

Ivanti Device and Application Control 5.2

Administrator Guide



Endpoint Security

powered by HEAT

Notices

Version Information

Ivanti Device and Application Control Administrator Guide - Ivanti Device and Application Control Version 5.2 - Published: July 2020
Document Number: 02_108_5.2

Copyright Information

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

Copyright® 2020, Ivanti. All rights reserved.

Ivanti and its logos are registered trademarks or trademarks of Ivanti, Inc. and its affiliates in the United States and/or other countries. Other brands and names may be claimed as the property of others.

Protected by patents, see www.ivanti.com/patents.

Table of Contents

Preface: About This Document.....	7
Typographical Conventions.....	7
Chapter 1: Ivanti Device and Application Control System Architecture.....	9
About Application Control and Device Control Architecture.....	9
The Database.....	11
The Application Server.....	11
The Management Console.....	12
The Client.....	13
Chapter 2: Understanding Cryptography.....	17
About Encryption.....	17
Advanced Encryption Standard.....	18
RSA Encryption.....	19
Chapter 3: Device Control Encryption Methodology.....	21
Encrypting Removable Storage Devices.....	21
Easy Exchange Encryption.....	22
Encrypting Media.....	22
Centralized Encryption.....	23
Decentralized Encryption.....	24
Chapter 4: Application Control Methodology.....	25
About File, Macro, and Script Execution Control.....	26
Identifying DLL Dependencies.....	26
Maintaining Application Control.....	27
Operating System Updates and Patches.....	27
Frequently Changing Software Use.....	27
Software Updates.....	28
New Software Installations.....	28
Macros and Other Changing Files.....	28
Deleting Local Authorization Files.....	29
Chapter 5: Understanding Application Server-Client Communications.....	31
About Application Server-Client Communications.....	31
Application Server Communications.....	32
Changing Licenses for the Application Server.....	33
Changing the SysLog Server.....	33
About Application Server-Client Communication Encryption.....	34
Digital Signatures and Certificate Authorities (CA).....	34
Digital Signatures.....	34
About Application Server-Client Proxy Communications.....	35
Configuring a Static Proxy on Clients.....	35
Configuring WPAD for Application Server-Client Proxy Communication.....	36

Chapter 6: Deploying the Client.....41
 About Ivanti Device and Application Control Client Deployment..... 41
 Deploying the Client with Windows Group Policy.....42
 Deploying the Client with Other Tools.....43

Chapter 7: Controlling Administrative Rights.....45
 About Ivanti Device and Application Control Access Control Rights.....45
 Defining User Access.....46
 Using the Access Control Visual Basic Script.....47

Chapter 8: Using File Tools.....49
 About File Tools.....49
 Using the **Ivanti Device and Application Control** Authorization Service Tool.....49
 Using the Application Control File Tool.....52
 Using the SXDomain Tool.....54

Chapter 9: Managing Registry Keys.....57
 Database Connection Loss Registry Key.....58
 Authorization Service Registry Key.....58
 Debugging Registry Key.....59
 General Registry Keys.....60
 Security Registry Keys.....61
 Configure MaxSockets.....63
 Configuring MaxSockets and TLSMaxSockets.....64
 Command & Control Registry Key.....64
 Client Kernel Registry Key.....66
 Software Registry Key.....67
 Application Server Registry Key.....67
 Authorization Wizard Registry Key.....68
 Resolving Driver Conflicts.....68
 Preventing a Security Audit.....69



Preface

About This Document

This Administrator Guide is a resource written for all users of Ivanti Device and Application Control 5.2. This document defines the concepts and procedures for installing, configuring, implementing, and using Ivanti Device and Application Control 5.2.

Tip: Ivanti documentation is updated on a regular basis. To acquire the latest version of this or any other published document, please refer to [Ivanti Product Documentation \(https://help.ivanti.com\)](https://help.ivanti.com).

Typographical Conventions

The following conventions are used throughout this documentation to help you identify various information types.

Table 1: Typographical Conventions

Convention	Usage
bold	Buttons, menu items, window and screen objects.
<i>bold italics</i>	Wizard names, window names, and page names.
<i>italics</i>	New terms, options, and variables.
MONOSPACE UPPERCASE	Keyboard keys.
BOLD UPPERCASE	SQL Commands.
monospace	File names, path names, programs, executables, command syntax, and property names.

Chapter 1

Ivanti Device and Application Control System Architecture

In this chapter:

- About Application Control and Device Control Architecture
- The Database
- The Application Server
- The Management Console
- The Client

The Ivanti Device and Application Control software application is based on a multi-tier software architecture that processes and stores data for Application Control and Device Control. Users can interact with the application through the client interface. A separate Management Console provides a user interface for network administrators.

Application Control and Device Control architecture is a 3-tier structure composed of an Application Server, database, and one or more clients. A Management Console provides the administrative interface between the system administrator and Application Server.

About Application Control and Device Control Architecture

Ivanti Device and Application Control architecture is a multi-tier system composed of a database, an Application Server, and a client.

The three primary components that form the basis of Ivanti Device and Application Control system architecture interact as follows. The Application Server component runs as a service that:

- Keeps track of the connected clients and their status.
- Coordinates data flow between the Application Server(s) and the SQL database.

The database component serves as the central repository of the authorization and permission policies. The client component stores file authorization and device permission policies locally, and controls user access from the endpoint to software applications and all connected devices. To change authorization and permission policies, an administrator uses a Management Console to interface with the Application Server(s), which then communicate with the database and the client(s).

Ivanti Device and Application Control system architecture is scalable to meet the needs of any enterprise. You can use multiple Application Server(s) and clients.

The Application Control and Device Control architecture components are:

- The database which serves as the central repository of authorization information for devices and applications.
- One or more Application Servers that communicate between the database, the protected clients, and the Management Console.
- The client (installed on each computer, endpoint or server that you want to protect).
- The Management Console, which provides the administrative user interface for the Application Server.

The following figure illustrates the relationships between the components.

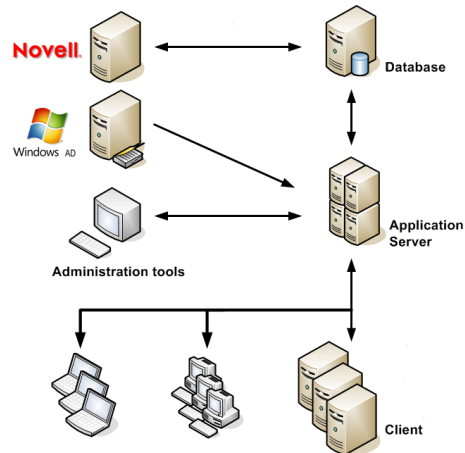


Figure 1: Endpoint Security Architecture

The Database

The database serves as the central repository of software and device permission data. Administrator and user access logs are also stored in the database.

The database uses one of the following Microsoft® SQL® Server products:

- Microsoft SQL Server 2012, Standard, Enterprise, Express Edition (32-bit and 64-bit)
- Microsoft SQL Server 2014, Standard, Enterprise, Express Edition (32-bit and 64-bit)
- Microsoft SQL Server 2016, Standard, Enterprise, Express Edition (64-bit only)
- Microsoft SQL Server 2017, Standard, Enterprise, Express Edition (64-bit only)
- Microsoft SQL Server 2019, Standard, Enterprise, Express Edition (64-bit only)

For evaluations and test environments Microsoft SQL Server 2008 Express (or greater) is sufficient. Enterprise implementation must use the Microsoft SQL Server 2008 (or greater) edition.

The Application Server

The Application Server runs as a Windows® service that coordinates and tracks data flow between Application Server(s), connected clients, and the SQL® database.

The Application Server service runs under any domain account capable of reading domain users, user groups, and computer accounts from the domain controller. The Application Server performs the following functions:

- Retrieves user access and device permission policies from the database which are stored in the Application Server cache.
- Signs and/or encrypts the user access and/or device permission list, compresses the list, and communicates updated user access and/or device permission lists to client servers and computers, where the permission policies are stored locally. Permission policy updates only communicate changes to the existing user access and device permission policies, rather than retransmitting entire policies.
- Saves a log of administrator actions and, optionally, users actions including information about when application or device access is denied.

Each Ivanti Device and Application Control product installation requires at least one Application Server and a corresponding `DataFileDirectory` (DFD). The DFD can reside on the same computer or a shared network resource, to store log information. All servers can write to a shared `DataFileDirectory` or to a different directory for each Application Server, depending upon the unique architecture of your network environment.

Up to three Application Servers can be defined during client installation. Additional servers can be assigned by:

- Changing the **Server Address** default option in the Management Console **Tools** menu, as outlined in the [Ivanti Device Control User Guide \(https://help.ivanti.com\)](https://help.ivanti.com) or [Ivanti Application Control User Guide \(https://help.ivanti.com\)](https://help.ivanti.com) .
- Modifying the `Server` parameter for the [Command & Control Registry Key](#) on page 64

The Application Server sends user access and device permission changes to users when:

- A user logs in.
- An administrator sends updated information to all computers, specific computers, or export changes to a file.
- A user requests updated information from a client computer.

An administrator uses the Management Console to interact with Application Server.

The Management Console

The Management Console is the administrative interface for the Application Server. This product component is used to configure Application Control and Device Control.

You can install the console on one or more computers, including the database or Application Server hosts.

The Management Console does not connect directly to the database. All communication with the database is conducted via the Application Server(s). The Management Console and Application Server use Remote Procedure Call (RPC) protocol authenticate encrypted communications.

You use the console to:

- Define administrator roles.
- Monitor system administrator and user activity logs and options.
- Define default settings for administrative tools.
- Generate standard or custom reports.

When you are using Device Control, you can:

- Manage access to removable storage devices.
- Authorize user access to specific CD/DVD media for use with CD/DVD drives.
- Encrypt removable media and devices.
- View lists of files and data transferred using authorized media.
- View the content of files transferred using authorized removable storage devices.
- View information about user attempts to access or connect unauthorized removable storage devices.

When you are using Application Control, you can:

- Build and manage centrally authorized lists of executable files, scripts, and macros.
- Organize authorized application software files into logical file groups.
- Assign file groups to users and user groups.
- Manage and maintain the software authorization database.

The Client

The client is installed on each client computer to enforce file authorization and device permissions policies.

You can install the client on the same computer as the Management Console when you are using Device Control. Each protected computer can maintain local authorization and device permission files, so that routine application requests do not have to traverse the network. Only log files and periodic differential updates are sent from client computers to the using the Management Console.

When a client requests a connection to the Application Server listed by DNS name, the name is resolved and the first IP address returned is selected, as required by round-robin DNS conventions for a server-client connection attempt. This behavior is controlled by the `FirstServer` registry key.

If the connection attempt fails, the client selects the next server name from the list and repeats the process. After reaching the end of the list and no server-client connection is established, the client uses the locally cached user access and device permission list. The first connected server receives client logs and shadow files in a compressed format that is stored in the `DataFileDirectory` (DFD) defined during installation.

When you are using Device Control, the client:

- Ensures that only removable storage devices and media that the user is authorized for can be accessed. Any attempt to access unauthorized devices or media is denied, regardless of the computer the user logs into.

When you are using Application Control, the client performs the following:

- Calculates the digital signature, or *hash*, for files requiring authorization.
- Checks the digital signature against the locally stored authorization list for a matching digital signature.
- Denies and logs any user attempts to run unauthorized files.
- Allows, when expressly permitted, a user to locally authorize a denied file.
- Generates log records of all application access attempts, approved and denied. The **Log Access Denied** option is enabled by default.

The client is composed of the following components:

- Kernel driver (`Sk.sys`), that runs on supported operating systems to enforce defined policies for determining which applications and/or devices users can access.
- Communication service (`scomc.exe`), which provides communication with the Application Server(s). This component, Ivanti Device and Application Control Command & Control (SCC), runs as a service, that sends log data to the Application Server that can be viewed via the Management Console.
- User interface (**RtNotify**), which informs the user of updated policy changes completed by the administrator (these messages can be deactivated). **RtNotify** is used by the client to retrieve user certificates on demand.
- Auxiliary dynamically linked library (DLL) files, which provide additional features to the core components. These files contain support for RtNotify localization information, 16-bit application control, and macro and script protection.

The following illustrates the relationships between the client component layers.

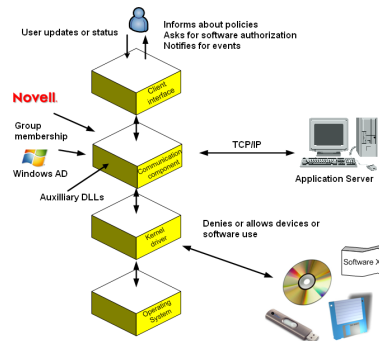


Figure 2: Client Component Layers

After the client is installed, **RTNotify** appears as an icon in the system tray. Through **RTNotify** the user receives information about permission changes via pop-up messages. End-users can interact with the client to:

- Locally authorize software application files
- Manage user access to removable storage devices
- Update permissions changes when the client receives event notifications

Important: A user cannot change any Application Control or Device Control administrative settings or permissions.

The administrator can query the client to obtain the **salt** value used for endpoint maintenance, when the computer is not connected to the network and this value cannot be obtained using the Management Console.

When an Application Server is unavailable at login, the client uses the locally cached permission list from the last successful Application Server connection. If no permission list exists, the client denies user access to all device and application requests. Permissions lists can be imported to a computer, as necessary, when no server connection is available or the computer is disconnected from the network.

A key security principle of Application Control or Device Control is that, even when the communication service or user interface is disabled, the kernel driver always protects the client computer. Protection remains in force and the **least privilege principle** applies. This principle denies user access to applications or devices that are not expressly permitted. Client components use client hardening functionality as described in the [Ivanti Application Control User Guide \(https://help.ivanti.com\)](https://help.ivanti.com) to protect against user tampering.

Chapter 2

Understanding Cryptography

In this chapter:

- About Encryption
- Advanced Encryption Standard
- RSA Encryption

Cryptography is a field of mathematics concerned with the study of algorithms for encrypting and decrypting data.

Encryption transforms a plain text message using an algorithm, or cipher, into a coded message called ciphertext. Only designated users possessing special knowledge, usually referred to as a key, can access the data.

Cryptography involves two processes:

- Encryption
- Decryption

Encryption can be used to protect computer data, such as files on computers and removable storage devices. The purpose of encryption is to prevent third parties from accessing the sensitive information. This is particularly important for sensitive data. Encrypting such data helps protect it, should physical security measures fail.

Decryption is the process of decoding data that is encrypted using a secret key or password. If the password is not correct, it will be impossible to get the encryption key and subsequently decrypt the encoded information.

About Encryption

The Application Control and Device Control applications use two encryption algorithms to secure data in transit and data at rest.

The encryption algorithms used are:

- Advanced Encryption Standard (AES) 256-bit encryption
- RSA (for Rivest, Shamir, and Adleman)

Using these algorithms Application Control and Device Control accomplish the primary criteria used to measure the effectiveness of a cryptographic method:

- Confidentiality (privacy), only an authorized recipient can extract the original data from the encrypted message.
- Integrity, a recipient can determine whether the data was altered during the transmission.
- Authentication, a recipient can unmistakably identify the sender and verify who actually sent the data.
- Non-repudiation, a sender cannot deny sending the data.

Advanced Encryption Standard

Device Control uses the Advanced Encryption Standard (AES) 256-bit encryption standard, which provides a powerful, unbreakable encryption method to ensure data is always protected.

AES is based on a design principle known as a substitution permutation network and has a fixed block size of 128 bits and a key size of 128, 192, or 256 bits. The AES cipher is specified as a number of repetitions of transformation rounds that convert the input plain text into the final output of cipher text.

Assuming one byte equals 8 bits, the fixed block size of 128 bits is $128 \div 8 = 16$ bytes. AES operates on a 4x4 array of bytes, termed the state. Each turn generates a new state from the previous state. The final state after all rounds contains the ciphered text.

Each round consists of several processing steps, including one that depends on the encryption key. A set of reverse rounds is applied to transform cipher text back into the original plain text using the same encryption key. Ivanti works with a 256-bit block. In this case, the algorithm uses 8x6 matrices as states and sub keys. The 256-bit algorithm executes 14 rounds.

To ensure that the symmetric AES key is not visible when stored in the database, and cannot be read by anyone who has access to the database, Device Control uses public-private key pair-based encryption to encode a symmetric encryption key. This algorithm uses the same key for encryption and decryption.

The Application Server and kernel clients contain a default embedded encryption key pair that is only used for software evaluation purposes. You create your own key pair before deploying the client in your environment using the **Key Pair Generator** tool. If a higher level of protection is required, Ivanti strongly recommends storing the private key external to the Application Server. Only the public key should be available to the clients. The private key should only be available to the Application Server, internally or externally.

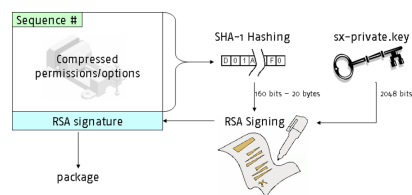


Figure 3: Device Control Key Encryption

RSA Encryption

In cryptography, RSA (named after its developers, Rivest, Shamir, and Adelman) is an algorithm for public-key cryptography. It is an algorithm known to be suitable for signing messages, as well as encryption. RSA is believed to be secure given sufficiently long keys.

RSA involves a public key and a private key. The public key can be known to everyone and is used for encrypting messages. Messages encrypted with the public key can only be decrypted using the private key.

The RSA algorithm bases its security on the difficulty of factoring large prime numbers. RSA keys are typically 1024–2048 bits long. In cryptography, key size or key length is the size, usually measured in bits or bytes, of the key used in a cryptographic algorithm. Ivanti Device and Application Control uses the RSA algorithm with a key size of 2048 bits, making it very difficult to compromise. It is extremely important to use a strong random number generator for the symmetric key, otherwise an eavesdropper wanting to see a message could bypass RSA by guessing the symmetric key.

Chapter

3

Device Control Encryption Methodology

In this chapter:

- Encrypting Removable Storage Devices

Device Control uses a combination of encryption algorithms to protect data communications between the product components and to protect data transferred to removable storage devices and CD/DVD media. Encryption methods are combined with user access control to enforce device permission use policies.

Device Control supports several encryption methods depending upon the needs of users and the policies established for your enterprise network. Each method uses the Advanced Encryption Standard (AES) 256-bit encryption standard. Device Control encryption can be performed centrally, by a network administrator using the Management Console, or decentrally, by a user with permitted access using the client.

Encrypting Removable Storage Devices

Device Control creates encrypted files in virtual memory, and then writes the files to physical media available in various formats, such as removable storage devices and CD/DVDs. Centralized and decentralized encryption provide an administrator with the flexibility to centrally encrypt removable media, enable users to encrypt removable media using the client, and enforce the use of encrypted media.

Device Control supports centralized (from the Management Console) and decentralized (from the client) encryption methods for ciphering data copied to removable storage media. The following methods are available for encrypting removable storage devices and CD/DVD media using:

- *Easy Exchange* encryption, which encrypts devices for portable use. Portable use means that a user can use the encrypted device with a password and the encryption key, without having to connect to the network through a computer running the client.
- Full and slow encryption, which encrypts devices for non-portable use. Non-portable use means that a user can only use the encrypted device with a password when connected to the network through a computer running the client.

Easy Exchange Encryption

Easy Exchange encryption is volume-based. The entire volume of the removable storage media is used for ciphering existing data and all sectors on the volume and installing the Secure Volume Browser (SVolBro.exe) deciphering program.

Devices encrypted using the *Easy Exchange* method do not require a password or encryption key when attached to a computer running the client. These encrypted devices are transparently deciphered when users attach the device to a computer running the client, and there is a Microsoft® Certificate Authority (CA) available from the network for authentication.

Important: When there is no Microsoft Enterprise CA installed in the network, users can only access encrypted data using a password and a public encryption key.

When a user is working outside your network, they must use the installed Secure Volume Browser to access encrypted data. The Secure Volume Browser does not require local administrative rights, however a password and a public encryption key are required. The Secure Volume Browser program is automatically copied on to the media when it is encrypted.

The administrator also has an option during encryption to export the public key to the media or to an external file, depending on enterprise network security policies and procedures.

Important: If the encryption key is not exported to the encrypted media, then an administrator must send the key in a separate file to the user before the decryption process can start.

The *Easy Exchange* encryption method is used for centralized and decentralized encryption because this method uses the Secure Volume Browser to unlock the medium for user access.

Encrypting Media

Encrypting media from the client uses the **Encrypt Medium** utility. The rules governing the behavior of the encryption options depend upon the **Export** permissions assigned by the administrator for user access.

Encryption from the client offers both portable encryption and encryption for external use options. These options offer control over exporting the medium encryption key.

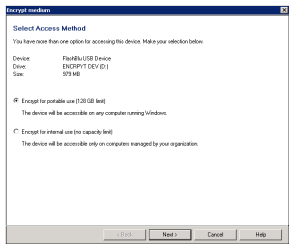


Figure 4: Encrypt Medium - Select Encryption Method Page

Standard User Options Rules

The default behavior for the **Encrypt Medium** utility options are governed by the following rules.

- When a user selects encryption for external use, the **Windows user** option is disabled when selecting **Add** to add users, user can only have one passphrase.
- When a user selects encryption for external use, the **Erase unused space** and **Retain existing data on device** options are enabled and selected by default, unless **Erase unused space on media** is forced by the administrator through the Management Console. When selecting an options, the remaining option is deselected.

For the list of users granted access:

- When a user does not have valid certificate, the user name is displayed in red and disabled.
- When a user is added, the domain and account name are displayed to distinguish between users having similar names in different contexts.
- When a user selects encryption for external use the user can add only one passphrase user.

When a user selects portable encryption:

- User can add any number of **Passphrase users**.
- User can add any number of **Windows users**.

When a user selects encryption for external use a user may not add users.

Encryption options for portable encryption are:

- Enabled when the device size is less than 128GB.

Encryption options for encryption for external use are:

- Enabled when the device size is greater than or equal 128GB.

Data options are set as follows:

- The **Retain existing data on device** is not selected and is disabled if no recognized file system has been found on the media.
- The **Erase unused space on media** option is selected and disabled if this option is set by the administrator in the Management Console.

Centralized Encryption

Centralized encryption is encryption performed at the Management Console by a network administrator. Centralized encryption offers users that have a Microsoft Enterprise Certificate Authority installed transparent device use within the network.

A user encrypting a removable storage device, ciphered using centralized encryption does not perceive that the device is encrypted. Users can freely use their removable storage device with any computer on the enterprise network, with delegated permission. There is no need for the user to have the encryption key or know the password. Authentication automatically takes place in the background, between the client and the certification authority. Even if the user loses the device, data protection is ensured.

Decentralized Encryption

Decentralized encryption enables a user to perform device encryption at a computer workstation without requiring network administrator rights. The user is forced to cipher and administer their removable storage devices, based on user access and device permissions established centrally by the network administrator.

Decentralized encryption is defined by an administrator using a central rule that establishes which users have access to removable storage devices, whether a user is forced to encrypt their removable storage devices, and whether they are allowed to access unencrypted devices. Depending upon the rule, a user may be able to:

- Read and/or write data to a removable storage device.
- Encrypt a device.
- Format a device.

Users encrypt their devices using the Easy Exchange method, where all existing data is erased and the remaining storage volume is encrypted. Removable storage devices encrypted using decentralized encryption can also be used outside the enterprise network, when necessary.

When a user has the necessary permissions formats or modifies an encrypted removable storage device, the Security Identification (SID) changes. The new SID that is not recognized by the Application Server because there is no matching record in the database. Therefore, access to the new device is restricted. This ensures that no data, encrypted or not, can leave the enterprise network using unauthorized removable storage devices. As an additional security measure when a removable storage device is used outside the network, an administrator can choose to export the public key to an external file that can be sent separately to the user, instead of storing the public key on the removable storage device.

Encryption from the client provides several options:

- Passphrase users can use encrypted media with an encryption key stored on the device at the time of encryption.
- Passphrase users can use encrypted media with an encryption key accessed from a file that is stored separately from the media at the time of encryption.
- Windows Active Directory users can use encrypted media with an encryption key protected by a Certificate Authority.

Chapter

4

Application Control Methodology

In this chapter:

- About File, Macro, and Script Execution Control
- Identifying DLL Dependencies
- Maintaining Application Control

Application Control operates on the basis that software application use is explicitly denied unless previously authorized by an administrator.

Application Control is an operating system software application extension that enforces strict control over which executable files, scripts, and macros can be run. An administrator initially creates, and then maintains, a centralized list of authorized applications that users or user groups are explicitly designated to run.

This ensures that only applications that have been previously identified and authorized by a network administrator can be run by users. Any unauthorized software application, known or unknown, cannot run.

The initial central Application Control authorization list can be constructed using a combination of tools available in the Management Console including:

- **Authorization Wizard**
- **Scan Explorer** templates
- *Standard File Definitions*

However, there are other types of executable files, scripts, and macros that have unique Application Control authorization requirements, including embedded macros, scripts, and Dynamic-Link Library (DLL) executable files.

About File, Macro, and Script Execution Control

You can specify which executable files, scripts, and macros a user is permitted to use.

Application Control recognizes the following types of scripts and macros:

- VBScript (short for Visual Basic Scripting Edition) that runs within a host environment, including Windows Script Host (WSH) and Internet Explorer (IE) Internet Information Services (IIS). The script file must have the `.vbs` file extension to be recognized by Application Control.
- JScript (short for JavaScript) that runs as a Windows Script engine plugged into any application that supports Windows Script, such as Internet Explorer, Active Server Pages, and Windows Script Host. The script file must have the `.js` file extension to be recognized by Application Control.
- Visual Basic for Applications (VBA) macros that may be embedded and run from many Microsoft® Office formats, such as `.doc`, `.dot`, `.xls`, and `.ppt` files. A hash is created for the whole file, not just the macro content. VBA is closely related to Visual Basic, but generally only runs code within a host application rather than as a standalone application.

Identifying DLL Dependencies

Dynamic-Link Libraries (DLLs) are executable software applications that cannot run independently. These libraries usually have the file extension `.dll`, `.ocx` (for libraries containing ActiveX controls), or `.drv` (for legacy system drivers). A dependency is the degree to which a program module depends on another to run. When one program depends on another, both of them must be installed and authorized to work together.

You must independently identify all DLL dependencies to authorize the DLLs required for your software applications to work. The **Authorization Wizard** and **Standard File Definitions** (SFD) import tools do not identify software DLL dependencies for authorization. After you identify DLL dependencies, you can identify and assign all required files to the corresponding application file group.

Danger: You must identify and authorize DLLs separately for Application Control, rather than scanning for these files with **Scan Explorer** templates or using the **Authorization Wizard**, because malicious software applications can be hidden in DLLs.

DLLs require independent identification because applications automatically load DLL files:

- Listed in an application DLL/EXE import table before loading a software application, to resolve DLL references.
- Explicitly by the application at run time.
- As dependencies of DLLs that use other DLLs.

Maintaining Application Control

Administrators must perform frequent maintenance tasks for operating system patches and service packs, software updates, new software installations, and frequently changing software application uses in any network environment. These tasks also require frequent updates to the Application Control central file authorization list.

Application Control provides different tools that address:

- [Operating System Updates and Patches](#)
- [Frequently Changing Software Use](#)
- [Software Updates](#)
- [New Software Installations](#)
- [Macros and Other Changing Files](#)

Operating System Updates and Patches

Operating systems are subject to continual updates and upgrades.

Microsoft® provides Windows® Server Update Services (WSUS) for downloading, approving, and managing the distribution of Windows Operating System (OS) updates for all computers in your network. You can use the Ivanti Device and Application Control **Authorization Service** tool to monitor and authorize OS changes and create updates for the central authorization list using Microsoft SUS or WSUS, thereby minimizing the network administration burden.

Frequently Changing Software Use

Some software applications must be updated daily, such as antivirus, antispam, or antispyware applications; other applications only receive periodic upgrades or updates. Using the **Path Rules** feature in the Management Console, combined with trusted ownership, you can allow software updates and modifications of frequently updated applications.

When software application updates are automatically allowed by your network administration policies, you can create specific path rules for each application that is frequently updated. Instead of individually authorizing application files using file groups, you can administer file authorization updates using path rules, if all of the following policies are true:

- You trust the source of the file updates.
- You are confident the update mechanisms can be trusted.

Finally, you can combine the path rules with trusted ownership verification to complete the Application Control authorization schema.

Software Updates

Periodic or single instance software application updates can modify a few application files or constitute a completely new installation. The methods for authorizing these types of software application updates vary, depending upon the source of the update and the computers and users targeted for the update.

Depending on the type of update and the targeted computers you can:

- Create an application-specific template to assign authorized users and user groups, using the **Scan Explorer**.
- Use the **Log Explorer** to identify the software application and assign users and user groups to corresponding authorized file groups.
- Use the **Authorization Wizard**.

New Software Installations

All enterprise network systems routinely deploy new software installations. For planned, known, and trusted source installations of new application software, the **Authorization Wizard** is the most administratively efficient method for authorizing new software application installations.

New software application installations are generally deployed from the following locations:

- Deployment from a central file server repository.
- Using Microsoft Systems Management Server (SMS) packages.
- Directly on a client computer using a CD/DVD source.

Regardless of the new software application deployment method, you can use the **Authorization Wizard** to scan, identify, and authorize the new application software executable files.

Macros and Other Changing Files

As an administrator you may be constantly challenged with authorizing files containing embedded macros. The content of such files may change frequently because any user can edit the macro content stored in the file.

Macros embedded in files with a previously calculated hash, for example Microsoft Word or Microsoft Excel files, are unique. These are legal files that are authorized to run. However, after a file is modified and re-saved in the system, the file no longer corresponds with the hash calculated when the file was initially authorized.

The next time a user attempts to run the modified file, access will be denied because the hash does not match the hash originally calculated. A good practice is to discourage modifying the macro content embedded in files and assign these types of files Windows® **Read-Only** file permission.

Deleting Local Authorization Files

Local authorization should only be granted to trusted users. Depending upon your security policies, you may need to periodically delete local user authorization lists.

You can delete locally authorized software applications when:

- An application is locally authorized by numerous users and merits control by central authorization, instead of local authorization.
- Policy changes require that all file authorizations must be centrally controlled.
- A user can no longer be considered trusted.
- A user mistakenly authorizes an application file.

Local authorizations are stored in a local file on the client. To remove local authorization files, delete the `.locauth` files stored in `%WINDOWS%\system32\lxdata` folder. You can perform local authorization file maintenance on a per-user basis or delete file batches at startup by defining a task in the Windows Scheduler.

Globally Disable Local Authorization

You can disable locally authorized executable files, scripts, or macros using file group assignments.

When you do not want to delete local authorization files, you can still disable access to locally authorized files through user file group assignments.

1. Create a file group named **Not Allowed**.
2. Add all applications that are not allowed to run to the **Not Allowed** file group.
Do not assign this file group to any user/user group.
3. Send updates to all computers.

Chapter

5

Understanding Application Server-Client Communications

In this chapter:

- About Application Server-Client Communications
- About Application Server-Client Communication Encryption
- About Application Server-Client Proxy Communications

The Application Server communicates between the database and the protected client computers.

The Application Server uses a TCP/IP server. When the Application Server is not reachable because of a firewall on the client, the client can initiate communication through a proxy server.

About Application Server-Client Communications

Ivanti Device and Application Control is based on standard TCP/IP protocols for all communication between clients and servers.

TCP/IP was chosen due to its pervasive implementation throughout most IT infrastructures. They are the most widely used open-system (non-proprietary) protocols since they are equally well suited for LAN or WAN communication. Using the TCP/IP protocol offers some clear advantages over other protocols, including the following:

- Allows enterprise networking connectivity between Windows and non-Windows based computers.
- Can be used to create client-server applications.

Currently Ivanti Device and Application Control uses only two configurable ports for full two-way communication between the client and server components. As with other TCP-based services, Application Servers cannot handle clients connecting through a firewall or proxy unless the required ports are opened. By default:

- The server uses port 65129 or 65229 (for the TLS protocol) to listen to clients or other Application Server requests.
- Clients use port 33115 by default to receive information and respond if the Application Server initiated the communication.

These three ports are required for full two-way communication. You can configure these ports as required by your environment.

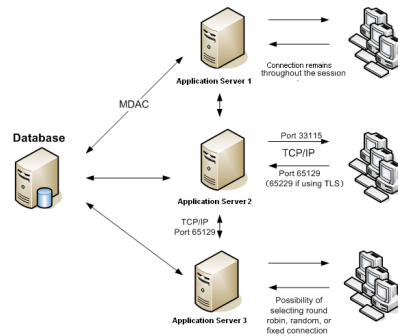


Figure 5: Application Server-Client Communication

Application Server Communications

The Application Server consists internally of two distinct subsystems. The first subsystem handles requests from administrative clients and exposes services via a secure, authenticated Remote Procedure Call (RPC). The second subsystem communicates with the clients.

The Application Server RPC is used to expose administrative functionality (the interfaces required to browse and manage the hashes and file groups in the database) and to offer control over driver behavior. Internally, the Application Server uses a thread pool to perform mass updates. It connects to each client individually according to the driver state. The database keeps track of drivers and users that are on-line. This is more than broadcasting, but offers the advantage of guaranteed delivery, a feature not found in broadcast-capable protocols.

The Application Server uses a TCP/IP server based on Microsoft Windows Input Output Completion Ports (IOCP). The most important server tasks are responding to login and log off notification messages from the clients, such as processing start (boot) and stop (shutdown) messages from clients, and creating and dispatching hash-lists requested by clients.

The TCP/IP client built into the Application Server serves primarily to push updates to clients. When an administrator makes changes to options or permissions, clients may need immediate updates. For permission changes, this typically invalidates the existing hash-list cache.

Forced updates create a need for multiple Application Server instances to communicate intra-server. In particular, when an administrator requests an immediate hash-list update, the instruction to flush the hash-list cache must be relayed to every server to keep the caches coherent. Since all servers share a common database, they register themselves in the database. Intra-server notifications are sent through the respective TCP/IP channel.

Changing Licenses for the Application Server

When you need to change the license for the Application Server, you must stop the server, change the license, then restart the server.

1. Select **Start > Run**.
2. Type `cmd` in the **Open:** field.
3. On the command line, type: `net stop sxs`.
4. Copy the new Ivanti Device and Application Control license file to the `\\Windows\System32` or `\\Windows\SysWOW64` folder, and rename the file to `endpoint.lic`.
5. Select **Start > Run**.
6. Type `cmd` in the **Open:** field.
7. On the command line, type: `net start sxs`.

Result: The Application Server restarts with the new license information.

Changing the SysLog Server

During the SXS installation you can enter a SysLog server address specify whether you want to log audit events, system events, client events, or some combination thereof. You may change the SysLog server address and specifications after installation.

You can modify the following Application Server registry keys as described by the [General Registry Keys](#) section.

1. Select **Start > Run**.
2. Type `regedit` in the **Open:** field.
3. Select **HKEY_LOCAL_MACHINE > SYSTEM > CurrentControlSet > Services > sxs > parameters** from the registry keys.

Step Result: The **Registry Editor** window opens.

4. Select any one or all of the following registry key parameters:
 - `SysLogServerAddress` - specifies the name of the SysLog server.
 - `SysLogGenerateMsg` - specifies the type of log event(s) sent to the SysLog server.
5. Close the **Registry Editor** window.

Result: SysLog behaviour changes based on the registry key data values that you specify.

About Application Server-Client Communication Encryption

The Application Server uses a symmetric encryption system to communicate with the client.

There are several methods that generate symmetrical public-private key pairs. The most well known method is based on the RSA algorithm. The security this algorithm provides relies on the difficulty of factoring large prime numbers. Device Control uses the RSA algorithm with a key size of 2048 bits, making it extremely difficult to compromise. The security of a strong cryptographic system depends on the secrecy of the key and key size.

The need for strong cryptographic security underscores the importance of generating your own key pair during installation and using a long *seed* value, before deploying Device Control clients in your environment. The private key should not be communicated to the clients and should reside on the Application Server computer or stored on an external medium for added security.

Digital Signatures and Certificate Authorities (CA)

For complete data security, digital signatures are combined with digital certificates that authenticate the identity of a sender.

Using a digital signature ensures, to a certain extent, the authenticity of the sender. Since only the public key of the sender can decrypt the digital signature, this only ensures that the sender has the private key corresponding to the public key used to decrypt the digital signature. To confirm the authenticity of the sender, a digital certificate is used. A digital certificate is an electronic document that certifies that a particular user owns a certain public key. A third party, called the certificate authority (CA), signs this document. You need to install the Microsoft CA service to create your own CA to use with Device Control encryption.

Digital Signatures

Device Control uses digital signatures to ensure the integrity of the private and public key pair. A digital signature is a stamp attached to a data transmission that can be used to determine whether an intervening malicious user tampered with the transmission.

The digital signature for a message is generated as follows:

1. A message digest or *hash* is generated using a set of hashing algorithms. A message digest is:
 - A summary of the data transmitted.
 - Always smaller than the message.
 - Detects even the slightest change in the data that produces a different hash.
2. The private key of the sender is used to encrypt the message hash that is the digital signature.
3. The digital signature is attached to the message which is then sent to the recipient.

The recipient then performs the following:

1. Uses the public key of the sender to decrypt the digital signature and obtain the message hash generated by the sender.
2. Uses the same message hash algorithm as the sender to generate a message hash for the received message.

3. Compares the two message hashes. If the message hashes are not exactly the same, a third-party tampered with the message or there was a problem with the data transmission.

You can be assured that the digital signature was originated by the sender, not by a malicious user, because only the public key of the sender can decrypt the digital signature. If the decryption using the public key renders a faulty signature hash, either the signature, or the data, is not exactly what the sender originally transmitted.

About Application Server-Client Proxy Communications

The client communicates with the Application Server using Fully Qualified Domain Name (FQDN) address(es) configured during the client setup. The FQDN addresses may not be reachable when the client initiates communication, particularly when using remote clients via a Virtual Private Network (VPN) connection that does not have a physical connection to the Application Server or a firewall is blocking the required ports when they should not be open for security reasons.

A client may use a proxy configured for the Internet Explorer to reach the Application Server, when the client cannot establish communication using a defined fully qualified domain name (FQDN).

When the Application Server is not reachable for the aforementioned reasons, all communication is accomplished using the Internet through a proxy that acts as a barrier between the internal network and the Internet. Many enterprises use proxy servers to manage a variety of communication protocols and add a higher level of security to their network environment. Data transferred via proxy connections is very resistant to eavesdropping and interception.

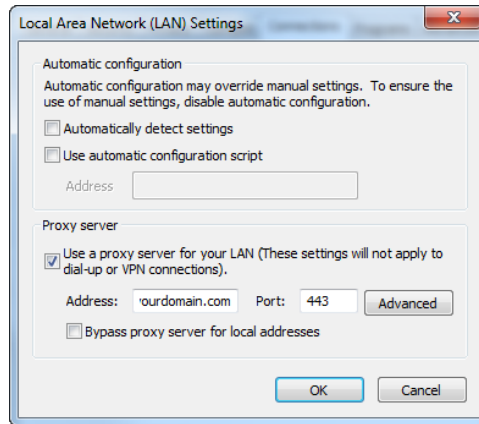
Configuring a Static Proxy on Clients

You can manually configure Clients in your LAN to use a proxy server with a secure Hypertext Transfer Protocol Secure (HTTPS) address when communicating with the Application Server.

Prerequisites:

- Ensure the Client is using a supported version of Microsoft Internet Explorer (IE).
 - Make note of the fully qualified domain name of the proxy server you want to use.
-

1. Open Internet Explorer on the Client.
2. Select **Tools > Internet Options**.
3. Select the **Connections** tab.
4. Click **LAN settings**.
5. Unselect the default **Automatically detect settings** option, if selected.
6. Select **Use a proxy server for your LAN**.
7. In the **Address** field, enter the fully qualified domain name of the proxy server you want to use.
For example, `proxyserver.yourdomain.com`.
8. In the **Port** field, enter 443.
This port enables you to enforce SSL connections.



9. Click **OK**, and close **Internet Options**.

Result: The Client is configured to communicate with the Application Server using a static proxy.

Configuring WPAD for Application Server-Client Proxy Communication

You can configure your Clients to automatically find and use a proxy by setting an option on your Dynamic Host Configuration Protocol (DHCP) Web server that points Web browsers to a `wpad.dat` configuration file.

Prerequisites:

- Install Clients on a supported operating system using the TLS encryption protocol.
- Set the Application Server TLS port registry key value to 443. This port is used for secure web browser communications and should be configured for the client, Application Server, and proxy.
- Install a valid certificate authority.
- Use a supported version of Microsoft Internet Explorer (IE).
- Create a `wpad.dat` configuration file. Basic example:

```
function FindProxyForURL(url, host) {
  return "PROXY proxyserver.example.com:8080"
}
```

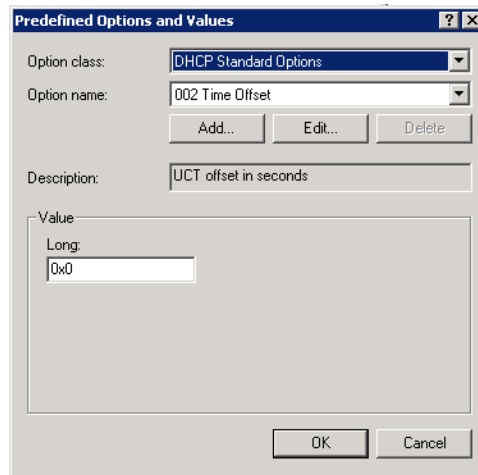
- Make note of the fully qualified domain name for the Web server where `wpad.dat` is located, including the port used.

Create a DHCP option that directs it to your WPAD configuration file.

1. Log on to the DHCP server as a user with administrative access rights.
2. Open the **Server Manager**.
Using either the start screen or start menu, search for **Server Manager** and open it.
3. Expand the console tree on the left to **Roles > DHCP Server > <Computer Name> > IPv4**.

4. Right-click **IPv4** and select **Set Predefined Options**.

The **Predefined Options and Values** dialog opens.



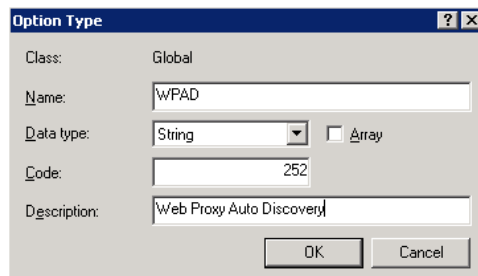
5. Select **DHCP Standard Options** from the **Option class** list.

6. Click **Add**.

Step Result: The **Option Type** dialog opens.

7. Enter the following information:

- Name: **WPAD**
- Data type: **String**
- Code: **252**
- Description: **Web Proxy Auto Discovery**



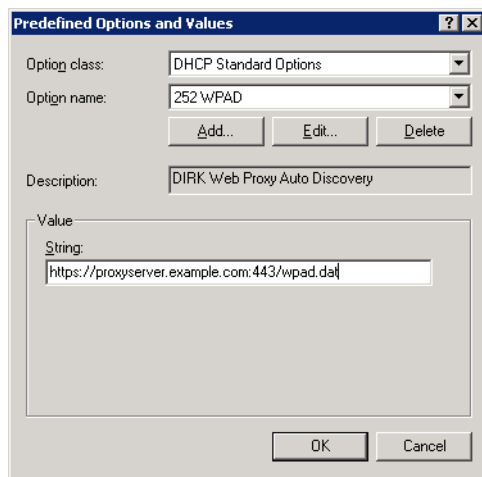
8. Click **OK**.

Step Result: The **Option Type** dialog closes and you return to the **Predefined Options and Values** dialog.

9. Select **252 WPAD** from the **Option name** list.

10. Enter `http://<computer_FQDN>:<port>/wpad.dat` in the **String** field. The string must be in lowercase.

Use the fully qualified domain name of the Web server where `wpad.dat` is located. By default, automatic discovery information is published to port 8080.



11. Click **OK**.

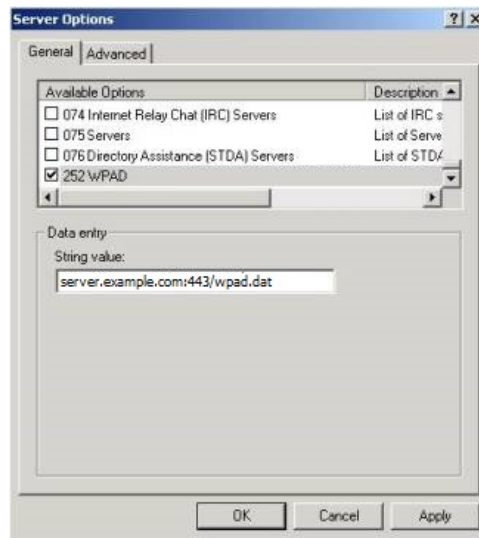
Step Result: You have created a DHCP option that directs your server toward your WPAD configuration file.

Set the DHCP Server to use the DHCP option you just created (252 WPAD).

12. Expand the Server Manager console tree to **Roles > DHCP Server > <Computer Name> > IPv4 > Server Options**.

13. Right-click on **Server Options** and select **Configuration Options**.

14. On the **General** tab, select **252 WPAD** in the **Server Options** window.



Result: You have created a DHCP option (252 WPAD) for your WPAD configuration file and set the DHCP server to use it.

After Completing This Task:

[Configure your Clients to automatically detect your DHCP server.](#)

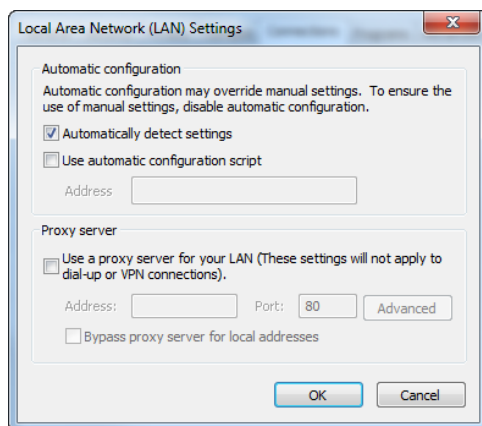
Configuring Clients to Automatically Detect Your DHCP Server

Ensure all Clients are using the default **Automatically detect settings** option found on the LAN Settings interface in Internet Explorer to enable WPAD Application Server-Client Proxy Communication.

Prerequisites:

[You have created a DHCP option \(252 WPAD\) for your WPAD configuration file and set the DHCP server to use it.](#)

1. Open Internet Explorer.
2. Select **Tools > Internet Options**.
3. Select the **Connections** tab.
4. Click **LAN settings**.
5. Ensure only **Automatically detect settings** is selected.



6. Click **OK**.

Result: The Client is configured to communicate with the Application Server using WPAD. Repeat this procedure for each Client.

Chapter

6

Deploying the Client

In this chapter:

- About Ivanti Device and Application Control Client Deployment
- Deploying the Client with Windows Group Policy
- Deploying the Client with Other Tools

You can deploy the client using additional methods and software deployment tools.

The **Ivanti Device and Application Control Client Deployment** tool is generally used to silently deploy the client to a group of computers, after you create deployment packages. Depending upon unique network environment requirements for your organization, you can perform unattended client installations using alternate deployment methods. For information regarding the use of the **Ivanti Device and Application Control Client Deployment** tool, refer to the [Ivanti Device Control User Guide \(https://help.ivanti.com\)](https://help.ivanti.com) or [Ivanti Application Control User Guide \(https://help.ivanti.com\)](https://help.ivanti.com), depending upon the application you are using.

About Ivanti Device and Application Control Client Deployment

You can deploy the client using methods other than the **Ivanti Device and Application Control Client Deployment** tool, depending upon your environment and company policies.

Client deployment methods, in addition to the **Ivanti Device and Application Control Client Deployment** tool, include using:

- A ghost image.
- Windows Group Policy.
- Third-party software deployment tools.

Deploying the Client with Windows Group Policy

You can implement a computer-based group policy for all computers in a domain. Group policies can be applied to sites, domains, or Active Directory organizational units (OU), based on individual system requirements and the types of computers in the groups.

Prerequisites:

Before using a third-party software deployment tool to deploy clients, you must:

1. Create a deployment package using the **Ivanti Device and Application Control Client Deployment** tool. Refer to the [Ivanti Device Control User Guide \(https://help.ivanti.com\)](https://help.ivanti.com) or [Ivanti Application Control User Guide \(https://help.ivanti.com\)](https://help.ivanti.com) for instructions about creating deployment packages.
2. Copy the entire deployment package folder to a local directory on the server that will be used to deploy the client. This directory should include the *.msi installation file and the `sx-public.key` public key file.

The following procedure demonstrates how to apply Windows® group policy for deploying clients.

Note: Execution of this procedure may vary according to individual operating system requirements. For the purposes of this example, the operating system reference is Windows 2003 Server.

1. Select **Start > All Programs > Administrative Tools > Active Directory Users and Computers**.

Step Result: The **Active Directory Users and Computers** dialog opens.

2. In the console tree, right-click the **DomainName**, where the **DomainName** is the name of your domain.
3. Click **Properties**.
4. Select the **Group Policy** tab.
5. Click **New** to create a new group policy.
6. Click **Edit**.
7. Under **Computer Configuration**, expand the **Software Settings** folder.
8. Right-click **Software Installation**.
9. Select **New > Package**.

10. In the **Open** dialog, type the full Universal Naming Convention (UNC) path of the shared installer package that you want to install using the `\\file server\share\client.msi` format.

Important: Do not use **Browse** to access the location. Make sure that you use the UNC path to the shared installer package.

11. Click **Open**.

12. Click **Advanced**.

13. Select the **Modifications** tab.

14. Point to the MST file in the package.

Make sure the path is in UNC format.

15. Click **OK**.

Step Result: The package is listed in the right pane of the **Group Policy** window.

16. Close the **Group Policy** tab.

17. Click **OK**.

18. Close the **Active Directory Users and Computers** dialog.

Result: When the client computer restarts, the managed software package is automatically installed.

Deploying the Client with Other Tools

You can use a third-party software deployment tool to deploy the Ivanti Device and Application Control client for a group of computers in your network.

Prerequisites:

Before using a third-party software deployment tool to deploy clients, you must:

1. Create a deployment package using the **Ivanti Device and Application Control Client Deployment** tool. Refer to the [Ivanti Device Control User Guide \(https://help.ivanti.com\)](https://help.ivanti.com) or [Ivanti Application Control User Guide \(https://help.ivanti.com\)](https://help.ivanti.com) for instructions about creating deployment packages.
 2. Copy the entire deployment package folder to a local directory on the server that will be used to deploy the client. This directory should include the *.msi installation file and the sx-public.key public key file.
-

1. Open the third-party tool.
2. Run the following command-line in the appropriate entry: `Client.mst`
3. Deploy the client using the third-party tool.

Result: The third-party software deployment tool installs the client on the specified target computers.

Chapter 7

Controlling Administrative Rights

In this chapter:

- About Ivanti Device and Application Control Access Control Rights
- Defining User Access
- Using the Access Control Visual Basic Script

You can create delegation rights that can be assigned to Active Directory organizational units (OUs) using a Visual Basic script file.

Access privileges are set in the system to allow or restrict administrative privileges. Administrative privileges are shown in the **Connection** panel of the **Management Console**. The system recognizes two types of administrators:

- Enterprise administrators, who have complete, unrestricted administrative privileges.
- Administrators, who have restricted administrative privileges.

About Ivanti Device and Application Control Access Control Rights

You can use the `ctrlacx.vbs` script to manage and show the user access control rights defined in the Active Directory hierarchy.

You can use the Visual Basic® script file, `ctrlacx.vbs`, to set, view, or modify the **Manage Ivanti Device and Application Control Settings** control rights in the Active Directory. This script allows Active Directory administrators to delegate administrative access to computers, users, user groups, and organizational units without allowing other task management, which is required by default, to the administrators.

Defining User Access

The Management Console can only be accessed by authorized network administrators.
To control user access to the Management Console, you can define two types of administrators:

- An *Enterprise Administrator* has full access to all management functions.

Note: Initially, any member of the Windows `Administrators` group for a Application Server has the privileges of a *Enterprise Administrator*. After an *Enterprise Administrator* is designated, administrative privileges are automatically restricted for the members of the local `Administrators` group.

- An *Administrator* has restricted access to Management Console functions as defined by the *Enterprise Administrator*.

An *Enterprise Administrator* can delegate administrative rights to other administrators using Active Directory Organizational Units. These rights are described in the following table.

Table 2: Ivanti Device and Application Control Administrator Rights

Administrative Rights	Administrator Type	Limitations	Ivanti Device and Application Control Application
View all device permissions and file authorizations	All Ivanti Device and Application Control administrators	NA	Application Control; Device Control
Modify file authorizations	<i>Enterprise Administrators</i>	NA	Application Control
Modify global-level device permissions	<i>Enterprise Administrators</i>	NA	Device Control
	Members of the Settings (Device Control) role	Only users the administrator is allowed to manage	
Modify computer-level device permissions	<i>Enterprise Administrators</i>	NA	Device Control
	Members of the Settings (Device Control) role	Only for the computers that the administrator is allowed to manage	
Modify computer-group device permissions	<i>Enterprise Administrators</i>	NA	Device Control
	Members of the Settings (Device Control) role	Only if the administrator is allowed to manage all the computers in the computer group for all accounts	



Administrative Rights	Administrator Type	Limitations	Ivanti Device and Application Control Application
Manage built-in accounts (Everyone, LocalSystem, and so forth)	<i>Enterprise Administrators</i>	NA	Application Control; Device Control

Initially, any administrator with password access to a Application Server and the Management Console can use the Management Console.

Before using Ivanti Device and Application Control, Ivanti recommends setting up administrators who have access to the Management Console. You can assign different roles to administrators, but you must define at least one *Enterprise Administrator*.

The following rules apply to administrative user roles:

- You must always designate one *Enterprise Administrator* before you modify the list of administrators.
- All Application Servers share the same database, so some administrative rights set for an administrator can be used for other Application Servers.
- Local computer users cannot manage the Management Console, even if assigned as an *Enterprise Administrator*, because they cannot connect to an Application Server.

Using the Access Control Visual Basic Script

You use the Visual Basic® script file, `ctrlacx.vbs`, to set, view, or modify the **Manage Ivanti Device and Application Control Settings** control rights in the Active Directory.

Prerequisites:

- Install the Windows® Script Host (WSH) interpreter. See [Script Host \(http://msdn.microsoft.com/en-us/library/ec0wcxh3\(VS.85\).aspx\)](http://msdn.microsoft.com/en-us/library/ec0wcxh3(VS.85).aspx) for additional information about the Windows Script Host.
- Schedule domain synchronization.

When `ctrlacx.vbs` runs, the script creates a special entry in the permissions list of the AD organization unit named **Manage Ivanti Device and Application Control Settings**. This entry only affects Device Control administrators and the devices they control permissions for. If you assign this setting to a specific user, who is also an *Administrator* defined using the **User Access Manager** dialog in the Management Console, this *Administrator* can only manage, directly from the Management Console, the designated users, user groups, and computers that the *Administrator* has assigned rights for. Administrator access rights are described by *Defining User Access* in the [Ivanti Device Control User Guide \(https://help.ivanti.com\)](https://help.ivanti.com) or [Ivanti Application Control User Guide \(https://help.ivanti.com\)](https://help.ivanti.com).

1. Select **Start > Run**.

- 2. Type: `cscript ctrlacx.vbs [parameter from following list]>filename.txt`
- 3. Add any of the following optional parameters, individually or in combination, to the parameters list command line:

Parameter	Description
-	Shows a brief description for each available parameter.
-e	Lists all access control rights, with condensed output.
-v	Lists all access control rights, with detailed output.
-q cn	Shows control rights by canonical name.
-s	Shows Manage Ivanti Device and Application Control Settings rights.
-create	Creates or updates Manage Ivanti Device and Application Control Settings rights.
-delete	Deletes Manage Ivanti Device and Application Control Settings rights.

- 4. Click **OK**.

Result: The delegation rights you create can be assigned to Active Directory organizational units (OUs).

Example:

To list all control access rights in condensed mode redirecting the output to MyFile.txt file, type:

```
cscript ctrlacx.vbs -e > MyFile.txt
```

To show the **Manage Ivanti Device and Application Control Settings** rights interactively, type:

```
ctrlacx.vbs -s
```

After Completing This Task:

You can assign the delegation rights by using the *Windows Management Services and MMC* when you run the script with `-create` parameter. See [Windows Management Services and MMC \(http://technet.microsoft.com/en-us/library/bb742441.aspx#XSLTsection123121120120\)](http://technet.microsoft.com/en-us/library/bb742441.aspx#XSLTsection123121120120) for additional information about assigning delegation rights.



Chapter

8

Using File Tools

In this chapter:

- About File Tools
- Using the Ivanti Device and Application Control Authorization Service Tool
- Using the Application Control File Tool
- Using the SXDomain Tool

The Ivanti Device and Application Control product suite is equipped with several features that ease the administrative burden associated with managing files for central file authorization, as the database file volume grows.

As you work with the Application Control and Device Control products, the number of files in the database increases over time, increasing the administrative burden associated with managing large volumes of files. Ivanti Device and Application Control provides several administrative functions that allow an administrator to manage multiple files automatically or in a single instance.

About File Tools

Ivanti Device and Application Control provides tools for administrators to manage large volumes of files in the Application Control database.

You can use the following tools to manage large volumes of files for Application Control, including the:

- Authorization Service
- Versatile File Processor
- File Import/Export
- SXDomain

Using the Ivanti Device and Application Control Authorization Service Tool

You can use the **Application Control Authorization Service Tool** to monitor approved and synchronized changes to executable file authorization policies using Microsoft® Update Services (SUS) and Windows® Server Update Services (WSUS).

The **Ivanti Device and Application Control Authorization Service Tool** service authorizes all approved Microsoft® updates and fixes, creates corresponding hash files, and updates the database.

1. From the location where you saved the Ivanti Device and Application Control application software, run the `\server\authsrv\setup.exe` file.

Step Result: The **Welcome** page opens.

2. Click **Next**.

Step Result: The **License Agreement** page opens.

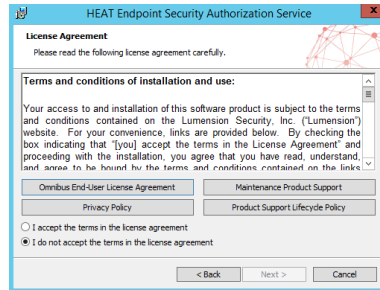


Figure 6: License Agreement Page

3. Review the license agreement and, if you agree, select **I accept the terms in the license agreement**.
4. Click **Next**.
5. From the **Ivanti Device and Application Control Application Server** dialog, enter an Application Server IP address in the corresponding field.
6. Click **Next**.

Step Result: The **Software Update Services** dialog opens.

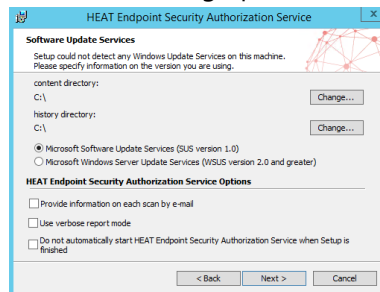


Figure 7: Software Update Services Dialog

7. Select one of the following options:
 - **Microsoft Software Update Services (SUS version 1.0)**
 - **Microsoft Windows Server Update Services (WSUS version 2.0 and greater)**

8. Select any of the following **Ivanti Device and Application Control Authorization Service Options**.

- **Provide information on each scan by e-mail**
- **Use verbose report mode**
- **Do not automatically start Ivanti Device and Application Control Authorization Service when Setup is finished**

Step Result: If you choose to **Provide information on each scan by e-mail**, the **E-mail configuration** dialog opens. The **Ivanti Device and Application Control Authorization Service** tool does not support **Outlook Express** or **Internet Information Server (IIS)** as clients for sending email messages. If there is already an account for these types of clients, the SMTP IP address is transferred directly to the Ivanti Device and Application Control Authorization Service configuration.

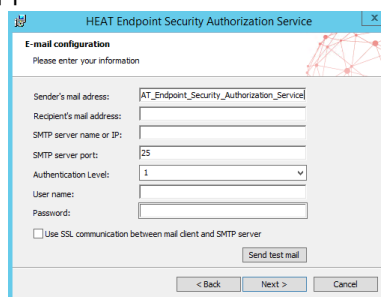


Figure 8: E-mail Configuration Dialog

Note: If you select **Do not automatically start Ivanti Device and Application Control Authorization Service when Setup is finished**, the service will automatically start when one of the following events occurs:

- A change is made by WSUS in the default update folder.
- An administrator approves the SUS updates using the Management Console.
- One hour elapses after installing the Ivanti Device and Application Control Authorization Service.

9. Click **Next**.

10. From the **Destination Folder** dialog, click **Change** to choose an installation destination folder other than the default folder `C:\Program Files\Ivanti\Device and Application Control\`.

11. Click **Next**.

Step Result: The **Ready to Install the Program** dialog opens.

12. Click **Install**.

A progress bar runs on the page, showing installation progress.

Step Result: The **Completed** page opens.

13. Click Finish.

Result: The Ivanti Device and Application Control **Database Explorer** window shows the specified installation directory(s) in the **Files** tab.

After Completing This Task:

After installing the **Ivanti Device and Application Control Authorization Service Tool** you must configure and synchronize WSUS. See [Configure and Synchronize WSUS](http://technet.microsoft.com/en-us/library/cc708455.aspx) (<http://technet.microsoft.com/en-us/library/cc708455.aspx>) for more information about configuring WSUS or SUS.

You can modify **Ivanti Device and Application Control Authorization Service Tool** by rerunning the **Authorization Service Tool**.

Using the Application Control File Tool

You use the Versatile File Processor tool to scan files in specific locations.

The **Application Control File Tool** tool consists of two files:

- The `filetool.dll` that provides the underlying functionality. This file is used by other Ivanti tools, such as the Authorization Service Tool.
- The `filetool.exe` command line executable file that uses the `filetool.dll`.

The **Application Control File Tool** tool works in one of two modes depending on the parameters you select. The operation modes are:

- Online mode - The online mode is used to scan specific files. The **Application Control File Tool** connects, by default, to the local Application Server using the login identity of the current user and scans file locations to assign the scanned files to file groups. The files can be assigned automatically, using Application Server suggestions.

Note: You can override the user login default using the command line parameters `-s <server>` and `-u <user/password>`. After the assignment, and if the `-p` option is specified, the tool can request Application Server to notify all the clients (drivers).

- Offline mode - The offline mode produces scan files. Even though the scan files have exactly the same format as the scan files produced by the driver, these cannot be used directly. Offline mode requires an output file name only if using the `-o` parameter. File assignments cannot be made in offline mode. You can copy the resulting scan files, from the directory that you specified in the command line instruction, to the Application Server `%/DataFileDirectory/%` to compare your results with previously scanned files.

Note: The **Application Control File Tool** tool can scan files contained in archives files with the following extensions: `.cab`, `.zip`, `.rar`, `.ace`, `.jar`, and `.cs`. However, some `.cab` archives do not have a standard cabinet structure despite having a `.cab` extension so it may not be possible to scan them correctly.

1. Select **Start > Run**.

2. Browse to the location where you saved the Ivanti Device and Application Control application software, and select `\bin\tools\FileTool.exe` file.
3. Add any of the following optional parameters, individually or in combination, to the parameters list command line.

Tip: If you execute the **Application Control File Tool** without parameters, help output is displayed.

Parameter	Description
-s	Application Server (The default is the current Application Server).
-u	User/password to connect to the Application Server. (The default is the current user).
-o	Offline mode; generate a scan
-d0	Delta mode off. (This is the default value).
-d1	Delta mode on, avoid rescanning files already scanned.
-d2	Delta mode on, clear the list then memorize files already scanned.
-f	Access failure, retry (n) times with a least (m) seconds between attempts. Example: -f 10, 3 (10 retry attempts with 3 seconds) between attempts.
-v	Verbose report; generate an xml report that you save to a location you specify in the command line after the parameter.
-r	Report; generate an xml report, errors only, that you save to a location you specify in the command line after the parameter.
-i	Ignore archive contents.
target	<p>File or directory to scan.</p> <p>Tip: To avoid recursive scan on directory, terminate the target entry with \\.The target may also be entered using one of the following keywords in brackets:</p> <ul style="list-style-type: none"> • [drives] (All hard drives) • [media] (All removable media) • [all] (All hard drives and removable media)
-e	An option wildcard mask.
-c0	File group creation, use only existing groups.
-c1	File group creation, create if necessary. (This is the default value).

Parameter	Description
-a0	Keep existing assignment, automatically assign new files to existing groups, and assign remaining files to groups you select. Tip: Accepts a list of file groups using the format FileGroup ₁ ,FileGroup ₂ ...FileGroup _n
-a1	Keep existing assignment, assign new files to groups you select.
-a2	Assign existing and new files to group.
-x0	Process all files. Tip: Be very cautious when using this parameter because all files are scanned, even if the files are not executable.
-x1	Only process executable files.
-x2	Only process executable files having a valid digital signature.
-p	Push updates to all online clients.

Result: The specified file locations are scanned and the files designated by the parameters are assigned to file groups.

Using the SXDomain Tool

The SxDomain utility provides a method to automatically schedule domain synchronization, using the Windows **Task Scheduler**.

You can schedule domain synchronizations with a task scheduler, such as the Windows **Task Scheduler**. You create a batch file that contains a list domains to synchronize.

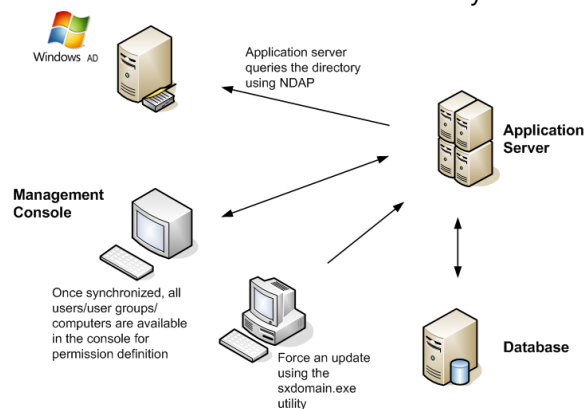


Figure 9: Synchronization Script Process

1. Navigate to C:\Program Files\Ivanti\Device and Application Control\SXTools.
 2. Create a batch file named `sxsync.bat` containing the following command line: `CMD/C SXDOMAIN-s SXS_Server -i -e <mydomains.txt> error_list.txt.`
 3. Navigate to the Windows **Control Panel**, select **Scheduled Tasks**.
 4. Select **Add Scheduled Tasks**.
Step Result: The **Scheduled Task Wizard** dialog opens.
 5. Click **Next**.
 6. Select the `sxsynch.bat` file from files shown.
 7. Click **Next**.
 8. Type a name for your scheduled task at the prompt.
 9. Select a schedule frequency from the options listed from **Perform this task**.
 10. Click **Next**.
 11. Select the day and time you want to perform the task.
 12. Click **Next**.
Step Result: A user name and password information dialog opens.
 13. Type the user name in the **User Name** field.
 14. Type the associated password in the **Password** and **Confirm Password** fields.
 15. Click **Next**.
Step Result: A dialog opens showing the name of the scheduled task and the date and time the task is scheduled to perform.
 16. Click **Finish**.
- Result:** Domain synchronization is scheduled to perform according to your specifications.

Chapter

9

Managing Registry Keys

In this chapter:

- Database Connection Loss Registry Key
- Authorization Service Registry Key
- Debugging Registry Key
- General Registry Keys
- Security Registry Keys
- Command & Control Registry Key
- Client Kernel Registry Key
- Software Registry Key
- Application Server Registry Key
- Authorization Wizard Registry Key
- Resolving Driver Conflicts
- Preventing a Security Audit

Network administrators can modify Ivanti Device and Application Control registry keys, so that Application Control and Device Control can operate more effectively or efficiently in unique network environments or according to specific enterprise network policies.

Registry keys can be used to adapt Application Control and Device Control according to enterprise-specific network environment policies and requirements. Registry keys that can be modified include:

- Application Server keys
- Client keys
- Management Console keys

Database Connection Loss Registry Key

The database connection registry key parameters allow you to define Application Server behavior during periods of database connectivity loss. The Application Server can run with intermittent database connectivity, ignoring the lack of database connection for specified periods. During these periods, the Application Server retries connecting to the database. After repeatedly attempting to establish database connectivity, the Application Server declines further connectivity requests from the client and console, until database connectivity is restored.

The following table describes the `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\sxs\parameters` registry key parameters for the Application Server. All registry key entries are of the type: `REG_SZ` (= string value), unless designated otherwise.

Table 3: Database Connection Loss Registry Key Parameters

Parameter Name	Description	Default Value
DbConnectionCount	Shows the number of database connections in the connection pool.	20
DbConnectionString	Shows the driver, server, database, and trusted connection or user name and password.	Provider=sqloledb; DataSource= ; InitialCatalog=sx; Trusted_Connection=yes
DbInitializationDelay	Shows the period in seconds that the Application Server waits before contacting the SQL server.	300

Authorization Service Registry Key

The parameters for the Ivanti Device and Application Control Authorization Service tool registry key parameters are shown in the following table.

Table 4: Authorization Service Tool Registry Key Parameters

Key Name	Description	Default Values
CmdLineBlockParams	Allows user to setup assignment mode, group name(s), and filters in the <code>FileTool.exe</code> command line parameters.	-a1 = Microsoft Update Files
Log file name	Depends upon the <code>Log to file</code> value.	authSrvHlpr.log
Log to file	Sends debug message to the log file yes=1	no = 0



Key Name	Description	Default Values
OutputDirectory	Directory path where output XML reports are located.	C:\Program Files\Ivanti\Device and Application Control\ \ Authorization Service\Logs\
SendMail	The service sends e-mail at the end of the scan, which includes the command line and attached XML report.	no = 0
SXSServer	Name or IP address of the Application Server.	Not applicable.
VerboseReport	Report mode. no=normal report mode	yes = verbose report mode
WSUSContentDirectory	Absolute directory path where WSUS files are located.	C:\

Debugging Registry Key

You can use the debugging registry key parameters to specify the behavior for debugging the Application Server.

The following table describes the `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\sxs\parameters` registry key parameters for the Application Server. All registry key entries are of the type: `REG_SZ` (= string value), unless designated otherwise.

Table 5: Debugging Registry Key Parameters

Key Name	Description	Default Value
Debug	For the Application Server running as a service, a debugger will launch and attach to the server when the value is set to yes = 1.	no = 0
Log file name	Shows the name of the log file written when Log to file = yes .	sxs.log
Log to console	Sends debug messages to the console when the value is set to yes = 1.	no = 0
Log to dbwin	Sends debug messages to the DBwin32 when the value is set to yes = 1.	no = 0

Key Name	Description	Default Value
Log to file	Sends debug messages to the log file when the value is set to yes = 1.	no = 0
Log file size	Sets the maximum size of a log file (in bytes). When the file size is reached, a new log file with an incremented filename is created. The limit may be exceeded by a small amount.	16777216
Log file count	Sets the maximum number of log files to retain.	8
VerboseSyncLogging	The Application Server logs the important object attributes retrieved during domain synchronization.	no = 0

General Registry Keys

The general registry key parameters govern the general behavior of the Application Server.

The following table describes the `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\sxs\parameters` registry key parameters for the Application Server. All registry key entries are of the type: `REG_SZ` (= string value), unless designated otherwise.

Table 6: General Registry Key Parameters

Parameter Name	Description	Default Value
AdoVersion	Specifies a string representing the version of Active Directory objects used.	No value.
Concurrency	Shows how many running threads are allowed for the input/output connection port (IOCP). Zero is equivalent to one thread per central processing unit (CPU). Maximum number equals <code>MaxThreads</code> .	0=auto
DataFileDirectory	Shows the name of the base directory where the Application Server stores data files.	C: \DataFileDirectory
SysLogGenerateMsg	Determines what type of logs are sent to the SysLog server. 1 = Audit logs only 2 = System logs only	3 = Audit and system logs
SysLogServerAddress	The SysLog server that the Application Server sends server and audit log events to.	Specified during initial client installation.

Parameter Name	Description	Default Value
SyncContinueOnError	When set to <code>true</code> , configures the Application Server to log duplicate user or machine SIDs (or other ACL errors), instead of abort, during a directory synchronization. Errors are recorded to the Application Server log (default <code>sxs.log</code>).	No default value. Key must be manually created and set to <code>true</code> .

Security Registry Keys

The security registry key parameters govern the security configuration for the Application Server.

The following table describes the `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\sxs\parameters` registry key parameters for the Application Server. All registry key entries are of the type: `REG_SZ` (= string value), unless designated otherwise.

Table 7: Security Registry Key Parameters

Parameter Name	Description	Default Value
CommVer	Specifies the client-server communication protocol that the Application Server uses as follows: 1 = Client versions 3.1 or 3.2 2 = Client version 4.0 or higher 3 = Client version 4.0 or higher with TLS enabled	3
MaxSockets	Specifies the maximum number of TCP connections allowed at any time.	5000
Port	Specifies the TCP port where the socket-based Application Server listens for new connections. Minimum value = 1 Maximum value = 65534	65129

Parameter Name	Description	Default Value
RpcProtectionLevel	Specifies whether the RPC server requires RPC clients to authenticate. 0 = OS selects protection level 1 = No protection 2 = Client identify is verified when connecting to the Ivanti Device and Application Control Application Server 3 = Examines client credentials for TCP connection. 4 = Examines client credentials for TCP connection, adds cryptographic signature to every packet. 5 = Examines client credentials for TCP connection, adds additional cryptographic signature to every packet. 6 = Examines client credentials for TCP connection, adds cryptographic signature to every packet, and encrypts data in both directions	6
SecureInterSxs	Specifies that inter-Ivanti Device and Application Control Application Server communication use the TLS protocol.	no
SndPort	Specifies the TCP where the Ivanti Device and Application Control client listens for new connections. Minimum value = 1 Maximum value = 65534	33115
SxdConnectTimeoutMSec	Specifies the time in milliseconds that the Application Server waits to accept a TCP connection for the client. The value should be between 500 and 120000 ms.	5000
SxdPort	Specifies the TCP port where the built-in client server listens for new connections. Minimum value = 1 Maximum value = 65534	33115
TLSTMaxSockets	Specifies the maximum number of TCP connections allowed when using the TLS protocol.	0

Parameter Name	Description	Default Value
TLSPort	Specifies the TLS port where the socket-based Application Server listens for new connections. Minimum value = 1 Maximum value = 65534	65229

Configure MaxSockets

When certain security registry key parameters interact, some of the combinations formed are not valid as shown in the following table.

Table 8: Configure MaxSockets

Secure InterSxs	CommVer	TLSMax Sockets	MaxSockets	Combined Result
No	<3	0	0	Not valid
		0	>0	Valid
		>0	0	Not valid
		>0	>0	Valid
	3	0	0	Not valid
		0	>0	Not valid
		>0	0	Not valid
		>0	>0	Valid
Yes	<3	0	0	Not valid
		0	>0	Not valid
		>0	0	Valid
		>0	>0	Valid
	3	0	0	Not valid
		0	>0	Not valid
		>0	0	Valid
		>0	>0	Valid

Configuring MaxSockets and TLSMaxSockets

The following table describes the parameters used for the `MaxSockets` and `TLSMaxSockets` configuration rules.

Table 9: Configuring MaxSockets and TLSMaxSockets

TLSMaxSockets and MaxSockets Values	Description
<code>TLSMaxSockets>0 AND MaxSockets=0</code>	Only TLS connections are available for Application Server-client communication using the <code>TLSPort</code> port specification.
<code>TLSMaxSockets=0 AND MaxSockets>0</code>	Only non-TLS connections are available for Application Server-client communication using the <code>Port</code> port specification.
<code>TLSMaxSockets>0 AND MaxSockets>0</code>	Both TLS and non-TLS connections are available for Application Server-client communication using the <code>TLSPort</code> and <code>Port</code> port specifications.

Command & Control Registry Key

The registry key parameters for Ivanti Device and Application Control Command & Control (SCC) that govern the behavior for all communication between the server, the client(s), and the Certificate Authority (CA) server are shown in the following table.

The following table describes the `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\scomc\parameters` registry key parameters for the Ivanti Device and Application Control Command & Control module. All registry key entries are of the type: `REG_SZ` (= string value), unless designated otherwise.

Table 10: Ivanti Device and Application Control Command and Control (SCC) Registry Key Parameters

Key Name	Description	Default Value
<code>CertGeneration</code>	<code>Yes</code> =Client is in automatic mode and requests CA certificate. <code>No</code> =Client is in manual mode and a user certificate must be generated manually.	Defined during client install.
<code>ImportDir</code>	Shows the directory used import the policies file.	<code>C:\Program Files\Ivanti\Device and Application Control\\Import</code>



Key Name	Description	Default Value
ListenPort	Show a port number value set between 1025 and 65536 (ports numbers 0 to 1024 are reserved for privileged services).	Port number 33115 is the default value.
Log file name	Shows the name of the log file.	scomc.log
Log to console	Yes=Sends debug message to console. No=No debug messages sent to console.	No default value.
Log to dbwin	Yes=Sends debug message to dbwin. No=No debug messages sent to dbwin.	No
Log to file	Yes=Sends debug message to log file. No=No debug messages sent to log file.	No
Log file size	Sets the maximum size of a log file (in bytes). When the file size is reached, a new log file with an incremented filename is created. The limit may be exceeded by a small amount.	16777216
Log file count	Sets the maximum number of log files to retain.	8
Servers	Shows a list of Application Server names by FDQN or IP address.	Defined during server install.
SyncPeriod	Shows the time in milliseconds for synchronization periods between the Application Server and the client.	3,600,000 ms (one hour) is the default value.
TicketDir	Shows directory where endpoint maintenance tickets are stored.	Cannot modify this key.
UseTLS	Yes=TLS protocol used No=TLS protocol not used	Defined during server install.

Client Kernel Registry Key

The client kernel parameter subentries are shown in the following table.

The following table describes the `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\sk\parameters` registry key parameters for the client kernel. All registry key entries are of the type: `REG_SZ` (= string value), unless designated otherwise.

Table 11: Client Kernel Registry Key Parameters

Parameter Name	Description	Default Value
EncryptionMode	EncryptionMode which SK runs one of the following masks: encLpc = 1 encFips = 2	encDefault = 0
NDISInstallation	Specifies how SCOMC installs SK-NDIS . 1 = Uninstall only an existing SK-NDIS driver. 2 = Uninstall an existing SK-NDIS driver, then install a new SK-NDIS driver.	None.



Software Registry Key

The registry key parameters that govern the application software configuration are shown in the following table.

The following table describes the `HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows NT\RPC` registry key parameters for the Management Console module. All registry key entries are of the type: `REG_SZ` (= string value), unless designated otherwise.

Table 12: Software Registry Key Parameters

Parameter Name	Description	Default Value
<code>EnableAuthEpResolution</code>	<p>Sets the <code>RestrictRemoteClients</code> functionality status for the RPC interface. Remote and anonymous RPC call is available only if:</p> <ul style="list-style-type: none"> • RPC registry key and <code>EnableAuthEpResolution</code> value described are created during setup. • The <code>EnableAuthEpResolution</code> registry value is set to 0. 	0

Application Server Registry Key

The Application Server registry key parameters allow you to define default server and number of servers connected to the database and console.

The following table describes the `HKEY_CURRENT_USER\SOFTWARE\Lumension\Endpoint Security\Server` registry key parameters for the Application Server module. All registry key entries are of the type: `REG_SZ` (= string value), unless designated otherwise.

Table 13: Application Server Registry Key Parameters

Parameter Name	Description	Default Value
<code>DefaultServer</code>	Name of the last Application Server that the Management Console connected to.	None.

Parameter Name	Description	Default Value
Server List	Lists the names of the Application Servers that the Management Console connected to. Server names are separated by commas.	None.

Authorization Wizard Registry Key

The Authorization Wizard registry key parameters allow you to define the behavior for communication between the Application Server and the wizard.

The following table describes the `HKEY_CURRENT_USER\SOFTWARE\Lumension\Endpoint Security\SecureEXE\AuthWiz\Parameters` registry key parameters for the **Authorization Wizard** module. All registry key entries are of the type: `REG_SZ` (= string value), unless designated otherwise.

Table 14: Authorization Wizard Registry Key Parameters

Parameter Name	Description	Default Value
AutoAssign	Specifies whether the Application Server should assign the hashed files to a file group.	Yes
DefaultServer	The Application Server that the Authorization Wizard sends a result to. The value is the Application Server IP address.	127.0.0.1
DefaultTemDir	The file location where the Authorization Wizard stores results before uploading to an Application Server.	%TEMP%

Resolving Driver Conflicts

Conflicts may occur between drivers that exist on managed endpoints and the Ivanti Device and Application Control driver. In these cases, you can create registry entries to exclude the conflicting driver from protection by Ivanti Device and Application Control.

Note: You can repeat this procedure to create multiple registry entries for each driver conflict.

1. Open the Windows registry on the client computer.
2. Navigate to the following registry subkey: `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\sk\Parameters`.



3. Select **New > DWORD (32-bit) Value**.

Step Result: A new entry is created.

4. In the **Name** column for the new entry, type the file path to the driver that you want to exclude.
5. Press Enter.
6. Right-click the new registry entry.
7. Select **Modify**.
8. Enter 0 in the **Value data** field.
9. Click **OK**.

Step Result: The value for the new registry entry is set. The driver specified in the in the registry entry name will be excluded from protection by Ivanti Device and Application Control.

Preventing a Security Audit

Installing Ivanti Device and Application Control on Windows 7 operating systems may trigger a security audit resulting in a false error message reporting invalid hashes. Adding a registry key will prevent the audit and the resulting error message from occurring.

1. Open the Windows registry on the client computer.
2. Navigate to the following registry subkey: `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\sk\Parameters`.
3. Select **New > DWORD (32-bit) Value**.

Step Result: A new entry is created.

4. In the **Name** column for the new entry, type `ExcludeProtectedProcesses`.
5. Press Enter.
6. Right-click the new registry entry.
7. Select **Modify**.
8. Enter 1 in the **Value data** field.
9. Click **OK**.

Step Result: The value for the new registry entry is set.

