Privilege Management in Ivanti Application Control

Technical Reviewers Guide

Contents

Preface 4
The Case for Comprehensive Privilege Management 4
How Ivanti Delivers on Privilege Management 4
Ivanti offers two solutions in this area to address privilege management. 4
Common Scenarios and Capabilities 5
Elevating privileges for applications 5
Elevating privileges for control panel applets 5
Reducing user-rights privileges for applications and control panel applets 5
Privilege Management and the Internet 5
Website-level web installation rights 5
Application-level web installations 5
Pre-configured web installation rights application templates 6
Trusted Ownership 6
Understanding Your Existing Local Administrator Landscape 6
Introducing Rights Discovery Mode 6
Self-elevation 7
Simple user experience 7
Granular Control of Self-elevation 7
Who can self-elevate and when 7
What a user can self-elevate 7
Auditing & Reporting of Self-elevation 8
Self-justification 8
Centralized reporting and events log 8
Ivanti Application Control and Internal App Stores 8
How and why 8
Granular control 8
Secure Dialog Boxes 8
Secure dialog-box control 8
Thinking inside the box 9
Child Process Control 9
Conclusion 9

Appendix 9
How it works: Ivanti Application Control and Local Security Authority (LSA)tokens 9
Functional Matrix 10
Differences between Ivanti Application Control and Ivanti Application Control – User Rights Management Edition 12

Preface

This paper provides an in-depth discussion of Ivanti® Application Control and how it works to help customers achieve a least privilege security stance in their organization.

For an overview and background on what this product does and the business value behind the concept of effective privilege management, please refer to the white paper titled “Ivanti Privilege Management.”

The Case for Comprehensive Privilege Management

Providing users with administrative privilege on their desktop provides them with access to areas of the desktop that, if misused, can result in high support costs and a compromised user experience. In many cases, security is compromised through the loss of data or attack from malicious software.

However, there are many scenarios where users do require local admin rights in order to be able to work effectively. Many applications, including newly released applications, allow changes to be made to hardware settings or network adapters, and all require administrative privilege in order to execute. This also includes web application updates, the installation of Active-X components, Adobe/Flash/Java updates, or installation of printer drivers.

For any of these common tasks, a user requires local administrative privilege on their desktop. Windows 7 attempted to solve this issue with its implementation of User Account Control (UAC), though it does not solve the complete granular needs of enterprise management.

With Ivanti Privilege Management, the privilege level of a user, group, or role can be elevated or reduced for application and control panel applets. A user with standard user access only to their PC can be given specific elevated rights to add a printer driver for their home printer (a task that requires administrator access to the machine). Alternatively, a user with administrative access to their Windows desktop can have their rights reduced for antivirus settings.

By controlling user privileges throughout the user session, IT can now provide users with the accessibility they require to perform their job, while protecting the desktop and the environment and reducing desktop-management costs.

How Ivanti Delivers on Privilege Management

The perfect balance between user productivity, security, and lower desktop TCO is to control user privilege—not at a session or account level, but instead, at an application or individual task level.

With Ivanti Application Control, access to applications and tasks is managed dynamically by managing user rights on-demand in response to user actions. The alternative is to revert back to the provision of a local admin account that ultimately increases the overall cost of support as the laptop falls vulnerable to misuse, malware, and user downtime.

Elevated privilege can be applied to a named application or control panel applet for a particular user or user group. Or, the rights of an administrative user can be reduced to that of a standard user account. By controlling user privileges throughout the user session, IT can now provide users with the accessibility they require to perform their job, while protecting the desktop and the environment and reducing desktop management costs.

Ivanti offers two solutions in this area to address privilege management.

The full-functioned Ivanti Application Control delivers a world-class industry solution to:

- Control user rights and privileges granularly
- Manage application access
- Implement whitelisting and blacklisting
- Enforce device-based licensing
- Block untrusted software and code
- Control network access

A reduced functionality solution, Ivanti Application Control - User Rights Management edition, implements the ability to purely control user rights and privileges.

For more information and a detailed matrix on specific functionality, please refer to the appendix.

Common Scenarios and Capabilities

Elevating privileges for applications

Scenario: Specify applications that need to be run with elevated rights within a standard, non-administrative user desktop session. This capability assists in remediating application compatibility scenarios relating to least privilege.

Benefit: Here, the user account or desktop session does not have admin rights; however, the user is able to run specific applications under an elevated context (with no explicit consent prompts required) when the application needs administrative rights to run. This is especially relevant if it is a legacy application written assuming that all users had admin rights (and a common scenario for Windows XP deployments).

Elevating privileges for control panel applets

Scenario: Many roaming users require the ability to install printers and change network and firewall settings; all require Control Panel applets to be accessed with admin rights.

Benefit: Privilege management can elevate privileges for individual applets, meaning a standard user can still make the required changes to their desktop in order to do their job without having administrative access to the whole desktop.

Reducing user-rights privileges for applications and control panel applets

Scenario: Specify applications or control panel applets to run with reduced privileges. The user has admin rights by default, but specified applications such as Task Manager, Firewall settings, or RegEdit are forced to run with reduced privileges as a non-administrative user.

Benefit: Useful where a business is forced to implement admin rights by default, such as in a legacy Windows XP deployment, but where the business wants to ensure that certain system and applications cannot be changed.

Privilege Management and the Internet

Ivanti also extends the privilege management capabilities through the control of web-based software installations. IT teams can now enable non-administrative users to install and update pre-approved, web-delivered software in a controlled manner without the need to grant full administrative user rights to the desktop. This increases user productivity while minimizing IT support costs.

Website-level web installation rights

Allowing end users to initiate web- based software installations is a potentially risky proposition, even when it can be accomplished without granting full administrative rights. For every legitimate business application source, there are countless other websites hosting malware or other software that are not approved for corporate use such as non-productivity or unlicensed software.

With Ivanti Application Control, IT administrators can “whitelist” approved websites from which users are authorized automatically to install approved software.

As an example, the IT team can preauthorize web-based software installations only from known sites such as **www.adobe.com** and **www.gotomeeting.com**. This provides end users with immediate access to common business applications such as Adobe Reader, Adobe Air, Adobe Flash Player, and the GoToMeeting or WebEx web conferencing clients without IT application-delivery bottlenecks and inefficiencies.

In addition to enabling the user population to access required personal-productive applications that aren’t delivered by IT, security and integrity is maintained at the same time as users are blocked from performing web-based software installation from all other sources. This reduces exposure to security vulnerabilities or system instability and associated costs through unauthorized software installation.

Application-level web installations

While website-level software installation policies may represent a good balance between security and simplicity for many organizations, some organizations may require more granular control over the applications that users can install from the specifically approved websites.

To revisit the earlier example, an IT administrator may wish to allow installation of Adobe Reader but block Adobe Air, Adobe Flash Player, and/or any other applications from www.adobe.com. In this scenario, Ivanti Application Control provides IT administrators with the required controls to accomplish this, providing the option to whitelist specific applications by version and ActiveX control class ID within the named website as needed. This provides the assurance that only trusted versions of specific applications may be installed from the web by end users.

This also helps with managing application version control within enterprise environments.

Pre-configured web installation rights application templates

While Ivanti Application Control offers granular user-rights policy creation and customization capabilities to address a wide range of complex scenarios, it also simplifies managing common application policy scenarios, thereby reducing implementation time.

IT teams extending web-based software installation rights to their end users can draw from a library of pre-defined templates for common business applications. Applying these templates saves time and provides the assurance that users are only gaining the minimum level of rights escalation required to be productive.

Ivanti Application Control includes the option of enabling select users to “self-elevate” the execution of an application to include elevated privileges. In addition, this ensures that any application elevated remains secure, access to Windows Explorer or any connected dialogs remains as a standard user, and that administrator rights are not inherited from within the elevated application.

Self-elevation can be configured by the system administrator so that only a predefined list of authorized applications can be elevated, or alternatively, a list of prohibited applications can be specified to prevent them from being self-elevated, e.g. CMD.exe or RegEdit.exe.

Trusted Ownership

Trusted Ownership prevents the execution of any code, even unknown, introduced by a non-trusted owner (example: a typical user account). This unique concept of “Trusted Ownership” is a key element of the Ivanti application-entitlement policy management approach. Trusted Ownership provides an additional degree of security and control by only allowing those applications that are associated or “owned” by a pre-defined and trusted user account to execute. An example of a “trusted owner” is the Local System administrator or Domain administrator account and typically an account that manages the desktop.

The Ivanti Application Control capabilities are fully compatible with the existing Ivanti Trusted Ownership policy framework. IT administrators may specify that approved web-based installations such as Adobe Reader and GoToMeeting will be configured automatically so that file ownership is assigned to an existing trusted owner account rather than the initiating user’s account. This allows approved, web- installed applications to execute even in environments with tightly managed Trusted Ownership policies.

Understanding Your Existing Local Administrator Landscape

Removing unnecessary local admin rights is a major objective for many IT departments as it lowers the overall cost of maintaining a desktop environment while also ensuring users have access to the applications and OS features they require to remain as productive as possible.

Unfortunately, not understanding your existing local admin landscape first can make implementing a privilege management solution a somewhat lengthy process.

Fortunately, however, by using user Rights Discovery Mode (RDM), enterprises can quickly understand which applications and OS features require elevated permissions across the entire desktop estate, to simplify and quicken rollout.

Introducing Rights Discovery Mode

Rights Discovery Mode (RDM) enables system administrators to monitor, analyze, and report concurrently on tens of thousands of endpoints to identify what applications and tasks require administrative privileges. RDM generates reports that can be grouped by user, computer, or application, making it easy to identify trends. Applications and tasks identified by RDM can be quickly added to AM-URM configurations with just a few clicks.

Default/STANDARD rights policies enable customers to simply identify the target application that requires an elevation policy and assign the application automatically to an elevation rule to further simplify the configuration and quicken the implementation of AM or AM-URM.

Figure 1 below shows a single computer screenshot of an RDM report, grouped by application, with the administrator adding an application very quickly and easily that has been identified as requiring admin rights to a Default Rights Elevation Policy for all users.

This can also be done by file name, or, for extra security, by using an SHA1 hash for only that specific application.

Note: Referring to the ‘List View’ (top right of the screenshot), it can be seen how many times each application was run, and from there you can also see on which machines it was run. This application-details view provides you with the opportunity to see the user, machine, command line, and start time for each instance of that application.

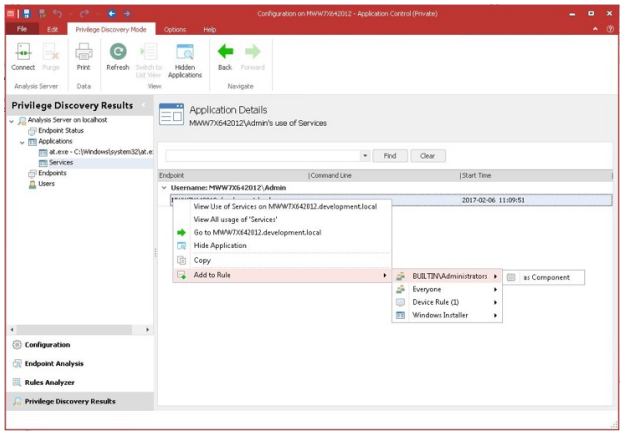

Figure 1 – User-Rights Discovery Mode screenshot

Self-elevation

Power users often exist in enterprise environments. Ivanti Application Control provides the system administrator with the option to enable selected (power) users to self-elevate their rights from their default standard-user account to that of a local administrator. This enables these important users to remain productive and reduces calls to the service desk and associated costs in relation to end-user requests.

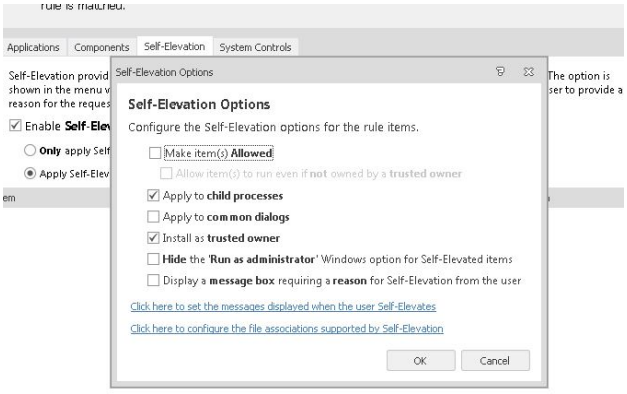

Figure 2 – Self-elevation of user rights option

Simple user experience

The self-elevation process is extremely simple for the user and integrates seamlessly with the Windows desktop; the user simply right-clicks on the application in question and selects “Run with Administrative Rights (Audited)” from the context menu.

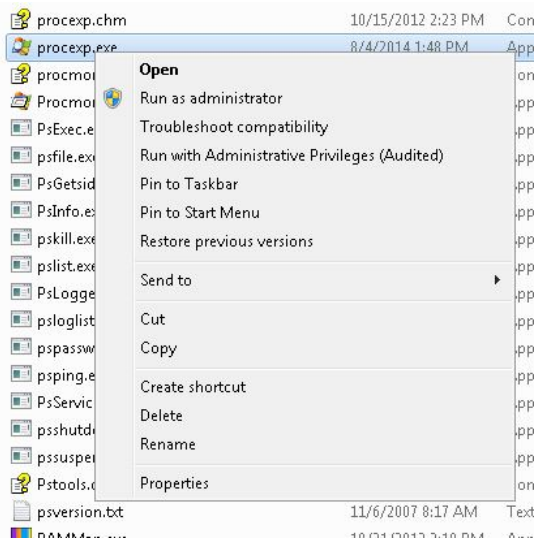

Figure 3 – Option to remove the Windows ‘Run as administrator’ option

To simplify the user experience and also ensure that any self-elevation actions are captured and audited, the standard in-built Windows ‘Run as administrator’ option can also be removed from the right-click context menu leaving only the Ivanti self-elevation option via a simple tick box in the Ivanti console.

Granular Control of Self-elevation

To provide the system administrator with granular control, this extremely configurable feature is able to modify the configuration dynamically based on the user, device, location, and time and date context.

Who can self-elevate and when

Self-elevation can be granted based on many of the existing Ivanti Application Control rules, including: user name, user group, device name, and IP address. This provides a dynamic configuration that changes automatically in relation to the context of the user, device, location, and time and date to ensure self-elevation only occurs when permitted and specified by the system administrator.

What a user can self-elevate

When providing the user with the option to self-elevate their rights, you may wish to restrict this ability to certain applications or actions. To achieve this, Ivanti Application Control provides the system administrator with two options that can be used together to provide enhanced granularity and control:

- Whitelist - Specify the applications a user is authorized to self-elevate.

- Blacklist - Specify the applications a user is prohibited to self-elevate.

For example, the system administrator may wish to create a configuration that whitelists Notepad.exe as an authorized application and blacklists RegEdit.exe as a prohibited application.

Auditing & Reporting of Self-elevation

A common problem for the IT support team is clarifying "who" and "why" users require administrative rights. The reporting options within Ivanti Application Control can help with the rights-discovery process.

Self-justification

When a user chooses to self-elevate their privileges to "administrator" and is authorized to do so, an option exists to ask the user to provide a reason why they are self-elevating their rights. A message alert is presented to the user with a simple text box to enter their reason.

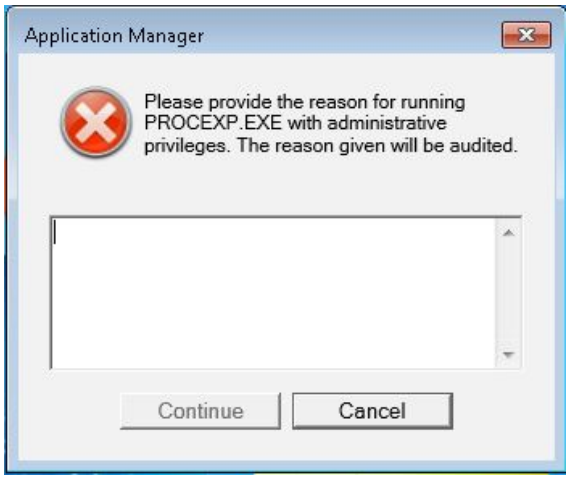

Figure 4 – Self-elevation justification prompt for auditing and tracking

Centralized reporting and events log

The centralized reporting and events log also captures the user name, application name, time, and date of the self-elevation request and the machine it is made from. It also contains any self-justification messages the user has provided.

Ivanti Application Control and Internal App Stores

Ivanti Application Control enables organizations to develop their own internal, private application stores and enable selected users to introduce applications as they require them, without burdening IT and waiting for the request to be fulfilled.

How and why

As part of the "when" and "why" a process or application is elevated, Ivanti Application Control provides the option for the system administrator to specify network drives or known good locations that can be set up so that when the user attempts to install an application from said location, Ivanti Application Control elevates the rights for the installation to enable a secure desktop, but allowing known good applications to be delivered via user self-service.

This not only improves the end-user experience since they are now able to self-service and fulfill their own IT service delivery requests, but it also reduces IT service costs and allows IT to focus on delivering the core business and corporate applications.

Granular control

Departmental app stores can be set up by taking advantage of the user and user-group rules so that, depending on user name or group membership, they are either able or unable to install the applications in relation to which app store they are accessing.

Device-based rules also ensure that the predefined applications housed in the authorized known good locations can only be installed onto permitted (e.g. corporate) devices.

Secure Dialog Boxes

When an application has been launched with elevated rights, it is imperative that this application cannot be used as a gateway to access the underlying desktop, operating system, other applications, or files under the context of the administrator.

Secure dialog-box control

The Ivanti Application Control secure dialog-box control capability ensures that only the application is elevated to "administrator" and any dialog box such as File Open or File Save, which access the complete file structure of the computer, remains under the context of the standard user, preventing them from accessing or modifying restricted files and folders.

In the example in Figure 5 below, Microsoft Word is elevated to "administrator," although access through a secure dialog box must not also be run under the context of the administrator. This prevents the user navigating to areas of the operating system or spawning new processes under the context of an elevated local administrator.

Thinking inside the box

The secure dialog box functionality is on by default and protects the system and data, stopping the user from altering files maliciously or accidentally. This includes:

- Renaming or deleting files not owned by the current user
- Dragging and dropping files to and from locked-down directories
- Copying and pasting to and from locked-down directories
- Changing the security permissions on a file

Child Process Control

Much like the secure dialog boxes, it is imperative that any new process spawned from an elevated process or application not inherit the elevated privileges, regardless of whether the spawned process is launched manually by the user or not. If this were to happen, it would be a major security risk as it leaves the application and desktop vulnerable to misuse while the app is elevated.

Example problem: An application is raised to have administrative privilege and the user navigates to the File > Open dialog. The user browses the file system to cmd.exe and launches the application. This application is now launched as a child process of the elevated Word process.

How Ivanti Application Control solves it: Without child process control, this application would now also run as a local administrator where the user could cause severe operating system damage.

Ivanti Application Control avoids this security risk by providing the option for the system administrator to prevent any child processes from adopting the elevated rights from the parent process.

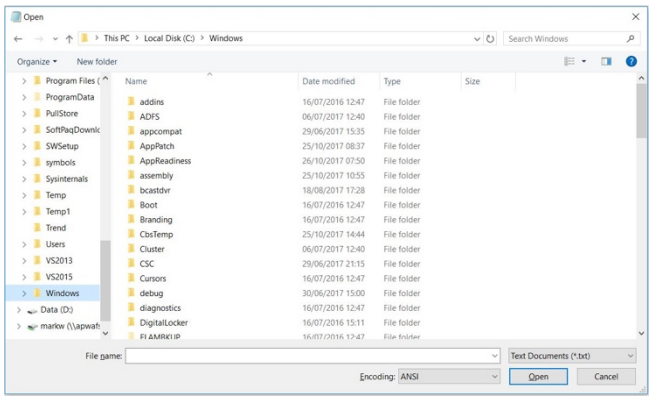

Figure 5 – A dialog box showing access to the file system from within an application

Conclusion

Improper control of privilege management is a serious issue in many organizations today that creates undue business risk, and significantly adds to the support cost.

Analysis by Gartner highlights that companies can save up to $1,278* with effective control of privileges and reduction in admin rights.

**Gartner Inc.; Desktop Total Cost of Ownership; 2011 update; ID: 1470342*

Ivanti Application Control and Ivanti Application Control User Rights Management Edition deliver IT with more flexibility and the ability to solve access and privilege issues on a granular basis, rather than using a sledge hammer approach.

This greater control ultimately reduces desktop management costs, security risks, and service requests, and improves the end-user experience and employee productivity.

For further information, please visit https://www.ivanti.com/solutions/needs/grant-the-correct-privileges-to-my-employees

Appendix

How it works: Ivanti Application Control and Local Security Authority (LSA) tokens

In a Microsoft Windows computing environment when an application-execution request is made, the application requests a security ticket as part of the application-launch approval process. This ticket 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 Ivanti Application Control is configured to manage an application, the security ticket that is requested will be modified dynamically to have permissions elevated or lowered, and therefore the application can now either run (or be blocked) with elevated or reduced privileges.

Some control-panel applets, including network adapter functions, are typically controlled by the explorer shell process. Elevating explorer.exe to run in the context of a local administrator is definitely not recommended as this opens the entire operating system to attack—compromising the entire purpose of least privilege. To resolve this effectively and enable the user to access the applet functionality under the context of an administrator, Ivanti Application Control provides the user with an Ivanti-spawned window containing the control panel applet,

which can now be controlled at an access level specific to the function, without changing any of the rights associated with the explorer shell.

Ivanti Privilege management integrates seamlessly with existing security principals, and does not create any new local administrator accounts, which are then referenced or used as part of applying an elevated right to a defined application.

Ivanti Application Control intercepts execution calls using detoured hooks, which are called before the real application (in question) is started. Ivanti Application Control then examines the parameters of the application request with a series of rules checks to determine if they match any of the heuristic rules and policies within the Ivanti Application Control configuration. If true, Ivanti Application Control then requests a unique access token for only the specific process from the Local Security Authority (LSA).

Note: Unlike other solutions, Ivanti Application Control ensures system security, as it does not modify the primary user token.

Now, Ivanti Application Control, with its custom authentication package, impersonates the newly created unique token with administrative rights from the administrators group member Security Identifier (SID). Ivanti Application Control then holds the primary token and injects the new token, for the specific process only. This also enables Ivanti to control secure dialog boxes and child process securely.

Functional Matrix

Below is a functionality matrix detailing the capabilities of Ivanti Application Control as covered in this document:

Rights discovery	
User-rights discovery mode	Ability to identify existing applications that make administrative calls to the operating system
Grouping of user rights discovery results	By application, machine, or user
Ability to manage applications identified in discovery reports	Add applications to configuration quickly to speed up implementation
Contextual control of when elevation occurs	
Contextual control of what is elevated	Define individual or groups of applications, control panel applets, and system tasks
Contextual control of when elevation occurs	Configurable rules based on context of user, device location, and time/date
Contextual control of the locations where elevation occurs	Dynamic rules based on IP address of device or location of application
Contextual control of devices on which elevation occurs	Permit elevation to only occur on predefined devices
What can be elevated	
Elevate rights for applications	Create whitelists and blacklists for individual or groups of applications
Elevate rights for control-panel applets	Create whitelists and blacklists for individual or groups of control panels applets
Elevate rights for system tasks	Create whitelists and blacklists for individual or groups of system tasks

Contextual control of when reduction occurs	
Contextual control of what is reduced	Define individual or groups of applications, control-panel applets, and system tasks
Contextual control of when reduction occurs	Configurable rules based on context of user, device location, and time/date
Contextual control of locations where reduction occurs	Dynamic rules based on IP address of device or location of application
Contextual control of devices on which reduction occurs	Enforce reduction to always occur on predefined devices
What can be reduced	
Reduce rights for applications	Create whitelists and blacklists for individual or groups of applications
Reduce rights for control panel applets	Create whitelists and blacklists for individual or groups of control panels applets
Reduce rights for system settings	Create whitelists and blacklists for individual or groups of system tasks

Internet controls and web installations	
Enable website-level application installations	Specify trusted websites where users can install from
Enable application-specific installations from defined websites	Within trusted website, specify only the authorized applications users can install
Assign Trusted Ownership to application installation	To enable seamless integration with AppSense Trusted Ownership application access
Provide pre-configured website templates	Simplify and quicken implementation with out-of-the-box templates
Self-elevation	
User self-elevation	Enable authorized users to self-elevate without IT service request
User justification for self-elevation	User is prompted to provide audited reason as to why self-elevation
Auditing of self-elevation	Report on who is elevating, what they are elevating, and on what device
Contextual control of who can self-elevate	Based on user name, group, or pre-live directory
Contextual control of what can be self-elevated	Whitelist or blacklist what can be self-elevated
Contextual control of when self-elevation can occur	Based on time and day or other contextual rules
Contextual control of where self-elevation can occur	Based on device location or where the application resides
Contextual control of which devices self-elevation can occur from	Permit any nominated devices to allow self-elevation
Internal app stores	
Create known good locations to install from	Only allow elevation for application installations initiated from predefined locations
Additional security controls	
Ability to secure dialog boxes	Prevent elevated application accessing underlying desktop
Ability to control child processes	Ensure new child processes inherit elevated rights

Differences between Ivanti Application Control and Ivanti Application Control – User Rights Management Edition

Capabilities	Ivanti Application Control URM Edition	Ivanti Application Control
User-rights discovery mode	✔	✔
Contextual control of rights management and invocation	✔	✔
Granular metadata-based configuration option	✔	✔
Web- and application-level installation controls	✔	✔
User self-service and self-elevation options	✔	✔
Control of dialog boxes and child processes	✔	✔
Digital signature-validation controls	✔	✔
Auditing and reporting capabilities	✔	✔
Application whitelisting and blacklisting for user rights	✔	✔
Application whitelisting and blacklisting for app access		✔
Device license enforcement		✔
Dynamic application access entitlement		✔
Block untrusted code execution (Trusted Ownership)		✔
Context-aware application termination		✔
Archiving of unauthorized or self-approved applications		✔
Granular URL filtering and re-direction options		✔
Network access control		✔
Application time limits		✔
Number of application instance controls		✔
Script, Batch File, and Registry controls		✔
Passive reporting for user-introduced executable code		✔
Endpoint analysis and application usage scans		✔
Pre-built Ivanti configuration templates		✔