever the parent OneDrive executables run, any child executables or dlls that they call will also be allowed to run.

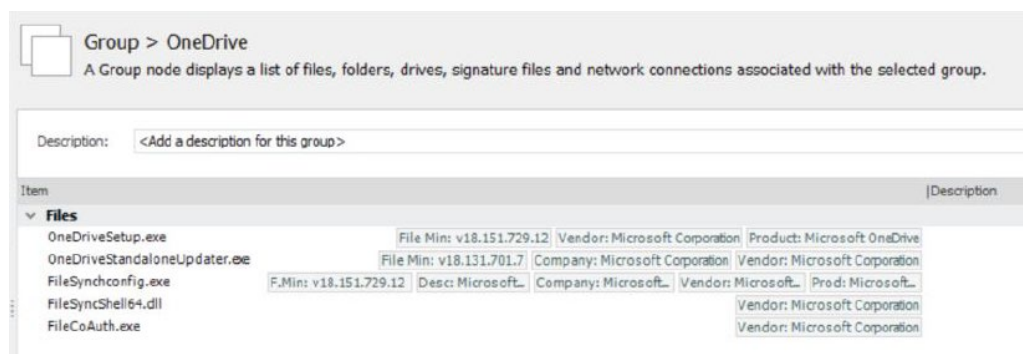


You can further secure and restrict these rules by adding file metadata (e.g. if the dlls are actually signed) and possibly even include the specific dll names but remember the more specific you make the rule, the greater the risk that the rule will need to be updated if the vendor changes something.

## Configuration Snippets

Ivanti has developed configuration snippets for a number of common applications that typically fail Trusted Ownership checking. These include applications such as Microsoft OneDrive, Microsoft Teams and GoToMeeting. These snippets and associated documentation are available on the [Ivanti Marketplace](#).

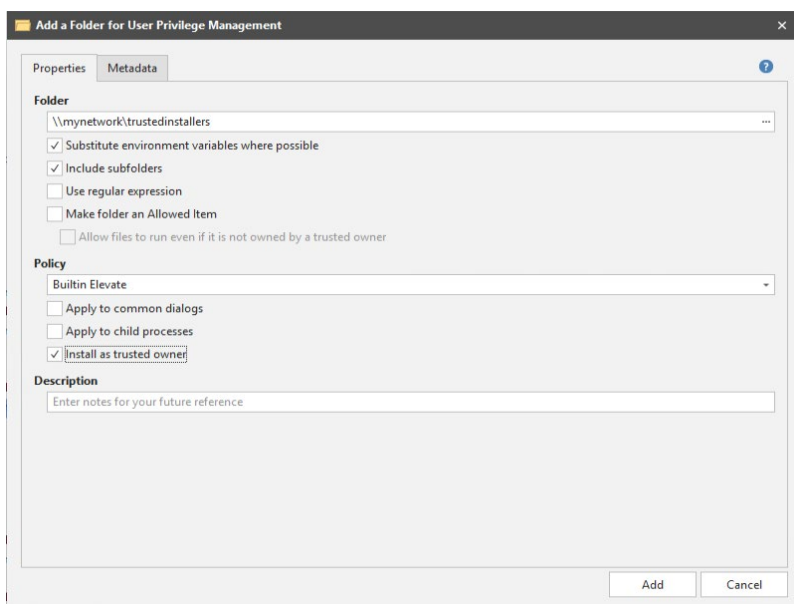
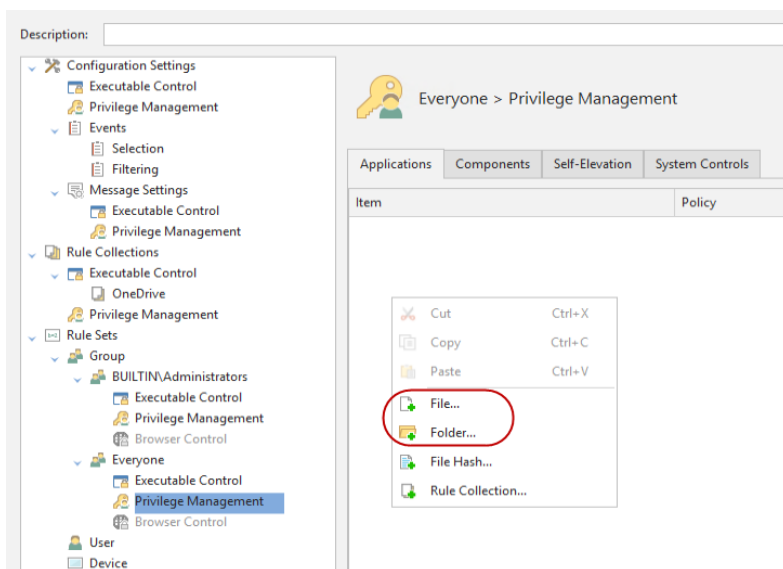
It is not currently possible to import these directly into Ivanti Security Controls Application Control. This is planned for a future release. However, each snippet contains a pdf guide which provides the associated configuration details (see example below) and this can be used both to understand which files need to be included in the rule and the associated metadata settings.



## What about Installers?

If a user downloads an installer from the Internet, it will be blocked via trusted ownership checking, because the installer will be owned by the logged-in user. But, if you have a repository of software that is approved for employee use, you may want to enable users to install this approved software on their own endpoints rather than requiring IT to install it for them. To achieve this, two separate configuration rules are required:

- 1) **Trusted Fileshare** - The first step is to create a folder rule for a fileshare on your network (e.g. \\mynetwork\trustedinstallers). This network share should also be secured so that files can only be added to this location by members of the IT team. When a user tries to run an installer from the network share, it will be allowed to execute due to this folder rule.
- 2) **Elevated Application Rule** - While the addition of the trusted fileshare will allow the installers to run, the user also needs elevated privileges to install applications. As best practice requires that users do not have local admin privileges on their endpoints, the users will typically be configured as standard users, so installation privileges will be required to complete the install. This can be achieved by adding either the network folder or the individual installers as elevated applications under Privilege Management, as shown in the following graphic. Make sure that you check the "Install as trusted owner" checkbox to ensure that the installed applications will be allowed to run via trusted ownership checking.



## Are you ready for Restricted Mode?

While in audit mode, you will continue to review the events and update the rules on a daily basis to identify applications that would be blocked if the endpoints were in restricted mode. As each of these rules gets added, they will then be used to authorize subsequent executions of these applications, so they are allowed to run. As a result, the number of events should reduce over time.

Once the events have reduced to a trickle or you've experienced a few days without any new events, you are now ready to move this group of endpoints into restricted mode and/or expand the number of systems that are in audit mode.

# Restricted Mode

In restricted mode, unlike in audit mode, applications will be blocked unless they are authorized by Trusted Ownership or one of the configuration rules added by the administrator, and the users will receive a blocked message dialog.



For this reason, it is important to ensure that you have spent sufficient time in audit mode to identify any regularly used applications that will be blocked in restricted mode. Otherwise you face the prospect of having frustrated users, lost productivity and a flood of help desk tickets.

## Communicating with Users

In advance of moving endpoints into restricted mode, make sure you communicate with the associated users to let them know what to expect, and also what to do in the event that an application is blocked, which they need to access to do their job.

## Silent Deny

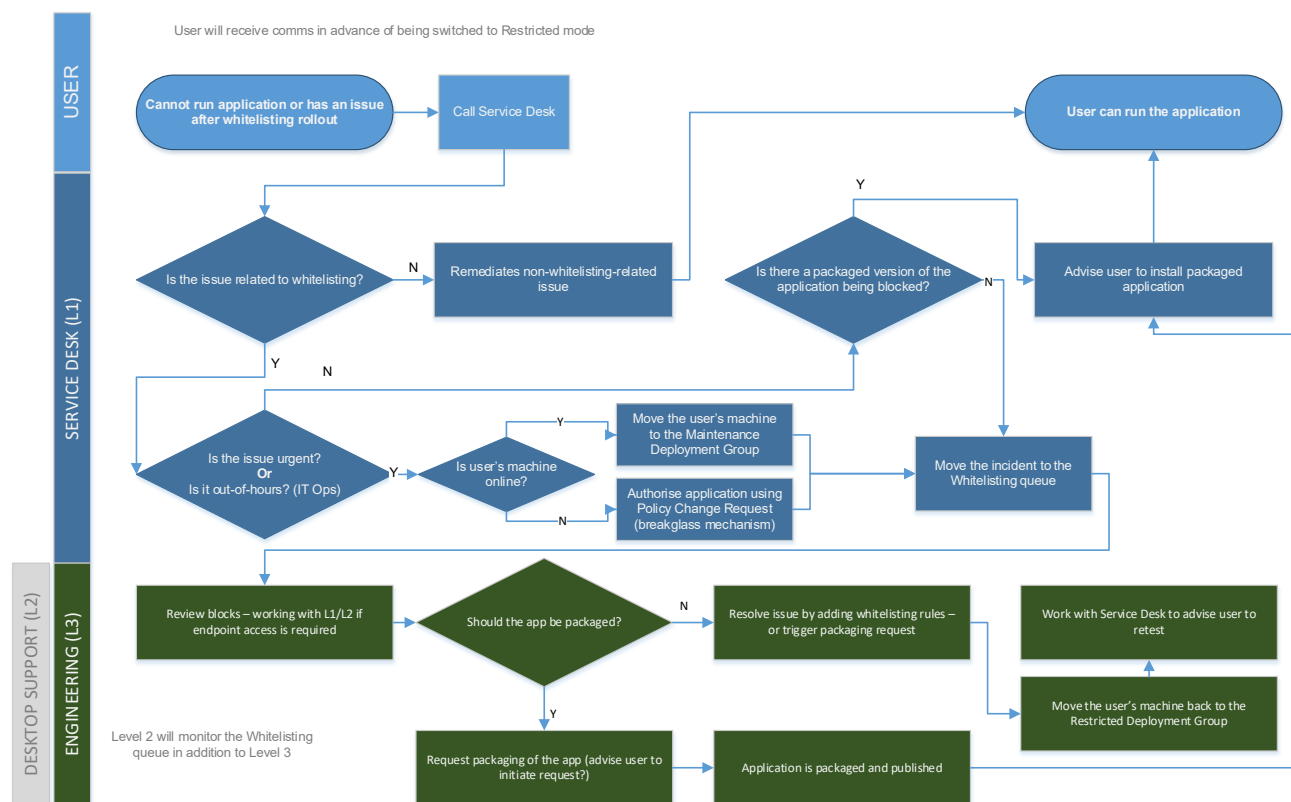
Every time an application is blocked, the user receives a blocked dialog. In the case of some applications, such as application updaters, which run on a schedule rather than being activated by user action, the correct behavior may be for these to be blocked if you want to centrally control when applications are allowed to update. However, every time an application is blocked, the user receives a blocked message dialog and, even though this is expected behavior, these can become an annoyance for users. To prevent this from occurring for these types of applications, you should select the “silent deny” option when creating a denied rule so that the access denied message is not shown when these executables are blocked.



Note that if an application updater is blocked by Trusted Owner rather than by a denied rule, this option doesn't apply and the user will still receive a blocked message.

## Exception Process Workflow

When an application that a user needs in order to be able to do their job is blocked, you need to have an effective exception workflow which will enable the user to notify IT that they need an exception, along with the ability to review and action the request in a timely manner. This workflow will typically be integrated into your normal help desk workflow and the associated details should be included in the communication to users. The following is an example of a support workflow associated with Application Control.

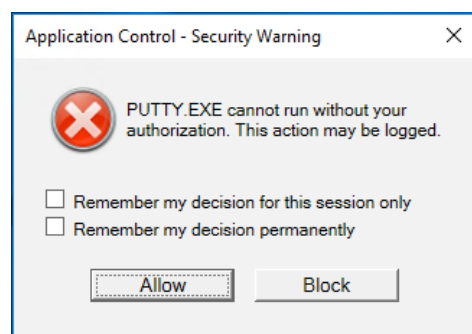


## Self-Authorizing Mode as an Interim Step for Application Control

While getting to restricted mode is the desired state, you can also use self-authorizing mode as a stepping stone between audit mode and restricted mode.

Security Level: ☐ Unrestricted ☐ Audit Only ☒ Self Authorizing ☐ Restricted

In self-authorizing mode, an application, which would otherwise be blocked, is instead presented to the user for authorization with options to authorize just for the current session or permanently.



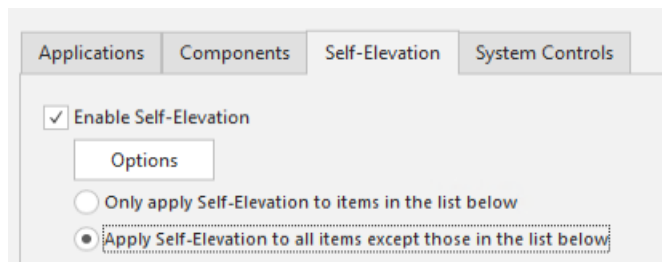
Using self-authorizing mode as an interim step between audit and restricted modes has a number of benefits:

- 1) While you've been reviewing events and updating rules on a daily basis during audit mode and you think that you are now ready to take the next step but are concerned that something might happen to cause a flood of new events which, in restricted mode, would cause applications to be blocked and user productivity impacted. Customers may decide to wait a bit longer before moving out of audit mode due to this concern. Self-authorizing mode alleviates this concern by providing users with the ability to self-authorize any blocked applications and get on with their job.  
You can continue to monitor log events while users are in self-authorizing mode, so you understand how many such authorization requests were presented to users and the allow/block action that they took. You will continue to update your rules based on these log events and, if there are very few events raised in the days following self-authorizing mode being introduced, you know that you are ready to move into restricted mode.
- 2) While in audit mode, when users open applications, they are unaware whether the application would have been blocked if they had been in restricted mode because the application just opens as normal. However, once in self-authorizing mode, applications that would have been blocked are now presented to them to authorize. This starts to create the awareness that things are changing which will be helpful when you subsequently go into restricted mode.
- 3) Self-authorizing mode also limits the spread of potential malware in your environment. As each user has to authorize any new/untrusted applications, it limits the infection to just their endpoints.

## Self-Elevation Mode as an Interim Step for Privilege Management

You can use self-elevation as an interim step for Privilege Management when removing admin accounts. While you will have already conducted discovery to identify any applications that need to be run as elevated, and will have updated the rules to cater for these. You could find that, once you have converted your admin users to standard user accounts, they are unable to use certain applications because they didn't use them during the monitoring period and rules have not been created to allow for these applications to run as elevated.

To alleviate any concerns about removing admin privileges, you can enable self-elevation for a period of time to allow users to run applications with elevated user privileges.



Applications Components **Self-Elevation** System Controls

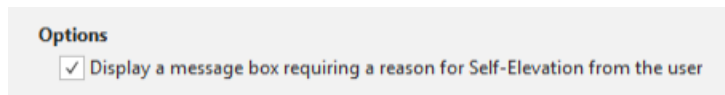
☒ Enable Self-Elevation

Options

☐ Only apply Self-Elevation to items in the list below

☒ Apply Self-Elevation to all items except those in the list below

In the privilege management configuration settings, you can also request the user to provide a reason for self-elevation.

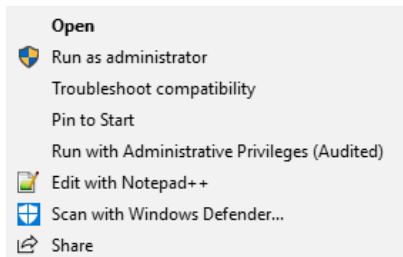


Options

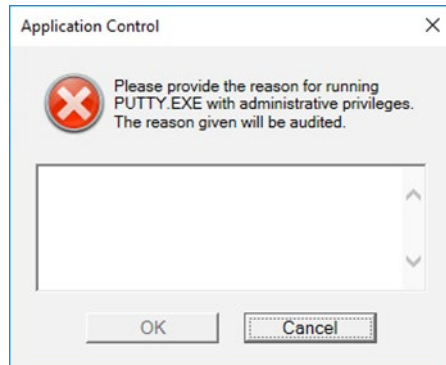
☒ Display a message box requiring a reason for Self-Elevation from the user



If a user needs elevated privileges, they have an additional option to run the application with administrative privileges:



If the “reason” option has been enabled, they will then be requested to provide a reason before continuing:



You will continue to receive application user privileges changed notifications and can update the configuration to elevate these applications. If the users do not have a legitimate need for self-elevation in the longer term, you can then remove this as an option.

## Potential Pitfalls

Some potential pitfalls to watch out for as you move into restricted mode:

- Support
  - Timeliness - You need to ensure that your support process can respond to Application Control related support tickets in a timely manner. You won't have addressed everything in audit mode and you should expect to receive requests for exceptions over time. Users will become frustrated if these are not dealt with in a timely manner and will request to be removed from restricted mode if delays persist.
  - Lack of clear roles and responsibilities - When Application Control related support tickets are raised requesting exceptions for blocked applications, there needs to be clear ownership for the associated decision making. Make sure you clearly identify who the decision makers are and that the tickets are routed to them.
- Skills
  - Ensure that you are not reliant on a single subject matter expert. Think about what happens if that resource is on vacation or out sick. Plan to have a backup, or multiple resources, who can perform the necessary tasks, whether related to decision making or rules creation.
  - Avoid “swiss cheesing” your configuration whereby you make it full of holes to support individual exceptions. Just like a firewall, once you allow everything it is much harder to go back and impose restrictions.

The majority of issues you are likely to face in Restricted mode will be process/support based rather than technical issues around the product, so make sure you address these potential pitfalls before moving users into Restricted.

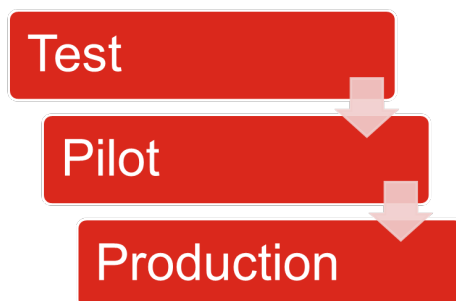
# Monitor and Refine

Now that your endpoints are in restricted mode, you are in control of what software is allowed to run in your environment and have added a very effective layer of protection against malware and unwanted applications.

## Ongoing Configuration Refinement and Deployment

As changes take place in your environment, you will have an ongoing need to make configuration rule updates to support new applications and possibly new versions of applications where the existing rules no longer work due to some changes in these applications.

As you make these ongoing changes, you will need to have a process to test and deploy these in a controlled manner across your environment. This will involve testing the rule changes in your development environment initially, then deploying the updated configuration to a broader test environment before finally pushing it out to the entire population. These groups (development, test and production) could be the same groups that you used as you rolled Application Control out in your environment initially or you may need to refine these groups now that you are using Application Control at scale. Either way, the best practice is to have three different rings or waves within your population to ensure that configuration updates are pushed out in a controlled manner and don't cause any productivity issues.



# Appendix 1 - Features & Benefits

The following section provides an overview of the features and associated benefits of the Application Control feature set in Ivanti Security Controls.

## Features & Benefits Table

Feature	What it does	Benefit
Windows Privilege Management	<p>This functionality allows IT admins to specify which applications, control panel applets and system controls (e.g. process termination &amp; application uninstall) can run with admin privileges for specific users. Full admin privileges can then be removed from the user, yet the user can continue to perform needed tasks, with elevated privileges only where required.</p> <p>Both elevation and restriction options are supported allowing for a standard user to have elevated privileges or an admin user to have privileges removed for situations where removing admin users is problematic.</p>	<p>IT &amp; System administrators prefer their users not have full administrative rights, as this can open up endpoints to security vulnerabilities.</p> <p>By removing users' full admin rights and providing them with elevated privileges for just the apps or tasks they need, you can simplify endpoint security, reduce support calls, and lower TCO.</p>
Self-Elevation	<p>Used in conjunction with Privilege Management, self-elevation provides an option from the Windows Explorer shortcut menu to run an item with elevated rights.</p> <p>When a user attempts to elevate a specified item, a prompt displays to request that the user enters a reason for the elevation before it is applied.</p> <p>Self-elevated items are audited so that the administrator can review and update the configuration to cater for this application.</p>	<p>Maintain productivity by empowering some users with the ability to elevate specified applications when required to perform administrative functions.</p>
Trusted Ownership Checking	<p>Trusted Ownership checking is the default, out-of-the-box security method used by Application Control.</p> <p>It relies on examining the NTFS owner of an application. If an application is introduced on to an endpoint, and hence owned by a 'Trusted Owner', e.g. an administrator or a software deployment system such as Microsoft SCCM, then everyone can execute the application unless otherwise stated. If the application was introduced, and hence owned, by a non-Trusted owner, e.g. a standard user, then no one can execute the application.</p>	<p>Trusted Ownership Checking protects endpoints automatically without the need for complex configurations and constant management.</p>

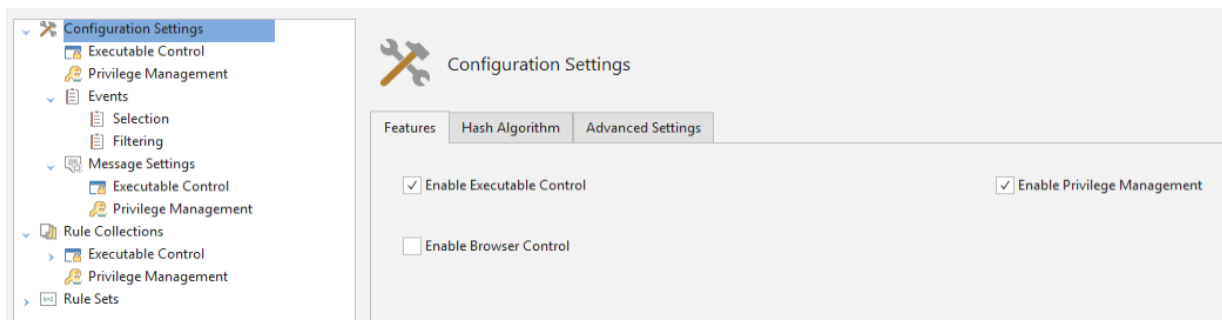
Feature	What it does	Benefit
Executable Control (Whitelisting and Blacklisting)	Define a list of denied items (blacklists) to protect against known threats and problem applications, or create a list of allowed items (whitelists) to guarantee that known and trusted applications can execute on a system. Specific files, folders, drives, file hashes and rule collections can be added to create rules for allowed or denied items. File metadata such as product name, certificate info and company name can be included to make the rules even more specific. Applications that users should not have access to, such as administrator-owned tools like cmd.exe or ftp.exe, are denied automatically.	IT can use allowed and denied items in conjunction with Trusted Ownership Checking to manage exceptions. Whitelist and blacklist configurations can also be used to provide an alternate application control method to Ivanti's out-of-the-box Trusted Ownership Checking to ensure a rigid, locked-down environment for highly secure environments.
File Hash Checking	Digital Signatures check the cryptographic hash (think electronic fingerprint) of files against blacklists or whitelists for advanced security and assign SHA-1, SHA-2, or Adler32 digital signatures to applications and files to ensure application integrity. Modified or spoofed applications are prevented from executing.	Provides IT with peace of mind knowing that applications and files installed on a system remain unaltered, maintaining system integrity. Digital signatures provide the ultimate security control as they enable a way of identifying an individual file very easily.
Passive Monitoring (Audit Mode)	Monitors unauthorized execution attempts without preventing users from running the applications. Unauthorized executions are logged and the log event can be used to update the configuration to authorize this application (if required) once the endpoint is in restricted mode.	Provides an extremely useful tool to accurately track user behavior prior to full implementation, or to understand application usage for software license management.
Self-Authorization	Any user configured as self-authorizing will have the option of allowing an untrusted executable to run, either once, every time during the current session, or always. Comprehensive auditing details information such as application name, time and date of execution, and device.	In some environments it's necessary for users to add new executables to a computer, for example, developers who constantly update or test internal software, or power users who require access to new or unknown applications. Self-Authorization allows nominated power users to execute applications they have introduced into the system. These power users can add applications to a secured endpoint while outside the office without relying on IT support.
Rule Collections	Rule Collections is a library for compiling reusable groups of files, folders, drives, signatures and network connections that can be associated with rules in the configuration.	Simplify rule management by grouping rules into collections and managing at the collection rather than the individual rule level.
Rule Sets	Rule sets enable administrators to create rules targeting specific users, groups and devices. The following five rule sets are supported: <ul style="list-style-type: none"> <li>Group Rules</li> <li>User Rules</li> <li>Device Rules</li> <li>Scripted Rules</li> <li>Process Rules</li> </ul>	Provides flexibility to develop rules to suit business needs.

Feature	What it does	Benefit
Process Rules	Implementing process rules allows IT to determine what processes (children) can be run by an application (the parent).	Process rules help automate application updates and other situations where child processes are spawned which would otherwise fail Trusted Ownership checking.
Scripted Rules	Scripted rules allow custom rules to be created using Windows PowerShell or VB Scripts. The success or failure of the Script determines whether the security level, Allowed Items, and Denied Items that are part of the rule apply to the user. For example, a customer might create a script rule to determine if a user is a member of a certain OU.	Enables customers to develop custom rules that makes sense for their business.
Application Termination	Application Termination allows you to control triggers, behavior, and warning messages for terminating applications on managed computers. You can also control the manner in which applications are terminated and how the user is notified.	In addition to controlling when applications are allowed to execute, IT can also cause applications to terminate if certain conditions are met, such as a change to the computer IP address or when a new configuration is applied.
Application Limits & Time Restrictions	Limits the number of instances allowed to run on a per-user, per-device, or per-application basis. A further level of control over application access can be achieved by applying time restrictions so users can only run programs during certain hours and for a certain length of time.	Application limits can be used to enforce corporate license policies, ensuring only authorized users can run business applications and only during certain time periods.
Browser Control	URL Redirection can be used to create a list of blacklisted URLs and presents the user with a URL-denied warning page or redirects them to an alternative URL when they attempt to access specified URLs. Alternatively, by creating a redirection that blocks access to all internet sites, this feature can be used to implement a whitelist approach by creating a list of web sites that can then be accessed by users.	Provides IT with the ability to control access to the Internet using either a blacklisting or whitelisting approach.

# Appendix 2 – Configuration Settings

This appendix provides an overview of the main product configuration settings and the best practices for these settings.

## Features



### Enable Executable Control

This option determines whether or not Executable Control is enabled. By disabling Executable Control, Application Control will no longer block (or validate) executable content defined within the rules set out by the configuration file. This means that Allowed Items and Trusted Ownership will not function.

Best practice is to ensure Executable Control is enabled and an appropriate configuration file is applied following a period of audit-only mode to ensure the correct allowed items have been defined within a configuration file.

### Enable Privilege Management

This option defines whether or not Privilege Management should be applied. Privilege Management allows the Application Control agent to temporarily grant processes the privileges required to perform defined operations.

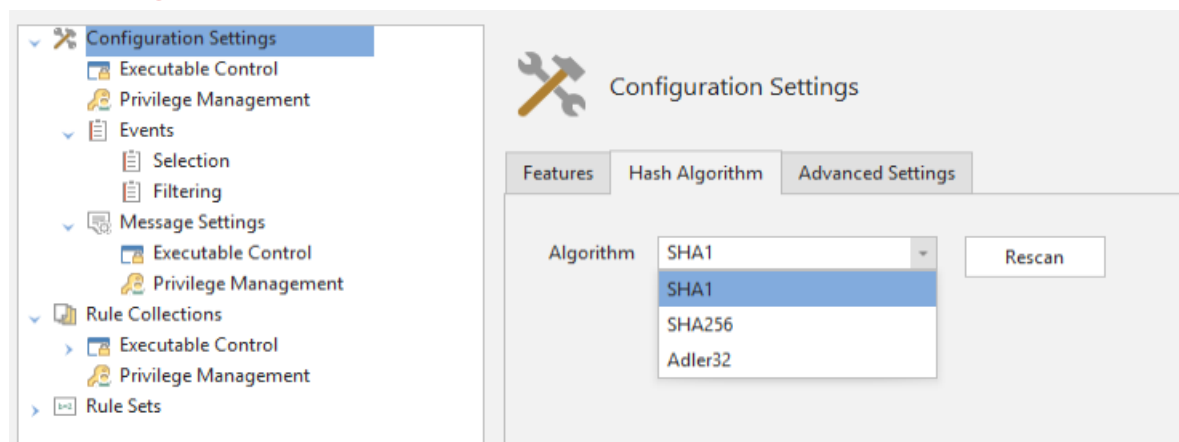
An example use case would be to allow an end user to change the date and time, install a printer, or install software, without needing to be granted Administrator rights on their endpoint.

Best practice is to leave this setting enabled except where the functionality is not required.

### Enable Browser Control

This option defines whether or not Browser Control is enabled. Browser Control provides the ability to control access to the Internet using either a blacklisting or whitelisting approach. It is disabled by default but should be enabled if you have a need to block or redirect URLs.

## Hash Algorithm



Application Control provides administrators with the ability to make files accessible, elevate or prohibit them based on the file hash. This hash is then used to guarantee the integrity of a file before allowing a user to execute this file.

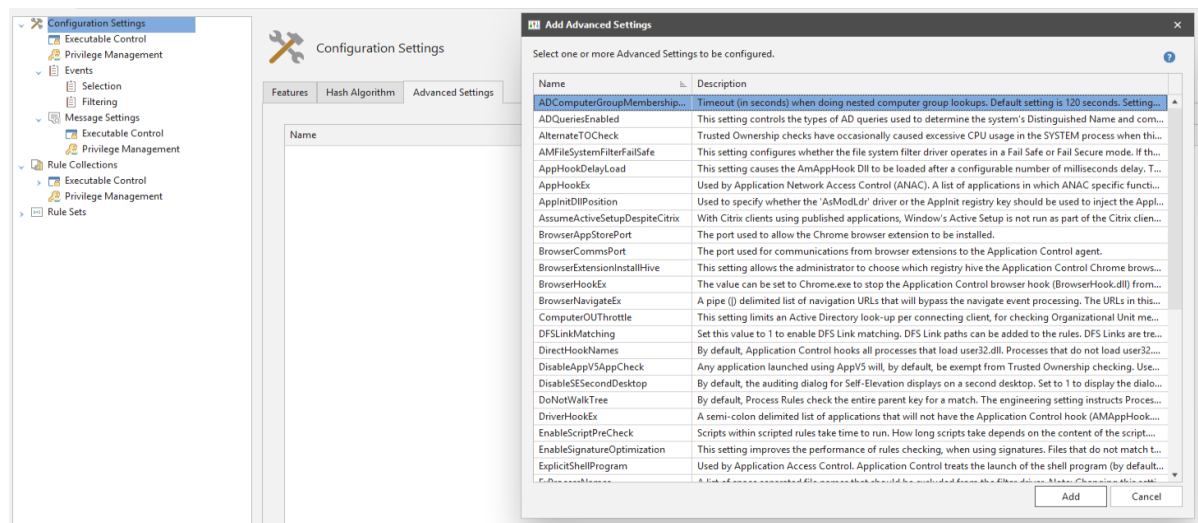
Three algorithm options are available within Application Control, namely:

- SHA1
- SHA256
- ADLER32

By default, SHA1 is selected to maintain compatibility with customers upgrading or migrating from older product versions. However, if you plan to use this feature, the best practice is to select SHA256 as the algorithm with which to scan files.

Note also that you can only use one hash type throughout a configuration so all hash rules will use the same algorithm.

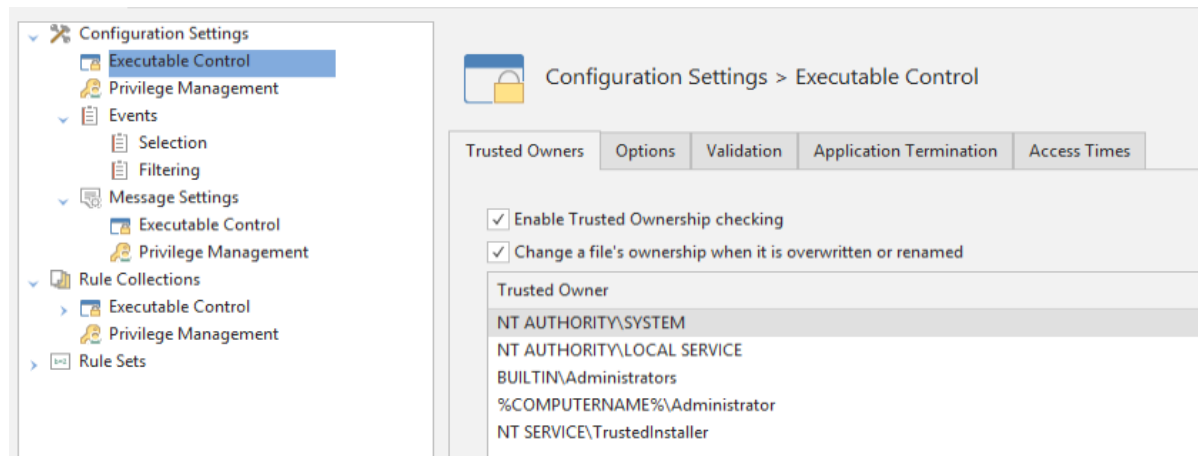
## Advanced Settings



Advanced Settings allows you to configure additional settings which may be required to resolve compatibility issues with other security products or to enable some custom settings. These settings are documented in the online help and should only be required if you encounter product compatibility or certain functionality issues during the commissioning phase. In particular, this is where exclusions for AV products are added, if required. If you think you are encountering any such issues, contact Ivanti support for assistance regarding how to configure these advanced settings. You can view the list of available settings by right-clicking in the Advanced Settings panel.

## Executable Control Configuration Settings

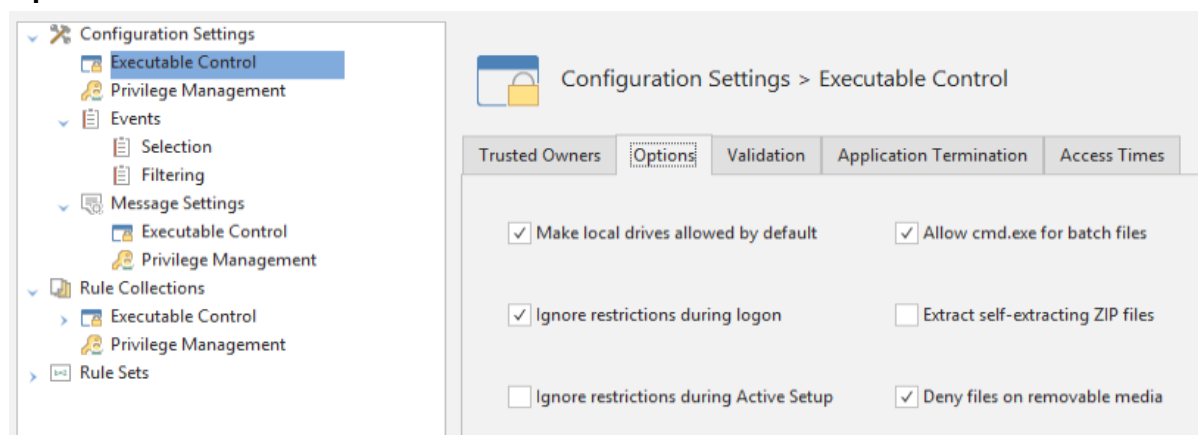
### Trusted Owners



Trusted Ownership is a security method that is enabled by default. Executable Control uses a secure filter driver as well as Microsoft NTFS security policies to intercept all execution requests. Application entitlement is determined by the NTFS ownership of the application.

Best practice is to ensure that any deployment accounts be added to the Trusted Ownership list, the gold build be analyzed to ensure all installed executables are owned by a trusted owner and that trusted ownership remain enabled on all configurations.

### Options



#### Make local drives allowed by default

The "make local drives accessible by default" feature defines whether executables residing on fixed hard drives installed in an endpoint should be accessible by default or not. Disabling the check-box will mean that administrators will need to create accessible item rules for all fixed hard drives and every file that should be allowed to run from those fixed drives for every endpoint.

While this is the most secure configuration option for Executable Control, this will add significant administrative overhead to the management of the Executable Control configuration. A better approach is to leave the setting enabled, create prohibited items rules for applications which the Administrator doesn't want users to be able to execute, and then use Trusted Ownership to block access to any files owned by users not defined in the trusted owners list.

Best practice is to enable this setting in all Executable Control Configurations.



### Ignore restrictions during logon

During the logon process several essential processes are executed. These applications are essential to providing end users with a satisfactory logon speed as well as a usable desktop environment. For this reason, the "ignore restrictions during logon" is enabled by default.

Best practice is to enable the "ignore restrictions during logon" general feature in all Executable Control Configurations.

### Ignore restrictions during Active Setup

All applications that run during the active setup phase of the logon process are subject to the rules defined within Executable Control. These applications include items like IE4UInit.exe (responsible for defining Internet Explorer settings), Rundll32.exe (used to configure the .Net Framework).

By default, the "ignore restrictions during active setup" general feature that controls this is not enabled.

Best practice is to leave this setting disabled and resolve any file ownership issues or create specific accessible item rules for any items blocked during the active setup process.

### Allow cmd.exe for batch files

Ivanti expects that most customers will create a prohibited item rule for cmd.exe within their Executable Control configuration to prevent end users from being able to launch the command prompt. If this is the case, the "allow cmd.exe for batch files" general feature will allow cmd.exe to run for any logon batch files whilst still preventing the users from being able to manually run cmd.exe. If cmd.exe is prohibited and this setting is not enabled, any batch files that the user executes will be denied. This could prevent logon scripts, logoff scripts and some application deployment tools from working as expected.

Best practice is to enable this setting and ensure cmd.exe is added as a prohibited item. Batch files (.bat) go through trusted ownership checking and will typically be owned by the administrator so no additional configuration is required. However, if the batch files are owned by a non-trusted owner they will need to be added as Allowed Items within the Executable Control configuration so that they are allowed to execute.

### Extract self-extracting ZIP files

Self-extracting ZIP files are typically executable files that contain one or more zip files along with a small application used for decompressing the zip file(s). Enabling the "extract self-extracting zip files" setting allows Executable Control to extract the file regardless of the rules in place on the file to a local hard disk drive. Once the extraction is complete, all executable files are still subject to the Executable Control rules that are in place (e.g. Trusted Ownership, etc.). If the "extract self-extracting zip files" general feature is disabled, the application is treated as a standard application and is subject to the Executable Control rules that are in place.

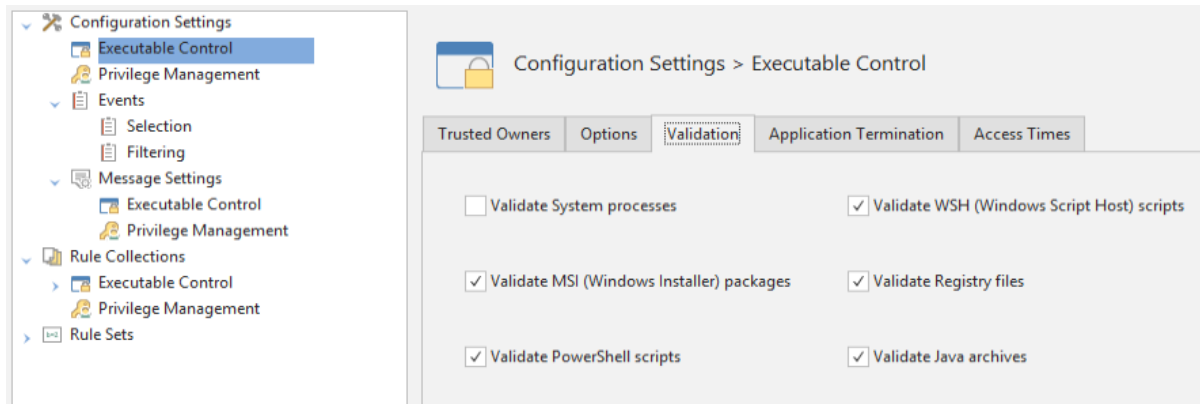
Best practice is to enable this setting for all configurations to allow scenarios where documents, etc., are shared using a self-extracting archive.

### Deny files on removable media

Removable media (e.g. USB sticks) are a potential attack vector for malware and this setting ensures that executable files on removable media are blocked by default.

Best practice is to leave this setting enabled and add any executables that are required as Allowed Items within the Executable Control configuration. Note that the only way to allow a file on removable media is via a file hash rule.

## Validation



### Validate System Processes

This setting requires that Executable Control validates all processes executed by the System as well as by the user. Effectively every single process that runs on an endpoint will be validated which will have a significant impact on the endpoint as a result of the sheer volume of validation required.

Best practice is for this setting to remain disabled on all configurations.

### Validate MSI (Windows Installer) packages

This setting defines whether Executable Control should validate Windows Installer packages that end users are attempting to install. By default, this setting is enabled which means that all MSI files are subject to rule validation. If this setting is disabled, the Windows Installer (msiexec.exe) is no longer blocked by default. In addition, msiexec.exe as well as the MSI file that this is trying to run are no longer subject to Executable Control rules processing.

Best practice is to enable this setting within all Executable Control configurations. In addition, users should ideally be prohibited from installing applications onto an endpoint as this is likely to result in trusted ownership failures. Where required, Executable Control can be configured to provide application store type capabilities whereby users can install applications as a trusted owner from a pre-defined software repository.

### Validate PowerShell scripts

This setting defines whether scripts run by powershell.exe and powershell\_ise.exe are subject to rule validation. By enabling this setting, administrators are forcing the script file as well as the command line contents of the script file to be validated.

Disabling the setting will mean that powershell.exe and powershell\_ise.exe are no longer blocked and the script (PS1 file) which these commands are attempting to execute will no longer be subject to validation.

Best practice is for this setting is to remain enabled, because scripts are regularly used to introduce malicious code into an operating system. Approved scripts will typically be owned by the administrator so no additional configuration is required as script files go through trusted ownership checking. However, if the script files are owned by a non-trusted owner they will need to be added as Allowed Items within the Executable Control configuration so that they are allowed to execute.

## Validate WSH (Windows Script Host) scripts

This setting defines whether scripts run by cscript.exe or wscript.exe are subject to rule validation. By enabling this setting, Administrators are forcing the script file as well as the command line contents of the script file to be validated.

Disabling the setting will mean that cscript.exe and wscript.exe are no longer blocked and the VBS file which these commands are attempting to execute will no longer be subject to validation.

Best practice is for this setting to remain enabled, because scripts are regularly used to introduce malicious code into an operating system. Approved scripts will typically be owned by the administrator, so no additional configuration is required as script files go through trusted ownership checking. However, if the script files are owned by a non-trusted owner they will need to be added as Allowed Items within the Executable Control configuration so that they are allowed to execute.

## Validate Registry files

This setting defines whether regedit.exe and regini.exe require validation. Disabling this setting means that regedit.exe, regini.exe and the REG file that these applications are attempting to execute are not subject to validation. Enabling this setting ensures that regedit.exe, regini.exe and the corresponding REG file are subject to the standard validation process.

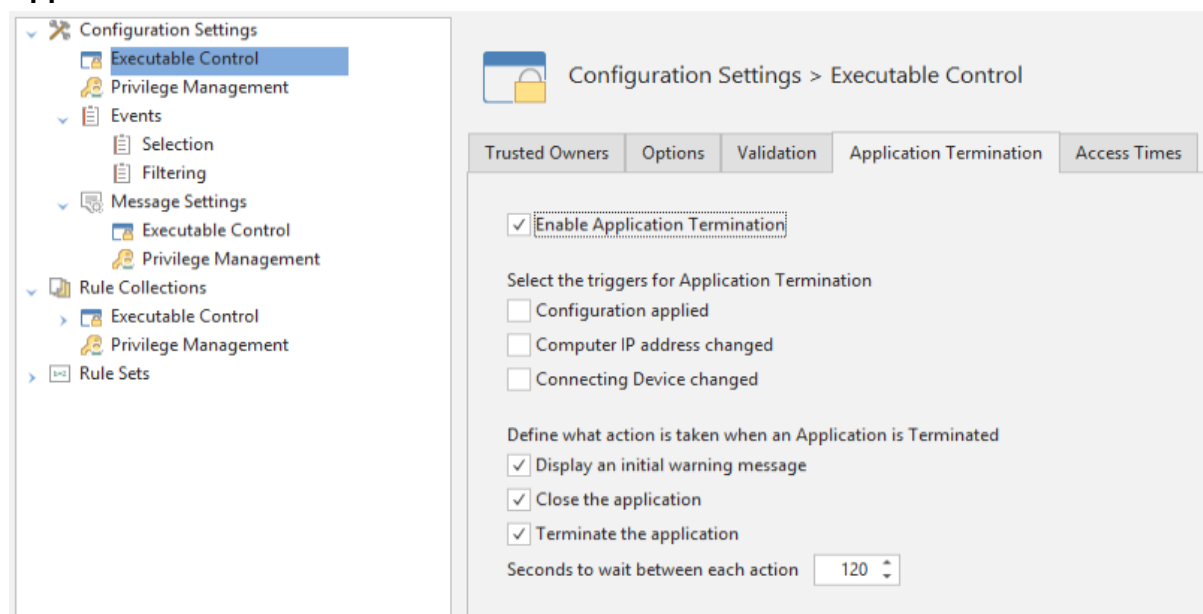
Best practice is to prohibit users from directly editing the registry and, as such, this option should remain enabled on all configurations.

## Validate Java archives

This setting defines whether Java archives (JAR files) run by java.exe and javaw.exe are subject to rule validation. By enabling this setting, administrators are forcing the Java archives to be validated.

Best practice is for this setting to remain enabled to provide additional protection against Java-based attacks.

## Application Termination



Application termination can be used to close or terminate applications within a user's session when one of the following triggers occurs:

- Configuration change
- Computer IP address change
- Connecting device change

An example use case could be a policy preventing end users from accessing financial systems whilst not in a corporate office. A user could have been working on the financial system from the office and went home without closing the application. When the user opens their laptop at home (or reconnects to a remote desktop) the IP address or connecting device will have changed, which will determine that the user is no longer in the office.

At this point one of three actions, predefined by the administrator will occur. These are:

- Display an initial warning message
- Close the application gracefully allowing users to save any work they were busy with
- Terminate the application losing any unsaved work

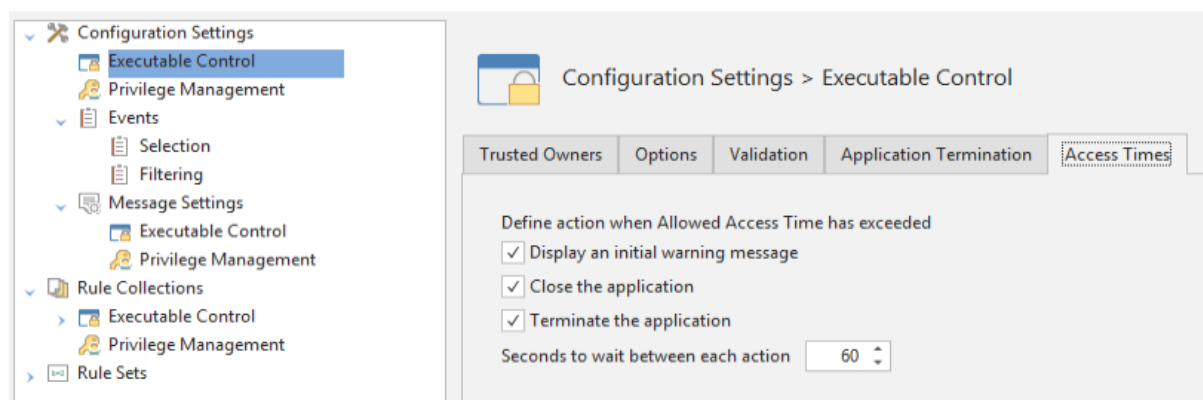
By default, the agent will work from the top down allowing 120 seconds (default value but configurable) between each action. In the event the administrator deems an action inappropriate these can be disabled within the options dialog.

In addition, a configurable message will be displayed allowing the administrator to customize the notification the end user will see when the trigger is met.

By default, Application Termination is not enabled.

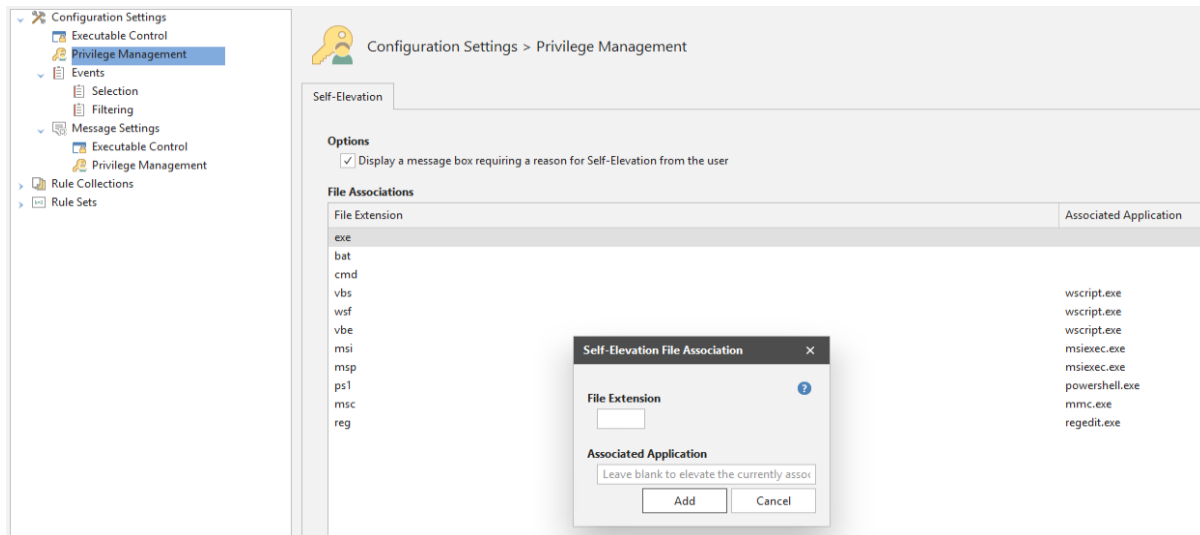
Best practice is to leave Application Termination disabled, unless a specific use case exists for this feature. Several regulated sectors have requirements where users are not permitted to work on specific systems in specific scenarios. Application termination can be used alongside other Executable Control capabilities to enforce this.

## Access Times



The Access Times feature allows you to control the specific times when an application is allowed to run. A common use case for this is to only allow access to an application during office hours. This can be configured using an Allowed File, Folder or File Hash rule. The screenshot above shows the options that can be specified when allowed access expires; for example, you can specify whether the user is allowed to save their work before closing the application, or to just close the application.

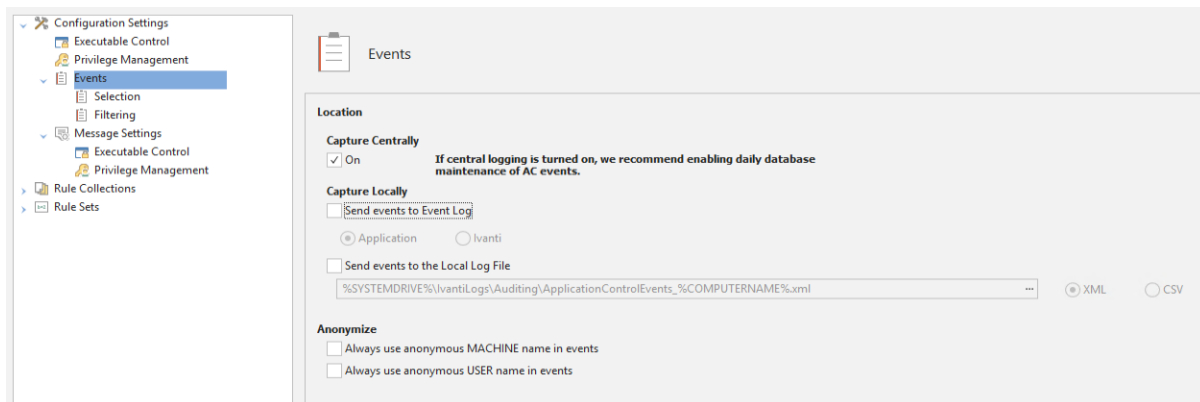
## Privilege Management Configuration Settings



### Self-Elevation

The self-elevation settings include an option to display a message requiring the user to enter a reason when they use the self-elevation feature. The best practice is to leave this enabled so that you can review the self-elevation activity via the Event Viewer and use these reasons to determine, for example, whether to create an elevation rule so that self-elevation is no longer required for this application.

### Events Configuration Settings

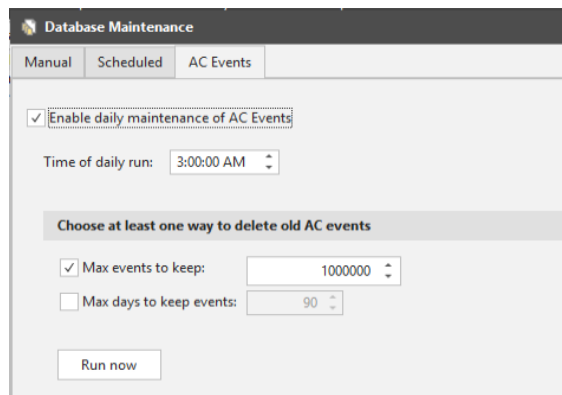


These settings allow you to define what log events get captured and where these log events are stored.

Best practice is to select the “Capture Centrally” checkbox so that log events are sent back to the database and can be viewed in the Event Viewer. There are additional settings to define how events are captured locally on the endpoint and you should select the options which best meet your needs.

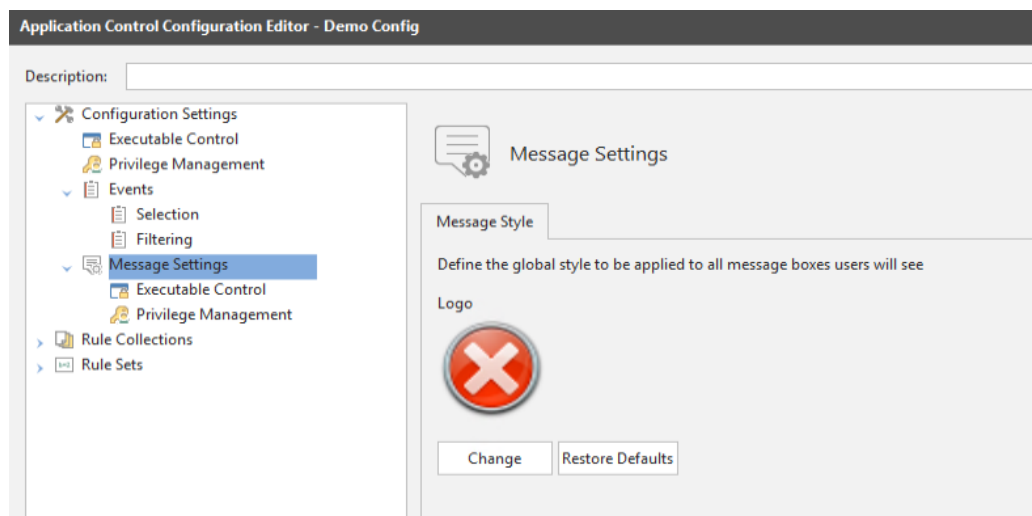
In the Selection and Filtering menus, a number of default settings have been defined for the event IDs (e.g. 9000, 9002) to be raised, and for the file types (e.g. \*.exe, \*.dll) for which they should be raised. Best practice is to leave these settings unchanged until you become more familiar with the Event Viewer from daily usage, at which point you may decide to tweak these settings depending on your needs.

Note that when you opt to capture events centrally, events will start to be returned to the database based on your event selection and filtering settings. Over time the database will start to fill up. In order to prevent this from occurring, you should perform daily Database Maintenance for Application Control Events (Manage > Database Maintenance).



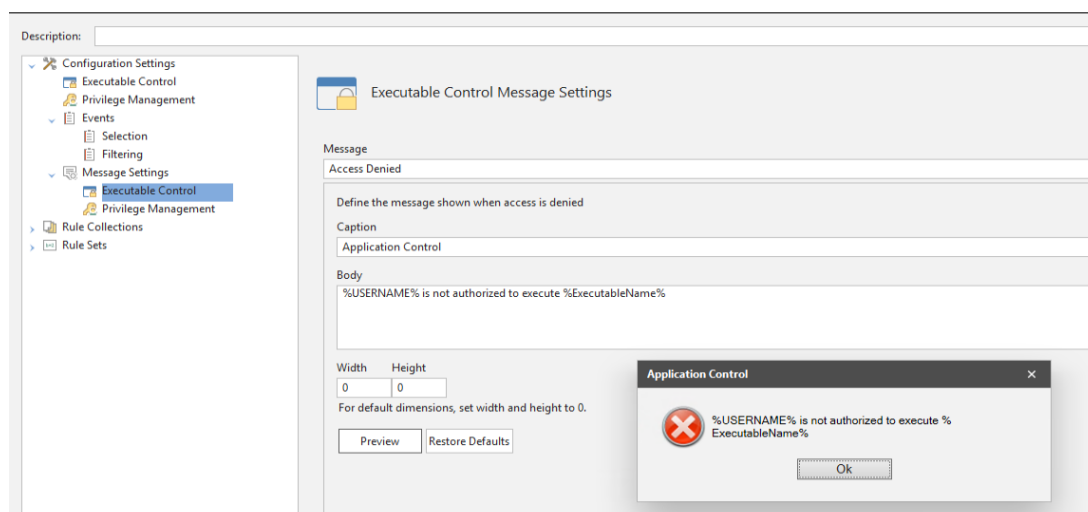
The Database Maintenance window has three tabs: Manual, Scheduled, and AC Events. The AC Events tab is selected. It contains a checkbox for 'Enable daily maintenance of AC Events' which is checked. Below it is a 'Time of daily run' field set to 3:00:00 AM. A section titled 'Choose at least one way to delete old AC events' contains two options: 'Max events to keep' (checked, set to 1000000) and 'Max days to keep events' (unchecked, set to 90). A 'Run now' button is at the bottom.

## Message Settings Configuration Settings



The Application Control Configuration Editor - Demo Config window shows a tree view on the left with 'Message Settings' selected. The main pane displays 'Message Settings' with a 'Message Style' tab. It includes a 'Define the global style to be applied to all message boxes users will see' section with a 'Logo' field showing a red 'X' icon. 'Change' and 'Restore Defaults' buttons are at the bottom.

These settings enable you to define the settings used to display messages to users for Application Control actions (e.g. Access Denied). The settings provide you with a set of defaults and enable you to customize these to meet your needs, including the logo that will be used, along with the text that gets displayed to the user.



The Executable Control Message Settings window shows a tree view on the left with 'Executable Control' selected. The main pane displays 'Executable Control Message Settings' with a 'Message' section. It includes a 'Caption' field set to 'Application Control' and a 'Body' field containing '%USERNAME% is not authorized to execute %ExecutableName%'. Below these are 'Width' and 'Height' fields set to 0. A 'Preview' button is next to them. A preview window titled 'Application Control' shows a red 'X' icon and the message '%USERNAME% is not authorized to execute %ExecutableName%' with an 'Ok' button.

# Appendix 3 – How does it work?

## How does Application Control Work?

The Application Control module requires the Ivanti Security Controls agent to be installed on the endpoint along with an associated configuration. When Application Control is enabled, the associated components are added to the agent framework.

The agent installs a Windows Service (the Application Control Service), a filter driver, and a hook. The hook sits above the driver and intercepts all executables. It does not intercept DLLs, unlike the driver. If an executable is not intercepted by the hook it is intercepted by the driver.

The driver intercepts execution requests that are made within the operating system that pass from the I/O Manager to the drive and the device subsystems, for example NTF5.SYS or the LanMan Redirector for Microsoft Networking Services. The driver does not intercept ordinary file access such as the opening of a document or text file.

Every intercepted create process request is intercepted by the hook. When the request is intercepted by the hook, the request is passed on to the Application Control Agent Service for validation against the configuration settings, which returns an execution granted or denied response that is dealt with by the hook or driver, depending on which sent the request. If the response is granted, the request is passed on to the relevant file system driver to continue with the application loading from disk.

In the case of a denied executable or script, the agent replaces the original path with Application Control's customizable message box. This effectively blocks access to the original requested executable and instead displays a message to the user. In the event of a DLL being blocked, no message is displayed, and the default operating system message is displayed.

## How does Privilege Management Work?

Privilege Management allows you to apply the principle of least privilege. This principle requires that users are provided the minimum privileges to do their job, without giving the user full administrator privileges. The experience is seamless to the user.

In a Microsoft Windows computing environment, as part of the application launch process, when an execution request is made, the application requests a security token as part of the application launch approval process. This token details the rights and permissions given to the application and these rights can be used to interact with the operating system or other applications.

When Privilege Management is configured to manage an application, the security token that is requested is dynamically modified to have permissions elevated or restricted, thus allowing the application to be run or blocked.

## How does Browser Control Work?

The Browser Control feature is currently supported on Internet Explorer and Google Chrome browsers.

CascadeBHO.dll is an Application Control Browser Helper Object (BHO) loaded by Internet Explorer that is used as part of the URL Redirection feature. If the Browser Control feature is enabled and the configuration contains a URL redirection rule, the Cascade BHO is enabled, and therefore loaded, by Internet Explorer. If there are no URL Redirection rules, the BHO is disabled.

The BHO is loaded by only Internet Explorer. A separate extension that provides the same functionality, AppSense Cascade, is loaded by Chrome.

There is no equivalent for the Microsoft Edge browser at this time, so the Edge browser is not currently supported.