Contents

Optimizing Office 365 traffic with the Windows 10 VPN client

Identity and access management

4/29/2021 • 2 minutes to read • Edit Online

Learn more about identity and access management technologies in Windows 10 and Windows 10 Mobile.

SECTION	DESCRIPTION
Technical support policy for lost or forgotten passwords	Outlines the ways in which Microsoft can help you reset a lost or forgotten password, and provides links to instructions for doing so.
Access control	Describes access control in Windows, which is the process of authorizing users, groups, and computers to access objects on the network or computer. Key concepts that make up access control are permissions, ownership of objects, inheritance of permissions, user rights, and object auditing.
Configure S/MIME for Windows 10 and Windows 10 Mobile	In Windows 10, S/MIME lets users encrypt outgoing messages and attachments so that only intended recipients who have a digital identification (ID), also known as a certificate, can read them. Users can digitally sign a message, which provides the recipients with a way to verify the identity of the sender and that the message hasn't been tampered with.
Install digital certificates on Windows 10 Mobile	Digital certificates bind the identity of a user or computer to a pair of keys that can be used to encrypt and sign digital information. Certificates are issued by a certification authority (CA) that vouches for the identity of the certificate holder, and they enable secure client communications with websites and services.
Protect derived domain credentials with Credential Guard	Introduced in Windows 10 Enterprise, Credential Guard uses virtualization-based security to isolate secrets so that only privileged system software can access them. Unauthorized access to these secrets can lead to credential theft attacks, such as Pass-the-Hash or Pass-The-Ticket. Credential Guard helps prevent these attacks by protecting NTLM password hashes and Kerberos Ticket Granting Tickets.
Protect Remote Desktop credentials with Remote Credential Guard	Remote Credential Guard helps you protect your credentials over a Remote Desktop connection by redirecting the Kerberos requests back to the device that's requesting the connection.
User Account Control	Provides information about User Account Control (UAC), which helps prevent malware from damaging a PC and helps organizations deploy a better-managed desktop. UAC can help block the automatic installation of unauthorized apps and prevent inadvertent changes to system settings.

SECTION	DESCRIPTION
Virtual Smart Cards	Provides information about deploying and managing virtual smart cards, which are functionally similar to physical smart cards and appear in Windows as smart cards that are always-inserted. Virtual smart cards use the Trusted Platform Module (TPM) chip that is available on computers in many organizations, rather than requiring the use of a separate physical smart card and reader.
VPN technical guide	Virtual private networks (VPN) let you give your users secure remote access to your company network. Windows 10 adds useful new VPN profile options to help you manage how users connect.
Smart Cards	Provides a collection of references topics about smart cards, which are tamper-resistant portable storage devices that can enhance the security of tasks such as authenticating clients, signing code, securing e-mail, and signing in with a Windows domain account.
Windows Hello for Business	In Windows 10, Windows Hello replaces passwords with strong two-factor authentication on PCs and mobile devices. This authentication consists of a new type of user credential that is tied to a device and a biometric or PIN.
Windows 10 Credential Theft Mitigation Guide Abstract	Learn more about credential theft mitigation in Windows 10.

Technical support policy for lost or forgotten passwords

12/3/2019 • 2 minutes to read • Edit Online

Microsoft takes security seriously. This is for your protection. Microsoft accounts, the Windows operating system, and other Microsoft products include passwords to help secure your information. This article provides some options that you can use to reset or recover your password if you forget it. Be aware that, if these options don't work, Microsoft support engineers can't help you retrieve or circumvent a lost or forgotten password.

If you lose or forget a password, you can use the links in this article to find published support information that will help you reset the password.

How to reset a password for a domain account

If you lose or forget the password for a domain account, contact your IT administrator or Helpdesk. For more information, see Change or reset your Windows password.

How to reset a password for a Microsoft account

If you lose or forget the password for your Microsoft Account, use the Recover your account wizard.

This wizard requests your security proofs. If you have forgotten your security proofs, or no longer have access to them, select **I no longer have these anymore**. After you select this option, fill out a form for the Microsoft Account team. Provide as much information as you can on this form. The Microsoft Account team reviews the information that you provide to determine whether you are the account holder. This decision is final. Microsoft does not influence the team's choice of action.

How to reset a password for a local account on a Windows device

Local accounts on a device include the device's Administrator account.

Windows 10

If you lose or forget the password for a local account on a device that runs Windows 10, see Reset your Windows 10 local account password.

Windows 8.1 or Windows 7

If you lose or forget the password for a local account on a device that runs Windows 8.1 or Windows 7, see Change or reset your Windows password. In that article, you can select your operating system version from the **Select Product Version** menu.

How to reset a hardware BIOS password

If you lose or forget the password for the hardware BIOS of a device, contact the device manufacturer for help and support. If you do contact the manufacturer online, make sure that you visit the manufacturer website and not the website of some third party.

How to reset a password for an individual file

Some applications let you password-protect individual files. If you lose or forget such a password, you can rely on that application only to reset or recover it. Microsoft support engineers cannot help you reset, retrieve, or

circumvent such passwords.

Using third-party password tools

Some third-party companies claim to be able to circumvent passwords that have been applied to files and features that Microsoft programs use. For legal reasons, we cannot recommend or endorse any one of these companies. If you want help to circumvent or reset a password, you can locate and contact a third party for this help. However, you use such third-party products and services at your own risk.

Access Control Overview

3/26/2021 • 6 minutes to read • Edit Online

Applies to

- Windows 10
- Windows Server 2016

This topic for the IT professional describes access control in Windows, which is the process of authorizing users, groups, and computers to access objects on the network or computer. Key concepts that make up access control are permissions, ownership of objects, inheritance of permissions, user rights, and object auditing.

Feature description

Computers that are running a supported version of Windows can control the use of system and network resources through the interrelated mechanisms of authentication and authorization. After a user is authenticated, the Windows operating system uses built-in authorization and access control technologies to implement the second phase of protecting resources: determining if an authenticated user has the correct permissions to access a resource.

Shared resources are available to users and groups other than the resource's owner, and they need to be protected from unauthorized use. In the access control model, users and groups (also referred to as security principals) are represented by unique security identifiers (SIDs). They are assigned rights and permissions that inform the operating system what each user and group can do. Each resource has an owner who grants permissions to security principals. During the access control check, these permissions are examined to determine which security principals can access the resource and how they can access it.

Security principals perform actions (which include Read, Write, Modify, or Full control) on objects. Objects include files, folders, printers, registry keys, and Active Directory Domain Services (AD DS) objects. Shared resources use access control lists (ACLs) to assign permissions. This enables resource managers to enforce access control in the following ways:

- Deny access to unauthorized users and groups
- Set well-defined limits on the access that is provided to authorized users and groups

Object owners generally grant permissions to security groups rather than to individual users. Users and computers that are added to existing groups assume the permissions of that group. If an object (such as a folder) can hold other objects (such as subfolders and files), it is called a container. In a hierarchy of objects, the relationship between a container and its content is expressed by referring to the container as the parent. An object in the container is referred to as the child, and the child inherits the access control settings of the parent. Object owners often define permissions for container objects, rather than individual child objects, to ease access control management.

This content set contains:

- Dynamic Access Control Overview
- Security identifiers
- Security Principals
    - Local Accounts

- Active Directory Accounts
- Microsoft Accounts
- Service Accounts
- Active Directory Security Groups

Practical applications

Administrators who use the supported version of Windows can refine the application and management of access control to objects and subjects to provide the following security:

- Protect a greater number and variety of network resources from misuse.
- Provision users to access resources in a manner that is consistent with organizational policies and the requirements of their jobs.
- Enable users to access resources from a variety of devices in numerous locations.
- Update users' ability to access resources on a regular basis as an organization's policies change or as users' jobs change.
- Account for a growing number of use scenarios (such as access from remote locations or from a rapidly expanding variety of devices, such as tablet computers and mobile phones).
- Identify and resolve access issues when legitimate users are unable to access resources that they need to perform their jobs.

Permissions

Permissions define the type of access that is granted to a user or group for an object or object property. For example, the Finance group can be granted Read and Write permissions for a file named Payroll.dat.

By using the access control user interface, you can set NTFS permissions for objects such as files, Active Directory objects, registry objects, or system objects such as processes. Permissions can be granted to any user, group, or computer. It is a good practice to assign permissions to groups because it improves system performance when verifying access to an object.

For any object, you can grant permissions to:

- Groups, users, and other objects with security identifiers in the domain.
- Groups and users in that domain and any trusted domains.
- Local groups and users on the computer where the object resides.

The permissions attached to an object depend on the type of object. For example, the permissions that can be attached to a file are different from those that can be attached to a registry key. Some permissions, however, are common to most types of objects. These common permissions are:

- Read
- Modify
- Change owner
- Delete

When you set permissions, you specify the level of access for groups and users. For example, you can let one user read the contents of a file, let another user make changes to the file, and prevent all other users from

accessing the file. You can set similar permissions on printers so that certain users can configure the printer and other users can only print.

When you need to change the permissions on a file, you can run Windows Explorer, right-click the file name, and click **Properties**. On the **Security** tab, you can change permissions on the file. For more information, see Managing Permissions.

Note Another kind of permissions, called share permissions, is set on the Sharing tab of a folder's **Properties** page or by using the Shared Folder Wizard. For more information see Share and NTFS Permissions on a File Server.

Ownership of objects

An owner is assigned to an object when that object is created. By default, the owner is the creator of the object. No matter what permissions are set on an object, the owner of the object can always change the permissions. For more information, see Manage Object Ownership.

Inheritance of permissions

Inheritance allows administrators to easily assign and manage permissions. This feature automatically causes objects within a container to inherit all the inheritable permissions of that container. For example, the files within a folder inherit the permissions of the folder. Only permissions marked to be inherited will be inherited.

User rights

User rights grant specific privileges and sign-in rights to users and groups in your computing environment. Administrators can assign specific rights to group accounts or to individual user accounts. These rights authorize users to perform specific actions, such as signing in to a system interactively or backing up files and directories.

User rights are different from permissions because user rights apply to user accounts, and permissions are associated with objects. Although user rights can apply to individual user accounts, user rights are best administered on a group account basis. There is no support in the access control user interface to grant user rights. However, user rights assignment can be administered through **Local Security Settings**.

For more information about user rights, see User Rights Assignment.

Object auditing

With administrator's rights, you can audit users' successful or failed access to objects. You can select which object access to audit by using the access control user interface, but first you must enable the audit policy by selecting **Audit object access** under **Local Policies** in **Local Security Settings**. You can then view these security-related events in the Security log in Event Viewer.

For more information about auditing, see Security Auditing Overview.

See also

- For more information about access control and authorization, see Access Control and Authorization Overview.

Dynamic Access Control Overview

3/26/2021 • 8 minutes to read • Edit Online

Applies to

- Windows Server 2016

This overview topic for the IT professional describes Dynamic Access Control and its associated elements, which were introduced in Windows Server 2012 and Windows 8.

Domain-based Dynamic Access Control enables administrators to apply access-control permissions and restrictions based on well-defined rules that can include the sensitivity of the resources, the job or role of the user, and the configuration of the device that is used to access these resources.

For example, a user might have different permissions when they access a resource from their office computer versus when they are using a portable computer over a virtual private network. Or access may be allowed only if a device meets the security requirements that are defined by the network administrators. When Dynamic Access Control is used, a user's permissions change dynamically without additional administrator intervention if the user's job or role changes (resulting in changes to the user's account attributes in AD DS). For more detailed examples of Dynamic Access Control in use, see the scenarios described in Dynamic Access Control: Scenario Overview.

Dynamic Access Control is not supported in Windows operating systems prior to Windows Server 2012 and Windows 8. When Dynamic Access Control is configured in environments with supported and non-supported versions of Windows, only the supported versions will implement the changes.

Features and concepts associated with Dynamic Access Control include:

- Central access rules
- Central access policies
- Claims
- Expressions
- Proposed permissions

Central access rules

A central access rule is an expression of authorization rules that can include one or more conditions involving user groups, user claims, device claims, and resource properties. Multiple central access rules can be combined into a central access policy.

If one or more central access rules have been defined for a domain, file share administrators can match specific rules to specific resources and business requirements.

Central access policies

Central access policies are authorization policies that include conditional expressions. For example, let's say an organization has a business requirement to restrict access to personally identifiable information (PII) in files to only the file owner and members of the human resources (HR) department who are allowed to view PII information. This represents an organization-wide policy that applies to PII files wherever they are located on file servers across the organization. To implement this policy, an organization needs to be able to:

- Identify and mark the files that contain the PII.

- Identify the group of HR members who are allowed to view the PII information.
- Add the central access policy to a central access rule, and apply the central access rule to all files that contain the PII, wherever they are located amongst the file servers across the organization.

Central access policies act as security umbrellas that an organization applies across its servers. These policies are in addition to (but do not replace) the local access policies or discretionary access control lists (DACLs) that are applied to files and folders.

Claims

A claim is a unique piece of information about a user, device, or resource that has been published by a domain controller. The user's title, the department classification of a file, or the health state of a computer are valid examples of a claim. An entity can involve more than one claim, and any combination of claims can be used to authorize access to resources. The following types of claims are available in the supported versions of Windows:

- **User claims** Active Directory attributes that are associated with a specific user.
- **Device claims** Active Directory attributes that are associated with a specific computer object.
- **Resource attributes** Global resource properties that are marked for use in authorization decisions and published in Active Directory.

Claims make it possible for administrators to make precise organization- or enterprise-wide statements about users, devices, and resources that can be incorporated in expressions, rules, and policies.

Expressions

Conditional expressions are an enhancement to access control management that allow or deny access to resources only when certain conditions are met, for example, group membership, location, or the security state of the device. Expressions are managed through the Advanced Security Settings dialog box of the ACL Editor or the Central Access Rule Editor in the Active Directory Administrative Center (ADAC).

Expressions help administrators manage access to sensitive resources with flexible conditions in increasingly complex business environments.

Proposed permissions

Proposed permissions enable an administrator to more accurately model the impact of potential changes to access control settings without actually changing them.

Predicting the effective access to a resource helps you plan and configure permissions for those resources before implementing those changes.

Additional changes

Additional enhancements in the supported versions of Windows that support Dynamic Access Control include:

Support in the Kerberos authentication protocol to reliably provide user claims, device claims, and device groups.

By default, devices running any of the supported versions of Windows are able to process Dynamic Access Control-related Kerberos tickets, which include data needed for compound authentication. Domain controllers are able to issue and respond to Kerberos tickets with compound authentication-related information. When a domain is configured to recognize Dynamic Access Control, devices receive claims from domain controllers during initial authentication, and they receive compound authentication tickets when submitting service ticket requests. Compound authentication results in an access token that includes the identity of the user and the device on the resources that recognize Dynamic Access Control.

Support for using the Key Distribution Center (KDC) Group Policy setting to enable Dynamic Access Control for a domain.

Every domain controller needs to have the same Administrative Template policy setting, which is located at **Computer Configuration\Policies\Administrative Templates\System\KDC\Support Dynamic Access Control and Kerberos armoring**.

Support in Active Directory to store user and device claims, resource properties, and central access policy objects.

Support for using Group Policy to deploy central access policy objects.

The following Group Policy setting enables you to deploy central access policy objects to file servers in your organization: **Computer Configuration\Policies\ Windows Settings\Security Settings\File System\Central Access Policy**.

Support for claims-based file authorization and auditing for file systems by using Group Policy and Global Object Access Auditing

You must enable staged central access policy auditing to audit the effective access of central access policy by using proposed permissions. You configure this setting for the computer under **Advanced Audit Policy Configuration** in the **Security Settings** of a Group Policy Object (GPO). After you configure the security setting in the GPO, you can deploy the GPO to computers in your network.

Support for transforming or filtering claim policy objects that traverse Active Directory forest trusts

You can filter or transform incoming and outgoing claims that traverse a forest trust. There are three basic scenarios for filtering and transforming claims:

- **Value-based filtering** Filters can be based on the value of a claim. This allows the trusted forest to prevent claims with certain values from being sent to the trusting forest. Domain controllers in trusting forests can use value-based filtering to guard against an elevation-of-privilege attack by filtering the incoming claims with specific values from the trusted forest.
- **Claim type-based filtering** Filters are based on the type of claim, rather than the value of the claim. You identify the claim type by the name of the claim. You use claim type-based filtering in the trusted forest, and it prevents Windows from sending claims that disclose information to the trusting forest.
- **Claim type-based transformation** Manipulates a claim before sending it to the intended target. You use claim type-based transformation in the trusted forest to generalize a known claim that contains specific information. You can use transformations to generalize the claim-type, the claim value, or both.

Software requirements

Because claims and compound authentication for Dynamic Access Control require Kerberos authentication extensions, any domain that supports Dynamic Access Control must have enough domain controllers running the supported versions of Windows to support authentication from Dynamic Access Control-aware Kerberos clients. By default, devices must use domain controllers in other sites. If no such domain controllers are available, authentication will fail. Therefore, you must support one of the following conditions:

- Every domain that supports Dynamic Access Control must have enough domain controllers running the supported versions of Windows Server to support authentication from all devices running the supported versions of Windows or Windows Server.
- Devices running the supported versions of Windows or that do not protect resources by using claims or compound identity, should disable Kerberos protocol support for Dynamic Access Control.

For domains that support user claims, every domain controller running the supported versions of Windows server must be configured with the appropriate setting to support claims and compound authentication, and to provide Kerberos armoring. Configure settings in the KDC Administrative Template policy as follows:

- **Always provide claims** Use this setting if all domain controllers are running the supported versions of Windows Server. In addition, set the domain functional level to Windows Server 2012 or higher.

- **Supported** When you use this setting, monitor domain controllers to ensure that the number of domain controllers running the supported versions of Windows Server is sufficient for the number of client computers that need to access resources protected by Dynamic Access Control.

If the user domain and file server domain are in different forests, all domain controllers in the file server's forest root must be set at the Windows Server 2012 or higher functional level.

If clients do not recognize Dynamic Access Control, there must be a two-way trust relationship between the two forests.

If claims are transformed when they leave a forest, all domain controllers in the user's forest root must be set at the Windows Server 2012 or higher functional level.

A file server running a server operating system that supports Dyamic Access Control must have a Group Policy setting that specifies whether it needs to get user claims for user tokens that do not carry claims. This setting is set by default to **Automatic**, which results in this Group Policy setting to be turned **On** if there is a central policy that contains user or device claims for that file server. If the file server contains discretionary ACLs that include user claims, you need to set this Group Policy to **On** so that the server knows to request claims on behalf of users that do not provide claims when they access the server.

See also

- Access control overview

Security identifiers

7/21/2021 • 31 minutes to read • Edit Online

Applies to

- Windows 10
- Windows Server 2016

This topic for the IT professional describes security identifiers and how they work in regards to accounts and groups in the Windows operating system.

What are security identifiers?

A security identifier (SID) is used to uniquely identify a security principal or security group. Security principals can represent any entity that can be authenticated by the operating system, such as a user account, a computer account, or a thread or process that runs in the security context of a user or computer account.

Each account or group, or process running in the security context of the account, has a unique SID that is issued by an authority, such as a Windows domain controller. It is stored in a security database. The system generates the SID that identifies a particular account or group at the time the account or group is created. When a SID has been used as the unique identifier for a user or group, it can never be used again to identify another user or group.

Each time a user signs in, the system creates an access token for that user. The access token contains the user's SID, user rights, and the SIDs for any groups the user belongs to. This token provides the security context for whatever actions the user performs on that computer.

In addition to the uniquely created, domain-specific SIDs that are assigned to specific users and groups, there are well-known SIDs that identify generic groups and generic users. For example, the Everyone and World SIDs identify a group that includes all users. Well-known SIDs have values that remain constant across all operating systems.

SIDs are a fundamental building block of the Windows security model. They work with specific components of the authorization and access control technologies in the security infrastructure of the Windows Server operating systems. This helps protect access to network resources and provides a more secure computing environment.

The content in this topic applies to computers that are running the supported versions of the Windows operating system as designated in the **Applies To** list at the beginning of this topic.

How security identifiers work

Users refer to accounts by using the account name, but the operating system internally refers to accounts and processes that run in the security context of the account by using their security identifiers (SIDs). For domain accounts, the SID of a security principal is created by concatenating the SID of the domain with a relative identifier (RID) for the account. SIDs are unique within their scope (domain or local), and they are never reused.

The operating system generates a SID that identifies a particular account or group at the time the account or group is created. The SID for a local account or group is generated by the Local Security Authority (LSA) on the computer, and it is stored with other account information in a secure area of the registry. The SID for a domain account or group is generated by the domain security authority, and it is stored as an attribute of the User or Group object in Active Directory Domain Services.

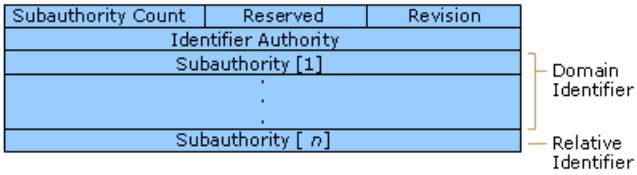
For every local account and group, the SID is unique for the computer where it was created. No two accounts or

groups on the computer ever share the same SID. Likewise, for every domain account and group, the SID is unique within an enterprise. This means that the SID for an account or group that is created in one domain will never match the SID for an account or group created in any other domain in the enterprise.

SIDs always remain unique. Security authorities never issue the same SID twice, and they never reuse SIDs for deleted accounts. For example, if a user with a user account in a Windows domain leaves her job, an administrator deletes her Active Directory account, including the SID that identifies the account. If she later returns to a different job at the same company, an administrator creates a new account, and the Windows Server operating system generates a new SID. The new SID does not match the old one; so none of the user's access from her old account is transferred to the new account. Her two accounts represent two completely different security principals.

Security identifier architecture

A security identifier is a data structure in binary format that contains a variable number of values. The first values in the structure contain information about the SID structure. The remaining values are arranged in a hierarchy (similar to a telephone number), and they identify the SID-issuing authority (for example, "NT Authority"), the SID-issuing domain, and a particular security principal or group. The following image illustrates the structure of a SID.



The individual values of a SID are described in the following table.

COMMENT	DESCRIPTION
Revision	Indicates the version of the SID structure that is used in a particular SID.
Identifier authority	Identifies the highest level of authority that can issue SIDs for a particular type of security principal. For example, the identifier authority value in the SID for the Everyone group is 1 (World Authority). The identifier authority value in the SID for a specific Windows Server account or group is 5 (NT Authority).
Subauthorities	>Holds the most important information in a SID, which is contained in a series of one or more subauthority values. All values up to, but not including, the last value in the series collectively identify a domain in an enterprise. This part of the series is called the domain identifier. The last value in the series, which is called the relative identifier (RID), identifies a particular account or group relative to a domain.

The components of a SID are easier to visualize when SIDs are converted from a binary to a string format by using standard notation:

```
S-R-X-Y1-Y2-Yn-1-Yn
```

In this notation, the components of a SID are represented as shown in the following table.

COMMENT	DESCRIPTION
S	Indicates that the string is a SID
R	Indicates the revision level
X	Indicates the identifier authority value
Y	Represents a series of subauthority values, where *n* is the number of values

The SID's most important information is contained in the series of subauthority values. The first part of the series (-Y1-Y2-Y*n*-1) is the domain identifier. This element of the SID becomes significant in an enterprise with several domains, because the domain identifier differentiates SIDs that are issued by one domain from SIDs that are issued by all other domains in the enterprise. No two domains in an enterprise share the same domain identifier.

The last item in the series of subauthority values (-Y*n*) is the relative identifier. It distinguishes one account or group from all other accounts and groups in the domain. No two accounts or groups in any domain share the same relative identifier.

For example, the SID for the built-in Administrators group is represented in standardized SID notation as the following string:

```
S-1-5-32-544
```

This SID has four components:

- A revision level (1)
- An identifier authority value (5, NT Authority)
- A domain identifier (32, Builtin)
- A relative identifier (544, Administrators)

SIDs for built-in accounts and groups always have the same domain identifier value: 32. This value identifies the domain **Builtin**, which exists on every computer that is running a version of the Windows Server operating system. It is never necessary to distinguish one computer's built-in accounts and groups from another computer's built-in accounts and groups because they are local in scope. They are local to a single computer, or in the case of domain controllers for a network domain, they are local to several computers that are acting as one.

Built-in accounts and groups need to be distinguished from one another within the scope of the **Builtin** domain. Therefore, the SID for each account and group has a unique relative identifier. A relative identifier value of 544 is unique to the built-in Administrators group. No other account or group in the **Builtin** domain has a SID with a final value of 544.

In another example, consider the SID for the global group, Domain Admins. Every domain in an enterprise has a Domain Admins group, and the SID for each group is different. The following example represents the SID for the Domain Admins group in the Contoso, Ltd. domain (Contoso\Domain Admins):

```
S-1-5-21-1004336348-1177238915-682003330-512
```

The SID for Contoso\Domain Admins has:

- A revision level (1)
- An identifier authority (5, NT Authority)
- A domain identifier (21-1004336348-1177238915-682003330, Contoso)
- A relative identifier (512, Domain Admins)

The SID for Contoso\Domain Admins is distinguished from the SIDs for other Domain Admins groups in the same enterprise by its domain identifier: 21-1004336348-1177238915-682003330. No other domain in the enterprise uses this value as its domain identifier. The SID for Contoso\Domain Admins is distinguished from the SIDs for other accounts and groups that are created in the Contoso domain by its relative identifier, 512. No other account or group in the domain has a SID with a final value of 512.

Relative identifier allocation

When accounts and groups are stored in an account database that is managed by a local Security Accounts Manager (SAM), it is fairly easy for the system to generate a unique relative identifier for each account and in a group that it creates on a stand-alone computer. The SAM on a stand-alone computer can track the relative identifier values that it has used before and make sure that it never uses them again.

In a network domain, however, generating unique relative identifiers is a more complex process. Windows Server network domains can have several domain controllers. Each domain controller stores Active Directory account information. This means that, in a network domain, there are as many copies of the account database as there are domain controllers. In addition to this, every copy of the account database is a master copy. New accounts and groups can be created on any domain controller. Changes that are made to Active Directory on one domain controller are replicated to all other domain controllers in the domain. The process of replicating changes in one master copy of the account database to all other master copies is called a multimaster operation.

The process of generating unique relative identifiers is a single-master operation. One domain controller is assigned the role of relative identifier (RID) master, and it allocates a sequence of relative identifiers to each domain controller in the domain. When a new domain account or group is created in one domain controller's replica of Active Directory, it is assigned a SID. The relative identifier for the new SID is taken from the domain controller's allocation of relative identifiers. When its supply of relative identifiers begins to run low, the domain controller requests another block from the RID master.

Each domain controller uses each value in a block of relative identifiers only once. The RID master allocates each block of relative identifier values only once. This process assures that every account and group created in the domain has a unique relative identifier.

Security identifiers and globally unique identifiers

When a new domain user or group account is created, Active Directory stores the account's SID in the **ObjectSID** property of a User or Group object. It also assigns the new object a globally unique identifier (GUID), which is a 128-bit value that is unique not only in the enterprise, but also across the world. GUIDs are assigned to every object that is created by Active Directory, not only User and Group objects. Each object's GUID is stored in its **ObjectGUID** property.

Active Directory uses GUIDs internally to identify objects. For example, the GUID is one of an object's properties that is published in the global catalog. Searching the global catalog for a User object GUID produces results if the user has an account somewhere in the enterprise. In fact, searching for any object by **ObjectGUID** might be the most reliable way of finding the object you want to locate. The values of other object properties can change, but the **ObjectGUID** property never changes. When an object is assigned a GUID, it keeps that value for life.

If a user moves from one domain to another, the user gets a new SID. The SID for a group object does not change because groups stay in the domain where they were created. However, if people move, their accounts

can move with them. If an employee moves from North America to Europe, but stays in the same company, an administrator for the enterprise can move the employee's User object from, for example, Contoso\NoAm to Contoso\Europe. If the administrator does this, the User object for the account needs a new SID. The domain identifier portion of a SID that is issued in NoAm is unique to NoAm; so the SID for the user's account in Europe has a different domain identifier. The relative identifier portion of a SID is unique relative to the domain; so if the domain changes, the relative identifier also changes.

When a User object moves from one domain to another, a new SID must be generated for the user account and stored in the **ObjectSID** property. Before the new value is written to the property, the previous value is copied to another property of a User object, **SIDHistory**. This property can hold multiple values. Each time a User object moves to another domain, a new SID is generated and stored in the **ObjectSID** property, and another value is added to the list of old SIDs in **SIDHistory**. When a user signs in and is successfully authenticated, the domain authentication service queries Active Directory for all the SIDs that are associated with the user, including the user's current SID, the user's old SIDs, and the SIDs for the user's groups. All these SIDs are returned to the authentication client, and they are included in the user's access token. When the user tries to gain access to a resource, any one of the SIDs in the access token (including one of the SIDs in **SIDHistory**), can allow or deny the user access.

If you allow or deny users' access to a resource based on their jobs, you should allow or deny access to a group, not to an individual. That way, when users change jobs or move to other departments, you can easily adjust their access by removing them from certain groups and adding them to others.

However, if you allow or deny an individual user access to resources, you probably want that user's access to remain the same no matter how many times the user's account domain changes. The **SIDHistory** property makes this possible. When a user changes domains, there is no need to change the access control list (ACL) on any resource. If an ACL has the user's old SID, but not the new one, the old SID is still in the user's access token. It is listed among the SIDs for the user's groups, and the user is granted or denied access based on the old SID.

Well-known SIDs

The values of certain SIDs are constant across all systems. They are created when the operating system or domain is installed. They are called well-known SIDs because they identify generic users or generic groups.

There are universal well-known SIDs that are meaningful on all secure systems that use this security model, including operating systems other than Windows. In addition, there are well-known SIDs that are meaningful only on Windows operating systems.

The following table lists the universal well-known SIDs.

VALUE	UNIVERSAL WELL-KNOWN SID	IDENTIFIES
S-1-0-0	Null SID	A group with no members. This is often used when a SID value is not known.
S-1-1-0	World	A group that includes all users.
S-1-2-0	Local	Users who log on to terminals that are locally (physically) connected to the system.
S-1-2-1	Console Logon	A group that includes users who are logged on to the physical console.

VALUE	UNIVERSAL WELL-KNOWN SID	IDENTIFIES
S-1-3-0	Creator Owner ID	A security identifier to be replaced by the security identifier of the user who created a new object. This SID is used in inheritable ACEs.
S-1-3-1	Creator Group ID	A security identifier to be replaced by the primary-group SID of the user who created a new object. Use this SID in inheritable ACEs.
S-1-3-2	Creator Owner Server	
S-1-3-3	Creator Group Server	
S-1-3-4	Owner Rights	A group that represents the current owner of the object. When an ACE that carries this SID is applied to an object, the system ignores the implicit READ_CONTROL and WRITE_DAC permissions for the object owner.
S-1-4	Non-unique Authority	A SID that represents an identifier authority.
S-1-5	NT Authority	A SID that represents an identifier authority.
S-1-5-80-0	All Services	A group that includes all service processes configured on the system. Membership is controlled by the operating system.

The following table lists the predefined identifier authority constants. The first four values are used with universal well-known SIDs, and the last value is used with well-known SIDs in Windows operating systems designated in the **Applies To** list.

IDENTIFIER AUTHORITY	VALUE	SID STRING PREFIX
SECURITY_NULL_SID_AUTHORITY	0	S-1-0
SECURITY_WORLD_SID_AUTHORITY	1	S-1-1
SECURITY_LOCAL_SID_AUTHORITY	2	S-1-2
SECURITY_CREATOR_SID_AUTHORITY	3	S-1-3

The following RID values are used with universal well-known SIDs. The Identifier authority column shows the prefix of the identifier authority with which you can combine the RID to create a universal well-known SID.

RELATIVE IDENTIFIER AUTHORITY	VALUE	IDENTIFIER AUTHORITY
SECURITY_NULL_RID	0	S-1-0

RELATIVE IDENTIFIER AUTHORITY	VALUE	IDENTIFIER AUTHORITY
SECURITY_WORLD_RID	0	S-1-1
SECURITY_LOCAL_RID	0	S-1-2
SECURITY_CREATOR_OWNER_RID	0	S-1-3
SECURITY_CREATOR_GROUP_RID	1	S-1-3

The SECURITY_NT_AUTHORITY (S-1-5) predefined identifier authority produces SIDs that are not universal and are meaningful only in installations of the Windows operating systems that are designated in the **Applies To** list at the beginning of this topic. The following table lists the well-known SIDs.

SID	DISPLAY NAME	DESCRIPTION
S-1-5-1	Dialup	A group that includes all users who are logged on to the system by means of a dial-up connection.
S-1-5-113	Local account	You can use this SID when restricting network logon to local accounts instead of "administrator" or equivalent. This SID can be effective in blocking network logon for local users and groups by account type regardless of what they are actually named.
S-1-5-114	Local account and member of Administrators group	You can use this SID when restricting network logon to local accounts instead of "administrator" or equivalent. This SID can be effective in blocking network logon for local users and groups by account type regardless of what they are actually named.
S-1-5-2	Network	A group that includes all users who are logged on by means of a network connection. Access tokens for interactive users do not contain the Network SID.
S-1-5-3	Batch	A group that includes all users who have logged on by means of a batch queue facility, such as task scheduler jobs.

SID	DISPLAY NAME	DESCRIPTION
S-1-5-4	Interactive	A group that includes all users who log on interactively. A user can start an interactive logon session by logging on directly at the keyboard, by opening a Remote Desktop Services connection from a remote computer, or by using a remote shell such as Telnet. In each case, the user's access token contains the Interactive SID. If the user signs in by using a Remote Desktop Services connection, the user's access token also contains the Remote Interactive Logon SID.
S-1-5-5- *X*-*Y*	Logon Session	The *X* and *Y* values for these SIDs uniquely identify a particular logon session.
S-1-5-6	Service	A group that includes all security principals that have signed in as a service.
S-1-5-7	Anonymous Logon	A user who has connected to the computer without supplying a user name and password. The Anonymous Logon identity is different from the identity that is used by Internet Information Services (IIS) for anonymous web access. IIS uses an actual account—by default, IUSR_ *ComputerName*, for anonymous access to resources on a website. Strictly speaking, such access is not anonymous because the security principal is known even though unidentified people are using the account. IUSR_ *ComputerName* (or whatever you name the account) has a password, and IIS logs on the account when the service starts. As a result, the IIS "anonymous" user is a member of Authenticated Users but Anonymous Logon is not.
S-1-5-8	Proxy	Does not currently apply: this SID is not used.
S-1-5-9	Enterprise Domain Controllers	A group that includes all domain controllers in a forest of domains.

SID	DISPLAY NAME	DESCRIPTION
S-1-5-10	Self	A placeholder in an ACE for a user, group, or computer object in Active Directory. When you grant permissions to Self, you grant them to the security principal that is represented by the object. During an access check, the operating system replaces the SID for Self with the SID for the security principal that is represented by the object.
S-1-5-11	Authenticated Users	A group that includes all users and computers with identities that have been authenticated. Authenticated Users does not include Guest even if the Guest account has a password. This group includes authenticated security principals from any trusted domain, not only the current domain.
S-1-5-12	Restricted Code	An identity that is used by a process that is running in a restricted security context. In Windows and Windows Server operating systems, a software restriction policy can assign one of three security levels to code: unrestricted, restricted, or disallowed. When code runs at the restricted security level, the Restricted SID is added to the user's access token.
S-1-5-13	Terminal Server User	A group that includes all users who sign in to a server with Remote Desktop Services enabled.
S-1-5-14	Remote Interactive Logon	A group that includes all users who log on to the computer by using a remote desktop connection. This group is a subset of the Interactive group. Access tokens that contain the Remote Interactive Logon SID also contain the Interactive SID.
S-1-5-15	This Organization	A group that includes all users from the same organization. Only included with Active Directory accounts and only added by a domain controller.
S-1-5-17	IIS_USRS	An account that is used by the default Internet Information Services (IIS) user.

SID	DISPLAY NAME	DESCRIPTION
S-1-5-18	System (or LocalSystem)	An identity that is used locally by the operating system and by services that are configured to sign in as LocalSystem. System is a hidden member of Administrators. That is, any process running as System has the SID for the built-in Administrators group in its access token. When a process that is running locally as System accesses network resources, it does so by using the computer's domain identity. Its access token on the remote computer includes the SID for the local computer's domain account plus SIDs for security groups that the computer is a member of, such as Domain Computers and Authenticated Users.
S-1-5-19	NT Authority (LocalService)	An identity that is used by services that are local to the computer, have no need for extensive local access, and do not need authenticated network access. Services that run as LocalService access local resources as ordinary users, and they access network resources as anonymous users. As a result, a service that runs as LocalService has significantly less authority than a service that runs as LocalSystem locally and on the network.
S-1-5-20	Network Service	An identity that is used by services that have no need for extensive local access but do need authenticated network access. Services running as NetworkService access local resources as ordinary users and access network resources by using the computer's identity. As a result, a service that runs as NetworkService has the same network access as a service that runs as LocalSystem, but it has significantly reduced local access.

SID	DISPLAY NAME	DESCRIPTION
S-1-5-*domain*-500	Administrator	A user account for the system administrator. Every computer has a local Administrator account and every domain has a domain Administrator account. The Administrator account is the first account created during operating system installation. The account cannot be deleted, disabled, or locked out, but it can be renamed. By default, the Administrator account is a member of the Administrators group, and it cannot be removed from that group.
S-1-5-*domain*-501	Guest	A user account for people who do not have individual accounts. Every computer has a local Guest account, and every domain has a domain Guest account. By default, Guest is a member of the Everyone and the Guests groups. The domain Guest account is also a member of the Domain Guests and Domain Users groups. Unlike Anonymous Logon, Guest is a real account, and it can be used to log on interactively. The Guest account does not require a password, but it can have one.
S-1-5-*domain*-502	krbtgt	A user account that is used by the Key Distribution Center (KDC) service. The account exists only on domain controllers.
S-1-5-*domain*-512	Domain Admins	A global group with members that are authorized to administer the domain. By default, the Domain Admins group is a member of the Administrators group on all computers that have joined the domain, including domain controllers. Domain Admins is the default owner of any object that is created in the domain's Active Directory by any member of the group. If members of the group create other objects, such as files, the default owner is the Administrators group.
S-1-5-*domain*-513	Domain Users	A global group that includes all users in a domain. When you create a new User object in Active Directory, the user is automatically added to this group.

SID	DISPLAY NAME	DESCRIPTION
S-1-5-*domain*-514	Domain Guests	A global group, which by default, has only one member: the domain's built-in Guest account.
S-1-5-*domain*-515	Domain Computers	A global group that includes all computers that have joined the domain, excluding domain controllers.
S-1-5-*domain*-516	Domain Controllers	A global group that includes all domain controllers in the domain. New domain controllers are added to this group automatically.
S-1-5-*domain*-517	Cert Publishers	A global group that includes all computers that host an enterprise certification authority. Cert Publishers are authorized to publish certificates for User objects in Active Directory.
S-1-5-*root domain*-518	Schema Admins	A group that exists only in the forest root domain. It is a universal group if the domain is in native mode, and it is a global group if the domain is in mixed mode. The Schema Admins group is authorized to make schema changes in Active Directory. By default, the only member of the group is the Administrator account for the forest root domain.
S-1-5-*root domain*-519	Enterprise Admins	A group that exists only in the forest root domain. It is a universal group if the domain is in native mode, and it is a global group if the domain is in mixed mode. The Enterprise Admins group is authorized to make changes to the forest infrastructure, such as adding child domains, configuring sites, authorizing DHCP servers, and installing enterprise certification authorities. By default, the only member of Enterprise Admins is the Administrator account for the forest root domain. The group is a default member of every Domain Admins group in the forest.

SID	DISPLAY NAME	DESCRIPTION
S-1-5-*domain*-520	Group Policy Creator Owners	A global group that is authorized to create new Group Policy Objects in Active Directory. By default, the only member of the group is Administrator. Objects that are created by members of Group Policy Creator Owners are owned by the individual user who creates them. In this way, the Group Policy Creator Owners group is unlike other administrative groups (such as Administrators and Domain Admins). Objects that are created by members of these groups are owned by the group rather than by the individual.
S-1-5-*domain*-553	RAS and IAS Servers	A local domain group. By default, this group has no members. Computers that are running the Routing and Remote Access service are added to the group automatically. Members of this group have access to certain properties of User objects, such as Read Account Restrictions, Read Logon Information, and Read Remote Access Information.
S-1-5-32-544	Administrators	A built-in group. After the initial installation of the operating system, the only member of the group is the Administrator account. When a computer joins a domain, the Domain Admins group is added to the Administrators group. When a server becomes a domain controller, the Enterprise Admins group also is added to the Administrators group.
S-1-5-32-545	Users	A built-in group. After the initial installation of the operating system, the only member is the Authenticated Users group.
S-1-5-32-546	Guests	A built-in group. By default, the only member is the Guest account. The Guests group allows occasional or one-time users to log on with limited privileges to a computer's built-in Guest account.
S-1-5-32-547	Power Users	A built-in group. By default, the group has no members. Power users can create local users and groups; modify and delete accounts that they have created; and remove users from the Power Users, Users, and Guests groups. Power users also can install programs; create, manage, and delete local printers; and create and delete file shares.

SID	DISPLAY NAME	DESCRIPTION
S-1-5-32-548	Account Operators	A built-in group that exists only on domain controllers. By default, the group has no members. By default, Account Operators have permission to create, modify, and delete accounts for users, groups, and computers in all containers and organizational units of Active Directory except the Builtin container and the Domain Controllers OU. Account Operators do not have permission to modify the Administrators and Domain Admins groups, nor do they have permission to modify the accounts for members of those groups.
S-1-5-32-549	Server Operators	Description: A built-in group that exists only on domain controllers. By default, the group has no members. Server Operators can log on to a server interactively; create and delete network shares; start and stop services; back up and restore files; format the hard disk of the computer; and shut down the computer.
S-1-5-32-550	Print Operators	A built-in group that exists only on domain controllers. By default, the only member is the Domain Users group. Print Operators can manage printers and document queues.
S-1-5-32-551	Backup Operators	A built-in group. By default, the group has no members. Backup Operators can back up and restore all files on a computer, regardless of the permissions that protect those files. Backup Operators also can log on to the computer and shut it down.
S-1-5-32-552	Replicators	A built-in group that is used by the File Replication service on domain controllers. By default, the group has no members. Do not add users to this group.
S-1-5-32-554	Builtin\Pre-Windows 2000 Compatible Access	An alias added by Windows 2000. A backward compatibility group that allows read access on all users and groups in the domain.
S-1-5-32-555	Builtin\Remote Desktop Users	An alias. Members in this group are granted the right to log on remotely.
S-1-5-32-556	Builtin\Network Configuration Operators	An alias. Members in this group can have some administrative privileges to manage configuration of networking features.

SID	DISPLAY NAME	DESCRIPTION
S-1-5-32-557	Builtin\Incoming Forest Trust Builders	An alias. Members of this group can create incoming, one-way trusts to this forest.
S-1-5-32-558	Builtin\Performance Monitor Users	An alias. Members of this group have remote access to monitor this computer.
S-1-5-32-559	Builtin\Performance Log Users	An alias. Members of this group have remote access to schedule logging of performance counters on this computer.
S-1-5-32-560	Builtin\Windows Authorization Access Group	An alias. Members of this group have access to the computed tokenGroupsGlobalAndUniversal attribute on User objects.
S-1-5-32-561	Builtin\Terminal Server License Servers	An alias. A group for Terminal Server License Servers. When Windows Server 2003 Service Pack 1 is installed, a new local group is created.
S-1-5-32-562	Builtin\Distributed COM Users	An alias. A group for COM to provide computer-wide access controls that govern access to all call, activation, or launch requests on the computer.
S-1-5-32-569	Builtin\Cryptographic Operators	A built-in local group. Members are authorized to perform cryptographic operations.
S-1-5-32-573	Builtin\Event Log Readers	A built-in local group. Members of this group can read event logs from local computer.
S-1-5-32-574	Builtin\Certificate Service DCOM Access	A built-in local group. Members of this group are allowed to connect to Certification Authorities in the enterprise.
S-1-5-32-575	Builtin\RDS Remote Access Servers	A built-in local group. Servers in this group enable users of RemoteApp programs and personal virtual desktops access to these resources. In Internet-facing deployments, these servers are typically deployed in an edge network. This group needs to be populated on servers running RD Connection Broker. RD Gateway servers and RD Web Access servers used in the deployment need to be in this group.

SID	DISPLAY NAME	DESCRIPTION
S-1-5-32-576	Builtin\RDS Endpoint Servers	A built-in local group. Servers in this group run virtual machines and host sessions where users RemoteApp programs and personal virtual desktops run. This group needs to be populated on servers running RD Connection Broker. RD Session Host servers and RD Virtualization Host servers used in the deployment need to be in this group.
S-1-5-32-577	Builtin\RDS Management Servers	A builtin local group. Servers in this group can perform routine administrative actions on servers running Remote Desktop Services. This group needs to be populated on all servers in a Remote Desktop Services deployment. The servers running the RDS Central Management service must be included in this group.
S-1-5-32-578	Builtin\Hyper-V Administrators	A built-in local group. Members of this group have complete and unrestricted access to all features of Hyper-V.
S-1-5-32-579	Builtin\Access Control Assistance Operators	A built-in local group. Members of this group can remotely query authorization attributes and permissions for resources on this computer.
S-1-5-32-580	Builtin\Remote Management Users	A built-in local group. Members of this group can access WMI resources over management protocols (such as WS-Management via the Windows Remote Management service). This applies only to WMI namespaces that grant access to the user.
S-1-5-64-10	NTLM Authentication	A SID that is used when the NTLM authentication package authenticated the client
S-1-5-64-14	SChannel Authentication	A SID that is used when the SChannel authentication package authenticated the client.
S-1-5-64-21	Digest Authentication	A SID that is used when the Digest authentication package authenticated the client.
S-1-5-80	NT Service	A SID that is used as an NT Service account prefix.

SID	DISPLAY NAME	DESCRIPTION
S-1-5-80-0	All Services	A group that includes all service processes that are configured on the system. Membership is controlled by the operating system. SID S-1-5-80-0 equals NT SERVICES\ALL SERVICES. This SID was introduced in Windows Server 2008 R2.
S-1-5-83-0	NT VIRTUAL MACHINE\Virtual Machines	A built-in group. The group is created when the Hyper-V role is installed. Membership in the group is maintained by the Hyper-V Management Service (VMMS). This group requires the **Create Symbolic Links** right (SeCreateSymbolicLinkPrivilege), and also the **Log on as a Service** right (SeServiceLogonRight).
S-1-16-0	Untrusted Mandatory Level	A SID that represents an untrusted integrity level.
S-1-16-4096	Low Mandatory Level	A SID that represents a low integrity level.
S-1-16-8192	Medium Mandatory Level	This SID represents a medium integrity level.
S-1-16-8448	Medium Plus Mandatory Level	A SID that represents a medium plus integrity level.
S-1-16-12288	High Mandatory Level	A SID that represents a high integrity level.
S-1-16-16384	System Mandatory Level	A SID that represents a system integrity level.
S-1-16-20480	Protected Process Mandatory Level	A SID that represents a protected-process integrity level.
S-1-16-28672	Secure Process Mandatory Level	A SID that represents a secure process integrity level.

The following RIDs are relative to each domain.

RID	DECIMAL VALUE	IDENTIFIES
DOMAIN_USER_RID_ADMIN	500	The administrative user account in a domain.
DOMAIN_USER_RID_GUEST	501	The guest-user account in a domain. Users who do not have an account can automatically sign in to this account.

RID	DECIMAL VALUE	IDENTIFIES
DOMAIN_GROUP_RID_USERS	513	A group that contains all user accounts in a domain. All users are automatically added to this group.
DOMAIN_GROUP_RID_GUESTS	514	The group Guest account in a domain.
DOMAIN_GROUP_RID_COMPUTERS	515	The Domain Computer group. All computers in the domain are members of this group.
DOMAIN_GROUP_RID_CONTROLLERS	516	The Domain Controller group. All domain controllers in the domain are members of this group.
DOMAIN_GROUP_RID_CERT_ADMINS	517	The certificate publishers' group. Computers running Active Directory Certificate Services are members of this group.
DOMAIN_GROUP_RID_SCHEMA_ADMINS	518	The schema administrators' group. Members of this group can modify the Active Directory schema.
DOMAIN_GROUP_RID_ENTERPRISE_ADMINS	519	The enterprise administrators' group. Members of this group have full access to all domains in the Active Directory forest. Enterprise administrators are responsible for forest-level operations such as adding or removing new domains.
DOMAIN_GROUP_RID_POLICY_ADMINS	520	The policy administrators' group.

The following table provides examples of domain-relative RIDs that are used to form well-known SIDs for local groups.

RID	DECIMAL VALUE	IDENTIFIES
DOMAIN_ALIAS_RID_ADMINS	544	Administrators of the domain.
DOMAIN_ALIAS_RID_USERS	545	All users in the domain.
DOMAIN_ALIAS_RID_GUESTS	546	Guests of the domain.
DOMAIN_ALIAS_RID_POWER_USERS	547	A user or a set of users who expect to treat a system as if it were their personal computer rather than as a workstation for multiple users.
DOMAIN_ALIAS_RID_BACKUP_OPS	551	A local group that is used to control the assignment of file backup-and-restore user rights.

RID	DECIMAL VALUE	IDENTIFIES
DOMAIN_ALIAS_RID_REPLICATOR	552	A local group that is responsible for copying security databases from the primary domain controller to the backup domain controllers. These accounts are used only by the system.
DOMAIN_ALIAS_RID_RAS_SERVERS	553	A local group that represents remote access and servers running Internet Authentication Service (IAS). This group permits access to various attributes of User objects.

Changes in security identifier's functionality

The following table describes changes in SID implementation in the Windows operating systems that are designated in the list.

CHANGE	OPERATING SYSTEM VERSION	DESCRIPTION AND RESOURCES
Most of the operating system files are owned by the TrustedInstaller security identifier (SID)	Windows Server 2008, Windows Vista	The purpose of this change is to prevent a process that is running as an administrator or under the LocalSystem account from automatically replacing the operating system files.
Restricted SID checks are implemented	Windows Server 2008, Windows Vista	When restricting SIDs are present, Windows performs two access checks. The first is the normal access check, and the second is the same access check against the restricting SIDs in the token. Both access checks must pass to allow the process to access the object.

Capability SIDs

Capability Security Identifiers (SIDs) are used to uniquely and immutably identify capabilities. Capabilities represent an unforgeable token of authority that grants access to resources (Examples: documents, camera, locations etc...) to Universal Windows Applications. An App that "has" a capability is granted access to the resource the capability is associated with, and one that "does not have" a capability is denied access to the resource.

All Capability SIDs that the operating system is aware of are stored in the Windows Registry in the path `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\SecurityManager\CapabilityClasses\AllCachedCapabilities'. Any Capability SID added to Windows by first or third-party applications will be added to this location.

Examples of registry keys taken from Windows 10, version 1909, 64-bit Enterprise edition

You may see the following registry keys under AllCachedCapabilities:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\SecurityManager\CapabilityClasses\AllCachedCapabilities\capabilityClass_DevUnlock

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\SecurityManager\CapabilityClasses\AllCachedCapabilities\capabilityClass_DevUnlock_Internal
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\SecurityManager\CapabilityClasses\AllCachedCapabilities\capabilityClass_Enterprise
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\SecurityManager\CapabilityClasses\AllCachedCapabilities\capabilityClass_General
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\SecurityManager\CapabilityClasses\AllCachedCapabilities\capabilityClass_Restricted
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\SecurityManager\CapabilityClasses\AllCachedCapabilities\capabilityClass_Windows

All Capability SIDs are prefixed by S-1-15-3

See also

- Access Control Overview

Security Principals

3/5/2021 • 10 minutes to read • Edit Online

Applies to

- Windows 10
- Windows Server 2016

This reference topic for the IT professional describes security principals in regards to Windows accounts and security groups, in addition to security technologies that are related to security principals.

What are security principals?

Security principals are any entity that can be authenticated by the operating system, such as a user account, a computer account, or a thread or process that runs in the security context of a user or computer account, or the security groups for these accounts. Security principals have long been a foundation for controlling access to securable resources on Windows computers. Each security principal is represented in the operating system by a unique security identifier (SID).

The following content applies to the versions of Windows that are designated in the **Applies To** list at the beginning of this topic.

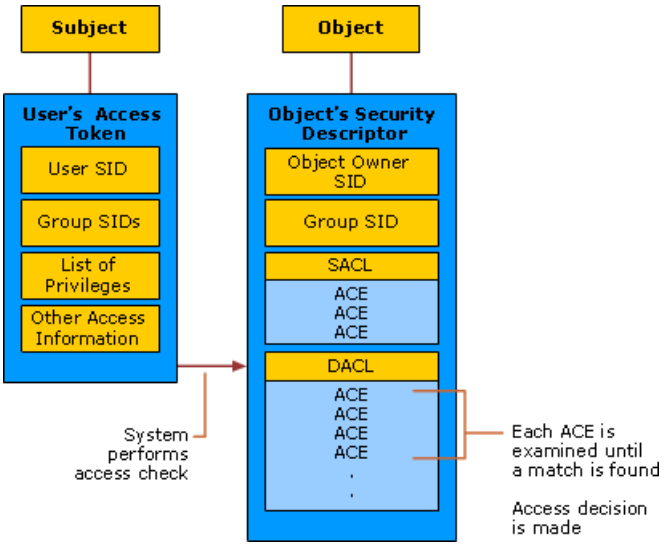
How security principals work

Security principals that are created in an Active Directory domain are Active Directory objects, which can be used to manage access to domain resources. Each security principal is assigned a unique identifier, which it retains for its entire lifetime. Local user accounts and security groups are created on a local computer, and they can be used to manage access to resources on that computer. Local user accounts and security groups are managed by the Security Accounts Manager (SAM) on the local computer.

Authorization and access control components

The following diagram illustrates the Windows authorization and access control process. In this diagram, the subject (a process that is initiated by a user) attempts to access an object, such as a shared folder. The information in the user's access token is compared to the access control entries (ACEs) in the object's security descriptor, and the access decision is made. The SIDs of security principals are used in the user's access token and in the ACEs in the object's security descriptor.

Authorization and access control process



Security principals are closely related to the following components and technologies:

- Security identifiers
- Access tokens
- Security descriptors and access control lists
- Permissions

Security identifiers

Security identifiers (SIDs) provide a fundamental building block of the Windows security model. They work with specific components of the authorization and access control technologies in the security infrastructure of the Windows Server operating systems. This helps protect access to network resources and provides a more secure computing environment.

A SID is a value of variable length that is used to uniquely identify a security principal that represents any entity that can be authenticated by the system. These entities include a user account, a computer account, or a thread or process that runs in the security context of a user or computer account. Each security principal is automatically assigned a SID when it is created. The SID is stored in a security database. When a SID is used as the unique identifier for a user or group, it can never be used to identify another user or group.

Each time a user signs in, the system creates an access token for that user. The access token contains the user's SID, user rights, and the SIDs for groups that the user belongs to. This token provides the security context for whatever actions the user performs on that computer.

In addition to the uniquely created, domain-specific SIDs that are assigned to specific users and groups, there are well-known SIDs that identify generic groups and generic users. For example, the Everyone and the World SIDs identify groups that includes all users. Well-known SIDs have values that remain constant across all operating systems.

Access tokens

An access token is a protected object that contains information about the identity and user rights that are associated with a user account.

When a user signs in interactively or tries to make a network connection to a computer running Windows, the sign-in process authenticates the user's credentials. If authentication is successful, the process returns a SID for the user and a list of SIDs for the user's security groups. The Local Security Authority (LSA) on the computer uses this information to create an access token (in this case, the primary access token). This includes the SIDs that are returned by the sign-in process and a list of user rights that are assigned by the local security policy to the user and to the user's security groups.

After the LSA creates the primary access token, a copy of the access token is attached to every thread and process that executes on the user's behalf. Whenever a thread or process interacts with a securable object or tries to perform a system task that requires user rights, the operating system checks the access token that is associated with the thread to determine the level of authorization.

There are two kinds of access tokens, primary and impersonation. Every process has a primary token that describes the security context of the user account that is associated with the process. A primary access token is typically assigned to a process to represent the default security information for that process. Impersonation tokens, on the other hand, are usually used for client and server scenarios. Impersonation tokens enable a thread to run in a security context that differs from the security context of the process that owns the thread.

Security descriptors and access control lists

A security descriptor is a data structure that is associated with each securable object. All objects in Active Directory and all securable objects on a local computer or on the network have security descriptors to help control access to the objects. Security descriptors include information about who owns an object, who can access it and in what way, and what types of access are audited. Security descriptors contain the access control list (ACL) of an object, which includes all of the security permissions that apply to that object. An object's security descriptor can contain two types of ACLs:

- A discretionary access control list (DACL), which identifies the users and groups who are allowed or denied access
- A system access control list (SACL), which controls how access is audited

You can use this access control model to individually secure objects and attributes such as files and folders, Active Directory objects, registry keys, printers, devices, ports, services, processes, and threads. Because of this individual control, you can adjust the security of objects to meet the needs of your organization, delegate authority over objects or attributes, and create custom objects or attributes that require unique security protections to be defined.

Permissions

Permissions enable the owner of each securable object, such as a file, Active Directory object, or registry key, to control who can perform an operation or a set of operations on the object or object property. Permissions are expressed in the security architecture as access control entries (ACEs). Because access to an object is at the discretion of the object's owner, the type of access control that is used in Windows is called discretionary access control.

Permissions are different from user rights in that permissions are attached to objects, and user rights apply to user accounts. Administrators can assign user rights to groups or users. These rights authorize users to perform specific actions, such as signing in to a system interactively or backing up files and directories.

On computers, user rights enable administrators to control who has the authority to perform operations that affect an entire computer, rather than a particular object. Administrators assign user rights to individual users or groups as part of the security settings for the computer. Although user rights can be managed centrally through Group Policy, they are applied locally. Users can (and usually do) have different user rights on different computers.

For information about which user rights are available and how they can be implemented, see User Rights Assignment.

Security context in authentication

A user account enables a user to sign in to computers, networks, and domains with an identity that can be authenticated by the computer, network, or domain.

In Windows, any user, service, group, or computer that can initiate action is a security principal. Security principals have accounts, which can be local to a computer or domain-based. For example, domain-joined

Windows client computers can participate in a network domain by communicating with a domain controller, even when no user is signed in.

To initiate communications, the computer must have an active account in the domain. Before accepting communications from the computer, the Local Security Authority on the domain controller authenticates the computer's identity and then defines the computer's security context just as it would for a user's security principal.

This security context defines the identity and capabilities of a user or service on a particular computer, or of a user, service, group or computer on a network. For example, it defines the resources (such as a file share or printer) that can be accessed and the actions (such as Read, Write, or Modify) that can be performed by a user, service, or computer on that resource.

The security context of a user or computer can vary from one computer to another, such as when a user authenticates to a server or a workstation other than the user's primary workstation. It can also vary from one session to another, such as when an administrator modifies the user's rights and permissions. In addition, the security context is usually different when a user or computer is operating on a stand-alone basis, in a mixed network domain, or as part of an Active Directory domain.

Accounts and security groups

Accounts and security groups that are created in an Active Directory domain are stored in the Active Directory database and managed by using Active Directory tools. These security principals are directory objects, and they can be used to manage access to domain resources.

Local user accounts and security groups are created on a local computer, and they can be used to manage access to resources on that computer. Local user accounts and security groups are stored in and managed by the Security Accounts Manager (SAM) on the local computer.

User accounts

A user account uniquely identifies a person who is using a computer system. The account signals the system to enforce the appropriate authorization to allow or deny that user access to resources. User accounts can be created in Active Directory and on local computers, and administrators use them to:

- Represent, identify, and authenticate the identity of a user. A user account enables a user to sign in to computers, networks, and domains with a unique identifier that can be authenticated by the computer, network, or domain.
- Authorize (grant or deny) access to resources. After a user has been authenticated, the user is authorized access to resources based on the permissions that are assigned to that user for the resource.
- Audit the actions that are carried out on a user account.

Windows and the Windows Server operating systems have built-in user accounts, or you can create user accounts to meet the requirements of your organization.

Security groups

A security group is a collection of user accounts, computer accounts, and other groups of accounts that can be managed as a single unit from a security perspective. In Windows operating systems, there are several built-in security groups that are preconfigured with the appropriate rights and permissions for performing specific tasks. Additionally, you can (and, typically, will) create a security group for each unique combination of security requirements that applies to multiple users in your organization.

Groups can be Active Directory-based or local to a particular computer:

- Active Directory security groups are used to manage rights and permissions to domain resources.

- Local groups exist in the SAM database on local computers (on all Windows-based computers) except domain controllers. You use local groups to manage rights and permissions only to resources on the local computer.

By using security groups to manage access control, you can:

- Simplify administration. You can assign a common set of rights, a common set of permissions, or both to many accounts at one time, rather than assigning them to each account individually. Also, when users transfer jobs or leave the organization, permissions are not tied to their user accounts, making permission reassignment or removal easier.
- Implement a role-based access-control model. You can use this model to grant permissions by using groups with different scopes for appropriate purposes. Scopes that are available in Windows include local, global, domain local, and universal.
- Minimize the size of access control lists (ACLs) and speed security checking. A security group has its own SID; therefore, the group SID can be used to specify permissions for a resource. In an environment with more than a few thousand users, if the SIDs of individual user accounts are used to specify access to a resource, the ACL of that resource can become unmanageably large, and the time that is needed for the system to check permissions to the resource can become unacceptable.

For descriptions and settings information about the domain security groups that are defined in Active Directory, see Active Directory Security Groups.

For descriptions and settings information about the Special Identities group, see Special Identities.

See also

- Access Control Overview

Local Accounts

3/26/2021 • 20 minutes to read • Edit Online

Applies to

- Windows 10
- Windows Server 2019
- Windows Server 2016

This reference topic for IT professionals describes the default local user accounts for servers, including how to manage these built-in accounts on a member or standalone server.

About local user accounts

Local user accounts are stored locally on the server. These accounts can be assigned rights and permissions on a particular server, but on that server only. Local user accounts are security principals that are used to secure and manage access to the resources on a standalone or member server for services or users.

This topic describes the following:

- Default local user accounts
  - Administrator account
  - Guest Account
  - HelpAssistant account (installed by using a Remote Assistance session)
  - DefaultAccount
- Default local system accounts
- How to manage local accounts
  - Restrict and protect local accounts with administrative rights
  - Enforce local account restrictions for remote access
  - Deny network logon to all local Administrator accounts
  - Create unique passwords for local accounts with administrative rights

For information about security principals, see Security Principals.

Default local user accounts

The default local user accounts are built-in accounts that are created automatically when you install Windows.

After Windows is installed, the default local user accounts cannot be removed or deleted. In addition, default local user accounts do not provide access to network resources.

Default local user accounts are used to manage access to the local server's resources based on the rights and permissions that are assigned to the account. The default local user accounts, and the local user accounts that you create, are located in the Users folder. The Users folder is located in the Local Users and Groups folder in the local Computer Management Microsoft Management Console (MMC). Computer Management is a collection of administrative tools that you can use to manage a single local or remote computer. For more information, see

How to manage local accounts later in this topic.

Default local user accounts are described in the following sections.

Administrator account

The default local Administrator account is a user account for the system administrator. Every computer has an Administrator account (SID S-1-5-*domain*-500, display name Administrator). The Administrator account is the first account that is created during the Windows installation.

The Administrator account has full control of the files, directories, services, and other resources on the local computer. The Administrator account can create other local users, assign user rights, and assign permissions. The Administrator account can take control of local resources at any time simply by changing the user rights and permissions.

The default Administrator account cannot be deleted or locked out, but it can be renamed or disabled.

In Windows 10 and Windows Server 2016, Windows setup disables the built-in Administrator account and creates another local account that is a member of the Administrators group. Members of the Administrators groups can run apps with elevated permissions without using the **Run as Administrator** option. Fast User Switching is more secure than using Runas or different-user elevation.

Account group membership

By default, the Administrator account is installed as a member of the Administrators group on the server. It is a best practice to limit the number of users in the Administrators group because members of the Administrators group on a local server have Full Control permissions on that computer.

The Administrator account cannot be deleted or removed from the Administrators group, but it can be renamed.

Security considerations

Because the Administrator account is known to exist on many versions of the Windows operating system, it is a best practice to disable the Administrator account when possible to make it more difficult for malicious users to gain access to the server or client computer.

You can rename the Administrator account. However, a renamed Administrator account continues to use the same automatically assigned security identifier (SID), which can be discovered by malicious users. For more information about how to rename or disable a user account, see Disable or activate a local user account and Rename a local user account.

As a security best practice, use your local (non-Administrator) account to sign in and then use **Run as administrator** to accomplish tasks that require a higher level of rights than a standard user account. Do not use the Administrator account to sign in to your computer unless it is entirely necessary. For more information, see Run a program with administrative credentials.

In comparison, on the Windows client operating system, a user with a local user account that has Administrator rights is considered the system administrator of the client computer. The first local user account that is created during installation is placed in the local Administrators group. However, when multiple users run as local administrators, the IT staff has no control over these users or their client computers.

In this case, Group Policy can be used to enable secure settings that can control the use of the local Administrators group automatically on every server or client computer. For more information about Group Policy, see Group Policy Overview.

Note Blank passwords are not allowed in the versions designated in the **Applies To** list at the beginning of this topic.

Important Even when the Administrator account has been disabled, it can still be used to gain access to a computer by using safe mode. In the Recovery Console or in safe mode, the Administrator account is

automatically enabled. When normal operations are resumed, it is disabled.

Guest account

The Guest account is disabled by default on installation. The Guest account lets occasional or one-time users, who do not have an account on the computer, temporarily sign in to the local server or client computer with limited user rights. By default, the Guest account has a blank password. Because the Guest account can provide anonymous access, it is a security risk. For this reason, it is a best practice to leave the Guest account disabled, unless its use is entirely necessary.

Account group membership

By default, the Guest account is the only member of the default Guests group (SID S-1-5-32-546), which lets a user sign in to a server. On occasion, an administrator who is a member of the Administrators group can set up a user with a Guest account on one or more computers.

Security considerations

When enabling the Guest account, only grant limited rights and permissions. For security reasons, the Guest account should not be used over the network and made accessible to other computers.

In addition, the guest user in the Guest account should not be able to view the event logs. After the Guest account is enabled, it is a best practice to monitor the Guest account frequently to ensure that other users cannot use services and other resources, such as resources that were unintentionally left available by a previous user.

HelpAssistant account (installed with a Remote Assistance session)

The HelpAssistant account is a default local account that is enabled when a Remote Assistance session is run. This account is automatically disabled when no Remote Assistance requests are pending.

HelpAssistant is the primary account that is used to establish a Remote Assistance session. The Remote Assistance session is used to connect to another computer running the Windows operating system, and it is initiated by invitation. For solicited remote assistance, a user sends an invitation from their computer, through e-mail or as a file, to a person who can provide assistance. After the user's invitation for a Remote Assistance session is accepted, the default HelpAssistant account is automatically created to give the person who provides assistance limited access to the computer. The HelpAssistant account is managed by the Remote Desktop Help Session Manager service.

Security considerations

The SIDs that pertain to the default HelpAssistant account include:

- SID: S-1-5-<domain>-13, display name Terminal Server User. This group includes all users who sign in to a server with Remote Desktop Services enabled. Note that, in Windows Server 2008, Remote Desktop Services are called Terminal Services.
- SID: S-1-5-<domain>-14, display name Remote Interactive Logon. This group includes all users who connect to the computer by using a remote desktop connection. This group is a subset of the Interactive group. Access tokens that contain the Remote Interactive Logon SID also contain the Interactive SID.

For the Windows Server operating system, Remote Assistance is an optional component that is not installed by default. You must install Remote Assistance before it can be used.

For details about the HelpAssistant account attributes, see the following table.

HelpAssistant account attributes

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-<domain>-13 (Terminal Server User), S-1-5-<domain>-14 (Remote Interactive Logon)
Type	User
Default container	CN=Users, DC=<domain>, DC=
Default members	None
Default member of	Domain Guests Guests
Protected by ADMINSDHOLDER?	No
Safe to move out of default container?	Can be moved out, but we do not recommend it.
Safe to delegate management of this group to non-Service admins?	No

DefaultAccount

The DefaultAccount, also known as the Default System Managed Account (DSMA), is a built-in account introduced in Windows 10 version 1607 and Windows Server 2016. The DSMA is a well-known user account type. It is a user neutral account that can be used to run processes that are either multi-user aware or user-agnostic. The DSMA is disabled by default on the desktop SKUs (full windows SKUs) and WS 2016 with the Desktop.

The DSMA has a well-known RID of 503. The security identifier (SID) of the DSMA will thus have a well-known SID in the following format: S-1-5-21-<ComputerIdentifier>-503

The DSMA is a member of the well-known group **System Managed Accounts Group**, which has a well-known SID of S-1-5-32-581.

The DSMA alias can be granted access to resources during offline staging even before the account itself has been created. The account and the group are created during first boot of the machine within the Security Accounts Manager (SAM).

How Windows uses the DefaultAccount

From a permission perspective, the DefaultAccount is a standard user account. The DefaultAccount is needed to run multi-user-manifested-apps (MUMA apps). MUMA apps run all the time and react to users signing in and signing out of the devices. Unlike Windows Desktop where apps run in context of the user and get terminated when the user signs off, MUMA apps run by using the DSMA.

MUMA apps are functional in shared session SKUs such as Xbox. For example, Xbox shell is a MUMA app. Today, Xbox automatically signs in as Guest account and all apps run in this context. All the apps are multi-user-aware and respond to events fired by user manager. The apps run as the Guest account.

Similarly, Phone auto logs in as a "DefApps" account which is akin to the standard user account in Windows but

with a few extra privileges. Brokers, some services and apps run as this account.

In the converged user model, the multi-user-aware apps and multi-user-aware brokers will need to run in a context different from that of the users. For this purpose, the system creates DSMA.

How the DefaultAccount gets created on domain controllers

If the domain was created with domain controllers that run Windows Server 2016, the DefaultAccount will exist on all domain controllers in the domain. If the domain was created with domain controllers that run an earlier version of Windows Server, the DefaultAccount will be created after the PDC Emulator role is transferred to a domain controller that runs Windows Server 2016. The DefaultAccount will then be replicated to all other domain controllers in the domain.

Recommendations for managing the Default Account (DSMA)

Microsoft does not recommend changing the default configuration, where the account is disabled. There is no security risk with having the account in the disabled state. Changing the default configuration could hinder future scenarios that rely on this account.

Default local system accounts

SYSTEM

The SYSTEM account is used by the operating system and by services that run under Windows. There are many services and processes in the Windows operating system that need the capability to sign in internally, such as during a Windows installation. The SYSTEM account was designed for that purpose, and Windows manages the SYSTEM account's user rights. It is an internal account that does not show up in User Manager, and it cannot be added to any groups.

On the other hand, the SYSTEM account does appear on an NTFS file system volume in File Manager in the **Permissions** portion of the **Security** menu. By default, the SYSTEM account is granted Full Control permissions to all files on an NTFS volume. Here the SYSTEM account has the same functional rights and permissions as the Administrator account.

Note To grant the account Administrators group file permissions does not implicitly give permission to the SYSTEM account. The SYSTEM account's permissions can be removed from a file, but we do not recommend removing them.

NETWORK SERVICE

The NETWORK SERVICE account is a predefined local account used by the service control manager (SCM). A service that runs in the context of the NETWORK SERVICE account presents the computer's credentials to remote servers. For more information, see NetworkService Account.

LOCAL SERVICE

The LOCAL SERVICE account is a predefined local account used by the service control manager. It has minimum privileges on the local computer and presents anonymous credentials on the network. For more information, see LocalService Account.

How to manage local user accounts

The default local user accounts, and the local user accounts that you create, are located in the Users folder. The Users folder is located in Local Users and Groups. For more information about creating and managing local user accounts, see Manage Local Users.

You can use Local Users and Groups to assign rights and permissions on the local server, and that server only, to limit the ability of local users and groups to perform certain actions. A right authorizes a user to perform certain actions on a server, such as backing up files and folders or shutting down a server. An access permission is a rule that is associated with an object, usually a file, folder, or printer. It regulates which users can have access to an

object on the server and in what manner.

You cannot use Local Users and Groups on a domain controller. However, you can use Local Users and Groups on a domain controller to target remote computers that are not domain controllers on the network.

Note You use Active Directory Users and Computers to manage users and groups in Active Directory.

You can also manage local users by using NET.EXE USER and manage local groups by using NET.EXE LOCALGROUP, or by using a variety of PowerShell cmdlets and other scripting technologies.

Restrict and protect local accounts with administrative rights

An administrator can use a number of approaches to prevent malicious users from using stolen credentials, such as a stolen password or password hash, for a local account on one computer from being used to authenticate on another computer with administrative rights; this is also called "lateral movement".

The simplest approach is to sign in to your computer with a standard user account, instead of using the Administrator account for tasks, for example, to browse the Internet, send email, or use a word processor. When you want to perform an administrative task, for example, to install a new program or to change a setting that affects other users, you don't have to switch to an Administrator account. You can use User Account Control (UAC) to prompt you for permission or an administrator password before performing the task, as described in the next section.

The other approaches that can be used to restrict and protect user accounts with administrative rights include:

- Enforce local account restrictions for remote access.
- Deny network logon to all local Administrator accounts.
- Create unique passwords for local accounts with administrative rights.

Each of these approaches is described in the following sections.

Note These approaches do not apply if all administrative local accounts are disabled.

Enforce local account restrictions for remote access

The User Account Control (UAC) is a security feature in Windows that has been in use in Windows Server 2008 and in Windows Vista, and the operating systems to which the **Applies To** list refers. UAC enables you to stay in control of your computer by informing you when a program makes a change that requires administrator-level permission. UAC works by adjusting the permission level of your user account. By default, UAC is set to notify you when applications try to make changes to your computer, but you can change how often UAC notifies you.

UAC makes it possible for an account with administrative rights to be treated as a standard user non-administrator account until full rights, also called elevation, is requested and approved. For example, UAC lets an administrator enter credentials during a non-administrator's user session to perform occasional administrative tasks without having to switch users, sign out, or use the **Run as** command.

In addition, UAC can require administrators to specifically approve applications that make system-wide changes before those applications are granted permission to run, even in the administrator's user session.

For example, a default feature of UAC is shown when a local account signs in from a remote computer by using Network logon (for example, by using NET.EXE USE). In this instance, it is issued a standard user token with no administrative rights, but without the ability to request or receive elevation. Consequently, local accounts that sign in by using Network logon cannot access administrative shares such as C$, or ADMIN$, or perform any remote administration.

For more information about UAC, see User Account Control.

The following table shows the Group Policy and registry settings that are used to enforce local account restrictions for remote access.

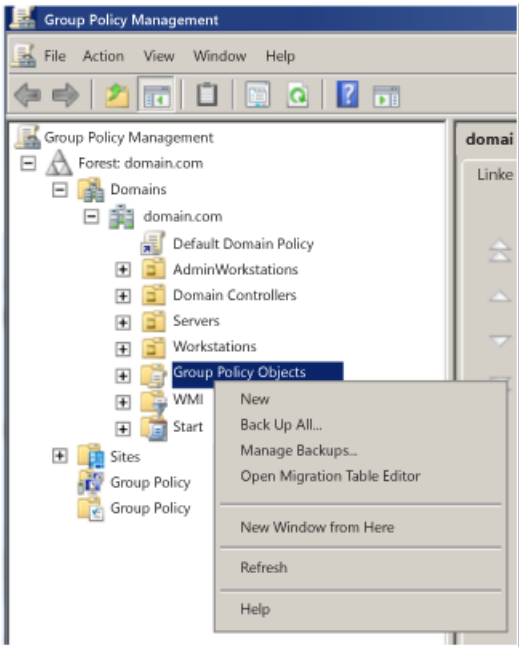
No.	Setting	Detailed Description
	Policy location	Computer Configuration\Windows Settings\Security Settings\Local Policies\Security Options
1	Policy name	User Account Control: Run all administrators in Admin Approval Mode
	Policy setting	Enabled
2	Policy location	Computer Configuration\Windows Settings\Security Settings\Local Policies\Security Options
	Policy name	User Account Control: Run all administrators in Admin Approval Mode
	Policy setting	Enabled
3	Registry key	**HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System**
	Registry value name	LocalAccountTokenFilterPolicy
	Registry value type	DWORD
	Registry value data	0

NOTE

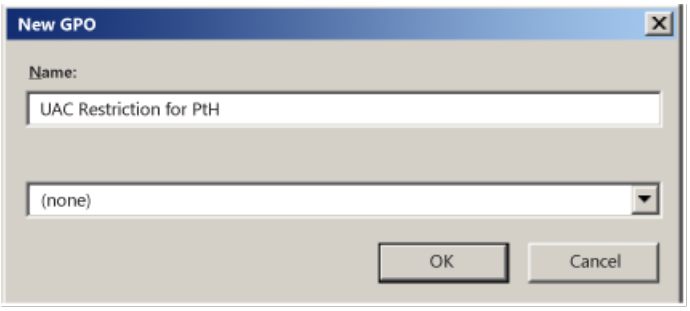
You can also enforce the default for LocalAccountTokenFilterPolicy by using the custom ADMX in Security Templates.

To enforce local account restrictions for remote access

1. Start the **Group Policy Management** Console (GPMC).
2. In the console tree, expand <*Forest*>\Domains\<*Domain*>, and then **Group Policy Objects** where *forest* is the name of the forest, and *domain* is the name of the domain where you want to set the Group Policy Object (GPO).
3. In the console tree, right-click **Group Policy Objects**, and > **New**.



4. In the **New GPO** dialog box, type <**gpo_name**>, and > **OK** where *gpo_name* is the name of the new GPO. The GPO name indicates that the GPO is used to restrict local administrator rights from being carried over to another computer.



5. In the details pane, right-click <**gpo_name**>, and > **Edit**.

6. Ensure that UAC is enabled and that UAC restrictions apply to the default Administrator account by doing the following:

   a. Navigate to the Computer Configuration\Windows Settings\Security Settings\Local Policies\, and > **Security Options**.

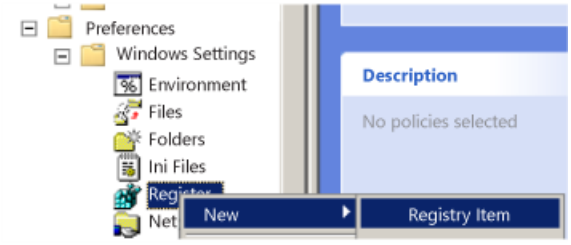
   b. Double-click **User Account Control: Run all administrators in Admin Approval Mode** > **Enabled** > **OK**.

   c. Double-click **User Account Control: Admin Approval Mode for the Built-in Administrator account** > **Enabled** > **OK**.

7. Ensure that the local account restrictions are applied to network interfaces by doing the following:

   a. Navigate to Computer Configuration\Preferences and Windows Settings, and > **Registry**.

   b. Right-click **Registry**, and > **New** > **Registry Item**.



c. In the **New Registry Properties** dialog box, on the **General** tab, change the setting in the **Action** box to **Replace**.

d. Ensure that the **Hive** box is set to **HKEY_LOCAL_MACHINE**.

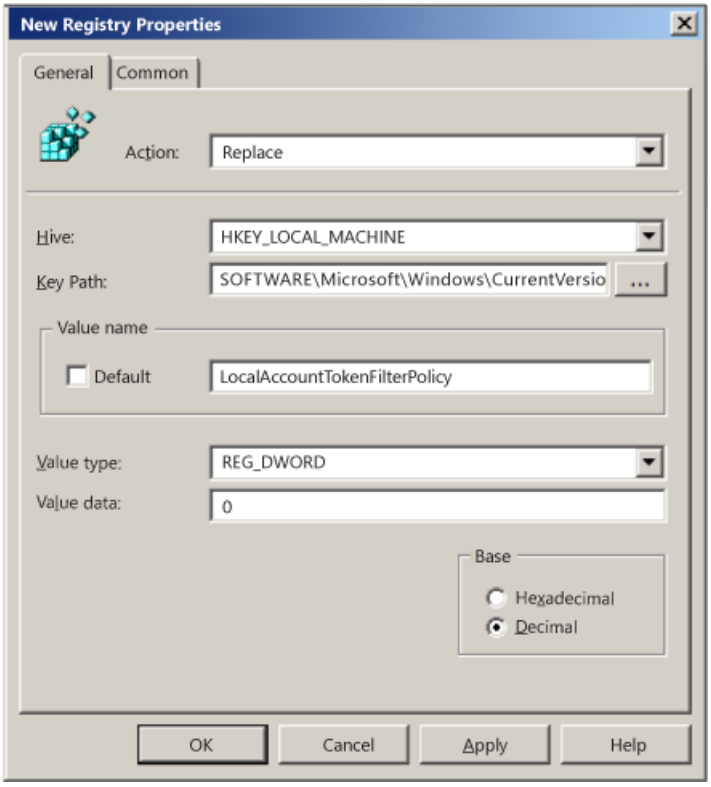
e. Click (...), browse to the following location for **Key Path** > **Select** for: **SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System**.

f. In the **Value name** area, type **LocalAccountTokenFilterPolicy**.

g. In the **Value type** box, from the drop-down list, select **REG_DWORD** to change the value.

h. In the **Value data** box, ensure that the value is set to **0**.

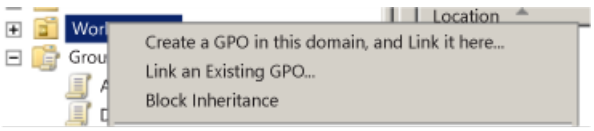
i. Verify this configuration, and > **OK**.



8. Link the GPO to the first **Workstations** organizational unit (OU) by doing the following:

a. Navigate to the <*Forest*>\Domains\<*Domain*>\OU path.

b. Right-click the **Workstations** OU, and > **Link an existing GPO**.



c. Select the GPO that you just created, and > **OK**.

9. Test the functionality of enterprise applications on the workstations in that first OU and resolve any issues caused by the new policy.

10. Create links to all other OUs that contain workstations.
11. Create links to all other OUs that contain servers.

Deny network logon to all local Administrator accounts

Denying local accounts the ability to perform network logons can help prevent a local account password hash from being reused in a malicious attack. This procedure helps to prevent lateral movement by ensuring that the credentials for local accounts that are stolen from a compromised operating system cannot be used to compromise additional computers that use the same credentials.

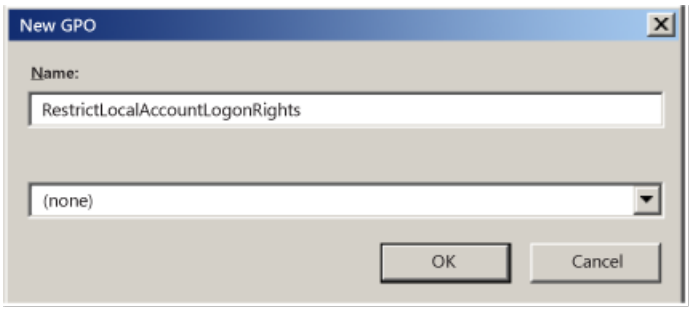
Note In order to perform this procedure, you must first identify the name of the local, default Administrator account, which might not be the default user name "Administrator", and any other accounts that are members of the local Administrators group.

The following table shows the Group Policy settings that are used to deny network logon for all local Administrator accounts.

No.	Setting	Detailed Description
	Policy location	Computer Configuration\Windows Settings\Security Settings\Local Policies\User Rights Assignment
1	Policy name	Deny access to this computer from the network
	Policy setting	Local account and member of Administrators group
2	Policy location	Computer Configuration\Windows Settings\Security Settings\Local Policies\User Rights Assignment
	Policy name	Deny log on through Remote Desktop Services
	Policy setting	Local account and member of Administrators group

To deny network logon to all local administrator accounts

1. Start the **Group Policy Management** Console (GPMC).
2. In the console tree, expand <*Forest*>\Domains\<*Domain*>, and then **Group Policy Objects**, where *forest* is the name of the forest, and *domain* is the name of the domain where you want to set the Group Policy Object (GPO).
3. In the console tree, right-click **Group Policy Objects**, and > **New**.
4. In the **New GPO** dialog box, type <**gpo_name**>, and then > **OK** where *gpo_name* is the name of the new GPO indicates that it is being used to restrict the local administrative accounts from interactively signing in to the computer.



5. In the details pane, right-click <**gpo_name**>, and > **Edit**.

6. Configure the user rights to deny network logons for administrative local accounts as follows:

   a. Navigate to the Computer Configuration\Windows Settings\Security Settings\, and > **User Rights Assignment**.

   b. Double-click **Deny access to this computer from the network**.

   c. Click **Add User or Group**, type **Local account and member of Administrators group**, and > **OK**.

7. Configure the user rights to deny Remote Desktop (Remote Interactive) logons for administrative local accounts as follows:

   a. Navigate to Computer Configuration\Policies\Windows Settings and Local Policies, and then click **User Rights Assignment**.

   b. Double-click **Deny log on through Remote Desktop Services**.

   c. Click **Add User or Group**, type **Local account and member of Administrators group**, and > **OK**.

8. Link the GPO to the first **Workstations** OU as follows:

   a. Navigate to the <*Forest*>\Domains\<*Domain*>\OU path.

   b. Right-click the **Workstations** OU, and > **Link an existing GPO**.

   c. Select the GPO that you just created, and > **OK**.

9. Test the functionality of enterprise applications on the workstations in that first OU and resolve any issues caused by the new policy.

10. Create links to all other OUs that contain workstations.

11. Create links to all other OUs that contain servers.

    Note You might have to create a separate GPO if the user name of the default Administrator account is different on workstations and servers.

Create unique passwords for local accounts with administrative rights

Passwords should be unique per individual account. While this is generally true for individual user accounts, many enterprises have identical passwords for common local accounts, such as the default Administrator account. This also occurs when the same passwords are used for local accounts during operating system deployments.

Passwords that are left unchanged or changed synchronously to keep them identical add a significant risk for organizations. Randomizing the passwords mitigates "pass-the-hash" attacks by using different passwords for

local accounts, which hampers the ability of malicious users to use password hashes of those accounts to compromise other computers.

Passwords can be randomized by:

- Purchasing and implementing an enterprise tool to accomplish this task. These tools are commonly referred to as "privileged password management" tools.
- Configuring Local Administrator Password Solution (LAPS) to accomplish this task.
- Creating and implementing a custom script or solution to randomize local account passwords.

See also

The following resources provide additional information about technologies that are related to local accounts.

- Security Principals
- Security Identifiers
- Access Control Overview

Active Directory Accounts

3/26/2021 • 34 minutes to read • Edit Online

Applies to

- Windows Server 2016

Windows Server operating systems are installed with default local accounts. In addition, you can create user accounts to meet the requirements of your organization. This reference topic for the IT professional describes the Windows Server default local accounts that are stored locally on the domain controller and are used in Active Directory.

This reference topic does not describe default local user accounts for a member or standalone server or for a Windows client. For more information, see Local Accounts.

About this topic

This topic describes the following:

- Default local accounts in Active Directory
  - Administrator account
  - Guest account
  - HelpAssistant account (installed with a Remote Assistance session)
  - KRBTGT account
- Settings for default local accounts in Active Directory
- Manage default local accounts in Active Directory
- Restrict and protect sensitive domain accounts
  - Separate administrator accounts from user accounts
  - Create dedicated workstation hosts without Internet and email access
  - Restrict administrator logon access to servers and workstations
  - Disable the account delegation right for administrator accounts
- Secure and manage domain controllers

Default local accounts in Active Directory

Default local accounts are built-in accounts that are created automatically when a Windows Server domain controller is installed and the domain is created. These default local accounts have counterparts in Active Directory. These accounts also have domain-wide access and are completely separate from the default local user accounts for a member or standalone server.

You can assign rights and permissions to default local accounts on a particular domain controller, and only on that domain controller. These accounts are local to the domain. After the default local accounts are installed, they are stored in the Users container in Active Directory Users and Computers. It is a best practice to keep the default local accounts in the User container and not attempt to move these accounts, for example, to a different

organizational unit (OU).

The default local accounts in the Users container include: Administrator, Guest, and KRBTGT. The HelpAssistant account is installed when a Remote Assistance session is established. The following sections describe the default local accounts and their use in Active Directory.

Primarily, default local accounts do the following:

- Let the domain represent, identify, and authenticate the identity of the user that is assigned to the account by using unique credentials (user name and password). It is a best practice to assign each user to a single account to ensure maximum security. Multiple users are not allowed to share one account. A user account lets a user sign in to computers, networks, and domains with a unique identifier that can be authenticated by the computer, network, or domain.
- Authorize (grant or deny) access to resources. After a user's credentials have been authenticated, the user is authorized to access the network and domain resources based on the user's explicitly assigned rights on the resource.
- Audit the actions that are carried out on a user account.

In Active Directory, default local accounts are used by administrators to manage domain and member servers directly and from dedicated administrative workstations. Active Directory accounts provide access to network resources. Active Directory User accounts and Computer accounts can represent a physical entity, such as a computer or person, or act as dedicated service accounts for some applications.

Each default local account is automatically assigned to a security group that is preconfigured with the appropriate rights and permissions to perform specific tasks. Active Directory security groups collect user accounts, computer accounts, and other groups into manageable units. For more information, see Active Directory Security Groups.

On an Active Directory domain controller, each default local account is referred to as a security principal. A security principal is a directory object that is used to secure and manage Active Directory services that provide access to domain controller resources. A security principal includes objects such as user accounts, computer accounts, security groups, or the threads or processes that run in the security context of a user or computer account. For more information, see Security Principals.

A security principal is represented by a unique security identifier (SID).The SIDs that are related to each of the default local accounts in Active Directory are described in the sections below.

Some of the default local accounts are protected by a background process that periodically checks and applies a specific security descriptor. A security descriptor is a data structure that contains security information that is associated with a protected object. This process ensures that any successful unauthorized attempt to modify the security descriptor on one of the default local accounts or groups is overwritten with the protected settings.

This security descriptor is present on the AdminSDHolder object. If you want to modify the permissions on one of the service administrator groups or on any of its member accounts, you must modify the security descriptor on the AdminSDHolder object to ensure that it is applied consistently. Be careful when making these modifications, because you are also changing the default settings that are applied to all of your protected accounts.

Administrator account

The Administrator account is a default account that is used in all versions of the Windows operating system on every computer and device. The Administrator account is used by the system administrator for tasks that require administrative credentials. This account cannot be deleted or locked out, but the account can be renamed or disabled.

The Administrator account gives the user complete access (Full Control permissions) of the files, directories, services, and other resources that are on that local server. The Administrator account can be used to create local users, and assign user rights and access control permissions. Administrator can also be used to take control of local resources at any time simply by changing the user rights and permissions. Although files and directories can be protected from the Administrator account temporarily, the Administrator account can take control of these resources at any time by changing the access permissions.

Account group membership

The Administrator account has membership in the default security groups as described in the Administrator account attributes table later in this topic.

The security groups ensure that you can control administrator rights without having to change each Administrator account. In most instances, you do not have to change the basic settings for this account. However, you might have to change its advanced settings, such as membership in particular groups.

Security considerations

After installation of the server operating system, your first task is to set up the Administrator account properties securely. This includes setting up an especially long, strong password, and securing the Remote control and Remote Desktop Services profile settings.

The Administrator account can also be disabled when it is not required. Renaming or disabling the Administrator account makes it more difficult for malicious users to try to gain access to the account. However, even when the Administrator account is disabled, it can still be used to gain access to a domain controller by using safe mode.

On a domain controller, the Administrator account becomes the Domain Admin account. The Domain Admin account is used to sign in to the domain controller and this account requires a strong password. The Domain Admin account gives you access to domain resources.

Note

When the domain controller is initially installed, you can sign in and use Server Manager to set up a local Administrator account, with the rights and permissions you want to assign. For example, you can use a local Administrator account to manage the operating system when you first install it. By using this approach, you can set up the operating system without getting locked out. Generally, you do not need to use the account after installation. You can only create local user accounts on the domain controller, before Active Directory Domain Services is installed, and not afterwards.

When Active Directory is installed on the first domain controller in the domain, the Administrator account is created for Active Directory. The Administrator account is the most powerful account in the domain. It is given domain-wide access and administrative rights to administer the computer and the domain, and it has the most extensive rights and permissions over the domain. The person who installs Active Directory Domain Services on the computer creates the password for this account during the installation.

Administrator account attributes

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-<domain>-500
Type	User
Default container	CN=Users, DC=<domain>, DC=

ATTRIBUTE	VALUE
Default members	N/A
Default member of	Administrators, Domain Admins, Enterprise Administrators, Domain Users. Note that the Primary Group ID of all user accounts is Domain Users. Group Policy Creator Owners, and Schema Admins in Active Directory Domain Users group
Protected by ADMINSDHOLDER?	Yes
Safe to move out of default container?	Yes
Safe to delegate management of this group to non-service administrators?	No

Guest account

The Guest account is a default local account that has limited access to the computer and is disabled by default. By default, the Guest account password is left blank. A blank password allows the Guest account to be accessed without requiring the user to enter a password.

The Guest account enables occasional or one-time users, who do not have an individual account on the computer, to sign in to the local server or domain with restricted rights and permissions. The Guest account can be enabled, and the password can be set up if needed, but only by a member of the Administrator group on the domain.

Account group membership

The Guest account has membership in the default security groups that are described in the following Guest account attributes table. By default, the Guest account is the only member of the default Guests group, which lets a user sign in to a server, and the Domain Guests global group, which lets a user sign in to a domain.

A member of the Administrators group or Domain Admins group can set up a user with a Guest account on one or more computers.

Security considerations

Because the Guest account can provide anonymous access, it is a security risk. It also has a well-known SID. For this reason, it is a best practice to leave the Guest account disabled, unless its use is required and then only with restricted rights and permissions for a very limited period of time.

When the Guest account is required, an Administrator on the domain controller is required to enable the Guest account. The Guest account can be enabled without requiring a password, or it can be enabled with a strong password. The Administrator also grants restricted rights and permissions for the Guest account. To help prevent unauthorized access:

- Do not grant the Guest account the Shut down the system user right. When a computer is shutting down or starting up, it is possible that a Guest user or anyone with local access, such as a malicious user, could gain unauthorized access to the computer.

- Do not provide the Guest account with the ability to view the event logs. After the Guest account is enabled, it is a best practice to monitor this account frequently to ensure that other users cannot use services and other resources, such as resources that were unintentionally left available by a previous user.
- Do not use the Guest account when the server has external network access or access to other computers.

If you decide to enable the Guest account, be sure to restrict its use and to change the password regularly. As with the Administrator account, you might want to rename the account as an added security precaution.

In addition, an administrator is responsible for managing the Guest account. The administrator monitors the Guest account, disables the Guest account when it is no longer in use, and changes or removes the password as needed.

For details about the Guest account attributes, see the following table.

Guest account attributes

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-<domain>-501
Type	User
Default container	CN=Users, DC=<domain>, DC=
Default members	None
Default member of	Guests, Domain Guests
Protected by ADMINSDHOLDER?	No
Safe to move out of default container?	Can be moved out, but we do not recommend it.
Safe to delegate management of this group to non-Service admins?	No

HelpAssistant account (installed with a Remote Assistance session)

The HelpAssistant account is a default local account that is enabled when a Remote Assistance session is run. This account is automatically disabled when no Remote Assistance requests are pending.

HelpAssistant is the primary account that is used to establish a Remote Assistance session. The Remote Assistance session is used to connect to another computer running the Windows operating system, and it is initiated by invitation. For solicited remote assistance, a user sends an invitation from their computer, through e-mail or as a file, to a person who can provide assistance. After the user's invitation for a Remote Assistance session is accepted, the default HelpAssistant account is automatically created to give the person who provides assistance limited access to the computer. The HelpAssistant account is managed by the Remote Desktop Help Session Manager service.

Security considerations

The SIDs that pertain to the default HelpAssistant account include:

- SID: S-1-5-<domain>-13, display name Terminal Server User. This group includes all users who sign in to a server with Remote Desktop Services enabled. Note that, in Windows Server 2008, Remote Desktop Services are called Terminal Services.
- SID: S-1-5-<domain>-14, display name Remote Interactive Logon. This group includes all users who connect to the computer by using a remote desktop connection. This group is a subset of the Interactive group. Access tokens that contain the Remote Interactive Logon SID also contain the Interactive SID.

For the Windows Server operating system, Remote Assistance is an optional component that is not installed by default. You must install Remote Assistance before it can be used.

For details about the HelpAssistant account attributes, see the following table.

HelpAssistant account attributes

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-<domain>-13 (Terminal Server User), S-1-5-<domain>-14 (Remote Interactive Logon)
Type	User
Default container	CN=Users, DC=<domain>, DC=
Default members	None
Default member of	Domain Guests Guests
Protected by ADMINSDHOLDER?	No
Safe to move out of default container?	Can be moved out, but we do not recommend it.
Safe to delegate management of this group to non-Service admins?	No

KRBTGT account

The KRBTGT account is a local default account that acts as a service account for the Key Distribution Center (KDC) service. This account cannot be deleted, and the account name cannot be changed. The KRBTGT account cannot be enabled in Active Directory.

KRBTGT is also the security principal name used by the KDC for a Windows Server domain, as specified by RFC 4120. The KRBTGT account is the entity for the KRBTGT security principal, and it is created automatically when a new domain is created.

Windows Server Kerberos authentication is achieved by the use of a special Kerberos ticket-granting ticket (TGT) enciphered with a symmetric key. This key is derived from the password of the server or service to which access

is requested. The TGT password of the KRBTGT account is known only by the Kerberos service. In order to request a session ticket, the TGT must be presented to the KDC. The TGT is issued to the Kerberos client from the KDC.

KRBTGT account maintenance considerations

A strong password is assigned to the KRBTGT and trust accounts automatically. Like any privileged service accounts, organizations should change these passwords on a regular schedule. The password for the KDC account is used to derive a secret key for encrypting and decrypting the TGT requests that are issued. The password for a domain trust account is used to derive an inter-realm key for encrypting referral tickets.

Resetting the password requires you either to be a member of the Domain Admins group, or to have been delegated with the appropriate authority. In addition, you must be a member of the local Administrators group, or you must have been delegated the appropriate authority.

After you reset the KRBTGT password, ensure that event ID 9 in the (Kerberos) Key-Distribution-Center event source is written to the System event log.

Security considerations

It is also a best practice to reset the KRBTGT account password to ensure that a newly restored domain controller does not replicate with a compromised domain controller. In this case, in a large forest recovery that is spread across multiple locations, you cannot guarantee that all domain controllers are shut down, and if they are shut down, they cannot be rebooted again before all of the appropriate recovery steps have been undertaken. After you reset the KRBTGT account, another domain controller cannot replicate this account password by using an old password.

An organization suspecting domain compromise of the KRBTGT account should consider the use of professional incident response services. The impact to restore the ownership of the account is domain-wide and labor intensive an should be undertaken as part of a larger recovery effort.

The KRBTGT password is the key from which all trust in Kerberos chains up to. Resetting the KRBTGT password is similar to renewing the root CA certificate with a new key and immediately not trusting the old key, resulting in almost all subsequent Kerberos operations will be affected.

For all account types (users, computers, and services)

- All the TGTs that are already issued and distributed will be invalid because the DCs will reject them. These tickets are encrypted with the KRBTGT so any DC can validate them. When the password changes, the tickets become invalid.
- All currently authenticated sessions that logged on users have established (based on their service tickets) to a resource (such as a file share, SharePoint site, or Exchange server) are good until the service ticket is required to re-authenticate.
- NTLM authenticated connections are not affected

Because it is impossible to predict the specific errors that will occur for any given user in a production operating environment, you must assume all computers and users will be affected.

Important

Rebooting a computer is the only reliable way to recover functionality as this will cause both the computer account and user accounts to log back in again. Logging in again will request new TGTs that are valid with the new KRBTGT, correcting any KRBTGT related operational issues on that computer.

For information about how to help mitigate the risks associated with a potentially compromised KRBTGT account, see KRBTGT Account Password Reset Scripts now available for customers.

Read-only domain controllers and the KRBTGT account

Windows Server 2008 introduced the read-only domain controller (RODC). The RODC is advertised as the Key Distribution Center (KDC) for the branch office. The RODC uses a different KRBTGT account and password than the KDC on a writable domain controller when it signs or encrypts ticket-granting ticket (TGT) requests. After an account is successfully authenticated, the RODC determines if a user's credentials or a computer's credentials can be replicated from the writable domain controller to the RODC by using the Password Replication Policy.

After the credentials are cached on the RODC, the RODC can accept that user's sign-in requests until the credentials change. When a TGT is signed with the KRBTGT account of the RODC, the RODC recognizes that it has a cached copy of the credentials. If another domain controller signs the TGT, the RODC forwards requests to a writable domain controller.

KRBTGT account attributes

For details about the KRBTGT account attributes, see the following table.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-<domain>-502
Type	User
Default container	CN=Users, DC=<domain>, DC=
Default members	None
Default member of	Domain Users group. Note that the Primary Group ID of all user accounts is Domain Users.
Protected by ADMINSDHOLDER?	Yes
Safe to move out of default container?	Can be moved out, but we do not recommend it.
Safe to delegate management of this group to non-Service admins?	No

Settings for default local accounts in Active Directory

Each default local account in Active Directory has a number of account settings that you can use to configure password settings and security-specific information, as described in the following table.

Settings for default local accounts in Active Directory

ACCOUNT SETTINGS	DESCRIPTION
User must change password at next logon	Forces a password change the next time that the user logs signs in to the network. Use this option when you want to ensure that the user is the only person to know his or her password.

ACCOUNT SETTINGS	DESCRIPTION
User cannot change password	Prevents the user from changing the password. Use this option when you want to maintain control over a user account, such as for a Guest or temporary account.
Password never expires	Prevents a user password from expiring. It is a best practice to enable this option with service accounts and to use strong passwords.
Store passwords using reversible encryption	Provides support for applications that use protocols requiring knowledge of the plaintext form of the user's password for authentication purposes. This option is required when using Challenge Handshake Authentication Protocol (CHAP) in Internet Authentication Services (IAS), and when using digest authentication in Internet Information Services (IIS).
Account is disabled	Prevents the user from signing in with the selected account. As an administrator, you can use disabled accounts as templates for common user accounts.
Smart card is required for interactive logon	Requires that a user has a smart card to sign on to the network interactively. The user must also have a smart card reader attached to their computer and a valid personal identification number (PIN) for the smart card. When this attribute is applied on the account, the effect is as follows: • The attribute only restricts initial authentication for interactive logon and Remote Desktop logon. When interactive or Remote Desktop logon requires a subsequent network logon, such as with a domain credential, an NT Hash provided by the domain controller is used to complete the smartcard authentication process • Each time the attribute is enabled on an account, the account's current password hash value is replaced with a 128-bit random number. This invalidates the use of any previously configured passwords for the account. The value does not change after that unless a new password is set or the attribute is disabled and re-enabled. • Accounts with this attribute cannot be used to start services or run scheduled tasks.

ACCOUNT SETTINGS	DESCRIPTION
Account is trusted for delegation	Lets a service running under this account perform operations on behalf of other user accounts on the network. A service running under a user account (also known as a service account) that is trusted for delegation can impersonate a client to gain access to resources, either on the computer where the service is running or on other computers. For example, in a forest that is set to the Windows Server 2003 functional level, this setting is found on the **Delegation** tab. It is available only for accounts that have been assigned service principal names (SPNs), which are set by using the **setspn** command from Windows Support Tools. This setting is security-sensitive and should be assigned cautiously.
Account is sensitive and cannot be delegated	Gives control over a user account, such as for a Guest account or a temporary account. This option can be used if this account cannot be assigned for delegation by another account.
Use DES encryption types for this account	Provides support for the Data Encryption Standard (DES). DES supports multiple levels of encryption, including Microsoft Point-to-Point Encryption (MPPE) Standard (40-bit and 56-bit), MPPE standard (56-bit), MPPE Strong (128-bit), Internet Protocol security (IPSec) DES (40-bit), IPSec 56-bit DES, and IPSec Triple DES (3DES). **Note** DES is not enabled by default in Windows Server operating systems starting with Windows Server 2008 R2, nor in Windows client operating systems starting with Windows 7. For these operating systems, computers will not use DES-CBC-MD5 or DES-CBC-CRC cipher suites by default. If your environment requires DES, then this setting might affect compatibility with client computers or services and applications in your environment. For more information, see Hunting down DES in order to securely deploy Kerberos.
Do not require Kerberos preauthentication	Provides support for alternate implementations of the Kerberos protocol. Because preauthentication provides additional security, use caution when enabling this option. Note that domain controllers running Windows 2000 or Windows Server 2003 can use other mechanisms to synchronize time.

Manage default local accounts in Active Directory

After the default local accounts are installed, these accounts reside in the Users container in Active Directory Users and Computers. Default local accounts can be created, disabled, reset, and deleted by using the Active Directory Users and Computers Microsoft Management Console (MMC) and by using command-line tools.

You can use Active Directory Users and Computers to assign rights and permissions on a given local domain

controller, and that domain controller only, to limit the ability of local users and groups to perform certain actions. A right authorizes a user to perform certain actions on a computer, such as backing up files and folders or shutting down a computer. In contrast, an access permission is a rule that is associated with an object, usually a file, folder, or printer, that regulates which users can have access to the object and in what manner.

For more information about creating and managing local user accounts in Active Directory, see Manage Local Users.

You can also use Active Directory Users and Computers on a domain controller to target remote computers that are not domain controllers on the network.

You can obtain recommendations from Microsoft for domain controller configurations that you can distribute by using the Security Compliance Manager (SCM) tool. For more information, see Microsoft Security Compliance Manager.

Some of the default local user accounts are protected by a background process that periodically checks and applies a specific security descriptor, which is a data structure that contains security information that is associated with a protected object. This security descriptor is present on the AdminSDHolder object.

This means, when you want to modify the permissions on a service administrator group or on any of its member accounts, you are also required to modify the security descriptor on the AdminSDHolder object. This approach ensures that the permissions are applied consistently. Be careful when you make these modifications, because this action can also affect the default settings that are applied to all of your protected administrative accounts.

Restrict and protect sensitive domain accounts

Restricting and protecting domain accounts in your domain environment requires you to adopt and implement the following best practices approach:

- Strictly limit membership to the Administrators, Domain Admins, and Enterprise Admins groups.
- Stringently control where and how domain accounts are used.

Member accounts in the Administrators, Domain Admins, and Enterprise Admins groups in a domain or forest are high-value targets for malicious users. It is a best practice to strictly limit membership to these administrator groups to the smallest number of accounts in order to limit any exposure. Restricting membership in these groups reduces the possibility that an administrator might unintentionally misuse these credentials and create a vulnerability that malicious users can exploit.

Moreover, it is a best practice to stringently control where and how sensitive domain accounts are used. Restrict the use of Domain Admins accounts and other administrator accounts to prevent them from being used to sign in to management systems and workstations that are secured at the same level as the managed systems. When administrator accounts are not restricted in this manner, each workstation from which a domain administrator signs in provides another location that malicious users can exploit.

Implementing these best practices is separated into the following tasks:

- Separate administrator accounts from user accounts
- Create dedicated workstation hosts for administrators
- Restrict administrator logon access to servers and workstations
- Disable the account delegation right for administrator accounts

Note that, to provide for instances where integration challenges with the domain environment are expected, each task is described according to the requirements for a minimum, better, and ideal implementation. As with all significant changes to a production environment, ensure that you test these changes thoroughly before you

implement and deploy them. Then stage the deployment in a manner that allows for a rollback of the change in case technical issues occur.

Separate administrator accounts from user accounts

Restrict Domain Admins accounts and other sensitive accounts to prevent them from being used to sign in to lower trust servers and workstations. Restrict and protect administrator accounts by segregating administrator accounts from standard user accounts, by separating administrative duties from other tasks, and by limiting the use of these accounts. Create dedicated accounts for administrative personnel who require administrator credentials to perform specific administrative tasks, and then create separate accounts for other standard user tasks, according to the following guidelines:

- **Privileged account**. Allocate administrator accounts to perform the following administrative duties only:
  - **Minimum**. Create separate accounts for domain administrators, enterprise administrators, or the equivalent with appropriate administrator rights in the domain or forest. Use accounts that have been granted sensitive administrator rights only to administer domain data and domain controllers.
  - **Better**. Create separate accounts for administrators that have reduced administrative rights, such as accounts for workstation administrators, and accounts with user rights over designated Active Directory organizational units (OUs).
  - **Ideal**. Create multiple, separate accounts for an administrator who has a variety of job responsibilities that require different trust levels. Set up each administrator account with significantly different user rights, such as for workstation administration, server administration and domain administration, to let the administrator sign in to given workstations, servers and domain controllers based strictly on his or her job responsibilities.
- **Standard user account**. Grant standard user rights for standard user tasks, such as email, web browsing, and using line-of-business (LOB) applications. These accounts should not be granted administrator rights.

Important
Ensure that sensitive administrator accounts cannot access email or browse the Internet as described in the following section.

Create dedicated workstation hosts without Internet and email access

Administrators need to manage job responsibilities that require sensitive administrator rights from a dedicated workstation because they do not have easy physical access to the servers. A workstation that is connected to the Internet and has email and web browsing access is regularly exposed to compromise through phishing, downloading, and other types of Internet attacks. Because of these threats, it is a best practice to set these administrators up by using workstations that are dedicated to administrative duties only, and not provide access to the Internet, including email and web browsing. For more information, see Separate administrator accounts from user accounts.

Note
If the administrators in your environment can sign in locally to managed servers and perform all tasks without elevated rights or domain rights from their workstation, you can skip this task.

- **Minimum**. Build dedicated administrative workstations and block Internet access on those workstations including web browsing and email. Use the following ways to block Internet access:
  - Configure authenticating boundary proxy services, if they are deployed, to disallow administrator accounts from accessing the Internet.
  - Configure boundary firewall or proxy services to disallow Internet access for the IP addresses that

are assigned to dedicated administrative workstations.

- Block outbound access to the boundary proxy servers in the Windows Firewall.

The instructions for meeting this minimum requirement are described in the following procedure.

- **Better**. Do not grant administrators membership in the local Administrator group on the computer in order to restrict the administrator from bypassing these protections.
- **Ideal**. Restrict workstations from having any network connectivity, except for the domain controllers and servers that the administrator accounts are used to manage. Alternately, use AppLocker application control policies to restrict all applications from running, except for the operating system and approved administrative tools and applications. For more information about AppLocker, see AppLocker.

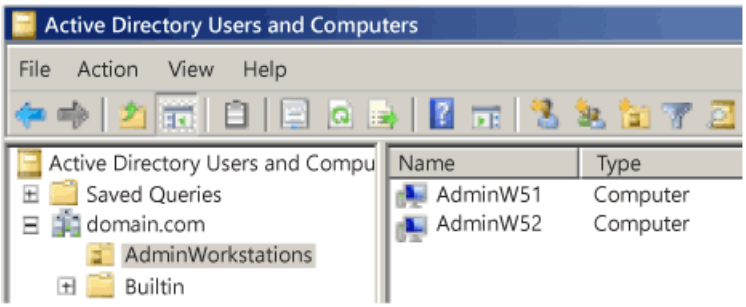
The following procedure describes how to block Internet access by creating a Group Policy Object (GPO) that configures an invalid proxy address on administrative workstations. These instructions apply only to computers running Internet Explorer and other Windows components that use these proxy settings.

Note
In this procedure, the workstations are dedicated to domain administrators. By simply modifying the administrator accounts to grant permission to administrators to sign in locally, you can create additional OUs to manage administrators that have fewer administrative rights to use the instructions described in the following procedure.

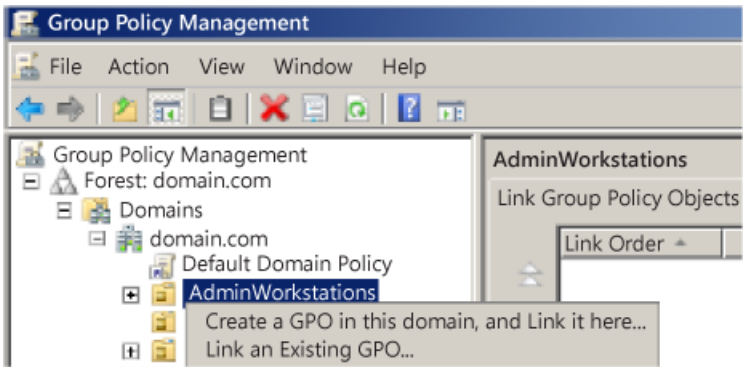
To install administrative workstations in a domain and block Internet and email access (minimum)

1. As a domain administrator on a domain controller, open Active Directory Users and Computers, and create a new OU for administrative workstations.
2. Create computer accounts for the new workstations.

> **Note** You might have to delegate permissions to join computers to the domain if the account that joins the workstations to the domain does not already have them. For more information, see Delegation of Administration in Active Directory.

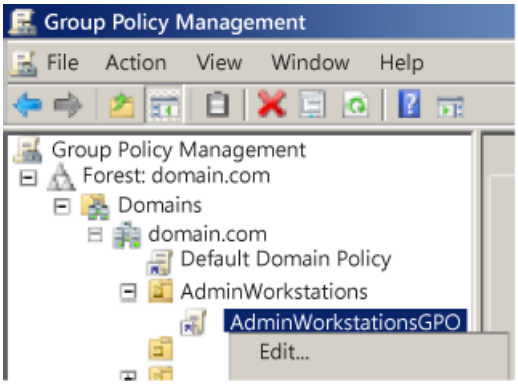


3. Close Active Directory Users and Computers.
4. Start the **Group Policy Management** Console (GPMC).
5. Right-click the new OU, and > **Create a GPO in this domain, and Link it here**.



6. Name the GPO, and > **OK**.

7. Expand the GPO, right-click the new GPO, and > **Edit**.



8. Configure which members of accounts can log on locally to these administrative workstations as follows:

   a. Navigate to Computer Configuration\Policies\Windows Settings\Local Policies, and then click **User Rights Assignment**.

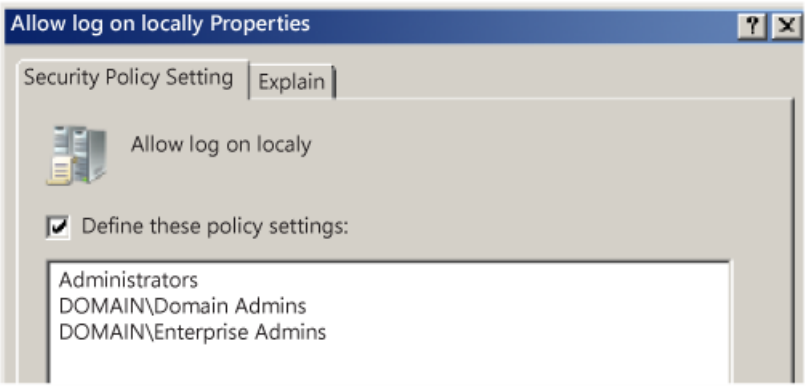
   b. Double-click **Allow log on locally**, and then select the **Define these policy settings** check box.

   c. Click **Add User or Group** > **Browse**, type **Enterprise Admins**, and > **OK**.

   d. Click **Add User or Group** > **Browse**, type **Domain Admins**, and > **OK**.

      Important
      These instructions assume that the workstation is to be dedicated to domain administrators.

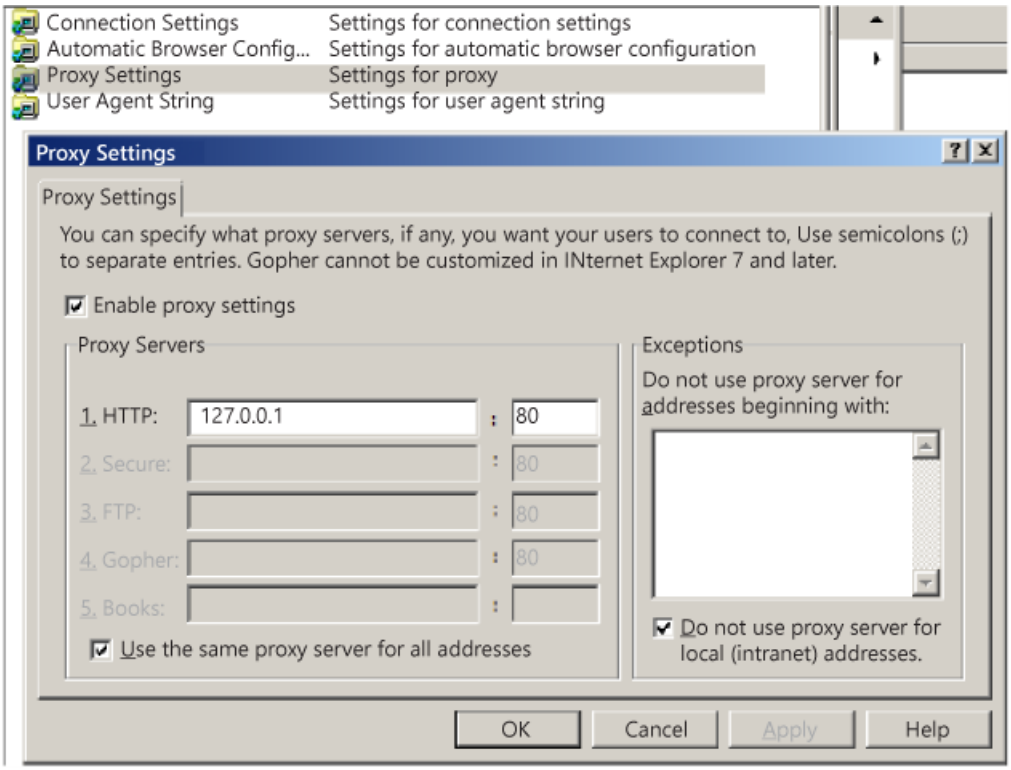
   e. Click **Add User or Group**, type **Administrators**, and > **OK**.



9. Configure the proxy configuration:

   a. Navigate to User Configuration\Policies\Windows Settings\Internet Explorer, and > **Connection**.

   b. Double-click **Proxy Settings**, select the **Enable proxy settings** check box, type **127.0.0.1** (the network Loopback IP address) as the proxy address, and > **OK**.



10. Configure the loopback processing mode to enable the user Group Policy proxy setting to apply to all users on the computer as follows:

    a. Navigate to Computer Configuration\Policies\Administrative Templates\System, and > **Group Policy**.

    b. Double-click **User Group Policy loopback policy processing mode**, and > **Enabled**.

    c. Select **Merge Mode**, and > **OK**.

11. Configure software updates as follows:

    a. Navigate to Computer Configuration\Policies\Administrative Templates\Windows Components, and then click **Windows Update**.

    b. Configure Windows Update settings as described in the following table.

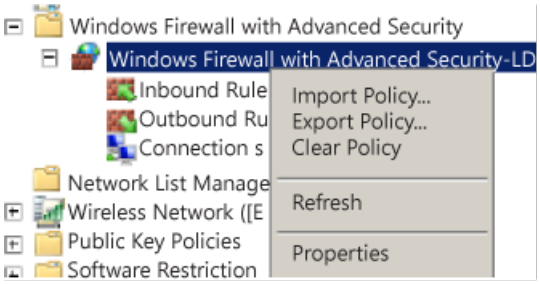
Windows Update Setting	Configuration
Allow Automatic Updates immediate installation	Enabled
Configure Automatic Updates	Enabled 4 - Auto download and schedule the installation 0 - Every day 03:00
Enable Windows Update Power Management to automatically wake up the system to install scheduled updates	Enabled

Specify intranet Microsoft Update service location	Enabled http://<WSUSServername> http://<WSUSServername> Where <WSUSServername> is the DNS name or IP address of the Windows Server Update Services (WSUS) in the environment.
Automatic Updates detection frequency	6 hours
Re-prompt for restart with scheduled installations	1 minute
Delay restart for scheduled installations	5 minutes

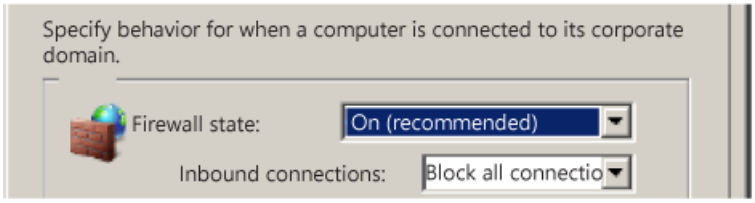
> **Note** This step assumes that Windows Server Update Services (WSUS) is installed and configured in the environment. You can skip this step if you use another tool to deploy software updates. Also, if the public Microsoft Windows Update service only is used on the Internet, then these administrative workstations no longer receive updates.

12. Configure the inbound firewall to block all connections as follows:

    a. Right-click **Windows Firewall with Advanced Security LDAP://path**, and > **Properties**.



    b. On each profile, ensure that the firewall is enabled and that inbound connections are set to **Block all connections**.



    c. Click **OK** to complete the configuration.

13. Close the Group Policy Management Console.

14. Install the Windows operating system on the workstations, give each workstation the same names as the computer accounts assigned to them, and then join them to the domain.

Restrict administrator logon access to servers and workstations

It is a best practice to restrict administrators from using sensitive administrator accounts to sign in to lower-trust servers and workstations. This restriction prevents administrators from inadvertently increasing the risk of credential theft by signing in to a lower-trust computer.

Important

Ensure that you either have local access to the domain controller or that you have built at least one dedicated

administrative workstation.

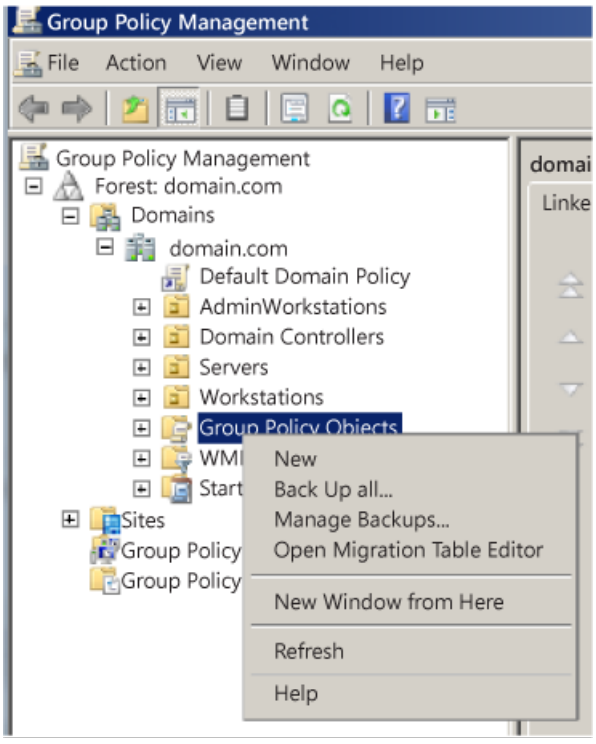
Restrict logon access to lower-trust servers and workstations by using the following guidelines:

- **Minimum**. Restrict domain administrators from having logon access to servers and workstations. Before starting this procedure, identify all OUs in the domain that contain workstations and servers. Any computers in OUs that are not identified will not restrict administrators with sensitive accounts from signing-in to them.
- **Better**. Restrict domain administrators from non-domain controller servers and workstations.
- **Ideal**. Restrict server administrators from signing in to workstations, in addition to domain administrators.

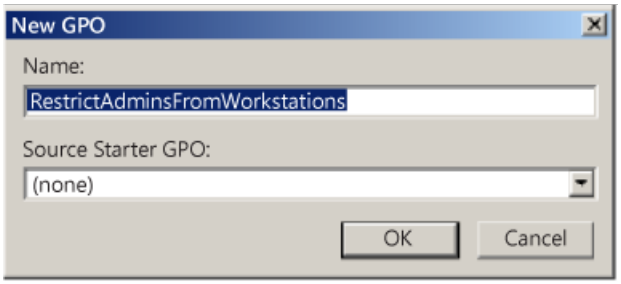
Note
For this procedure, do not link accounts to the OU that contain workstations for administrators that perform administration duties only, and do not provide Internet or email access. For more information, see Create dedicated workstation hosts for administrators

To restrict domain administrators from workstations (minimum)

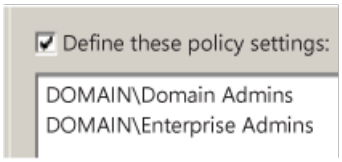
1. As a domain administrator, open the Group Policy Management Console (GPMC).
2. Open **Group Policy Management**, and expand *<forest>*\Domains*<domain>*, and then expand to **Group Policy Objects**.
3. Right-click **Group Policy Objects**, and > **New**.



4. In the **New GPO** dialog box, name the GPO that restricts administrators from signing in to workstations, and > **OK**.



5. Right-click **New GPO**, and > **Edit**.
6. Configure user rights to deny logon locally for domain administrators.
7. Navigate to Computer Configuration\Policies\Windows Settings\Local Policies, and then click **User Rights Assignment**, and perform the following:
   a. Double-click **Deny logon locally**, and > **Define these policy settings**.
   b. Click **Add User or Group**, click **Browse**, type **Enterprise Admins**, and > **OK**.
   c. Click **Add User or Group**, click **Browse**, type **Domain Admins**, and > **OK**.

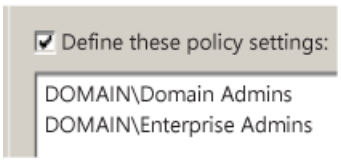


   Note
   You can optionally add any groups that contain server administrators who you want to restrict from signing in to workstations.

   d. Click **OK** to complete the configuration.
8. Configure the user rights to deny batch and service logon rights for domain administrators as follows:

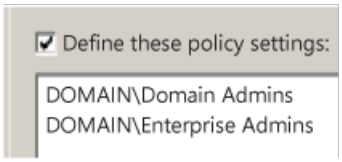
   Note
   Completing this step might cause issues with administrator tasks that run as scheduled tasks or services with accounts in the Domain Admins group. The practice of using domain administrator accounts to run services and tasks on workstations creates a significant risk of credential theft attacks and therefore should be replaced with alternative means to run scheduled tasks or services.

   a. Double-click **Deny logon as a batch job**, and > **Define these policy settings**.
   b. Click **Add User or Group** > **Browse**, type **Enterprise Admins**, and > **OK**.
   c. Click **Add User or Group** > **Browse**, type **Domain Admins**, and > **OK**.



   Note
   You can optionally add any groups that contain server administrators who you want to restrict from signing in to workstations.

   d. Double-click **Deny logon as a service**, and > **Define these policy settings**.
   e. Click **Add User or Group** > **Browse**, type **Enterprise Admins**, and > **OK**.
   f. Click **Add User or Group** > **Browse**, type **Domain Admins**, and > **OK**.

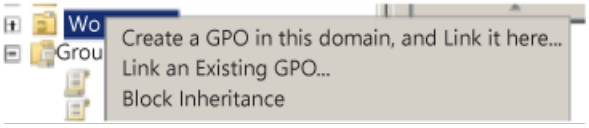


Note
You can optionally add any groups that contain server administrators who you want to restrict from signing in to workstations.

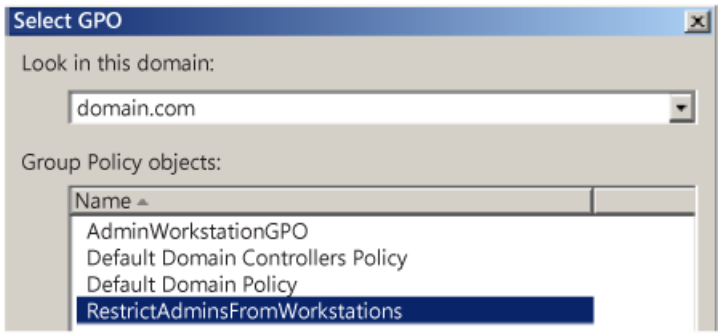
9. Link the GPO to the first Workstations OU.

   Navigate to the *<forest>*\Domains*<domain>*\OU Path, and then:

   a. Right-click the workstation OU, and then > **Link an Existing GPO**.

   b. Select the GPO that you just created, and > **OK**.

10. Test the functionality of enterprise applications on workstations in the first OU and resolve any issues caused by the new policy.

11. Link all other OUs that contain workstations.

    However, do not create a link to the Administrative Workstation OU if it is created for administrative workstations that are dedicated to administration duties only, and that are without Internet or email access. For more information, see Create dedicated workstation hosts for administrators.

    Important
    If you later extend this solution, do not deny logon rights for the **Domain Users** group. The **Domain Users** group includes all user accounts in the domain, including Users, Domain Administrators, and Enterprise Administrators.

Disable the account delegation right for sensitive administrator accounts

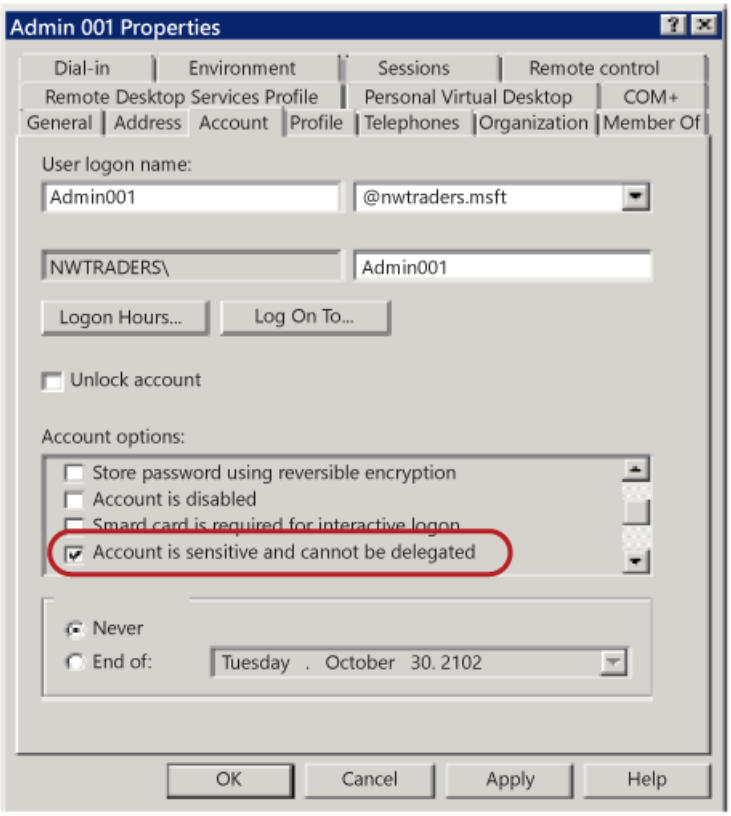
Although user accounts are not marked for delegation by default, accounts in an Active Directory domain can be trusted for delegation. This means that a service or a computer that is trusted for delegation can impersonate an account that authenticates to them to access other resources across the network.

For sensitive accounts, such as those belonging to members of the Administrators, Domain Admins, or Enterprise Admins groups in Active Directory, delegation can present a substantial risk of rights escalation. For example, if an account in the Domain Admins group is used to sign in to a compromised member server that is trusted for delegation, that server can request access to resources in the context of the Domain Admins account, and escalate the compromise of that member server to a domain compromise.

It is a best practice to configure the user objects for all sensitive accounts in Active Directory by selecting the **Account is sensitive and cannot be delegated** check box under **Account options** to prevent these

accounts from being delegated. For more information, see Setting for default local accounts in Active Directory.

As with any configuration change, test this enabled setting fully to ensure that it performs correctly before you implement it.



Secure and manage domain controllers

It is a best practice to strictly enforce restrictions on the domain controllers in your environment. This ensures that the domain controllers:

1. Run only required software
2. Required software is regularly updated
3. Are configured with the appropriate security settings

One aspect of securing and managing domain controllers is to ensure that the default local user accounts are fully protected. It is of primary importance to restrict and secure all sensitive domain accounts, as described in the preceding sections.

Because domain controllers store credential password hashes of all accounts in the domain, they are high-value targets for malicious users. When domain controllers are not well managed and secured by using restrictions that are strictly enforced, they can be compromised by malicious users. For example, a malicious user could steal sensitive domain administrator credentials from one domain controller, and then use these credentials to attack the domain and forest.

In addition, installed applications and management agents on domain controllers might provide a path for escalating rights that malicious users can use to compromise the management service or administrators of that service. The management tools and services, which your organization uses to manage domain controllers and their administrators, are equally important to the security of the domain controllers and the domain administrator accounts. Ensure that these services and administrators are fully secured with equal effort.

See also

- Security Principals
- Access Control Overview

Microsoft Accounts

3/26/2021 • 9 minutes to read • Edit Online

Applies to

- Windows 10

This topic for the IT professional explains how a Microsoft account works to enhance security and privacy for users, and how you can manage this consumer account type in your organization.

Microsoft sites, services, and properties, as well as computers running Windows 10, can use a Microsoft account as a means of identifying a user. Microsoft account was previously called Windows Live ID. It has user-defined secrets, and consists of a unique email address and a password.

When a user signs in with a Microsoft account, the device is connected to cloud services. Many of the user's settings, preferences, and apps can be shared across devices.

How a Microsoft account works

The Microsoft account allows users to sign in to websites that support this service by using a single set of credentials. Users' credentials are validated by a Microsoft account authentication server that is associated with a website. The Microsoft Store is an example of this association. When new users sign in to websites that are enabled to use Microsoft accounts, they are redirected to the nearest authentication server, which asks for a user name and password. Windows uses the Schannel Security Support Provider to open a Transport Level Security/Secure Sockets Layer (TLS/SSL) connection for this function. Users then have the option to use Credential Manager to store their credentials.

When users sign in to websites that are enabled to use a Microsoft account, a time-limited cookie is installed on their computers, which includes a triple DES encrypted ID tag. This encrypted ID tag has been agreed upon between the authentication server and the website. This ID tag is sent to the website, and the website plants another time-limited encrypted HTTP cookie on the user’s computer. When these cookies are valid, users are not required to supply a user name and password. If a user actively signs out of their Microsoft account, these cookies are removed.

Important Local Windows account functionality has not been removed, and it is still an option to use in managed environments.

How Microsoft accounts are created

To prevent fraud, the Microsoft system verifies the IP address when a user creates an account. A user who tries to create multiple Microsoft accounts with the same IP address is stopped.

Microsoft accounts are not designed to be created in batches, such as for a group of domain users within your enterprise.

There are two methods for creating a Microsoft account:

- **Use an existing email address**.

  Users are able to use their valid email addresses to sign up for Microsoft accounts. The service turns the requesting user's email address into a Microsoft account. Users can also choose their personal passwords.

- **Sign up for a Microsoft email address**.

Users can sign up for an email account with Microsoft's webmail services. This account can be used to sign in to websites that are enabled to use Microsoft accounts.

How the Microsoft account information is safeguarded

Credential information is encrypted twice. The first encryption is based on the account's password. Credentials are encrypted again when they are sent across the Internet. The data that is stored is not available to other Microsoft or non-Microsoft services.

- **Strong password is required**.

  Blank passwords are not allowed.

  For more information, see Microsoft Account Security Overview.

- **Secondary proof of identity is required**.

  Before user profile information and settings can be accessed on a second supported Windows computer for the first time, trust must established for that device by providing secondary proof of identity. This can be accomplished by providing Windows with a code that is sent to a mobile phone number or by following the instructions that are sent to an alternate email address that a user specifies in the account settings.

- **All user profile data is encrypted on the client before it is transmitted to the cloud**.

  User data does not roam over a wireless wide area network (WWAN) by default, thereby protecting profile data. All data and settings that leave a device are transmitted through the TLS/SSL protocol.

Microsoft account security information is added.

Users can add security information to their Microsoft accounts through the **Accounts** interface on computers running the supported versions of Windows. This feature allows the user to update the security information that they provided when they created their accounts. This security information includes an alternate email address or phone number so if their password is compromised or forgotten, a verification code can be sent to verify their identity. Users can potentially use their Microsoft accounts to store corporate data on a personal OneDrive or email app, so it is safe practice for the account owner to keep this security information up-to-date.

The Microsoft account in the enterprise

Although the Microsoft account was designed to serve consumers, you might find situations where your domain users can benefit by using their personal Microsoft account in your enterprise. The following list describes some advantages.

- **Download Microsoft Store apps**:

  If your enterprise chooses to distribute software through the Microsoft Store, your users can use their Microsoft accounts to download and use them on up to five devices running any version of Windows 10, Windows 8.1, Windows 8, or Windows RT.

- **Single sign-on**:

  Your users can use Microsoft account credentials to sign in to devices running Windows 10, Windows 8.1, Windows 8 or Windows RT. When they do this, Windows works with your Microsoft Store app to provide authenticated experiences for them. Users can associate a Microsoft account with their sign-in credentials for Microsoft Store apps or websites, so that these credentials roam across any devices running these supported versions.

- **Personalized settings synchronization**:

  Users can associate their most commonly used operating-system settings with a Microsoft account.

These settings are available whenever a user signs in with that account on any device that is running a supported version of Windows and is connected to the cloud. After a user signs in, the device automatically attempts to get the user's settings from the cloud and apply them to the device.

- **App synchronization**:

  Microsoft Store apps can store user-specific settings so that these settings are available to any device. As with operating system settings, these user-specific app settings are available whenever the user signs in with the same Microsoft account on any device that is running a supported version of Windows and is connected to the cloud. After the user signs in, that device automatically downloads the settings from the cloud and applies them when the app is installed.

- **Integrated social media services**:

  Contact information and status for your users' friends and associates automatically stay up-to-date from sites such as Hotmail, Outlook, Facebook, Twitter, and LinkedIn. Users can also access and share photos, documents, and other files from sites such as OneDrive, Facebook, and Flickr.

Managing the Microsoft account in the domain

Depending on your IT and business models, introducing Microsoft accounts into your enterprise might add complexity or it might provide solutions. You should address the following considerations before you allow the use of these account types in your enterprise:

- Restrict the use of the Microsoft account
- Configure connected accounts
- Provision Microsoft accounts in the enterprise
- Audit account activity
- Perform password resets
- Restrict app installation and usage

Restrict the use of the Microsoft account

The following Group Policy settings help control the use of Microsoft accounts in the enterprise:

- Block all consumer Microsoft account user authentication
- Accounts: Block Microsoft accounts

Block all consumer Microsoft account user authentication

This setting controls whether users can provide Microsoft accounts for authentication for applications or services.

If this setting is enabled, all applications and services on the device are prevented from using Microsoft accounts for authentication. This applies both to existing users of a device and new users who may be added.

However, any application or service that has already authenticated a user will not be affected by enabling this setting until the authentication cache expires. It is recommended to enable this setting before any user signs in to a device to prevent cached tokens from being present.

If this setting is disabled or not configured, applications and services can use Microsoft accounts for authentication. By default, this setting is **Disabled**.

This setting does not affect whether users can sign in to devices by using Microsoft accounts, or the ability for users to provide Microsoft accounts via the browser for authentication with web-based applications.

The path to this setting is:

Computer Configuration\Administrative Templates\Windows Components\Microsoft account

Accounts: Block Microsoft accounts

This setting prevents using the **Settings** app to add a Microsoft account for single sign-on (SSO) authentication for Microsoft services and some background services, or using a Microsoft account for single sign-on to other applications or services.

There are two options if this setting is enabled:

- **Users can't add Microsoft accounts** means that existing connected accounts can still sign in to the device (and appear on the Sign in screen). However, users cannot use the **Settings** app to add new connected accounts (or connect local accounts to Microsoft accounts).
- **Users can't add or log on with Microsoft accounts** means that users cannot add new connected accounts (or connect local accounts to Microsoft accounts) or use existing connected accounts through **Settings**.

This setting does not affect adding a Microsoft account for application authentication. For example, if this setting is enabled, a user can still provide a Microsoft account for authentication with an application such as **Mail**, but the user cannot use the Microsoft account for single sign-on authentication for other applications or services (in other words, the user will be prompted to authenticate for other applications or services).

By default, this setting is **Not defined**.

The path to this setting is:

Computer Configuration\Windows Settings\Security Settings\Local Policies\Security Options

Configure connected accounts

Users can connect a Microsoft account to their domain account and synchronize the settings and preferences between them. This enables users to see the same desktop background, app settings, browser history and favorites, and other Microsoft account settings on their other devices.

Users can disconnect a Microsoft account from their domain account at any time as follows: In **PC settings**, tap or click **Users**, tap or click **Disconnect**, and then tap or click **Finish**.

Note Connecting Microsoft accounts with domain accounts can limit access to some high-privileged tasks in Windows. For example, Task Scheduler will evaluate the connected Microsoft account for access and fail. In these situations, the account owner should disconnect the account.

Provision Microsoft accounts in the enterprise

Microsoft accounts are private user accounts. There are no methods provided by Microsoft to provision Microsoft accounts for an enterprise. Enterprises should use domain accounts.

Audit account activity

Because Microsoft accounts are Internet-based, Windows does not have a mechanism to audit their use until the account is associated with a domain account. But this association does not restrict the user from disconnecting the account or disjoining from the domain. It is not possible to audit the activity of accounts that are not associated with your domain.

Perform password resets

Only the owner of the Microsoft account can change the password. Passwords can be changed in the Microsoft account sign-in portal.

Restrict app installation and usage

Within your organization, you can set application control policies to regulate app installation and usage for Microsoft accounts. For more information, see AppLocker and Packaged Apps and Packaged App Installer Rules in AppLocker.

See also

- Managing Privacy: Using a Microsoft Account to Logon and Resulting Internet Communication
- Access Control Overview

Service Accounts

3/26/2021 • 6 minutes to read • Edit Online

Applies to

- Windows 10
- Windows Server 2016

This topic for the IT professional explains group and standalone managed service accounts, and the computer-specific virtual computer account, and it points to resources about these service accounts.

Overview

A service account is a user account that is created explicitly to provide a security context for services running on Windows Server operating systems. The security context determines the service's ability to access local and network resources. The Windows operating systems rely on services to run various features. These services can be configured through the applications, the Services snap-in, or Task Manager, or by using Windows PowerShell.

This topic contains information about the following types of service accounts:

- Standalone managed service accounts
- Group managed service accounts
- Virtual accounts

Standalone managed service accounts

A managed service account is designed to isolate domain accounts in crucial applications, such as Internet Information Services (IIS), and eliminate the need for an administrator to manually administer the service principal name (SPN) and credentials for the accounts.

To use managed service accounts, the server on which the application or service is installed must be running at least Windows Server 2008 R2. One managed service account can be used for services on a single computer. Managed service accounts cannot be shared between multiple computers, and they cannot be used in server clusters where a service is replicated on multiple cluster nodes. For this scenario, you must use a group managed service account. For more information, see Group Managed Service Accounts Overview.

In addition to the enhanced security that is provided by having individual accounts for critical services, there are four important administrative benefits associated with managed service accounts:

- You can create a class of domain accounts that can be used to manage and maintain services on local computers.
- Unlike domain accounts in which administrators must reset manually passwords, the network passwords for these accounts are automatically reset.
- You do not have to complete complex SPN management tasks to use managed service accounts.
- Administrative tasks for managed service accounts can be delegated to non-administrators.

Software requirements

Managed service accounts apply to the Windows operating systems that are designated in the **Applies To** list at the beginning of this topic.

Group managed service accounts

Group managed service accounts are an extension of the standalone managed service accounts, which were introduced in Windows Server 2008 R2. These are managed domain accounts that provide automatic password management and simplified service principal name (SPN) management, including delegation of management to other administrators.

The group managed service account provides the same functionality as a standalone managed service account within the domain, but it extends that functionality over multiple servers. When connecting to a service that is hosted on a server farm, such as Network Load Balancing, the authentication protocols that support mutual authentication require all instances of the services to use the same principal. When group managed service accounts are used as service principals, the Windows Server operating system manages the password for the account instead of relying on the administrator to manage the password.

The Microsoft Key Distribution Service (kdssvc.dll) provides the mechanism to securely obtain the latest key or a specific key with a key identifier for an Active Directory account. This service was introduced in Windows Server 2012, and it does not run on previous versions of the Windows Server operating system. The Key Distribution Service shares a secret, which is used to create keys for the account. These keys are periodically changed. For a group managed service account, the domain controller computes the password on the key that is provided by the Key Distribution Services, in addition to other attributes of the group managed service account.

Practical applications

Group managed service accounts provide a single identity solution for services running on a server farm, or on systems that use Network Load Balancing. By providing a group managed service account solution, services can be configured for the group managed service account principal, and the password management is handled by the operating system.

By using a group managed service account, services or service administrators do not need to manage password synchronization between service instances. The group managed service account supports hosts that are kept offline for an extended time period and the management of member hosts for all instances of a service. This means that you can deploy a server farm that supports a single identity to which existing client computers can authenticate without knowing the instance of the service to which they are connecting.

Failover clusters do not support group managed service account s. However, services that run on top of the Cluster service can use a group managed service account or a standalone managed service account if they are a Windows service, an App pool, a scheduled task, or if they natively support group managed service account or standalone managed service accounts.

Software requirements

Group managed service accounts can only be configured and administered on computers running at least Windows Server 2012, but they can be deployed as a single service identity solution in domains that still have domain controllers running operating systems earlier than Windows Server 2012. There are no domain or forest functional level requirements.

A 64-bit architecture is required to run the Windows PowerShell commands that are used to administer group managed service accounts.

A managed service account is dependent on encryption types supported by Kerberos. When a client computer authenticates to a server by using Kerberos protocol, the domain controller creates a Kerberos service ticket that is protected with encryption that the domain controller and the server support. The domain controller uses the account's **msDS-SupportedEncryptionTypes** attribute to determine what encryption the server supports, and if there is no attribute, it assumes that the client computer does not support stronger encryption types. The Advanced Encryption Standard (AES) should always be explicitly configured for managed service accounts. If computers that host the managed service account are configured to not support RC4, authentication will always fail.

Note Introduced in Windows Server 2008 R2, the Data Encryption Standard (DES) is disabled by default. For

more information about supported encryption types, see Changes in Kerberos Authentication.

Group managed service accounts are not applicable in Windows operating systems prior to Windows Server 2012.

Virtual accounts

Virtual accounts were introduced in Windows Server 2008 R2 and Windows 7, and are managed local accounts that provide the following features to simplify service administration:

- The virtual account is automatically managed.
- The virtual account can access the network in a domain environment.
- No password management is required. For example, if the default value is used for the service accounts during SQL Server setup on Windows Server 2008 R2, a virtual account that uses the instance name as the service name is established in the format NT SERVICE\<SERVICENAME>.

Services that run as virtual accounts access network resources by using the credentials of the computer account in the format <domain_name>\<computer_name>$.

For information about how to configure and use virtual service accounts, see Service Accounts Step-by-Step Guide.

Software requirements

Virtual accounts apply to the Windows operating systems that are designated in the **Applies To** list at the beginning of this topic.

See also

The following table provides links to additional resources that are related to standalone managed service accounts, group managed service accounts, and virtual accounts.

CONTENT TYPE	REFERENCES
Product evaluation	What's New for Managed Service Accounts Getting Started with Group Managed Service Accounts
Deployment	Windows Server 2012: Group Managed Service Accounts - Ask Premier Field Engineering (PFE) Platforms - Site Home - TechNet Blogs
Related technologies	Security Principals What's new in Active Directory Domain Services

Active Directory Security Groups

3/26/2021 • 57 minutes to read • Edit Online

Applies to

- Windows Server 2016

This reference topic for the IT professional describes the default Active Directory security groups.

There are two forms of common security principals in Active Directory: user accounts and computer accounts. These accounts represent a physical entity (a person or a computer). User accounts can also be used as dedicated service accounts for some applications. Security groups are used to collect user accounts, computer accounts, and other groups into manageable units.

In the Windows Server operating system, there are several built-in accounts and security groups that are preconfigured with the appropriate rights and permissions to perform specific tasks. For Active Directory, there are two types of administrative responsibilities:

- **Service administrators** Responsible for maintaining and delivering Active Directory Domain Services (AD DS), including managing domain controllers and configuring the AD DS.
- **Data administrators** Responsible for maintaining the data that is stored in AD DS and on domain member servers and workstations.

About Active Directory groups

Groups are used to collect user accounts, computer accounts, and other groups into manageable units. Working with groups instead of with individual users helps simplify network maintenance and administration.

There are two types of groups in Active Directory:

- **Distribution groups** Used to create email distribution lists.
- **Security groups** Used to assign permissions to shared resources.

Distribution groups

Distribution groups can be used only with email applications (such as Exchange Server) to send email to collections of users. Distribution groups are not security enabled, which means that they cannot be listed in discretionary access control lists (DACLs).

Security groups

Security groups can provide an efficient way to assign access to resources on your network. By using security groups, you can:

- Assign user rights to security groups in Active Directory.

  User rights are assigned to a security group to determine what members of that group can do within the scope of a domain or forest. User rights are automatically assigned to some security groups when Active Directory is installed to help administrators define a person's administrative role in the domain.

  For example, a user who is added to the Backup Operators group in Active Directory has the ability to back up and restore files and directories that are located on each domain controller in the domain. This is possible because, by default, the user rights **Backup files and directories** and **Restore files and**

directories are automatically assigned to the Backup Operators group. Therefore, members of this group inherit the user rights that are assigned to that group.

You can use Group Policy to assign user rights to security groups to delegate specific tasks. For more information about using Group Policy, see User Rights Assignment.

- Assign permissions to security groups for resources.

  Permissions are different than user rights. Permissions are assigned to the security group for the shared resource. Permissions determine who can access the resource and the level of access, such as Full Control. Some permissions that are set on domain objects are automatically assigned to allow various levels of access to default security groups, such as the Account Operators group or the Domain Admins group.

  Security groups are listed in DACLs that define permissions on resources and objects. When assigning permissions for resources (file shares, printers, and so on), administrators should assign those permissions to a security group rather than to individual users. The permissions are assigned once to the group, instead of several times to each individual user. Each account that is added to a group receives the rights that are assigned to that group in Active Directory, and the user receives the permissions that are defined for that group.

Like distribution groups, security groups can be used as an email entity. Sending an email message to the group sends the message to all the members of the group.

Group scope

Groups are characterized by a scope that identifies the extent to which the group is applied in the domain tree or forest. The scope of the group defines where the group can be granted permissions. The following three group scopes are defined by Active Directory:

- Universal
- Global
- Domain Local

> **NOTE**
> In addition to these three scopes, the default groups in the **Builtin** container have a group scope of Builtin Local. This group scope and group type cannot be changed.

The following table lists the three group scopes and more information about each scope for a security group.

Group scopes

SCOPE	POSSIBLE MEMBERS	SCOPE CONVERSION	CAN GRANT PERMISSIONS	POSSIBLE MEMBER OF

SCOPE	POSSIBLE MEMBERS	SCOPE CONVERSION	CAN GRANT PERMISSIONS	POSSIBLE MEMBER OF
Universal	Accounts from any domain in the same forest Global groups from any domain in the same forest Other Universal groups from any domain in the same forest	Can be converted to Domain Local scope if the group is not a member of any other Universal groups Can be converted to Global scope if the group does not contain any other Universal groups	On any domain in the same forest or trusting forests	Other Universal groups in the same forest Domain Local groups in the same forest or trusting forests Local groups on computers in the same forest or trusting forests
Global	Accounts from the same domain Other Global groups from the same domain	Can be converted to Universal scope if the group is not a member of any other global group	On any domain in the same forest, or trusting domains or forests	Universal groups from any domain in the same forest Other Global groups from the same domain Domain Local groups from any domain in the same forest, or from any trusting domain
Domain Local	Accounts from any domain or any trusted domain Global groups from any domain or any trusted domain Universal groups from any domain in the same forest Other Domain Local groups from the same domain Accounts, Global groups, and Universal groups from other forests and from external domains	Can be converted to Universal scope if the group does not contain any other Domain Local groups	Within the same domain	Other Domain Local groups from the same domain Local groups on computers in the same domain, excluding built-in groups that have well-known SIDs

Special identity groups

Special identities are generally referred to as groups. Special identity groups do not have specific memberships that can be modified, but they can represent different users at different times, depending on the circumstances. Some of these groups include Creator Owner, Batch, and Authenticated User.

For information about all the special identity groups, see Special Identities.

Default security groups

Default groups, such as the Domain Admins group, are security groups that are created automatically when you create an Active Directory domain. You can use these predefined groups to help control access to shared resources and to delegate specific domain-wide administrative roles.

Many default groups are automatically assigned a set of user rights that authorize members of the group to perform specific actions in a domain, such as logging on to a local system or backing up files and folders. For example, a member of the Backup Operators group has the right to perform backup operations for all domain controllers in the domain.

When you add a user to a group, the user receives all the user rights that are assigned to the group and all the permissions that are assigned to the group for any shared resources.

Default groups are located in the **Builtin** container and in the **Users** container in Active Directory Users and Computers. The **Builtin** container includes groups that are defined with the Domain Local scope. The **Users** includes contains groups that are defined with Global scope and groups that are defined with Domain Local scope. You can move groups that are located in these containers to other groups or organizational units (OU) within the domain, but you cannot move them to other domains.

Some of the administrative groups that are listed in this topic and all members of these groups are protected by a background process that periodically checks for and applies a specific security descriptor. This descriptor is a data structure that contains security information associated with a protected object. This process ensures that any successful unauthorized attempt to modify the security descriptor on one of the administrative accounts or groups will be overwritten with the protected settings.

The security descriptor is present on the **AdminSDHolder** object. This means that if you want to modify the permissions on one of the service administrator groups or on any of its member accounts, you must modify the security descriptor on the **AdminSDHolder** object so that it will be applied consistently. Be careful when you make these modifications because you are also changing the default settings that will be applied to all of your protected administrative accounts.

Active Directory default security groups by operating system version

The following tables provide descriptions of the default groups that are located in the **Builtin** and **Users** containers in each operating system.

DEFAULT SECURITY GROUP	WINDOWS SERVER 2016	WINDOWS SERVER 2012 R2	WINDOWS SERVER 2012	WINDOWS SERVER 2008 R2
Access Control Assistance Operators	Yes	Yes	Yes	
Account Operators	Yes	Yes	Yes	Yes
Administrators	Yes	Yes	Yes	Yes

DEFAULT SECURITY GROUP	WINDOWS SERVER 2016	WINDOWS SERVER 2012 R2	WINDOWS SERVER 2012	WINDOWS SERVER 2008 R2
Allowed RODC Password Replication Group	Yes	Yes	Yes	Yes
Backup Operators	Yes	Yes	Yes	Yes
Certificate Service DCOM Access	Yes	Yes	Yes	Yes
Cert Publishers	Yes	Yes	Yes	Yes
Cloneable Domain Controllers	Yes	Yes	Yes	
Cryptographic Operators	Yes	Yes	Yes	Yes
Denied RODC Password Replication Group	Yes	Yes	Yes	Yes
Device Owners	Yes	Yes	Yes	Yes
Distributed COM Users	Yes	Yes	Yes	Yes
DnsUpdateProxy	Yes	Yes	Yes	Yes
DnsAdmins	Yes	Yes	Yes	Yes
Domain Admins	Yes	Yes	Yes	Yes
Domain Computers	Yes	Yes	Yes	Yes
Domain Controllers	Yes	Yes	Yes	Yes

DEFAULT SECURITY GROUP	WINDOWS SERVER 2016	WINDOWS SERVER 2012 R2	WINDOWS SERVER 2012	WINDOWS SERVER 2008 R2
Domain Guests	Yes	Yes	Yes	Yes
Domain Users	Yes	Yes	Yes	Yes
Enterprise Admins	Yes	Yes	Yes	Yes
Enterprise Key Admins	Yes			
Enterprise Read-only Domain Controllers	Yes	Yes	Yes	Yes
Event Log Readers	Yes	Yes	Yes	Yes
Group Policy Creator Owners	Yes	Yes	Yes	Yes
Guests	Yes	Yes	Yes	Yes
Hyper-V Administrators	Yes	Yes	Yes	
IIS_IUSRS	Yes	Yes	Yes	Yes
Incoming Forest Trust Builders	Yes	Yes	Yes	Yes
Key Admins	Yes			
Network Configuration Operators	Yes	Yes	Yes	Yes
Performance Log Users	Yes	Yes	Yes	Yes
Performance Monitor Users	Yes	Yes	Yes	Yes

DEFAULT SECURITY GROUP	WINDOWS SERVER 2016	WINDOWS SERVER 2012 R2	WINDOWS SERVER 2012	WINDOWS SERVER 2008 R2
Pre–Windows 2000 Compatible Access	Yes	Yes	Yes	Yes
Print Operators	Yes	Yes	Yes	Yes
Protected Users	Yes	Yes		
RAS and IAS Servers	Yes	Yes	Yes	Yes
RDS Endpoint Servers	Yes	Yes	Yes	
RDS Management Servers	Yes	Yes	Yes	
RDS Remote Access Servers	Yes	Yes	Yes	
Read-only Domain Controllers	Yes	Yes	Yes	Yes
Remote Desktop Users	Yes	Yes	Yes	Yes
Remote Management Users	Yes	Yes	Yes	
Replicator	Yes	Yes	Yes	Yes
Schema Admins	Yes	Yes	Yes	Yes
Server Operators	Yes	Yes	Yes	Yes
Storage Replica Administrators	Yes			

DEFAULT SECURITY GROUP	WINDOWS SERVER 2016	WINDOWS SERVER 2012 R2	WINDOWS SERVER 2012	WINDOWS SERVER 2008 R2
System Managed Accounts Group	Yes			
Terminal Server License Servers	Yes	Yes	Yes	Yes
Users	Yes	Yes	Yes	Yes
Windows Authorization Access Group	Yes	Yes	Yes	Yes
WinRMRemoteWMIUsers_		Yes	Yes	

Access Control Assistance Operators

Members of this group can remotely query authorization attributes and permissions for resources on the computer.

The Access Control Assistance Operators group applies to versions of the Windows Server operating system listed in the Active Directory Default Security Groups table.

This security group has not changed since Windows Server 2008.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-32-579
Type	Builtin Local
Default container	CN=BuiltIn, DC=<domain>, DC=
Default members	None
Default member of	None
Protected by ADMINSDHOLDER?	No
Safe to move out of default container?	Cannot be moved
Safe to delegate management of this group to non-Service admins?	

ATTRIBUTE	VALUE
Default User Rights	None

Account Operators

The Account Operators group grants limited account creation privileges to a user. Members of this group can create and modify most types of accounts, including those of users, local groups, and global groups, and members can log in locally to domain controllers.

Members of the Account Operators group cannot manage the Administrator user account, the user accounts of administrators, or the Administrators, Server Operators, Account Operators, Backup Operators, or Print Operators groups. Members of this group cannot modify user rights.

The Account Operators group applies to versions of the Windows Server operating system listed in the Active Directory Default Security Groups table.

> **NOTE**
> By default, this built-in group has no members, and it can create and manage users and groups in the domain, including its own membership and that of the Server Operators group. This group is considered a service administrator group because it can modify Server Operators, which in turn can modify domain controller settings. As a best practice, leave the membership of this group empty, and do not use it for any delegated administration. This group cannot be renamed, deleted, or moved.

This security group has not changed since Windows Server 2008.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-32-548
Type	Builtin Local
Default container	CN=BuiltIn, DC=<domain>, DC=
Default members	None
Default member of	None
Protected by ADMINSDHOLDER?	Yes
Safe to move out of default container?	Cannot be moved
Safe to delegate management of this group to non-Service admins?	No
Default User Rights	Allow log on locally: SeInteractiveLogonRight

Administrators

Members of the Administrators group have complete and unrestricted access to the computer, or if the computer is promoted to a domain controller, members have unrestricted access to the domain.

The Administrators group applies to versions of the Windows Server operating system listed in the Active Directory Default Security Groups table.

> **NOTE**
> The Administrators group has built-in capabilities that give its members full control over the system. This group cannot be renamed, deleted, or moved. This built-in group controls access to all the domain controllers in its domain, and it can change the membership of all administrative groups.

Membership can be modified by members of the following groups: the default service Administrators, Domain Admins in the domain, or Enterprise Admins. This group has the special privilege to take ownership of any object in the directory or any resource on a domain controller. This account is considered a service administrator group because its members have full access to the domain controllers in the domain.

This security group includes the following changes since Windows Server 2008:

- Default user rights changes: **Allow log on through Terminal Services** existed in Windows Server 2008, and it was replaced by Allow log on through Remote Desktop Services.
- Remove computer from docking station was removed in Windows Server 2012 R2.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-32-544
Type	Builtin Local
Default container	CN=BuiltIn, DC=<domain>, DC=
Default members	Administrator, Domain Admins, Enterprise Admins
Default member of	None
Protected by ADMINSDHOLDER?	Yes
Safe to move out of default container?	Cannot be moved
Safe to delegate management of this group to non-Service admins?	No

ATTRIBUTE	VALUE
Default User Rights	Adjust memory quotas for a process: SeIncreaseQuotaPrivilege Access this computer from the network: SeNetworkLogonRight Allow log on locally: SeInteractiveLogonRight Allow log on through Remote Desktop Services: SeRemoteInteractiveLogonRight Back up files and directories: SeBackupPrivilege Bypass traverse checking: SeChangeNotifyPrivilege Change the system time: SeSystemTimePrivilege Change the time zone: SeTimeZonePrivilege Create a pagefile: SeCreatePagefilePrivilege Create global objects: SeCreateGlobalPrivilege Create symbolic links: SeCreateSymbolicLinkPrivilege Debug programs: SeDebugPrivilege Enable computer and user accounts to be trusted for delegation: SeEnableDelegationPrivilege Force shutdown from a remote system: SeRemoteShutdownPrivilege Impersonate a client after authentication: SeImpersonatePrivilege Increase scheduling priority: SeIncreaseBasePriorityPrivilege Load and unload device drivers: SeLoadDriverPrivilege Log on as a batch job: SeBatchLogonRight Manage auditing and security log: SeSecurityPrivilege Modify firmware environment values: SeSystemEnvironmentPrivilege Perform volume maintenance tasks: SeManageVolumePrivilege Profile system performance: SeSystemProfilePrivilege Profile single process: SeProfileSingleProcessPrivilege Remove computer from docking station: SeUndockPrivilege Restore files and directories: SeRestorePrivilege Shut down the system: SeShutdownPrivilege Take ownership of files or other objects: SeTakeOwnershipPrivilege

Allowed RODC Password Replication Group

The purpose of this security group is to manage a RODC password replication policy. This group has no members by default, and it results in the condition that new Read-only domain controllers do not cache user credentials. The Denied RODC Password Replication Group group contains a variety of high-privilege accounts and security groups. The Denied RODC Password Replication group supersedes the Allowed RODC Password

Replication group.

The Allowed RODC Password Replication group applies to versions of the Windows Server operating system listed in the Active Directory Default Security Groups table.

This security group has not changed since Windows Server 2008.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-21-<domain>-571
Type	Domain local
Default container	CN=Users DC=<domain>, DC=
Default members	None
Default member of	None
Protected by ADMINSDHOLDER?	No
Safe to move out of default container?	Cannot be moved
Safe to delegate management of this group to non-Service admins?	
Default User Rights	None

Backup Operators

Members of the Backup Operators group can back up and restore all files on a computer, regardless of the permissions that protect those files. Backup Operators also can log on to and shut down the computer. This group cannot be renamed, deleted, or moved. By default, this built-in group has no members, and it can perform backup and restore operations on domain controllers. Its membership can be modified by the following groups: default service Administrators, Domain Admins in the domain, or Enterprise Admins. It cannot modify the membership of any administrative groups. While members of this group cannot change server settings or modify the configuration of the directory, they do have the permissions needed to replace files (including operating system files) on domain controllers. Because of this, members of this group are considered service administrators.

The Backup Operators group applies to versions of the Windows Server operating system listed in the Active Directory Default Security Groups table.

This security group has not changed since Windows Server 2008.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-32-551

ATTRIBUTE	VALUE
Type	Builtin Local
Default container	CN=BuiltIn, DC=<domain>, DC=
Default members	None
Default member of	None
Protected by ADMINSDHOLDER?	Yes
Safe to move out of default container?	Cannot be moved
Safe to delegate management of this group to non-Service admins?	No
Default User Rights	Allow log on locally: SeInteractiveLogonRight Back up files and directories: SeBackupPrivilege Log on as a batch job: SeBatchLogonRight Restore files and directories: SeRestorePrivilege Shut down the system: SeShutdownPrivilege

Certificate Service DCOM Access

Members of this group are allowed to connect to certification authorities in the enterprise.

The Certificate Service DCOM Access group applies to versions of the Windows Server operating system listed in the Active Directory Default Security Groups table.

This security group has not changed since Windows Server 2008.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-32-<domain>-574
Type	Domain Local
Default container	CN=Builtin, DC=<domain>, DC=
Default members	None
Default member of	None

ATTRIBUTE	VALUE
Protected by ADMINSDHOLDER?	No
Safe to move out of default container?	Cannot be moved
Safe to delegate management of this group to non-Service admins?	
Default User Rights	None

Cert Publishers

Members of the Cert Publishers group are authorized to publish certificates for User objects in Active Directory.

The Cert Publishers group applies to versions of the Windows Server operating system listed in the Active Directory Default Security Groups table.

This security group has not changed since Windows Server 2008.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-21-<domain>-517
Type	Domain Local
Default container	CN=Users, DC=<domain>, DC=
Default members	None
Default member of	Denied RODC Password Replication Group
Protected by ADMINSDHOLDER?	No
Safe to move out of default container?	Cannot be moved
Safe to delegate management of this group to non-Service admins?	No
Default User Rights	None

Cloneable Domain Controllers

Members of the Cloneable Domain Controllers group that are domain controllers may be cloned. In Windows Server 2012 R2 and Windows Server 2012, you can deploy domain controllers by copying an existing virtual domain controller. In a virtual environment, you no longer have to repeatedly deploy a server image that is

prepared by using sysprep.exe, promote the server to a domain controller, and then complete additional configuration requirements for deploying each domain controller (including adding the virtual domain controller to this security group).

For more information, see Introduction to Active Directory Domain Services (AD DS) Virtualization (Level 100).

This security group was introduced in Windows Server 2012, and it has not changed in subsequent versions.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-21-<domain>-522
Type	Global
Default container	CN=Users, DC=<domain>, DC=
Default members	None
Default member of	None
Protected by ADMINSDHOLDER?	No
Safe to move out of default container?	Cannot be moved
Safe to delegate management of this group to non-Service admins?	
Default User Rights	None

Cryptographic Operators

Members of this group are authorized to perform cryptographic operations. This security group was added in Windows Vista Service Pack 1 (SP1) to configure Windows Firewall for IPsec in Common Criteria mode.

The Cryptographic Operators group applies to versions of the Windows Server operating system listed in the Active Directory Default Security Groups table.

This security group was introduced in Windows Vista Service Pack 1, and it has not changed in subsequent versions.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-32-569
Type	Builtin Local
Default container	CN=Builtin, DC=<domain>, DC=

ATTRIBUTE	VALUE
Default members	None
Default member of	None
Protected by ADMINSDHOLDER?	No
Safe to move out of default container?	Cannot be moved
Safe to delegate management of this group to non-Service admins?	
Default User Rights	None

Denied RODC Password Replication Group

Members of the Denied RODC Password Replication group cannot have their passwords replicated to any Read-only domain controller.

The purpose of this security group is to manage a RODC password replication policy. This group contains a variety of high-privilege accounts and security groups. The Denied RODC Password Replication Group supersedes the Allowed RODC Password Replication Group.

This security group includes the following changes since Windows Server 2008:

- Windows Server 2012 changed the default members to include Cert Publishers.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-21-<domain>-572
Type	Domain local
Default container	CN=Users, DC=<domain>, DC=
Default members	Cert Publishers Domain Admins Domain Controllers Enterprise Admins Group Policy Creator Owners krbtgt Read-only Domain Controllers Schema Admins

ATTRIBUTE	VALUE
Default member of	None
Protected by ADMINSDHOLDER?	No
Safe to move out of default container?	
Safe to delegate management of this group to non-Service admins?	
Default User Rights	None

Device Owners

This group is not currently used in Windows.

Microsoft does not recommend changing the default configuration where this security group has zero members. Changing the default configuration could hinder future scenarios that rely on this group.

The Device Owners group applies to versions of the Windows Server operating system listed in the Active Directory Default Security Groups table.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-32-583
Type	Builtin Local
Default container	CN=BuiltIn, DC=<domain>, DC=
Default members	None
Default member of	None
Protected by ADMINSDHOLDER?	No
Safe to move out of default container?	Can be moved out but it is not recommended
Safe to delegate management of this group to non-Service admins?	No

ATTRIBUTE	VALUE
Default User Rights	Allow log on locally: SeInteractiveLogonRight Access this computer from the network: SeNetworkLogonRight Bypass traverse checking: SeChangeNotifyPrivilege Change the time zone: SeTimeZonePrivilege

Distributed COM Users

Members of the Distributed COM Users group are allowed to launch, activate, and use Distributed COM objects on the computer. Microsoft Component Object Model (COM) is a platform-independent, distributed, object-oriented system for creating binary software components that can interact. Distributed Component Object Model (DCOM) allows applications to be distributed across locations that make the most sense to you and to the application. This group appears as a SID until the domain controller is made the primary domain controller and it holds the operations master role (also known as flexible single master operations or FSMO).

The Distributed COM Users group applies to versions of the Windows Server operating system listed in the Active Directory Default Security Groups table.

This security group has not changed since Windows Server 2008.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-32-562
Type	Builtin Local
Default container	CN=Builtin, DC=<domain>, DC=
Default members	None
Default member of	None
Protected by ADMINSDHOLDER?	No
Safe to move out of default container?	Cannot be moved
Safe to delegate management of this group to non-Service admins?	
Default User Rights	None

DnsUpdateProxy

Members of the DnsUpdateProxy group are DNS clients. They are permitted to perform dynamic updates on behalf of other clients (such as DHCP servers). A DNS server can develop stale resource records when a DHCP server is configured to dynamically register host (A) and pointer (PTR) resource records on behalf of DHCP

clients by using dynamic update. Adding clients to this security group mitigates this scenario.

However, to protect against unsecured records or to permit members of the DnsUpdateProxy group to register records in zones that allow only secured dynamic updates, you must create a dedicated user account and configure DHCP servers to perform DNS dynamic updates by using the credentials of this account (user name, password, and domain). Multiple DHCP servers can use the credentials of one dedicated user account. This group exists only if the DNS server role is or was once installed on a domain controller in the domain.

For information, see DNS Record Ownership and the DnsUpdateProxy Group.

This security group has not changed since Windows Server 2008.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-21-<domain>-<variable RID>
Type	Global
Default container	CN=Users, DC=<domain>, DC=
Default members	None
Default member of	None
Protected by ADMINSDHOLDER?	No
Safe to move out of default container?	Yes
Safe to delegate management of this group to non-Service admins?	
Default User Rights	None

DnsAdmins

Members of DNSAdmins group have access to network DNS information. The default permissions are as follows: Allow: Read, Write, Create All Child objects, Delete Child objects, Special Permissions. This group exists only if the DNS server role is or was once installed on a domain controller in the domain.

For more information about security and DNS, see DNSSEC in Windows Server 2012.

This security group has not changed since Windows Server 2008.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-21-<domain>-<variable RID>
Type	Builtin Local

ATTRIBUTE	VALUE
Default container	CN=Users, DC=<domain>, DC=
Default members	None
Default member of	None
Protected by ADMINSDHOLDER?	No
Safe to move out of default container?	Yes
Safe to delegate management of this group to non-Service admins?	
Default User Rights	None

Domain Admins

Members of the Domain Admins security group are authorized to administer the domain. By default, the Domain Admins group is a member of the Administrators group on all computers that have joined a domain, including the domain controllers. The Domain Admins group is the default owner of any object that is created in Active Directory for the domain by any member of the group. If members of the group create other objects, such as files, the default owner is the Administrators group.

The Domain Admins group controls access to all domain controllers in a domain, and it can modify the membership of all administrative accounts in the domain. Membership can be modified by members of the service administrator groups in its domain (Administrators and Domain Admins), and by members of the Enterprise Admins group. This is considered a service administrator account because its members have full access to the domain controllers in a domain.

The Domain Admins group applies to versions of the Windows Server operating system listed in the Active Directory Default Security Groups table.

This security group has not changed since Windows Server 2008.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-<domain>-512
Type	Global
Default container	CN=Users, DC=<domain>, DC=
Default members	Administrator

ATTRIBUTE	VALUE
Default member of	Administrators Denied RODC Password ReplicationGroup
Protected by ADMINSDHOLDER?	Yes
Safe to move out of default container?	Yes
Safe to delegate management of this group to non-Service admins?	No
Default User Rights	See Administrators See Denied RODC Password Replication Group

Domain Computers

This group can include all computers and servers that have joined the domain, excluding domain controllers. By default, any computer account that is created automatically becomes a member of this group.

The Domain Computers group applies to versions of the Windows Server operating system listed in the Active Directory Default Security Groups table.

This security group has not changed since Windows Server 2008.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-21-<domain>-515
Type	Global
Default container	CN=Users, DC=<domain>, DC=
Default members	All computers joined to the domain, excluding domain controllers
Default member of	None
Protected by ADMINSDHOLDER?	No
Safe to move out of default container?	Yes (but not required)
Safe to delegate management of this group to non-Service admins?	Yes

ATTRIBUTE	VALUE
Default User Rights	None

Domain Controllers

The Domain Controllers group can include all domain controllers in the domain. New domain controllers are automatically added to this group.

The Domain Controllers group applies to versions of the Windows Server operating system listed in the Active Directory Default Security Groups table.

This security group has not changed since Windows Server 2008.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-21-<domain>-516
Type	Global
Default container	CN=Users, DC=<domain>, DC=
Default members	Computer accounts for all domain controllers of the domain
Default member of	Denied RODC Password Replication Group
Protected by ADMINSDHOLDER?	Yes
Safe to move out of default container?	No
Safe to delegate management of this group to non-Service admins?	No
Default User Rights	None

Domain Guests

The Domain Guests group includes the domain's built-in Guest account. When members of this group sign in as local guests on a domain-joined computer, a domain profile is created on the local computer.

The Domain Guests group applies to versions of the Windows Server operating system listed in the Active Directory Default Security Groups table.

This security group has not changed since Windows Server 2008.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-21-<domain>-514
Type	Global
Default container	CN=Users, DC=<domain>, DC=
Default members	Guest
Default member of	Guests
Protected by ADMINSDHOLDER?	Yes
Safe to move out of default container?	Can be moved out but it is not recommended
Safe to delegate management of this group to non-Service admins?	No
Default User Rights	See Guests

Domain Users

The Domain Users group includes all user accounts in a domain. When you create a user account in a domain, it is automatically added to this group.

By default, any user account that is created in the domain automatically becomes a member of this group. This group can be used to represent all users in the domain. For example, if you want all domain users to have access to a printer, you can assign permissions for the printer to this group (or add the Domain Users group to a local group on the print server that has permissions for the printer).

The Domain Users group applies to versions of the Windows Server operating system listed in the Active Directory Default Security Groups table.

This security group has not changed since Windows Server 2008.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-21-<domain>-513
Type	Global
Default container	CN=Users, DC=<domain>, DC=

ATTRIBUTE	VALUE
Default members	Administrator krbtgt
Default member of	Users
Protected by ADMINSDHOLDER?	No
Safe to move out of default container?	Yes
Safe to delegate management of this group to non-Service admins?	No
Default User Rights	See Users

Enterprise Admins

The Enterprise Admins group exists only in the root domain of an Active Directory forest of domains. It is a Universal group if the domain is in native mode; it is a Global group if the domain is in mixed mode. Members of this group are authorized to make forest-wide changes in Active Directory, such as adding child domains.

By default, the only member of the group is the Administrator account for the forest root domain. This group is automatically added to the Administrators group in every domain in the forest, and it provides complete access for configuring all domain controllers. Members in this group can modify the membership of all administrative groups. Membership can be modified only by the default service administrator groups in the root domain. This is considered a service administrator account.

The Enterprise Admins group applies to versions of the Windows Server operating system listed in the Active Directory Default Security Groups table.

This security group has not changed since Windows Server 2008.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-21-<root domain>-519
Type	Universal (if Domain is in Native-Mode) else Global
Default container	CN=Users, DC=<domain>, DC=
Default members	Administrator
Default member of	Administrators Denied RODC Password Replication Group

ATTRIBUTE	VALUE
Protected by ADMINSDHOLDER?	Yes
Safe to move out of default container?	Yes
Safe to delegate management of this group to non-Service admins?	No
Default User Rights	See Administrators See Denied RODC Password Replication Group

Enterprise Key Admins

Members of this group can perform administrative actions on key objects within the forest.

The Enterprise Key Admins group was introduced in Windows Server 2016.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-21-<domain>-527
Type	Global
Default container	CN=Users, DC=<domain>, DC=
Default members	None
Default member of	None
Protected by ADMINSDHOLDER?	Yes
Safe to move out of default container?	Yes
Safe to delegate management of this group to non-Service admins?	No
Default User Rights	None

Enterprise Read-Only Domain Controllers

Members of this group are Read-Only Domain Controllers in the enterprise. Except for account passwords, a Read-only domain controller holds all the Active Directory objects and attributes that a writable domain controller holds. However, changes cannot be made to the database that is stored on the Read-only domain controller. Changes must be made on a writable domain controller and then replicated to the Read-only domain controller.

Read-only domain controllers address some of the issues that are commonly found in branch offices. These locations might not have a domain controller. Or, they might have a writable domain controller, but not the physical security, network bandwidth, or local expertise to support it.

For more information, see What Is an RODC?.

The Enterprise Read-Only Domain Controllers group applies to versions of the Windows Server operating system listed in the Active Directory Default Security Groups table.

This security group has not changed since Windows Server 2008.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-21-<domain>-498
Type	Universal
Default container	CN=Users, DC=<domain>, DC=
Default members	None
Default member of	None
Protected by ADMINSDHOLDER?	Yes
Safe to move out of default container?	
Safe to delegate management of this group to non-Service admins?	
Default User Rights	None

Event Log Readers

Members of this group can read event logs from local computers. The group is created when the server is promoted to a domain controller.

The Event Log Readers group applies to versions of the Windows Server operating system listed in the Active Directory Default Security Groups table.

This security group has not changed since Windows Server 2008.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-32-573
Type	Domain Local
Default container	CN=Users, DC=<domain>, DC=
Default members	None

ATTRIBUTE	VALUE
Default member of	None
Protected by ADMINSDHOLDER?	No
Safe to move out of default container?	Cannot be moved
Safe to delegate management of this group to non-Service admins?	
Default User Rights	None

Group Policy Creator Owners

This group is authorized to create, edit, or delete Group Policy Objects in the domain. By default, the only member of the group is Administrator.

For information about other features you can use with this security group, see Group Policy Overview.

The Group Policy Creator Owners group applies to versions of the Windows Server operating system listed in the Active Directory Default Security Groups table.

This security group has not changed since Windows Server 2008.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-21-<domain>-520
Type	Global
Default container	CN=Users, DC=<domain>, DC=
Default members	Administrator
Default member of	Denied RODC Password Replication Group
Protected by ADMINSDHOLDER?	No
Safe to move out of default container?	No
Safe to delegate management of this group to non-Service admins?	No
Default User Rights	See Denied RODC Password Replication Group

Guests

Members of the Guests group have the same access as members of the Users group by default, except that the Guest account has further restrictions. By default, the only member is the Guest account. The Guests group allows occasional or one-time users to sign in with limited privileges to a computer's built-in Guest account.

When a member of the Guests group signs out, the entire profile is deleted. This includes everything that is stored in the **%userprofile%** directory, including the user's registry hive information, custom desktop icons, and other user-specific settings. This implies that a guest must use a temporary profile to sign in to the system. This security group interacts with the Group Policy setting **Do not logon users with temporary profiles** when it is enabled. This setting is located under the following path:

Computer Configuration\Administrative Templates\System\User Profiles

> **NOTE**
> A Guest account is a default member of the Guests security group. People who do not have an actual account in the domain can use the Guest account. A user whose account is disabled (but not deleted) can also use the Guest account.

The Guest account does not require a password. You can set rights and permissions for the Guest account as in any user account. By default, the Guest account is a member of the built-in Guests group and the Domain Guests global group, which allows a user to sign in to a domain. The Guest account is disabled by default, and we recommend that it stay disabled.

The Guests group applies to versions of the Windows Server operating system listed in the Active Directory Default Security Groups table.

This security group has not changed since Windows Server 2008.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-32-546
Type	Builtin Local
Default container	CN=BuiltIn, DC=<domain>, DC=
Default members	Domain Guests Guest
Default member of	None
Protected by ADMINSDHOLDER?	No
Safe to move out of default container?	Cannot be moved
Safe to delegate management of this group to non-Service admins?	No

ATTRIBUTE	VALUE
Default User Rights	None

Hyper-V Administrators

Members of the Hyper-V Administrators group have complete and unrestricted access to all the features in Hyper-V. Adding members to this group helps reduce the number of members required in the Administrators group, and further separates access.

> **NOTE**
> Prior to Windows Server 2012, access to features in Hyper-V was controlled in part by membership in the Administrators group.

This security group was introduced in Windows Server 2012, and it has not changed in subsequent versions.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-32-578
Type	Builtin Local
Default container	CN=BuiltIn, DC=<domain>, DC=
Default members	None
Default member of	None
Protected by ADMINSDHOLDER?	No
Safe to move out of default container?	Cannot be moved
Safe to delegate management of this group to non-Service admins?	
Default User Rights	None

IIS_IUSRS

IIS_IUSRS is a built-in group that is used by Internet Information Services beginning with IIS 7.0. A built-in account and group are guaranteed by the operating system to always have a unique SID. IIS 7.0 replaces the IUSR_MachineName account and the IIS_WPG group with the IIS_IUSRS group to ensure that the actual names that are used by the new account and group will never be localized. For example, regardless of the language of the Windows operating system that you install, the IIS account name will always be IUSR, and the group name will be IIS_IUSRS.

For more information, see Understanding Built-In User and Group Accounts in IIS 7.

This security group has not changed since Windows Server 2008.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-32-568
Type	Builtin Local
Default container	CN=BuiltIn, DC=<domain>, DC=
Default members	IUSR
Default member of	None
Protected by ADMINSDHOLDER?	No
Safe to move out of default container?	
Safe to delegate management of this group to non-Service admins?	
Default User Rights	None

Incoming Forest Trust Builders

Members of the Incoming Forest Trust Builders group can create incoming, one-way trusts to this forest. Active Directory provides security across multiple domains or forests through domain and forest trust relationships. Before authentication can occur across trusts, Windows must determine whether the domain being requested by a user, computer, or service has a trust relationship with the logon domain of the requesting account.

To make this determination, the Windows security system computes a trust path between the domain controller for the server that receives the request and a domain controller in the domain of the requesting account. A secured channel extends to other Active Directory domains through interdomain trust relationships. This secured channel is used to obtain and verify security information, including security identifiers (SIDs) for users and groups.

> **NOTE**
> This group appears as a SID until the domain controller is made the primary domain controller and it holds the operations master role (also known as flexible single master operations or FSMO).

For more information, see How Domain and Forest Trusts Work: Domain and Forest Trusts.

The Incoming Forest Trust Builders group applies to versions of the Windows Server operating system listed in the Active Directory Default Security Groups table.

> **NOTE**
> This group cannot be renamed, deleted, or moved.

This security group has not changed since Windows Server 2008.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-32-557
Type	Builtin Local
Default container	CN=Builtin, DC=<domain>, DC=
Default members	None
Default member of	None
Protected by ADMINSDHOLDER?	No
Safe to move out of default container?	Cannot be moved
Safe to delegate management of this group to non-Service admins?	No
Default User Rights	None

Key Admins

Members of this group can perform administrative actions on key objects within the domain.

The Key Admins group applies to versions of the Windows Server operating system listed in the Active Directory Default Security Groups table.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-21-<domain>-526
Type	Global
Default container	CN=Users, DC=<domain>, DC=
Default members	None
Default member of	None
Protected by ADMINSDHOLDER?	Yes

ATTRIBUTE	VALUE
Safe to move out of default container?	Yes
Safe to delegate management of this group to non-Service admins?	No
Default User Rights	None

Network Configuration Operators

Members of the Network Configuration Operators group can have the following administrative privileges to manage configuration of networking features:

- Modify the Transmission Control Protocol/Internet Protocol (TCP/IP) properties for a local area network (LAN) connection, which includes the IP address, the subnet mask, the default gateway, and the name servers.
- Rename the LAN connections or remote access connections that are available to all the users.
- Enable or disable a LAN connection.
- Modify the properties of all of remote access connections of users.
- Delete all the remote access connections of users.
- Rename all the remote access connections of users.
- Issue **ipconfig**, **ipconfig /release**, or **ipconfig /renew** commands.
- Enter the PIN unblock key (PUK) for mobile broadband devices that support a SIM card.

> **NOTE**
> This group appears as a SID until the domain controller is made the primary domain controller and it holds the operations master role (also known as flexible single master operations or FSMO).

The Network Configuration Operators group applies to versions of the Windows Server operating system listed in the Active Directory Default Security Groups table.

> **NOTE**
> This group cannot be renamed, deleted, or moved.

This security group has not changed since Windows Server 2008.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-32-556
Type	Builtin Local
Default container	CN=Builtin, DC=<domain>, DC=

ATTRIBUTE	VALUE
Default members	None
Default member of	None
Protected by ADMINSDHOLDER?	No
Safe to move out of default container?	Cannot be moved
Safe to delegate management of this group to non-Service admins?	Yes
Default User Rights	None

Performance Log Users

Members of the Performance Log Users group can manage performance counters, logs, and alerts locally on the server and from remote clients without being a member of the Administrators group. Specifically, members of this security group:

- Can use all the features that are available to the Performance Monitor Users group.
- Can create and modify Data Collector Sets after the group is assigned the Log on as a batch job user right.

  > **WARNING**
  > If you are a member of the Performance Log Users group, you must configure Data Collector Sets that you create to run under your credentials.

- Cannot use the Windows Kernel Trace event provider in Data Collector Sets.

For members of the Performance Log Users group to initiate data logging or modify Data Collector Sets, the group must first be assigned the Log on as a batch job user right. To assign this user right, use the Local Security Policy snap-in in Microsoft Management Console.

> **NOTE**
> This group appears as a SID until the domain controller is made the primary domain controller and it holds the operations master role (also known as flexible single master operations or FSMO).

The Performance Log Users group applies to versions of the Windows Server operating system listed in the Active Directory Default Security Groups table.

> **NOTE**
> This account cannot be renamed, deleted, or moved.

This security group has not changed since Windows Server 2008.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-32-559
Type	Builtin Local
Default container	CN=Builtin, DC=<domain>, DC=
Default members	None
Default member of	None
Protected by ADMINSDHOLDER?	No
Safe to move out of default container?	Cannot be moved
Safe to delegate management of this group to non-Service admins?	Yes
Default User Rights	Log on as a batch job: SeBatchLogonRight

Performance Monitor Users

Members of this group can monitor performance counters on domain controllers in the domain, locally and from remote clients, without being a member of the Administrators or Performance Log Users groups. The Windows Performance Monitor is a Microsoft Management Console (MMC) snap-in that provides tools for analyzing system performance. From a single console, you can monitor application and hardware performance, customize what data you want to collect in logs, define thresholds for alerts and automatic actions, generate reports, and view past performance data in a variety of ways.

Specifically, members of this security group:

- Can use all the features that are available to the Users group.
- Can view real-time performance data in Performance Monitor.

  Can change the Performance Monitor display properties while viewing data.
- Cannot create or modify Data Collector Sets.

> **WARNING**
> You cannot configure a Data Collector Set to run as a member of the Performance Monitor Users group.

> **NOTE**
> This group appears as a SID until the domain controller is made the primary domain controller and it holds the operations master role (also known as flexible single master operations or FSMO). This group cannot be renamed, deleted, or moved.

The Performance Monitor Users group applies to versions of the Windows Server operating system listed in the Active Directory Default Security Groups table.

This security group has not changed since Windows Server 2008.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-32-558
Type	Builtin Local
Default container	CN=Builtin, DC=<domain>, DC=
Default members	None
Default member of	None
Protected by ADMINSDHOLDER?	No
Safe to move out of default container?	Cannot be moved
Safe to delegate management of this group to non-Service admins?	Yes
Default User Rights	None

Pre–Windows 2000 Compatible Access

Members of the Pre–Windows 2000 Compatible Access group have Read access for all users and groups in the domain. This group is provided for backward compatibility for computers running Windows NT 4.0 and earlier. By default, the special identity group, Everyone, is a member of this group. Add users to this group only if they are running Windows NT 4.0 or earlier.

> **WARNING**
> This group appears as a SID until the domain controller is made the primary domain controller and it holds the operations master role (also known as flexible single master operations or FSMO).

The Pre–Windows 2000 Compatible Access group applies to versions of the Windows Server operating system listed in the Active Directory Default Security Groups table.

This security group has not changed since Windows Server 2008.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-32-554
Type	Builtin Local
Default container	CN=Builtin, DC=<domain>, DC=
Default members	If you choose the Pre–Windows 2000 Compatible Permissions mode, Everyone and Anonymous are members, and if you choose the Windows 2000-only permissions mode, Authenticated Users are members.
Default member of	None
Protected by ADMINSDHOLDER?	No
Safe to move out of default container?	Cannot be moved
Safe to delegate management of this group to non-Service admins?	No
Default User Rights	Access this computer from the network: SeNetworkLogonRight Bypass traverse checking: SeChangeNotifyPrivilege

Print Operators

Members of this group can manage, create, share, and delete printers that are connected to domain controllers in the domain. They can also manage Active Directory printer objects in the domain. Members of this group can locally sign in to and shut down domain controllers in the domain.

This group has no default members. Because members of this group can load and unload device drivers on all domain controllers in the domain, add users with caution. This group cannot be renamed, deleted, or moved.

The Print Operators group applies to versions of the Windows Server operating system listed in the Active Directory Default Security Groups table.

This security group has not changed since Windows Server 2008. However, in Windows Server 2008 R2, functionality was added to manage print administration. For more information, see Assign Delegated Print Administrator and Printer Permission Settings in Windows Server 2012.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-32-550
Type	Builtin Local

ATTRIBUTE	VALUE
Default container	CN=Builtin, DC=<domain>, DC=
Default members	None
Default member of	None
Protected by ADMINSDHOLDER?	Yes
Safe to move out of default container?	Cannot be moved
Safe to delegate management of this group to non-Service admins?	No
Default User Rights	Allow log on locally: SeInteractiveLogonRight Load and unload device drivers: SeLoadDriverPrivilege Shut down the system: SeShutdownPrivilege

Protected Users

Members of the Protected Users group are afforded additional protection against the compromise of credentials during authentication processes.

This security group is designed as part of a strategy to effectively protect and manage credentials within the enterprise. Members of this group automatically have non-configurable protection applied to their accounts. Membership in the Protected Users group is meant to be restrictive and proactively secure by default. The only method to modify the protection for an account is to remove the account from the security group.

This domain-related, global group triggers non-configurable protection on devices and host computers, starting with the Windows Server 2012 R2 and Windows 8.1 operating systems. It also triggers non-configurable protection on domain controllers in domains with a primary domain controller running Windows Server 2012 R2 or Windows Server 2016. This greatly reduces the memory footprint of credentials when users sign in to computers on the network from a non-compromised computer.

Depending on the account's domain functional level, members of the Protected Users group are further protected due to behavior changes in the authentication methods that are supported in Windows.

- Members of the Protected Users group cannot authenticate by using the following Security Support Providers (SSPs): NTLM, Digest Authentication, or CredSSP. Passwords are not cached on a device running Windows 8.1 or Windows 10, so the device fails to authenticate to a domain when the account is a member of the Protected User group.
- The Kerberos protocol will not use the weaker DES or RC4 encryption types in the preauthentication process. This means that the domain must be configured to support at least the AES cipher suite.
- The user's account cannot be delegated with Kerberos constrained or unconstrained delegation. This means that former connections to other systems may fail if the user is a member of the Protected Users group.
- The default Kerberos ticket-granting tickets (TGTs) lifetime setting of four hours is configurable by using

Authentication Policies and Silos, which can be accessed through the Active Directory Administrative Center. This means that when four hours has passed, the user must authenticate again.

The Protected Users group applies to versions of the Windows Server operating system listed in the Active Directory Default Security Groups table.

This group was introduced in Windows Server 2012 R2. For more information about how this group works, see Protected Users Security Group.

The following table specifies the properties of the Protected Users group.

ATTRIBUTE	VALUE
Well-known SID/RID	S-1-5-21-<domain>-525
Type	Global
Default container	CN=Users, DC=<domain>, DC=
Default members	None
Default member of	None
Protected by ADMINSDHOLDER?	No
Safe to move out of default container?	Yes
Safe to delegate management of this group to non-service admins?	No
Default user rights	None

RAS and IAS Servers

Computers that are members of the RAS and IAS Servers group, when properly configured, are allowed to use remote access services. By default, this group has no members. Computers that are running the Routing and Remote Access service are added to the group automatically, such as IAS servers and Network Policy Servers. Members of this group have access to certain properties of User objects, such as Read Account Restrictions, Read Logon Information, and Read Remote Access Information.

The RAS and IAS Servers group applies to versions of the Windows Server operating system listed in the Active Directory Default Security Groups table.

This security group has not changed since Windows Server 2008.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-21-<domain>-553

ATTRIBUTE	VALUE
Type	Builtin Local
Default container	CN=Users, DC=<domain>, DC=
Default members	None
Default member of	None
Protected by ADMINSDHOLDER?	No
Safe to move out of default container?	Yes
Safe to delegate management of this group to non-Service admins?	Yes
Default User Rights	None

RDS Endpoint Servers

Servers that are members in the RDS Endpoint Servers group can run virtual machines and host sessions where user RemoteApp programs and personal virtual desktops run. This group needs to be populated on servers running RD Connection Broker. Session Host servers and RD Virtualization Host servers used in the deployment need to be in this group.

For information about Remote Desktop Services, see Host desktops and apps in Remote Desktop Services.

This security group was introduced in Windows Server 2012, and it has not changed in subsequent versions.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-32-576
Type	Builtin Local
Default container	CN=Builtin, DC=<domain>, DC=
Default members	None
Default member of	None
Protected by ADMINSDHOLDER?	No

ATTRIBUTE	VALUE
Safe to move out of default container?	Cannot be moved
Safe to delegate management of this group to non-Service admins?	
Default User Rights	None

RDS Management Servers

Servers that are members in the RDS Management Servers group can be used to perform routine administrative actions on servers running Remote Desktop Services. This group needs to be populated on all servers in a Remote Desktop Services deployment. The servers running the RDS Central Management service must be included in this group.

This security group was introduced in Windows Server 2012, and it has not changed in subsequent versions.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-32-577
Type	Builtin Local
Default container	CN=Builtin, DC=<domain>, DC=
Default members	None
Default member of	None
Protected by ADMINSDHOLDER?	No
Safe to move out of default container?	Cannot be moved
Safe to delegate management of this group to non-Service admins?	
Default User Rights	None

RDS Remote Access Servers

Servers in the RDS Remote Access Servers group provide users with access to RemoteApp programs and personal virtual desktops. In Internet facing deployments, these servers are typically deployed in an edge network. This group needs to be populated on servers running RD Connection Broker. RD Gateway servers and RD Web Access servers that are used in the deployment need to be in this group.

For more information, see Host desktops and apps in Remote Desktop Services.

This security group was introduced in Windows Server 2012, and it has not changed in subsequent versions.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-32-575
Type	Builtin Local
Default container	CN=Builtin, DC=<domain>, DC=
Default members	None
Default member of	None
Protected by ADMINSDHOLDER?	No
Safe to move out of default container?	Cannot be moved
Safe to delegate management of this group to non-Service admins?	
Default User Rights	None

Read-Only Domain Controllers

This group is comprised of the Read-only domain controllers in the domain. A Read-only domain controller makes it possible for organizations to easily deploy a domain controller in scenarios where physical security cannot be guaranteed, such as branch office locations, or in scenarios where local storage of all domain passwords is considered a primary threat, such as in an extranet or in an application-facing role.

Because administration of a Read-only domain controller can be delegated to a domain user or security group, an Read-only domain controller is well suited for a site that should not have a user who is a member of the Domain Admins group. A Read-only domain controller encompasses the following functionality:

- Read-only AD DS database
- Unidirectional replication
- Credential caching
- Administrator role separation
- Read-only Domain Name System (DNS)

For information about deploying a Read-only domain controller, see Understanding Planning and Deployment for Read-Only Domain Controllers.

This security group was introduced in Windows Server 2008, and it has not changed in subsequent versions.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-21-<domain>-521
Type	Global
Default container	CN=Users, DC=<domain>, DC=
Default members	None
Default member of	Denied RODC Password Replication Group
Protected by ADMINSDHOLDER?	Yes
Safe to move out of default container?	Yes
Safe to delegate management of this group to non-Service admins?	
Default User Rights	See Denied RODC Password Replication Group

Remote Desktop Users

The Remote Desktop Users group on an RD Session Host server is used to grant users and groups permissions to remotely connect to an RD Session Host server. This group cannot be renamed, deleted, or moved. It appears as a SID until the domain controller is made the primary domain controller and it holds the operations master role (also known as flexible single master operations or FSMO).

The Remote Desktop Users group applies to versions of the Windows Server operating system listed in the Active Directory Default Security Groups table.

This security group has not changed since Windows Server 2008.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-32-555
Type	Builtin Local
Default container	CN=Builtin, DC=<domain>, DC=
Default members	None
Default member of	None

ATTRIBUTE	VALUE
Protected by ADMINSDHOLDER?	No
Safe to move out of default container?	Cannot be moved
Safe to delegate management of this group to non-Service admins?	Yes
Default User Rights	None

Remote Management Users

Members of the Remote Management Users group can access WMI resources over management protocols (such as WS-Management via the Windows Remote Management service). This applies only to WMI namespaces that grant access to the user.

The Remote Management Users group is generally used to allow users to manage servers through the Server Manager console, whereas the WinRMRemoteWMIUsers_ group is allows remotely running Windows PowerShell commands.

For more information, see What's New in MI? and About WMI.

This security group was introduced in Windows Server 2012, and it has not changed in subsequent versions.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-32-580
Type	Builtin Local
Default container	CN=Builtin, DC=<domain>, DC=
Default members	None
Default member of	None
Protected by ADMINSDHOLDER?	No
Safe to move out of default container?	Cannot be moved
Safe to delegate management of this group to non-Service admins?	
Default User Rights	None

Replicator

Computers that are members of the Replicator group support file replication in a domain. Windows Server operating systems use the File Replication service (FRS) to replicate system policies and logon scripts stored in the System Volume (SYSVOL). Each domain controller keeps a copy of SYSVOL for network clients to access. FRS can also replicate data for the Distributed File System (DFS), synchronizing the content of each member in a replica set as defined by DFS. FRS can copy and maintain shared files and folders on multiple servers simultaneously. When changes occur, content is synchronized immediately within sites and by a schedule between sites.

> **WARNING**
> In Windows Server 2008 R2, FRS cannot be used for replicating DFS folders or custom (non-SYSVOL) data. A Windows Server 2008 R2 domain controller can still use FRS to replicate the contents of a SYSVOL shared resource in a domain that uses FRS for replicating the SYSVOL shared resource between domain controllers.

However, Windows Server 2008 R2 servers cannot use FRS to replicate the contents of any replica set apart from the SYSVOL shared resource. The DFS Replication service is a replacement for FRS, and it can be used to replicate the contents of a SYSVOL shared resource, DFS folders, and other custom (non-SYSVOL) data. You should migrate all non-SYSVOL FRS replica sets to DFS Replication. For more information, see:

- File Replication Service (FRS) Is Deprecated in Windows Server 2008 R2 (Windows)
- DFS Namespaces and DFS Replication Overview

This security group has not changed since Windows Server 2008.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-32-552
Type	Builtin Local
Default container	CN=Builtin, DC=<domain>, DC=
Default members	None
Default member of	None
Protected by ADMINSDHOLDER?	Yes
Safe to move out of default container?	Cannot be moved
Safe to delegate management of this group to non-Service admins?	
Default User Rights	None

Schema Admins

Members of the Schema Admins group can modify the Active Directory schema. This group exists only in the root domain of an Active Directory forest of domains. It is a Universal group if the domain is in native mode; it is a Global group if the domain is in mixed mode.

The group is authorized to make schema changes in Active Directory. By default, the only member of the group is the Administrator account for the forest root domain. This group has full administrative access to the schema.

The membership of this group can be modified by any of the service administrator groups in the root domain. This is considered a service administrator account because its members can modify the schema, which governs the structure and content of the entire directory.

For more information, see What Is the Active Directory Schema?: Active Directory.

The Schema Admins group applies to versions of the Windows Server operating system listed in the Active Directory Default Security Groups table.

This security group has not changed since Windows Server 2008.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-21-<root domain>-518
Type	Universal (if Domain is in Native-Mode) else Global
Default container	CN=Users, DC=<domain>, DC=
Default members	Administrator
Default member of	Denied RODC Password Replication Group
Protected by ADMINSDHOLDER?	Yes
Safe to move out of default container?	Yes
Safe to delegate management of this group to non-Service admins?	No
Default User Rights	See Denied RODC Password Replication Group

Server Operators

Members in the Server Operators group can administer domain controllers. This group exists only on domain controllers. By default, the group has no members. Members of the Server Operators group can sign in to a server interactively, create and delete network shared resources, start and stop services, back up and restore files, format the hard disk drive of the computer, and shut down the computer. This group cannot be renamed, deleted, or moved.

By default, this built-in group has no members, and it has access to server configuration options on domain controllers. Its membership is controlled by the service administrator groups Administrators and Domain Admins in the domain, and the Enterprise Admins group in the forest root domain. Members in this group

cannot change any administrative group memberships. This is considered a service administrator account because its members have physical access to domain controllers, they can perform maintenance tasks (such as backup and restore), and they have the ability to change binaries that are installed on the domain controllers. Note the default user rights in the following table.

The Server Operators group applies to versions of the Windows Server operating system listed in the Active Directory Default Security Groups table.

This security group has not changed since Windows Server 2008.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-32-549
Type	Builtin Local
Default container	CN=Builtin, DC=<domain>, DC=
Default members	None
Default member of	None
Protected by ADMINSDHOLDER?	Yes
Safe to move out of default container?	Cannot be moved
Safe to delegate management of this group to non-Service admins?	No
Default User Rights	Allow log on locally: SeInteractiveLogonRight Back up files and directories: SeBackupPrivilege Change the system time: SeSystemTimePrivilege Change the time zone: SeTimeZonePrivilege Force shutdown from a remote system: SeRemoteShutdownPrivilege Restore files and directories: Restore files and directories SeRestorePrivilege Shut down the system: SeShutdownPrivilege

Storage Replica Administrators

Members of this group have complete and unrestricted access to all features of Storage Replica.

The Storage Replica Administrators group applies to versions of the Windows Server operating system listed in the Active Directory Default Security Groups table.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-32-582
Type	Builtin Local
Default container	CN=BuiltIn, DC=<domain>, DC=
Default members	None
Default member of	None
Protected by ADMINSDHOLDER?	No
Safe to move out of default container?	Yes
Safe to delegate management of this group to non-Service admins?	No
Default User Rights	None

System Managed Accounts Group

Members of this group are managed by the system.

The System Managed Accounts group applies to versions of the Windows Server operating system listed in the Active Directory Default Security Groups table.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-32-581
Type	Builtin Local
Default container	CN=BuiltIn, DC=<domain>, DC=
Default members	Users
Default member of	None
Protected by ADMINSDHOLDER?	No
Safe to move out of default container?	Yes
Safe to delegate management of this group to non-Service admins?	No
Default User Rights	None

Terminal Server License Servers

Members of the Terminal Server License Servers group can update user accounts in Active Directory with information about license issuance. This is used to track and report TS Per User CAL usage. A TS Per User CAL gives one user the right to access a Terminal Server from an unlimited number of client computers or devices. This group appears as a SID until the domain controller is made the primary domain controller and it holds the

operations master role (also known as flexible single master operations or FSMO).

For more information about this security group, see Terminal Services License Server Security Group Configuration.

The Terminal Server License Servers group applies to versions of the Windows Server operating system listed in the Active Directory Default Security Groups table.

> **NOTE**
> This group cannot be renamed, deleted, or moved.

This security group only applies to Windows Server 2003 and Windows Server 2008 because Terminal Services was replaced by Remote Desktop Services in Windows Server 2008 R2.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-32-561
Type	Builtin Local
Default container	CN=Builtin, DC=<domain>, DC=
Default members	None
Default member of	None
Safe to move out of default container?	Cannot be moved
Protected by ADMINSDHOLDER?	No
Safe to delegate management of this group to non-Service admins?	Yes
Default User Rights	None

Users

Members of the Users group are prevented from making accidental or intentional system-wide changes, and they can run most applications. After the initial installation of the operating system, the only member is the Authenticated Users group. When a computer joins a domain, the Domain Users group is added to the Users group on the computer.

Users can perform tasks such as running applications, using local and network printers, shutting down the computer, and locking the computer. Users can install applications that only they are allowed to use if the installation program of the application supports per-user installation. This group cannot be renamed, deleted, or moved.

The Users group applies to versions of the Windows Server operating system listed in the Active Directory

Default Security Groups table.

This security group includes the following changes since Windows Server 2008:

- In Windows Server 2008 R2, INTERACTIVE was added to the default members list.
- In Windows Server 2012, the default **Member Of** list changed from Domain Users to none.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-32-545
Type	Builtin Local
Default container	CN=Builtin, DC=<domain>, DC=
Default members	Authenticated Users Domain Users INTERACTIVE
Default member of	None
Protected by ADMINSDHOLDER?	No
Safe to move out of default container?	Cannot be moved
Safe to delegate management of this group to non-Service admins?	No
Default User Rights	None

Windows Authorization Access Group

Members of this group have access to the computed token GroupsGlobalAndUniversal attribute on User objects. Some applications have features that read the token-groups-global-and-universal (TGGAU) attribute on user account objects or on computer account objects in Active Directory Domain Services. Some Win32 functions make it easier to read the TGGAU attribute. Applications that read this attribute or that call an API (referred to as a function) that reads this attribute do not succeed if the calling security context does not have access to the attribute. This group appears as a SID until the domain controller is made the primary domain controller and it holds the operations master role (also known as flexible single master operations or FSMO).

The Windows Authorization Access group applies to versions of the Windows Server operating system listed in the Active Directory Default Security Groups table.

> **NOTE**
> This group cannot be renamed, deleted, or moved.

This security group has not changed since Windows Server 2008.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-32-560
Type	Builtin Local
Default container	CN=Builtin, DC=<domain>, DC=
Default members	Enterprise Domain Controllers
Default member of	None
Protected by ADMINSDHOLDER?	No
Safe to move out of default container?	Cannot be moved
Safe to delegate management of this group to non-Service admins?	Yes
Default user rights	None

WinRMRemoteWMIUsers_

In Windows 8 and in Windows Server 2012, a **Share** tab was added to the Advanced Security Settings user interface. This tab displays the security properties of a remote file share. To view this information, you must have the following permissions and memberships, as appropriate for the version of Windows Server that the file server is running.

The WinRMRemoteWMIUsers_ group applies to versions of the Windows Server operating system listed in the Active Directory Default Security Groups table.

- If the file share is hosted on a server that is running a supported version of the operating system:
  - You must be a member of the WinRMRemoteWMIUsers__ group or the BUILTIN\Administrators group.
  - You must have Read permissions to the file share.
- If the file share is hosted on a server that is running a version of Windows Server that is earlier than Windows Server 2012:
  - You must be a member of the BUILTIN\Administrators group.
  - You must have Read permissions to the file share.

In Windows Server 2012, the Access Denied Assistance functionality adds the Authenticated Users group to the local WinRMRemoteWMIUsers__ group. Therefore, when the Access Denied Assistance functionality is enabled, all authenticated users who have Read permissions to the file share can view the file share permissions.

NOTE

The WinRMRemoteWMIUsers_ group allows running Windows PowerShell commands remotely whereas the Remote Management Users group is generally used to allow users to manage servers by using the Server Manager console.

This security group was introduced in Windows Server 2012, and it has not changed in subsequent versions.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-21-<domain>-1000
Type	Domain local
Default container	CN=Users, DC=<domain>, DC=
Default members	None
Default member of	None
Protected by ADMINSDHOLDER?	No
Safe to move out of default container?	Yes
Safe to delegate management of this group to non-Service admins?	
Default User Rights	None

See also

- Security Principals
- Special Identities
- Access Control Overview

Special Identities

6/28/2021 • 10 minutes to read • Edit Online

Applies to

- Windows Server 2016

This reference topic for the IT professional describes the special identity groups (which are sometimes referred to as security groups) that are used in Windows access control.

Special identity groups are similar to Active Directory security groups as listed in the users and built-in containers. Special identity groups can provide an efficient way to assign access to resources in your network. By using special identity groups, you can:

- Assign user rights to security groups in Active Directory.
- Assign permissions to security groups for the purpose of accessing resources.

Servers that are running the supported Windows Server operating systems designated in the **Applies To** list at the beginning of this topic include several special identity groups. These special identity groups do not have specific memberships that can be modified, but they can represent different users at different times, depending on the circumstances.

Although the special identity groups can be assigned rights and permissions to resources, the memberships cannot be modified or viewed. Group scopes do not apply to special identity groups. Users are automatically assigned to these special identity groups whenever they sign in or access a particular resource.

For information about security groups and group scope, see Active Directory Security Groups.

The special identity groups are described in the following tables:

- Anonymous Logon
- Authenticated User
- Batch
- Creator Group
- Creator Owner
- Dialup
- Digest Authentication
- Enterprise Domain Controllers
- Everyone
- Interactive
- Local Service
- LocalSystem
- Network
- Network Service

- NTLM Authentication
- Other Organization
- Principal Self
- Remote Interactive Logon
- Restricted
- SChannel Authentication
- Service
- Terminal Server User
- This Organization
- Window Manager\Window Manager Group

Anonymous Logon

Any user who accesses the system through an anonymous logon has the Anonymous Logon identity. This identity allows anonymous access to resources, such as a web page that is published on corporate servers. The Anonymous Logon group is not a member of the Everyone group by default.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-7
Object Class	Foreign Security Principal
Default Location in Active Directory	cn=WellKnown Security Principals, cn=Configuration, dc=<forestRootDomain>
Default User Rights	None

Authenticated Users

Any user who accesses the system through a sign-in process has the Authenticated Users identity. This identity allows access to shared resources within the domain, such as files in a shared folder that should be accessible to all the workers in the organization. Membership is controlled by the operating system.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-11
Object Class	Foreign Security Principal
Default Location in Active Directory	cn=System,cn=WellKnown Security Principals, cn=Configuration, dc=<forestRootDomain>
Default User Rights	Access this computer from the network: SeNetworkLogonRight Add workstations to domain: SeMachineAccountPrivilege Bypass traverse checking: SeChangeNotifyPrivilege

Batch

Any user or process that accesses the system as a batch job (or through the batch queue) has the Batch identity. This identity allows batch jobs to run scheduled tasks, such as a nightly cleanup job that deletes temporary files. Membership is controlled by the operating system.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-3
Object Class	Foreign Security Principal
Default Location in Active Directory	cn=WellKnown Security Principals, cn=Configuration, dc=<forestRootDomain>
Default User Rights	none

Creator Group

The person who created the file or the directory is a member of this special identity group. Windows Server operating systems use this identity to automatically grant access permissions to the creator of a file or directory.

A placeholder security identifier (SID) is created in an inheritable access control entry (ACE). When the ACE is inherited, the system replaces this SID with the SID for the primary group of the object's current owner. The primary group is used only by the Portable Operating System Interface for UNIX (POSIX) subsystem.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-3-1
Object Class	Foreign Security Principal
Default Location in Active Directory	cn=WellKnown Security Principals, cn=Configuration, dc=<forestRootDomain>
Default User Rights	none

Creator Owner

The person who created the file or the directory is a member of this special identity group. Windows Server operating systems use this identity to automatically grant access permissions to the creator of a file or directory. A placeholder SID is created in an inheritable ACE. When the ACE is inherited, the system replaces this SID with the SID for the object's current owner.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-3-0
Object Class	Foreign Security Principal
Default Location in Active Directory	cn=WellKnown Security Principals, cn=Configuration, dc=<forestRootDomain>

ATTRIBUTE	VALUE
Default User Rights	none

Dialup

Any user who accesses the system through a dial-up connection has the Dial-Up identity. This identity distinguishes dial-up users from other types of authenticated users.

ATTRIBUTE	VALUE	
Well-Known SID/RID	S-1-5-1	
Object Class	Foreign Security Principal	
Default Location in Active Directory	cn=WellKnown Security Principals, cn=Configuration, dc=<forestRootDomain>	
Default User Rights	none	

Digest Authentication

ATTRIBUTE	VALUE	
Well-Known SID/RID	S-1-5-64-21	
Object Class	Foreign Security Principal	
Default Location in Active Directory	cn=WellKnown Security Principals, cn=Configuration, dc=<forestRootDomain>	
Default User Rights	none	

Enterprise Domain Controllers

This group includes all domain controllers in an Active Directory forest. Domain controllers with enterprise-wide roles and responsibilities have the Enterprise Domain Controllers identity. This identity allows them to perform certain tasks in the enterprise by using transitive trusts. Membership is controlled by the operating system.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-9
Object Class	Foreign Security Principal
Default Location in Active Directory	cn=WellKnown Security Principals, cn=Configuration, dc=<forestRootDomain>

ATTRIBUTE	VALUE
Default User Rights	Access this computer from the network: SeNetworkLogonRight Allow log on locally: SeInteractiveLogonRight

Everyone

All interactive, network, dial-up, and authenticated users are members of the Everyone group. This special identity group gives wide access to system resources. Whenever a user logs on to the network, the user is automatically added to the Everyone group.

On computers running Windows 2000 and earlier, the Everyone group included the Anonymous Logon group as a default member, but as of Windows Server 2003, the Everyone group contains only Authenticated Users and Guest; and it no longer includes Anonymous Logon by default (although this can be changed, using Registry Editor, by going to the **Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa** key and setting the value of **everyoneincludesanonymous** DWORD to 1).

Membership is controlled by the operating system.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-1-0
Object Class	Foreign Security Principal
Default Location in Active Directory	cn=WellKnown Security Principals, cn=Configuration, dc=<forestRootDomain>
Default User Rights	Access this computer from the network: SeNetworkLogonRight Act as part of the operating system: SeTcbPrivilege Bypass traverse checking: SeChangeNotifyPrivilege

Interactive

Any user who is logged on to the local system has the Interactive identity. This identity allows only local users to access a resource. Whenever a user accesses a given resource on the computer to which they are currently logged on, the user is automatically added to the Interactive group. Membership is controlled by the operating system.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-4
Object Class	Foreign Security Principal
Default Location in Active Directory	cn=WellKnown Security Principals, cn=Configuration, dc=<forestRootDomain>
Default User Rights	None

Local Service

The Local Service account is similar to an Authenticated User account. The Local Service account has the same level of access to resources and objects as members of the Users group. This limited access helps safeguard your system if individual services or processes are compromised. Services that run as the Local Service account access network resources as a null session with anonymous credentials. The name of the account is NT AUTHORITY\LocalService. This account does not have a password.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-19
Object Class	Foreign Security Principal
Default Location in Active Directory	cn=WellKnown Security Principals, cn=Configuration, dc=<forestRootDomain>
Default User Rights	Adjust memory quotas for a process: SeIncreaseQuotaPrivilege Bypass traverse checking: SeChangeNotifyPrivilege Change the system time: SeSystemtimePrivilege Change the time zone: SeTimeZonePrivilege Create global objects: SeCreateGlobalPrivilege Generate security audits: SeAuditPrivilege Impersonate a client after authentication: SeImpersonatePrivilege Replace a process level token: SeAssignPrimaryTokenPrivilege

LocalSystem

This is a service account that is used by the operating system. The LocalSystem account is a powerful account that has full access to the system and acts as the computer on the network. If a service logs on to the LocalSystem account on a domain controller, that service has access to the entire domain. Some services are configured by default to log on to the LocalSystem account. Do not change the default service setting. The name of the account is LocalSystem. This account does not have a password.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-18
Object Class	Foreign Security Principal
Default Location in Active Directory	cn=WellKnown Security Principals, cn=Configuration, dc=<forestRootDomain>
Default User Rights	None

Network

This group implicitly includes all users who are logged on through a network connection. Any user who accesses the system through a network has the Network identity. This identity allows only remote users to access a resource. Whenever a user accesses a given resource over the network, the user is automatically added to the Network group. Membership is controlled by the operating system.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-2
Object Class	Foreign Security Principal
Default Location in Active Directory	cn=WellKnown Security Principals, cn=Configuration, dc=<forestRootDomain>
Default User Rights	None

Network Service

The Network Service account is similar to an Authenticated User account. The Network Service account has the same level of access to resources and objects as members of the Users group. This limited access helps safeguard your system if individual services or processes are compromised. Services that run as the Network Service account access network resources by using the credentials of the computer account. The name of the account is NT AUTHORITY\NetworkService. This account does not have a password.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-20
Object Class	Foreign Security Principal
Default Location in Active Directory	cn=WellKnown Security Principals, cn=Configuration, dc=<forestRootDomain>
Default User Rights	Adjust memory quotas for a process: SeIncreaseQuotaPrivilege Bypass traverse checking: SeChangeNotifyPrivilege Create global objects: SeCreateGlobalPrivilege Generate security audits: SeAuditPrivilege Impersonate a client after authentication: SeImpersonatePrivilege Replace a process level token: SeAssignPrimaryTokenPrivilege

NTLM Authentication

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-64-10
Object Class	Foreign Security Principal
Default Location in Active Directory	cn=WellKnown Security Principals, cn=Configuration, dc=<forestRootDomain>
Default User Rights	None

Other Organization

This group implicitly includes all users who are logged on to the system through a dial-up connection.

Membership is controlled by the operating system.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-1000
Object Class	Foreign Security Principal
Default Location in Active Directory	cn=WellKnown Security Principals, cn=Configuration, dc=<forestRootDomain>
Default User Rights	None

Principal Self

This identity is a placeholder in an ACE on a user, group, or computer object in Active Directory. When you grant permissions to Principal Self, you grant them to the security principal that is represented by the object. During an access check, the operating system replaces the SID for Principal Self with the SID for the security principal that is represented by the object.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-10
Object Class	Foreign Security Principal
Default Location in Active Directory	cn=WellKnown Security Principals, cn=Configuration, dc=<forestRootDomain>
Default User Rights	None

Remote Interactive Logon

This identity represents all users who are currently logged on to a computer by using a Remote Desktop connection. This group is a subset of the Interactive group. Access tokens that contain the Remote Interactive Logon SID also contain the Interactive SID.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-14
Object Class	Foreign Security Principal
Default Location in Active Directory	cn=WellKnown Security Principals, cn=Configuration, dc=<forestRootDomain>
Default User Rights	None

Restricted

Users and computers with restricted capabilities have the Restricted identity. This identity group is used by a process that is running in a restricted security context, such as running an application with the RunAs service. When code runs at the Restricted security level, the Restricted SID is added to the user's access token.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-12
Object Class	Foreign Security Principal
Default Location in Active Directory	cn=WellKnown Security Principals, cn=Configuration, dc=<forestRootDomain>
Default User Rights	None

SChannel Authentication

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-64-14
Object Class	Foreign Security Principal
Default Location in Active Directory	cn=WellKnown Security Principals, cn=Configuration, dc=<forestRootDomain>
Default User Rights	None

Service

Any service that accesses the system has the Service identity. This identity group includes all security principals that are signed in as a service. This identity grants access to processes that are being run by Windows Server services. Membership is controlled by the operating system.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-6
Object Class	Foreign Security Principal
Default Location in Active Directory	cn=WellKnown Security Principals, cn=Configuration, dc=<forestRootDomain>
Default User Rights	Create global objects: SeCreateGlobalPrivilege Impersonate a client after authentication: SeImpersonatePrivilege

Terminal Server User

Any user accessing the system through Terminal Services has the Terminal Server User identity. This identity allows users to access Terminal Server applications and to perform other necessary tasks with Terminal Server services. Membership is controlled by the operating system.

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-13

ATTRIBUTE	VALUE
Object Class	Foreign Security Principal
Default Location in Active Directory	cn=WellKnown Security Principals, cn=Configuration, dc=<forestRootDomain>
Default User Rights	None

This Organization

ATTRIBUTE	VALUE
Well-Known SID/RID	S-1-5-15
Object Class	Foreign Security Principal
Default Location in Active Directory	cn=WellKnown Security Principals, cn=Configuration, dc=<forestRootDomain>
Default User Rights	None

Window Manager\Window Manager Group

ATTRIBUTE	VALUE
Well-Known SID/RID	
Object Class	
Default Location in Active Directory	cn=WellKnown Security Principals, cn=Configuration, dc=<forestRootDomain>
Default User Rights	Bypass traverse checking: SeChangeNotifyPrivilege Increase a process working set: SeIncreaseWorkingSetPrivilege

See also

- Active Directory Security Groups
- Security Principals
- Access Control Overview

User Account Control

3/5/2021 • 2 minutes to read • Edit Online

Applies to

- Windows 10
- Windows Server 2016

User Account Control (UAC) helps prevent malware from damaging a PC and helps organizations deploy a better-managed desktop. With UAC, apps and tasks always run in the security context of a non-administrator account, unless an administrator specifically authorizes administrator-level access to the system. UAC can block the automatic installation of unauthorized apps and prevent inadvertent changes to system settings.

UAC allows all users to log on to their computers using a standard user account. Processes launched using a standard user token may perform tasks using access rights granted to a standard user. For instance, Windows Explorer automatically inherits standard user level permissions. Additionally, any apps that are started using Windows Explorer (for example, by double-clicking a shortcut) also run with the standard set of user permissions. Many apps, including those that are included with the operating system itself, are designed to work properly in this way.

Other apps, especially those that were not specifically designed with security settings in mind, often require additional permissions to run successfully. These types of apps are referred to as legacy apps. Additionally, actions such as installing new software and making configuration changes to the Windows Firewall, require more permissions than what is available to a standard user account.

When an app needs to run with more than standard user rights, UAC can restore additional user groups to the token. This enables the user to have explicit control of apps that are making system level changes to their computer or device.

Practical applications

Admin Approval Mode in UAC helps prevent malware from silently installing without an administrator's knowledge. It also helps protect from inadvertent system-wide changes. Lastly, it can be used to enforce a higher level of compliance where administrators must actively consent or provide credentials for each administrative process.

In this section

TOPIC	DESCRIPTION
How User Account Control works	User Account Control (UAC) is a fundamental component of Microsoft's overall security vision. UAC helps mitigate the impact of malware.
User Account Control security policy settings	You can use security policies to configure how User Account Control works in your organization. They can be configured locally by using the Local Security Policy snap-in (secpol.msc) or configured for the domain, OU, or specific groups by Group Policy.

TOPIC	DESCRIPTION
User Account Control Group Policy and registry key settings	Here's a list of UAC Group Policy and registry key settings that your organization can use to manage UAC.

How User Account Control works

3/5/2021 • 14 minutes to read • Edit Online

Applies to

- Windows 10

User Account Control (UAC) is a fundamental component of Microsoft's overall security vision. UAC helps mitigate the impact of malware.

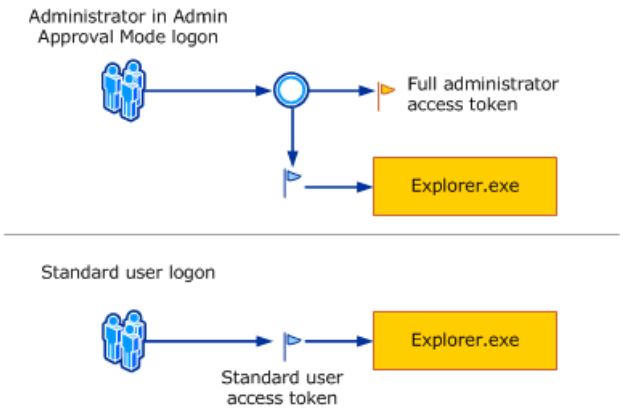
UAC process and interactions

Each app that requires the administrator access token must prompt for consent. The one exception is the relationship that exists between parent and child processes. Child processes inherit the user's access token from the parent process. Both the parent and child processes, however, must have the same integrity level.
Windows 10 protects processes by marking their integrity levels. Integrity levels are measurements of trust. A "high" integrity application is one that performs tasks that modify system data, such as a disk partitioning application, while a "low" integrity application is one that performs tasks that could potentially compromise the operating system, such as a Web browser. Apps with lower integrity levels cannot modify data in applications with higher integrity levels. When a standard user attempts to run an app that requires an administrator access token, UAC requires that the user provide valid administrator credentials.

In order to better understand how this process happens, let's look at the Windows logon process.

Logon process

The following shows how the logon process for an administrator differs from the logon process for a standard user.



By default, standard users and administrators access resources and run apps in the security context of standard users. When a user logs on to a computer, the system creates an access token for that user. The access token contains information about the level of access that the user is granted, including specific security identifiers (SIDs) and Windows privileges.

When an administrator logs on, two separate access tokens are created for the user: a standard user access token and an administrator access token. The standard user access token contains the same user-specific information as the administrator access token, but the administrative Windows privileges and SIDs are removed. The standard user access token is used to start apps that do not perform administrative tasks (standard user apps). The standard user access token is then used to display the desktop (explorer.exe). Explorer.exe is the parent process from which all other user-initiated processes inherit their access token. As a result, all apps run as a standard user unless a user provides consent or credentials to approve an app to use a full administrative

access token.

A user that is a member of the Administrators group can log on, browse the Web, and read e-mail while using a standard user access token. When the administrator needs to perform a task that requires the administrator access token, Windows 10 automatically prompts the user for approval. This prompt is called an elevation prompt, and its behavior can be configured by using the Local Security Policy snap-in (Secpol.msc) or Group Policy. For more info, see User Account Control security policy settings.

The UAC User Experience

When UAC is enabled, the user experience for standard users is different from that of administrators in Admin Approval Mode. The recommended and more secure method of running Windows 10 is to make your primary user account a standard user account. Running as a standard user helps to maximize security for a managed environment. With the built-in UAC elevation component, standard users can easily perform an administrative task by entering valid credentials for a local administrator account. The default, built-in UAC elevation component for standard users is the credential prompt.

The alternative to running as a standard user is to run as an administrator in Admin Approval Mode. With the built-in UAC elevation component, members of the local Administrators group can easily perform an administrative task by providing approval. The default, built-in UAC elevation component for an administrator account in Admin Approval Mode is called the consent prompt.

The consent and credential prompts

With UAC enabled, Windows 10 prompts for consent or prompts for credentials of a valid local administrator account before starting a program or task that requires a full administrator access token. This prompt ensures that no malicious software can be silently installed.

The consent prompt

The consent prompt is presented when a user attempts to perform a task that requires a user's administrative access token. The following is an example of the UAC consent prompt.

The credential prompt

The credential prompt is presented when a standard user attempts to perform a task that requires a user's administrative access token. Administrators can also be required to provide their credentials by setting the **User Account Control: Behavior of the elevation prompt for administrators in Admin Approval Mode** policy setting value to **Prompt for credentials**.

The following is an example of the UAC credential prompt.

UAC elevation prompts

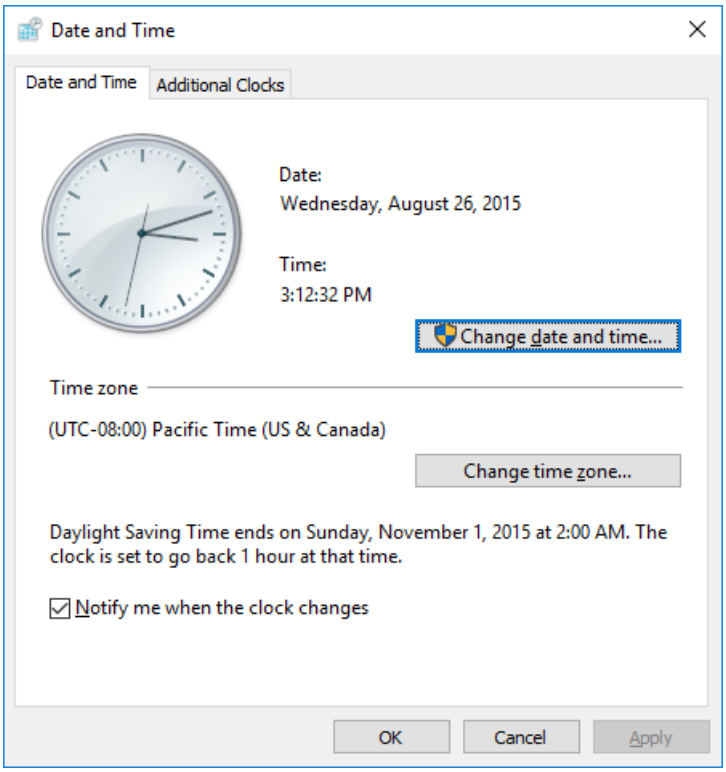
The UAC elevation prompts are color-coded to be app-specific, enabling for immediate identification of an application's potential security risk. When an app attempts to run with an administrator's full access token, Windows 10 first analyzes the executable file to determine its publisher. Apps are first separated into three categories based on the file's publisher: Windows 10, publisher verified (signed), and publisher not verified (unsigned). The following diagram illustrates how Windows 10 determines which color elevation prompt to present to the user.

The elevation prompt color-coding is as follows:

- Red background with a red shield icon: The app is blocked by Group Policy or is from a publisher that is blocked.
- Blue background with a blue and gold shield icon: The application is a Windows 10 administrative app, such as a Control Panel item.
- Blue background with a blue shield icon: The application is signed by using Authenticode and is trusted by the local computer.
- Yellow background with a yellow shield icon: The application is unsigned or signed but is not yet trusted by the local computer.

Shield icon

Some Control Panel items, such as **Date and Time Properties**, contain a combination of administrator and standard user operations. Standard users can view the clock and change the time zone, but a full administrator access token is required to change the local system time. The following is a screen shot of the **Date and Time Properties** Control Panel item.



The shield icon on the **Change date and time** button indicates that the process requires a full administrator access token and will display a UAC elevation prompt.

Securing the elevation prompt

The elevation process is further secured by directing the prompt to the secure desktop. The consent and credential prompts are displayed on the secure desktop by default in Windows 10. Only Windows processes can access the secure desktop. For higher levels of security, we recommend keeping the **User Account Control: Switch to the secure desktop when prompting for elevation** policy setting enabled.

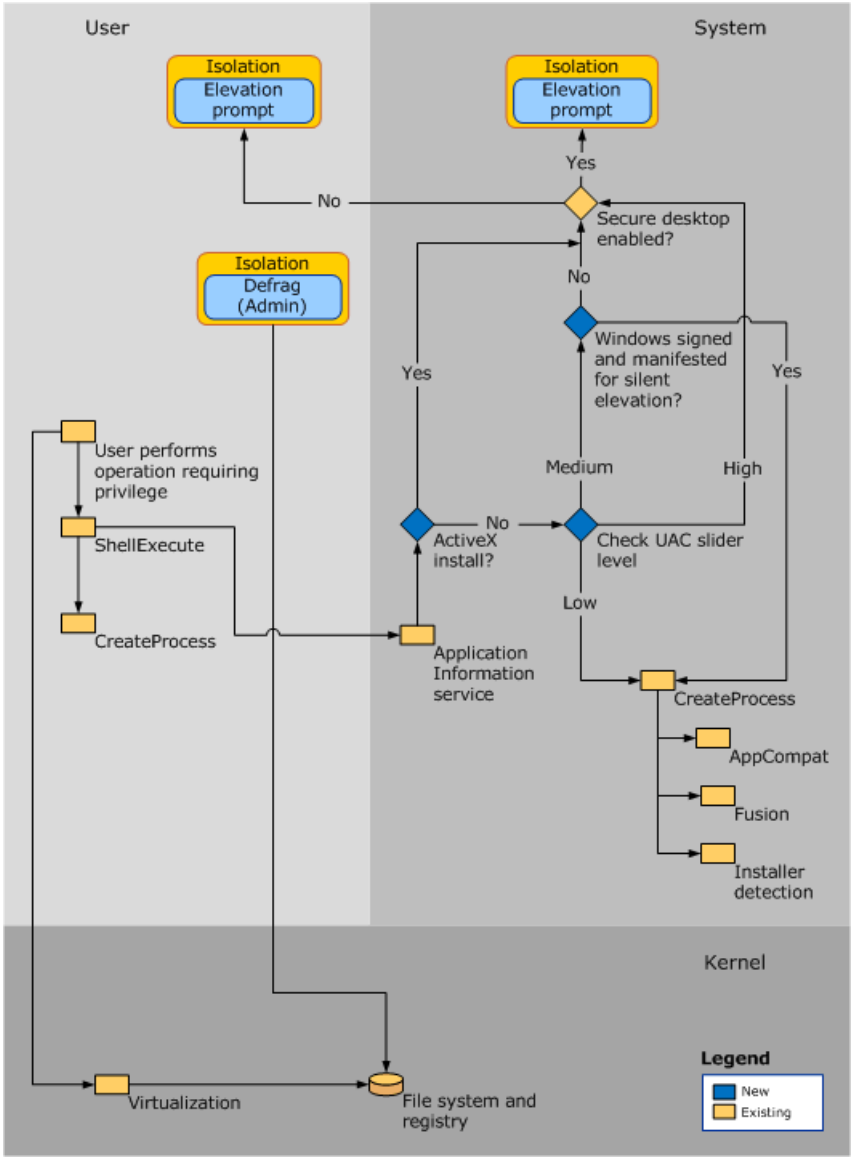
When an executable file requests elevation, the interactive desktop, also called the user desktop, is switched to the secure desktop. The secure desktop dims the user desktop and displays an elevation prompt that must be responded to before continuing. When the user clicks **Yes** or **No**, the desktop switches back to the user desktop.

Malware can present an imitation of the secure desktop, but when the **User Account Control: Behavior of the elevation prompt for administrators in Admin Approval Mode** policy setting is set to **Prompt for consent**, the malware does not gain elevation if the user clicks **Yes** on the imitation. If the policy setting is set to **Prompt for credentials**, malware imitating the credential prompt may be able to gather the credentials from the user. However, the malware does not gain elevated privilege and the system has other protections that mitigate malware from taking control of the user interface even with a harvested password.

While malware could present an imitation of the secure desktop, this issue cannot occur unless a user previously installed the malware on the PC. Because processes requiring an administrator access token cannot silently install when UAC is enabled, the user must explicitly provide consent by clicking **Yes** or by providing administrator credentials. The specific behavior of the UAC elevation prompt is dependent upon Group Policy.

UAC Architecture

The following diagram details the UAC architecture.



To better understand each component, review the table below:

COMPONENT	DESCRIPTION
USER	
User performs operation requiring privilege	If the operation changes the file system or registry, Virtualization is called. All other operations call ShellExecute.
ShellExecute	ShellExecute calls CreateProcess. ShellExecute looks for the ERROR_ELEVATION_REQUIRED error from CreateProcess. If it receives the error, ShellExecute calls the Application Information service to attempt to perform the requested task with the elevated prompt.
CreateProcess	If the application requires elevation, CreateProcess rejects the call with ERROR_ELEVATION_REQUIRED.
SYSTEM	

Application Information service	A system service that helps start apps that require one or more elevated privileges or user rights to run, such as local administrative tasks, and apps that require higher integrity levels. The Application Information service helps start such apps by creating a new process for the application with an administrative user's full access token when elevation is required and (depending on Group Policy) consent is given by the user to do so.
Elevating an ActiveX install	If ActiveX is not installed, the system checks the UAC slider level. If ActiveX is installed, the **User Account Control: Switch to the secure desktop when prompting for elevation** Group Policy setting is checked.

Check UAC slider level	UAC has a slider to select from four levels of notification. • **Always notify** will: ◦ Notify you when programs try to install software or make changes to your computer. ◦ Notify you when you make changes to Windows settings. ◦ Freeze other tasks until you respond. Recommended if you often install new software or visit unfamiliar websites. • **Notify me only when programs try to make changes to my computer** will: ◦ Notify you when programs try to install software or make changes to your computer. ◦ Not notify you when you make changes to Windows settings. ◦ Freeze other tasks until you respond. Recommended if you do not often install apps or visit unfamiliar websites. • **Notify me only when programs try to make changes to my computer (do not dim my desktop)** will: ◦ Notify you when programs try to install software or make changes to your computer. ◦ Not notify you when you make changes to Windows settings. ◦ Not freeze other tasks until you respond. Not recommended. Choose this only if it takes a long time to dim the desktop on your computer. • **Never notify (Disable UAC prompts)** will: ◦ Not notify you when programs try to install software or make changes to your computer. ◦ Not notify you when you make changes to Windows settings. ◦ Not freeze other tasks until you respond. Not recommended due to security concerns.
Secure desktop enabled	The **User Account Control: Switch to the secure desktop when prompting for elevation** policy setting is checked: • If the secure desktop is enabled, all elevation requests go to the secure desktop regardless of prompt behavior policy settings for administrators and standard users. • If the secure desktop is not enabled, all elevation requests go to the interactive user's desktop, and the per-user settings for administrators and standard users are used.

CreateProcess	CreateProcess calls AppCompat, Fusion, and Installer detection to assess if the app requires elevation. The file is then inspected to determine its requested execution level, which is stored in the application manifest for the file. CreateProcess fails if the requested execution level specified in the manifest does not match the access token and returns an error (ERROR_ELEVATION_REQUIRED) to ShellExecute.
AppCompat	The AppCompat database stores information in the application compatibility fix entries for an application.
Fusion	The Fusion database stores information from application manifests that describe the applications. The manifest schema is updated to add a new requested execution level field.
Installer detection	Installer detection detects setup files, which helps prevent installations from being run without the user's knowledge and consent.
KERNEL	
Virtualization	Virtualization technology ensures that non-compliant apps do not silently fail to run or fail in a way that the cause cannot be determined. UAC also provides file and registry virtualization and logging for applications that write to protected areas.
File system and registry	The per-user file and registry virtualization redirects per-computer registry and file write requests to equivalent per-user locations. Read requests are redirected to the virtualized per-user location first and to the per-computer location second.

The slider will never turn UAC completely off. If you set it to **Never notify**, it will:

- Keep the UAC service running.
- Cause all elevation request initiated by administrators to be auto-approved without showing a UAC prompt.
- Automatically deny all elevation requests for standard users.

> **IMPORTANT**
>
> In order to fully disable UAC you must disable the policy **User Account Control: Run all administrators in Admin Approval Mode**.

> **WARNING**
>
> Some Universal Windows Platform apps may not work when UAC is disabled.

Virtualization

Because system administrators in enterprise environments attempt to secure systems, many line-of-business

(LOB) applications are designed to use only a standard user access token. As a result, you do not need to replace the majority of apps when UAC is turned on.

Windows 10 includes file and registry virtualization technology for apps that are not UAC-compliant and that require an administrator's access token to run correctly. When an administrative apps that is not UAC-compliant attempts to write to a protected folder, such as Program Files, UAC gives the app its own virtualized view of the resource it is attempting to change. The virtualized copy is maintained in the user's profile. This strategy creates a separate copy of the virtualized file for each user that runs the non-compliant app.

Most app tasks operate properly by using virtualization features. Although virtualization allows a majority of applications to run, it is a short-term fix and not a long-term solution. App developers should modify their apps to be compliant as soon as possible, rather than relying on file, folder, and registry virtualization.

Virtualization is not an option in the following scenarios:

- Virtualization does not apply to apps that are elevated and run with a full administrative access token.
- Virtualization supports only 32-bit apps. Non-elevated 64-bit apps simply receive an access denied message when they attempt to acquire a handle (a unique identifier) to a Windows object. Native Windows 64-bit apps are required to be compatible with UAC and to write data into the correct locations.
- Virtualization is disabled if the app includes an app manifest with a requested execution level attribute.

Request execution levels

An app manifest is an XML file that describes and identifies the shared and private side-by-side assemblies that an app should bind to at run time. The app manifest includes entries for UAC app compatibility purposes. Administrative apps that include an entry in the app manifest prompt the user for permission to access the user's access token. Although they lack an entry in the app manifest, most administrative app can run without modification by using app compatibility fixes. App compatibility fixes are database entries that enable applications that are not UAC-compliant to work properly.

All UAC-compliant apps should have a requested execution level added to the application manifest. If the application requires administrative access to the system, then marking the app with a requested execution level of "require administrator" ensures that the system identifies this program as an administrative app and performs the necessary elevation steps. Requested execution levels specify the privileges required for an app.

Installer detection technology

Installation programs are apps designed to deploy software. Most installation programs write to system directories and registry keys. These protected system locations are typically writeable only by an administrator in Installer detection technology, which means that standard users do not have sufficient access to install programs. Windows 10 heuristically detects installation programs and requests administrator credentials or approval from the administrator user in order to run with access privileges. Windows 10 also heuristically detects updates and programs that uninstall applications. One of the design goals of UAC is to prevent installations from being run without the user's knowledge and consent because installation programs write to protected areas of the file system and registry.

Installer detection only applies to:

- 32-bit executable files.
- Applications without a requested execution level attribute.
- Interactive processes running as a standard user with UAC enabled.

Before a 32-bit process is created, the following attributes are checked to determine whether it is an installer:

- The file name includes keywords such as "install," "setup," or "update."
- Versioning Resource fields contain the following keywords: Vendor, Company Name, Product Name, File Description, Original Filename, Internal Name, and Export Name.

- Keywords in the side-by-side manifest are embedded in the executable file.
- Keywords in specific StringTable entries are linked in the executable file.
- Key attributes in the resource script data are linked in the executable file.
- There are targeted sequences of bytes within the executable file.

> **NOTE**
>
> The keywords and sequences of bytes were derived from common characteristics observed from various installer technologies.

> **NOTE**
>
> The User Account Control: Detect application installations and prompt for elevation policy setting must be enabled for installer detection to detect installation programs. For more info, see User Account Control security policy settings.

User Account Control security policy settings

3/5/2021 • 6 minutes to read • Edit Online

Applies to

- Windows 10

You can use security policies to configure how User Account Control works in your organization. They can be configured locally by using the Local Security Policy snap-in (secpol.msc) or configured for the domain, OU, or specific groups by Group Policy.

User Account Control: Admin Approval Mode for the Built-in Administrator account

This policy setting controls the behavior of Admin Approval Mode for the built-in Administrator account.

- **Enabled** The built-in Administrator account uses Admin Approval Mode. By default, any operation that requires elevation of privilege will prompt the user to approve the operation.
- **Disabled** (Default) The built-in Administrator account runs all applications with full administrative privilege.

User Account Control: Allow UIAccess application to prompt for elevation without using the secure desktop

This policy setting controls whether User Interface Accessibility (UIAccess or UIA) programs can automatically disable the secure desktop for elevation prompts used by a standard user.

- **Enabled** UIA programs, including Windows Remote Assistance, automatically disable the secure desktop for elevation prompts. If you do not disable the "User Account Control: Switch to the secure desktop when prompting for elevation" policy setting, the prompts appear on the interactive user's desktop instead of the secure desktop.
- **Disabled** (Default) The secure desktop can be disabled only by the user of the interactive desktop or by disabling the "User Account Control: Switch to the secure desktop when prompting for elevation" policy setting.

User Account Control: Behavior of the elevation prompt for administrators in Admin Approval Mode

This policy setting controls the behavior of the elevation prompt for administrators.

- **Elevate without prompting** Allows privileged accounts to perform an operation that requires elevation without requiring consent or credentials.

  > **Note:** Use this option only in the most constrained environments.

- **Prompt for credentials on the secure desktop** When an operation requires elevation of privilege, the user is prompted on the secure desktop to enter a privileged user name and password. If the user enters valid credentials, the operation continues with the user's highest available privilege.
- **Prompt for consent on the secure desktop** When an operation requires elevation of privilege, the user is prompted on the secure desktop to select either Permit or Deny. If the user selects Permit, the

operation continues with the user's highest available privilege.

- **Prompt for credentials** When an operation requires elevation of privilege, the user is prompted to enter an administrative user name and password. If the user enters valid credentials, the operation continues with the applicable privilege.
- **Prompt for consent** When an operation requires elevation of privilege, the user is prompted to select either Permit or Deny. If the user selects Permit, the operation continues with the user's highest available privilege.
- **Prompt for consent for non-Windows binaries** (Default) When an operation for a non-Microsoft application requires elevation of privilege, the user is prompted on the secure desktop to select either Permit or Deny. If the user selects Permit, the operation continues with the user's highest available privilege.

User Account Control: Behavior of the elevation prompt for standard users

This policy setting controls the behavior of the elevation prompt for standard users.

- **Prompt for credentials** (Default) When an operation requires elevation of privilege, the user is prompted to enter an administrative user name and password. If the user enters valid credentials, the operation continues with the applicable privilege.
- **Automatically deny elevation requests** When an operation requires elevation of privilege, a configurable access denied error message is displayed. An enterprise that is running desktops as standard user may choose this setting to reduce help desk calls.
- **Prompt for credentials on the secure desktop** When an operation requires elevation of privilege, the user is prompted on the secure desktop to enter a different user name and password. If the user enters valid credentials, the operation continues with the applicable privilege.

User Account Control: Detect application installations and prompt for elevation

This policy setting controls the behavior of application installation detection for the computer.

- **Enabled** (Default) When an app installation package is detected that requires elevation of privilege, the user is prompted to enter an administrative user name and password. If the user enters valid credentials, the operation continues with the applicable privilege.
- **Disabled** App installation packages are not detected and prompted for elevation. Enterprises that are running standard user desktops and use delegated installation technologies, such as Group Policy or Microsoft Endpoint Manager should disable this policy setting. In this case, installer detection is unnecessary.

User Account Control: Only elevate executable files that are signed and validated

This policy setting enforces public key infrastructure (PKI) signature checks for any interactive applications that request elevation of privilege. Enterprise administrators can control which applications are allowed to run by adding certificates to the Trusted Publishers certificate store on local computers.

- **Enabled** Enforces the certificate certification path validation for a given executable file before it is permitted to run.
- **Disabled** (Default) Does not enforce the certificate certification path validation before a given executable file is permitted to run.

User Account Control: Only elevate UIAccess applications that are installed in secure locations

This policy setting controls whether applications that request to run with a User Interface Accessibility (UIAccess) integrity level must reside in a secure location in the file system. Secure locations are limited to the following: - ...\Program Files\, including subfolders - ...\Windows\system32\ - ...\Program Files (x86)\, including subfolders for 64-bit versions of Windows

> **Note:** Windows enforces a digital signature check on any interactive app that requests to run with a UIAccess integrity level regardless of the state of this security setting.

- **Enabled** (Default) If an app resides in a secure location in the file system, it runs only with UIAccess integrity.
- **Disabled** An app runs with UIAccess integrity even if it does not reside in a secure location in the file system.

User Account Control: Turn on Admin Approval Mode

This policy setting controls the behavior of all User Account Control (UAC) policy settings for the computer. If you change this policy setting, you must restart your computer.

- **Enabled** (Default) Admin Approval Mode is enabled. This policy must be enabled and related UAC policy settings must also be set appropriately to allow the built-in Administrator account and all other users who are members of the Administrators group to run in Admin Approval Mode.
- **Disabled** Admin Approval Mode and all related UAC policy settings are disabled. Note: If this policy setting is disabled, the Security Center notifies you that the overall security of the operating system has been reduced.

User Account Control: Switch to the secure desktop when prompting for elevation

This policy setting controls whether the elevation request prompt is displayed on the interactive user's desktop or the secure desktop.

- **Enabled** (Default) All elevation requests go to the secure desktop regardless of prompt behavior policy settings for administrators and standard users.
- **Disabled** All elevation requests go to the interactive user's desktop. Prompt behavior policy settings for administrators and standard users are used.

User Account Control: Virtualize file and registry write failures to per-user locations

This policy setting controls whether application write failures are redirected to defined registry and file system locations. This policy setting mitigates applications that run as administrator and write run-time application data to %ProgramFiles%, %Windir%, %Windir%\system32, or HKLM\Software.

- **Enabled** (Default) App write failures are redirected at run time to defined user locations for both the file system and registry.
- **Disabled** Apps that write data to protected locations fail.

User Account Control Group Policy and registry key settings

3/5/2021 • 12 minutes to read • Edit Online

Applies to

- Windows 10
- Windows Server 2016

Group Policy settings

There are 10 Group Policy settings that can be configured for User Account Control (UAC). The table lists the default for each of the policy settings, and the following sections explain the different UAC policy settings and provide recommendations. These policy settings are located in **Security Settings\Local Policies\Security Options** in the Local Security Policy snap-in. For more information about each of the Group Policy settings, see the Group Policy description. For information about the registry key settings, see Registry key settings.

GROUP POLICY SETTING	REGISTRY KEY	DEFAULT
User Account Control: Admin Approval Mode for the built-in Administrator account	FilterAdministratorToken	Disabled
User Account Control: Allow UIAccess applications to prompt for elevation without using the secure desktop	EnableUIADesktopToggle	Disabled
User Account Control: Behavior of the elevation prompt for administrators in Admin Approval Mode	ConsentPromptBehaviorAdmin	Prompt for consent for non-Windows binaries
User Account Control: Behavior of the elevation prompt for standard users	ConsentPromptBehaviorUser	Prompt for credentials on the secure desktop
User Account Control: Detect application installations and prompt for elevation	EnableInstallerDetection	Enabled (default for home) Disabled (default for enterprise)
User Account Control: Only elevate executables that are signed and validated	ValidateAdminCodeSignatures	Disabled
User Account Control: Only elevate UIAccess applications that are installed in secure locations	EnableSecureUIAPaths	Enabled
User Account Control: Run all administrators in Admin Approval Mode	EnableLUA	Enabled

GROUP POLICY SETTING	REGISTRY KEY	DEFAULT
User Account Control: Switch to the secure desktop when prompting for elevation	PromptOnSecureDesktop	Enabled
User Account Control: Virtualize file and registry write failures to per-user locations	EnableVirtualization	Enabled

User Account Control: Admin Approval Mode for the built-in Administrator account

The **User Account Control: Admin Approval Mode for the built-in Administrator account** policy setting controls the behavior of Admin Approval Mode for the built-in Administrator account.

The options are:

- **Enabled.** The built-in Administrator account uses Admin Approval Mode. By default, any operation that requires elevation of privilege will prompt the user to approve the operation.
- **Disabled.** (Default) The built-in Administrator account runs all applications with full administrative privilege.

User Account Control: Allow UIAccess applications to prompt for elevation without using the secure desktop

The **User Account Control: Allow UIAccess applications to prompt for elevation without using the secure desktop** policy setting controls whether User Interface Accessibility (UIAccess or UIA) programs can automatically disable the secure desktop for elevation prompts used by a standard user.

The options are:

- **Enabled.** UIA programs, including Windows Remote Assistance, automatically disable the secure desktop for elevation prompts. If you do not disable the **User Account Control: Switch to the secure desktop when prompting for elevation** policy setting, the prompts appear on the interactive user's desktop instead of the secure desktop.
- **Disabled.** (Default) The secure desktop can be disabled only by the user of the interactive desktop or by disabling the **User Account Control: Switch to the secure desktop when prompting for elevation** policy setting.

UIA programs are designed to interact with Windows and application programs on behalf of a user. This policy setting allows UIA programs to bypass the secure desktop to increase usability in certain cases; however, allowing elevation requests to appear on the interactive desktop instead of the secure desktop can increase your security risk.

UIA programs must be digitally signed because they must be able to respond to prompts regarding security issues, such as the UAC elevation prompt. By default, UIA programs are run only from the following protected paths:

- ...\Program Files, including subfolders
- ...\Program Files (x86), including subfolders for 64-bit versions of Windows
- ...\Windows\System32

The **User Account Control: Only elevate UIAccess applications that are installed in secure locations** policy setting disables the requirement to be run from a protected path.

While this policy setting applies to any UIA program, it is primarily used in certain remote assistance scenarios, including the Windows Remote Assistance program in Windows 7.

If a user requests remote assistance from an administrator and the remote assistance session is established, any elevation prompts appear on the interactive user's secure desktop and the administrator's remote session is

paused. To avoid pausing the remote administrator's session during elevation requests, the user may select the **Allow IT Expert to respond to User Account Control prompts** check box when setting up the remote assistance session. However, selecting this check box requires that the interactive user respond to an elevation prompt on the secure desktop. If the interactive user is a standard user, the user does not have the required credentials to allow elevation.

If you enable this policy setting, requests for elevation are automatically sent to the interactive desktop (not the secure desktop) and also appear on the remote administrator's view of the desktop during a remote assistance session. This allows the remote administrator to provide the appropriate credentials for elevation.

This policy setting does not change the behavior of the UAC elevation prompt for administrators.

If you plan to enable this policy setting, you should also review the effect of the **User Account Control: Behavior of the elevation prompt for standard users** policy setting. If it is configured as **Automatically deny elevation requests**, elevation requests are not presented to the user.

User Account Control: Behavior of the elevation prompt for administrators in Admin Approval Mode

The **User Account Control: Behavior of the elevation prompt for administrators in Admin Approval Mode** policy setting controls the behavior of the elevation prompt for administrators.

The options are:

- **Elevate without prompting.** Allows privileged accounts to perform an operation that requires elevation without requiring consent or credentials.

  Note Use this option only in the most constrained environments.

- **Prompt for credentials on the secure desktop.** When an operation requires elevation of privilege, the user is prompted on the secure desktop to enter a privileged user name and password. If the user enters valid credentials, the operation continues with the user's highest available privilege.

- **Prompt for consent on the secure desktop.** When an operation requires elevation of privilege, the user is prompted on the secure desktop to select either **Permit** or **Deny**. If the user selects **Permit**, the operation continues with the user's highest available privilege.

- **Prompt for credentials.** When an operation requires elevation of privilege, the user is prompted to enter an administrative user name and password. If the user enters valid credentials, the operation continues with the applicable privilege.

- **Prompt for consent.** When an operation requires elevation of privilege, the user is prompted to select either **Permit** or **Deny**. If the user selects **Permit**, the operation continues with the user's highest available privilege.

- **Prompt for consent for non-Windows binaries.** (Default) When an operation for a non-Microsoft application requires elevation of privilege, the user is prompted on the secure desktop to select either **Permit** or **Deny**. If the user selects **Permit**, the operation continues with the user's highest available privilege.

User Account Control: Behavior of the elevation prompt for standard users

The **User Account Control: Behavior of the elevation prompt for standard users** policy setting controls the behavior of the elevation prompt for standard users.

The options are:

- **Automatically deny elevation requests.** When an operation requires elevation of privilege, a configurable access denied error message is displayed. An enterprise that is running desktops as standard user may choose this setting to reduce help desk calls.
- **Prompt for credentials on the secure desktop.** (Default) When an operation requires elevation of

privilege, the user is prompted on the secure desktop to enter a different user name and password. If the user enters valid credentials, the operation continues with the applicable privilege.

- **Prompt for credentials.** When an operation requires elevation of privilege, the user is prompted to enter an administrative user name and password. If the user enters valid credentials, the operation continues with the applicable privilege.

User Account Control: Detect application installations and prompt for elevation

The **User Account Control: Detect application installations and prompt for elevation** policy setting controls the behavior of application installation detection for the computer.

The options are:

- **Enabled.** (Default for home) When an application installation package is detected that requires elevation of privilege, the user is prompted to enter an administrative user name and password. If the user enters valid credentials, the operation continues with the applicable privilege.
- **Disabled.** (Default for enterprise) Application installation packages are not detected and prompted for elevation. Enterprises that are running standard user desktops and use delegated installation technologies such as Group Policy Software Installation or Systems Management Server (SMS) should disable this policy setting. In this case, installer detection is unnecessary.

User Account Control: Only elevate executables that are signed and validated

The **User Account Control: Only elevate executables that are signed and validated** policy setting enforces public key infrastructure (PKI) signature checks for any interactive applications that request elevation of privilege. Enterprise administrators can control which applications are allowed to run by adding certificates to the Trusted Publishers certificate store on local computers.

The options are:

- **Enabled.** Enforces the PKI certification path validation for a given executable file before it is permitted to run.
- **Disabled.** (Default) Does not enforce PKI certification path validation before a given executable file is permitted to run.

User Account Control: Only elevate UIAccess applications that are installed in secure locations

The **User Account Control: Only elevate UIAccess applications that are installed in secure locations** policy setting controls whether applications that request to run with a User Interface Accessibility (UIAccess) integrity level must reside in a secure location in the file system. Secure locations are limited to the following:

- ...\Program Files, including subfolders
- ...\Windows\system32
- ...\Program Files (x86), including subfolders for 64-bit versions of Windows

Note Windows enforces a PKI signature check on any interactive application that requests to run with a UIAccess integrity level regardless of the state of this security setting.

The options are:

- **Enabled.** (Default) If an application resides in a secure location in the file system, it runs only with UIAccess integrity.
- **Disabled.** An application runs with UIAccess integrity even if it does not reside in a secure location in the file system.

User Account Control: Run all administrators in Admin Approval Mode

The **User Account Control: Run all administrators Admin Approval Mode** policy setting controls the behavior of all UAC policy settings for the computer. If you change this policy setting, you must restart your computer.

The options are:

- **Enabled.** (Default) Admin Approval Mode is enabled. This policy must be enabled and related UAC policy settings must also be set appropriately to allow the built-in Administrator account and all other users who are members of the **Administrators** group to run in Admin Approval Mode.
- **Disabled.** Admin Approval Mode and all related UAC policy settings are disabled.

Note If this policy setting is disabled, the Security Center notifies you that the overall security of the operating system has been reduced.

User Account Control: Switch to the secure desktop when prompting for elevation

The **User Account Control: Switch to the secure desktop when prompting for elevation** policy setting controls whether the elevation request prompt is displayed on the interactive user's desktop or the secure desktop.

The options are:

- **Enabled.** (Default) All elevation requests go to the secure desktop regardless of prompt behavior policy settings for administrators and standard users.
- **Disabled.** All elevation requests go to the interactive user's desktop. Prompt behavior policy settings for administrators and standard users are used.

When this policy setting is enabled, it overrides the **User Account Control: Behavior of the elevation prompt for administrators in Admin Approval Mode** policy setting. The following table describes the behavior of the elevation prompt for each of the administrator policy settings when the **User Account Control: Switch to the secure desktop when prompting for elevation** policy setting is enabled or disabled.

ADMINISTRATOR POLICY SETTING	ENABLED	DISABLED
Prompt for credentials on the secure desktop	The prompt appears on the secure desktop.	The prompt appears on the secure desktop.
Prompt for consent on the secure desktop	The prompt appears on the secure desktop.	The prompt appears on the secure desktop.
Prompt for credentials	The prompt appears on the secure desktop.	The prompt appears on the interactive user's desktop.
Prompt for consent	The prompt appears on the secure desktop.	The prompt appears on the interactive user's desktop.
Prompt for consent for non-Windows binaries	The prompt appears on the secure desktop.	The prompt appears on the interactive user's desktop.

When this policy setting is enabled, it overrides the **User Account Control: Behavior of the elevation prompt for standard users** policy setting. The following table describes the behavior of the elevation prompt for each of the standard user policy settings when the **User Account Control: Switch to the secure desktop when prompting for elevation** policy setting is enabled or disabled.

STANDARD POLICY SETTING	ENABLED	DISABLED
Automatically deny elevation requests	No prompt. The request is automatically denied.	No prompt. The request is automatically denied.

STANDARD POLICY SETTING	ENABLED	DISABLED
Prompt for credentials on the secure desktop	The prompt appears on the secure desktop.	The prompt appears on the secure desktop.
Prompt for credentials	The prompt appears on the secure desktop.	The prompt appears on the interactive user's desktop.

User Account Control: Virtualize file and registry write failures to per-user locations

The **User Account Control: Virtualize file and registry write failures to per-user locations** policy setting controls whether application write failures are redirected to defined registry and file system locations. This policy setting mitigates applications that run as administrator and write run-time application data to %ProgramFiles%, %Windir%, %Windir%\system32, or HKLM\Software.

The options are:

- **Enabled.** (Default) Application write failures are redirected at run time to defined user locations for both the file system and registry.
- **Disabled.** Applications that write data to protected locations fail.

Registry key settings

The registry keys are found in **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System**. For information about each of the registry keys, see the associated Group Policy description.

REGISTRY KEY	GROUP POLICY SETTING	REGISTRY SETTING
FilterAdministratorToken	User Account Control: Admin Approval Mode for the built-in Administrator account	0 (Default) = Disabled 1 = Enabled
EnableUIADesktopToggle	User Account Control: Allow UIAccess applications to prompt for elevation without using the secure desktop	0 (Default) = Disabled 1 = Enabled
ConsentPromptBehaviorAdmin	User Account Control: Behavior of the elevation prompt for administrators in Admin Approval Mode	0 = Elevate without prompting 1 = Prompt for credentials on the secure desktop 2 = Prompt for consent on the secure desktop 3 = Prompt for credentials 4 = Prompt for consent 5 (Default) = Prompt for consent for non-Windows binaries
ConsentPromptBehaviorUser	User Account Control: Behavior of the elevation prompt for standard users	0 = Automatically deny elevation requests 1 = Prompt for credentials on the secure desktop 3 (Default) = Prompt for credentials
EnableInstallerDetection	User Account Control: Detect application installations and prompt for elevation	1 = Enabled (default for home) 0 = Disabled (default for enterprise)

REGISTRY KEY	GROUP POLICY SETTING	REGISTRY SETTING
ValidateAdminCodeSignatures	User Account Control: Only elevate executables that are signed and validated	0 (Default) = Disabled 1 = Enabled
EnableSecureUIAPaths	User Account Control: Only elevate UIAccess applications that are installed in secure locations	0 = Disabled 1 (Default) = Enabled
EnableLUA	User Account Control: Run all administrators in Admin Approval Mode	0 = Disabled 1 (Default) = Enabled
PromptOnSecureDesktop	User Account Control: Switch to the secure desktop when prompting for elevation	0 = Disabled 1 (Default) = Enabled
EnableVirtualization	User Account Control: Virtualize file and registry write failures to per-user locations	0 = Disabled 1 (Default) = Enabled

Protect derived domain credentials with Windows Defender Credential Guard

3/26/2021 • 2 minutes to read • Edit Online

Applies to

- Windows 10
- Windows Server 2016

Introduced in Windows 10 Enterprise and Windows Server 2016, Windows Defender Credential Guard uses virtualization-based security to isolate secrets so that only privileged system software can access them. Unauthorized access to these secrets can lead to credential theft attacks, such as Pass-the-Hash or Pass-The-Ticket. Windows Defender Credential Guard prevents these attacks by protecting NTLM password hashes, Kerberos Ticket Granting Tickets, and credentials stored by applications as domain credentials.

By enabling Windows Defender Credential Guard, the following features and solutions are provided:

- **Hardware security** NTLM, Kerberos, and Credential Manager take advantage of platform security features, including Secure Boot and virtualization, to protect credentials.
- **Virtualization-based security** Windows NTLM and Kerberos derived credentials and other secrets run in a protected environment that is isolated from the running operating system.
- **Better protection against advanced persistent threats** When Credential Manager domain credentials, NTLM, and Kerberos derived credentials are protected using virtualization-based security, the credential theft attack techniques and tools used in many targeted attacks are blocked. Malware running in the operating system with administrative privileges cannot extract secrets that are protected by virtualization-based security. While Windows Defender Credential Guard is a powerful mitigation, persistent threat attacks will likely shift to new attack techniques and you should also incorporate other security strategies and architectures.

Related topics

- Isolated User Mode in Windows 10 with Dave Probert (Channel 9)
- Isolated User Mode Processes and Features in Windows 10 with Logan Gabriel (Channel 9)
- More on Processes and Features in Windows 10 Isolated User Mode with Dave Probert (Channel 9)
- Mitigating Credential Theft using the Windows 10 Isolated User Mode (Channel 9)
- Protecting network passwords with Windows Defender Credential Guard
- Enabling Strict KDC Validation in Windows Kerberos
- What's New in Kerberos Authentication for Windows Server 2012
- Authentication Mechanism Assurance for AD DS in Windows Server 2008 R2 Step-by-Step Guide
- Trusted Platform Module

How Windows Defender Credential Guard works

3/5/2021 • 2 minutes to read • Edit Online

Applies to

- Windows 10
- Windows Server 2016

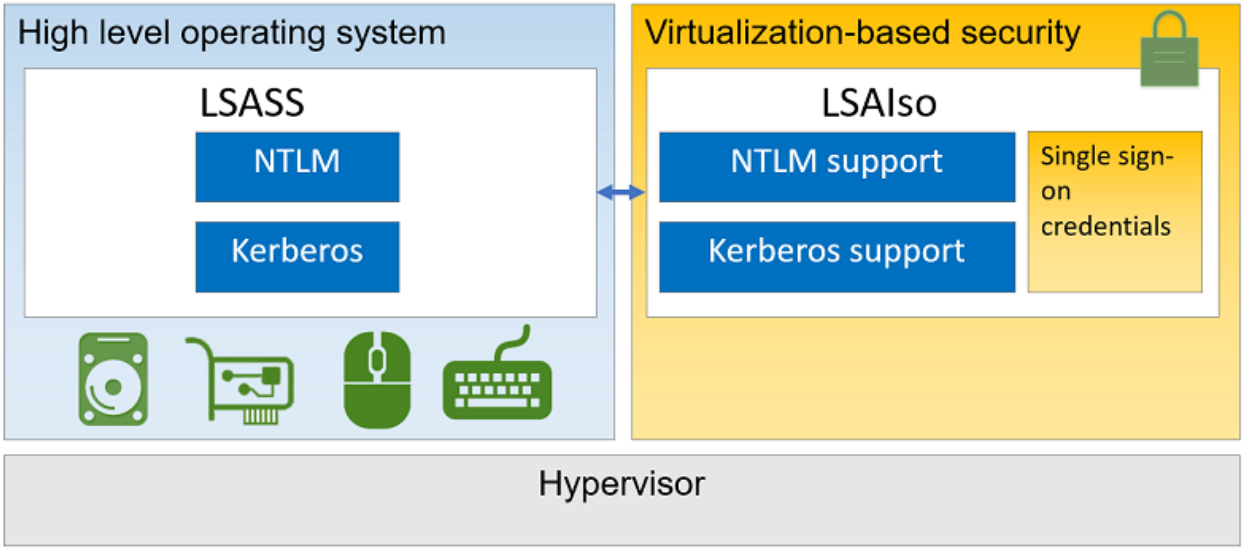
Kerberos, NTLM, and Credential manager isolate secrets by using virtualization-based security. Previous versions of Windows stored secrets in the Local Security Authority (LSA). Prior to Windows 10, the LSA stored secrets used by the operating system in its process memory. With Windows Defender Credential Guard enabled, the LSA process in the operating system talks to a new component called the isolated LSA process that stores and protects those secrets. Data stored by the isolated LSA process is protected using virtualization-based security and is not accessible to the rest of the operating system. LSA uses remote procedure calls to communicate with the isolated LSA process.

For security reasons, the isolated LSA process doesn't host any device drivers. Instead, it only hosts a small subset of operating system binaries that are needed for security and nothing else. All of these binaries are signed with a certificate that is trusted by virtualization-based security and these signatures are validated before launching the file in the protected environment.

When Windows Defender Credential Guard is enabled, NTLMv1, MS-CHAPv2, Digest, and CredSSP cannot use the signed-in credentials. Thus, single sign-on does not work with these protocols. However, applications can prompt for credentials or use credentials stored in the Windows Vault which are not protected by Windows Defender Credential Guard with any of these protocols. It is strongly recommended that valuable credentials, such as the sign-in credentials, not be used with any of these protocols. If these protocols must be used by domain or Azure AD users, secondary credentials should be provisioned for these use cases.

When Windows Defender Credential Guard is enabled, Kerberos does not allow unconstrained Kerberos delegation or DES encryption, not only for signed-in credentials, but also prompted or saved credentials.

Here's a high-level overview on how the LSA is isolated by using virtualization-based security:



See also

Related videos

Windows Defender Credential Guard: Requirements

3/26/2021 • 8 minutes to read • Edit Online

Applies to

- Windows 10 Enterprise
- Windows Server 2016

For Windows Defender Credential Guard to provide protection, the computers you are protecting must meet certain baseline hardware, firmware, and software requirements, which we will refer to as Hardware and software requirements. Additionally, Windows Defender Credential Guard blocks specific authentication capabilities, so applications that require such capabilities will break. We will refer to these requirements as Application requirements. Beyond these requirements, computers can meet additional hardware and firmware qualifications, and receive additional protections. Those computers will be more hardened against certain threats. For detailed information on baseline protections, plus protections for improved security that are associated with hardware and firmware options available in 2015, 2016, and 2017, refer to the tables in Security Considerations.

Hardware and software requirements

To provide basic protections against OS level attempts to read Credential Manager domain credentials, NTLM and Kerberos derived credentials, Windows Defender Credential Guard uses:

- Support for Virtualization-based security (required)
- Secure boot (required)
- Trusted Platform Module (TPM, preferred - provides binding to hardware) versions 1.2 and 2.0 are supported, either discrete or firmware
- UEFI lock (preferred - prevents attacker from disabling with a simple registry key change)

The Virtualization-based security requires:

- 64-bit CPU
- CPU virtualization extensions plus extended page tables
- Windows hypervisor (does not require Hyper-V Windows Feature to be installed)

Windows Defender Credential Guard deployment in virtual machines

Credential Guard can protect secrets in a Hyper-V virtual machine, just as it would on a physical machine. When Credential Guard is deployed on a VM, secrets are protected from attacks inside the VM. Credential Guard does not provide additional protection from privileged system attacks originating from the host.

Requirements for running Windows Defender Credential Guard in Hyper-V virtual machines

- The Hyper-V host must have an IOMMU, and run at least Windows Server 2016 or Windows 10 version 1607.
- The Hyper-V virtual machine must be Generation 2, have an enabled virtual TPM, and be running at least Windows Server 2016 or Windows 10.
  - TPM is not a requirement, but we recommend that you implement TPM.

For information about other host platforms, see Enabling Windows Server 2016 and Hyper-V virtualization based security features on other platforms.

For information about Windows Defender Remote Credential Guard hardware and software requirements, see

Windows Defender Remote Credential Guard requirements.

Application requirements

When Windows Defender Credential Guard is enabled, specific authentication capabilities are blocked, so applications that require such capabilities will break. Applications should be tested prior to deployment to ensure compatibility with the reduced functionality.

> **WARNING**
> Enabling Windows Defender Credential Guard on domain controllers is not supported. The domain controller hosts authentication services which integrate with processes isolated when Windows Defender Credential Guard is enabled, causing crashes.

> **NOTE**
> Windows Defender Credential Guard does not provide protections for the Active Directory database or the Security Accounts Manager (SAM). The credentials protected by Kerberos and NTLM when Windows Defender Credential Guard is enabled are also in the Active Directory database (on domain controllers) and the SAM (for local accounts).

Applications will break if they require:

- Kerberos DES encryption support
- Kerberos unconstrained delegation
- Extracting the Kerberos TGT
- NTLMv1

Applications will prompt and expose credentials to risk if they require:

- Digest authentication
- Credential delegation
- MS-CHAPv2

Applications may cause performance issues when they attempt to hook the isolated Windows Defender Credential Guard process.

Services or protocols that rely on Kerberos, such as file shares, remote desktop, or BranchCache, continue to work and are not affected by Windows Defender Credential Guard.

Security considerations

All computers that meet baseline protections for hardware, firmware, and software can use Windows Defender Credential Guard. Computers that meet additional qualifications can provide additional protections to further reduce the attack surface. The following tables describe baseline protections, plus protections for improved security that are associated with hardware and firmware options available in 2015, 2016, and 2017.

> **NOTE**
> Beginning with Windows 10, version 1607, Trusted Platform Module (TPM 2.0) must be enabled by default on new shipping computers.
>
> If you are an OEM, see PC OEM requirements for Windows Defender Credential Guard.

Baseline protections

BASELINE PROTECTIONS	DESCRIPTION	SECURITY BENEFITS
Hardware: **64-bit CPU**	A 64-bit computer is required for the Windows hypervisor to provide VBS.	
Hardware: **CPU virtualization extensions**, plus **extended page tables**	**Requirements**: - These hardware features are required for VBS: One of the following virtualization extensions: - VT-x (Intel) or - AMD-V And: - Extended page tables, also called Second Level Address Translation (SLAT).	VBS provides isolation of secure kernel from normal operating system. Vulnerabilities and Day 0s in normal operating system cannot be exploited because of this isolation.
Hardware: **Trusted Platform Module (TPM)**	**Requirement**: - TPM 1.2 or TPM 2.0, either discrete or firmware. TPM recommendations	A TPM provides protection for VBS encryption keys that are stored in the firmware. TPM helps protect against attacks involving a physically present user with BIOS access.
Firmware: **UEFI firmware version 2.3.1.c or higher with UEFI Secure Boot**	**Requirements**: - See the following Windows Hardware Compatibility Program requirement: System.Fundamentals.Firmware.UEFISecureBoot	UEFI Secure Boot helps ensure that the device boots only authorized code, and can prevent boot kits and root kits from installing and persisting across reboots.
Firmware: **Secure firmware update process**	**Requirements**: - UEFI firmware must support secure firmware update found under the following Windows Hardware Compatibility Program requirement: System.Fundamentals.Firmware.UEFISecureBoot.	UEFI firmware just like software can have security vulnerabilities that, when found, need to be patched through firmware updates. Patching helps prevent root kits from getting installed.
Software: Qualified **Windows operating system**	**Requirement**: - Windows 10 or Windows Server 2016.	Support for VBS and for management features that simplify configuration of Windows Defender Credential Guard.

IMPORTANT

Windows Server 2016 running as a domain controller does not support Windows Defender Credential Guard.

IMPORTANT

The following tables list additional qualifications for improved security. We strongly recommend meeting the additional qualifications to significantly strengthen the level of security that Windows Defender Credential Guard can provide.

2015 Additional security qualifications starting with Windows 10, version 1507, and Windows Server 2016 Technical Preview 4

PROTECTIONS FOR IMPROVED SECURITY	DESCRIPTION

PROTECTIONS FOR IMPROVED SECURITY	DESCRIPTION
Hardware: **IOMMU** (input/output memory management unit)	**Requirement**: - VT-D or AMD Vi IOMMU **Security benefits**: - An IOMMU can enhance system resiliency against memory attacks. For more information, see Advanced Configuration and Power Interface (ACPI) description tables
Firmware: **Securing Boot Configuration and Management**	**Requirements**: - BIOS password or stronger authentication must be supported. - In the BIOS configuration, BIOS authentication must be set. - There must be support for protected BIOS option to configure list of permitted boot devices (for example, "Boot only from internal hard drive") and boot device order, overriding BOOTORDER modification made by operating system. - In the BIOS configuration, BIOS options related to security and boot options (list of permitted boot devices, boot order) must be secured to prevent other operating systems from starting and to prevent changes to the BIOS settings.
Firmware: **Secure MOR, revision 2 implementation**	**Requirement**: - Secure MOR, revision 2 implementation

2016 Additional security qualifications starting with Windows 10, version 1607, and Windows Server 2016

> **IMPORTANT**
>
> The following tables list additional qualifications for improved security. Systems that meet these additional qualifications can provide more protections.

PROTECTIONS FOR IMPROVED SECURITY	DESCRIPTION	SECURITY BENEFITS
Firmware: **Hardware Rooted Trust Platform Secure Boot**	**Requirements**: - Boot Integrity (Platform Secure Boot) must be supported. See the Windows Hardware Compatibility Program requirements under System.Fundamentals.Firmware.CS.UEFISecureBoot.ConnectedStandby - The Hardware Security Test Interface (HSTI) must be implemented. See Hardware Security Testability Specification.	Boot Integrity (Platform Secure Boot) from Power-On provides protections against physically present attackers, and defense-in-depth against malware. - HSTI provides additional security assurance for correctly secured silicon and platform.
Firmware: **Firmware Update through Windows Update**	**Requirements**: - Firmware must support field updates through Windows Update and UEFI encapsulation update.	Helps ensure that firmware updates are fast, secure, and reliable.

PROTECTIONS FOR IMPROVED SECURITY	DESCRIPTION	SECURITY BENEFITS
Firmware: **Securing Boot Configuration and Management**	**Requirements**: - Required BIOS capabilities: Ability of OEM to add ISV, OEM, or Enterprise Certificate in Secure Boot DB at manufacturing time. - Required configurations: Microsoft UEFI CA must be removed from Secure Boot DB. Support for 3rd-party UEFI modules is permitted but should leverage ISV-provided certificates or OEM certificate for the specific UEFI software.	- Enterprises can choose to allow proprietary EFI drivers/applications to run. - Removing Microsoft UEFI CA from Secure Boot DB provides full control to enterprises over software that runs before the operating system boots.

2017 Additional security qualifications starting with Windows 10, version 1703

The following table lists qualifications for Windows 10, version 1703, which are in addition to all preceding qualifications.

PROTECTIONS FOR IMPROVED SECURITY	DESCRIPTION	SECURITY BENEFITS
Firmware: **VBS enablement of No-Execute (NX) protection for UEFI runtime services**	**Requirements**: - VBS will enable NX protection on UEFI runtime service code and data memory regions. UEFI runtime service code must support read-only page protections, and UEFI runtime service data must not be executable. UEFI runtime service must meet these requirements: - Implement UEFI 2.6 EFI_MEMORY_ATTRIBUTES_TABLE. All UEFI runtime service memory (code and data) must be described by this table. - PE sections must be page-aligned in memory (not required for in non-volatile storage). - The Memory Attributes Table needs to correctly mark code and data as RO/NX for configuration by the OS: - All entries must include attributes EFI_MEMORY_RO, EFI_MEMORY_XP, or both. - No entries may be left with neither of the above attributes, indicating memory that is both executable and writable. Memory must be either readable and executable or writable and non-executable. (**SEE IMPORTANT INFORMATION AFTER THIS TABLE**)	Vulnerabilities in UEFI runtime, if any, will be blocked from compromising VBS (such as in functions like UpdateCapsule and SetVariable) - Reduces the attack surface to VBS from system firmware.

PROTECTIONS FOR IMPROVED SECURITY	DESCRIPTION	SECURITY BENEFITS
Firmware: **Firmware support for SMM protection**	**Requirements**: - The Windows SMM Security Mitigations Table (WSMT) specification contains details of an ACPI table that was created for use with Windows operating systems that support Windows virtualization-based security (VBS) features.	- Protects against potential vulnerabilities in UEFI runtime services, if any, will be blocked from compromising VBS (such as in functions like UpdateCapsule and SetVariable) - Reduces the attack surface to VBS from system firmware. - Blocks additional security attacks against SMM.

IMPORTANT

Regarding **VBS enablement of NX protection for UEFI runtime services**:

- This only applies to UEFI runtime service memory, and not UEFI boot service memory.
- This protection is applied by VBS on OS page tables.

  Please also note the following:

  - Do not use sections that are both writable and executable
  - Do not attempt to directly modify executable system memory
  - Do not use dynamic code

Manage Windows Defender Credential Guard

4/8/2021 • 9 minutes to read • Edit Online

Applies to

- Windows 10 Enterprise or Education SKUs
- Windows Server 2016
- Windows Server 2019

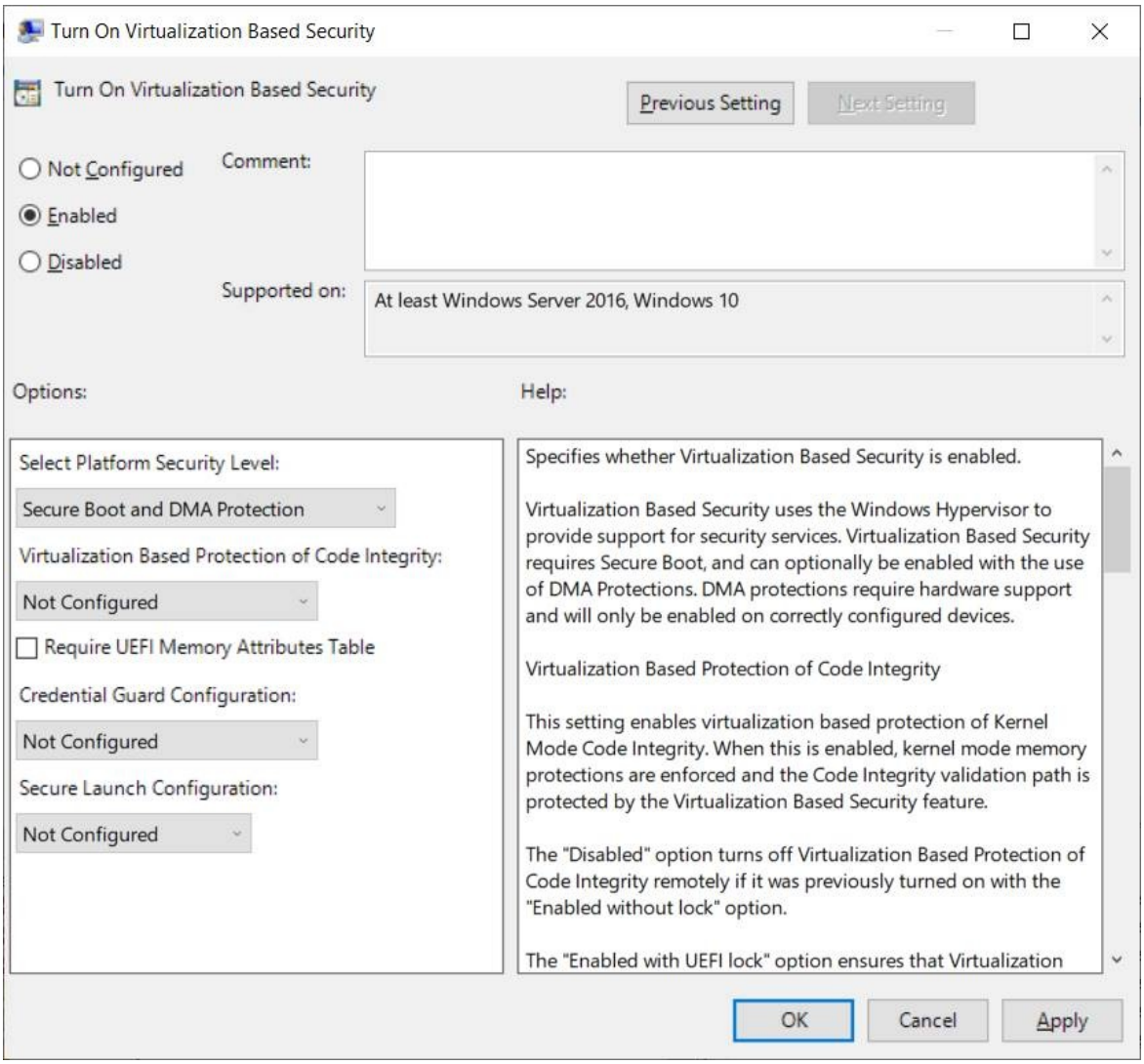
Enable Windows Defender Credential Guard

Windows Defender Credential Guard can be enabled either by using Group Policy, the registry, or the Hypervisor-Protected Code Integrity (HVCI) and Windows Defender Credential Guard hardware readiness tool. Windows Defender Credential Guard can also protect secrets in a Hyper-V virtual machine, just as it would on a physical machine. The same set of procedures used to enable Windows Defender Credential Guard on physical machines applies also to virtual machines.

Enable Windows Defender Credential Guard by using Group Policy

You can use Group Policy to enable Windows Defender Credential Guard. This will add and enable the virtualization-based security features for you if needed.

1. From the Group Policy Management Console, go to **Computer Configuration** -> **Administrative Templates** -> **System** -> **Device Guard**.
2. Double-click **Turn On Virtualization Based Security**, and then click the **Enabled** option.
3. In the **Select Platform Security Level** box, choose **Secure Boot** or **Secure Boot and DMA Protection**.
4. In the **Credential Guard Configuration** box, click **Enabled with UEFI lock**, and then click **OK**. If you want to be able to turn off Windows Defender Credential Guard remotely, choose **Enabled without lock**.
5. In the **Secure Launch Configuration** box, choose **Not Configured**, **Enabled** or **Disabled**. Check this article for more details.



6. Close the Group Policy Management Console.

To enforce processing of the group policy, you can run `gpupdate /force`.

Enable Windows Defender Credential Guard by using Intune

1. From **Home**, click **Microsoft Intune**.
2. Click **Device configuration**.
3. Click **Profiles** > **Create Profile** > **Endpoint protection** > **Windows Defender Credential Guard**.

> **NOTE**
> It will enable VBS and Secure Boot and you can do it with or without UEFI Lock. If you will need to disable Credential Guard remotely, enable it without UEFI lock.

> **TIP**
> You can also configure Credential Guard by using an account protection profile in endpoint security. See Account protection policy settings for endpoint security in Intune.

Enable Windows Defender Credential Guard by using the registry

If you don't use Group Policy, you can enable Windows Defender Credential Guard by using the registry. Windows Defender Credential Guard uses virtualization-based security features which have to be enabled first on some operating systems.

Add the virtualization-based security features

Starting with Windows 10, version 1607 and Windows Server 2016, enabling Windows features to use virtualization-based security is not necessary and this step can be skipped.

If you are using Windows 10, version 1507 (RTM) or Windows 10, version 1511, Windows features have to be enabled to use virtualization-based security. You can do this by using either the Control Panel or the Deployment Image Servicing and Management tool (DISM).

> **NOTE**
> If you enable Windows Defender Credential Guard by using Group Policy, the steps to enable Windows features through Control Panel or DISM are not required. Group Policy will install Windows features for you.

Add the virtualization-based security features by using Programs and Features

1. Open the Programs and Features control panel.
2. Click **Turn Windows feature on or off**.
3. Go to **Hyper-V** -> **Hyper-V Platform**, and then select the **Hyper-V Hypervisor** check box.
4. Select the **Isolated User Mode** check box at the top level of the feature selection.
5. Click **OK**.

Add the virtualization-based security features to an offline image by using DISM

1. Open an elevated command prompt.
2. Add the Hyper-V Hypervisor by running the following command:

```
dism /image:<WIM file name> /Enable-Feature /FeatureName:Microsoft-Hyper-V-Hypervisor /all
```

3. Add the Isolated User Mode feature by running the following command:

```
dism /image:<WIM file name> /Enable-Feature /FeatureName:IsolatedUserMode
```

> **NOTE**
> In Windows 10, version 1607 and later, the Isolated User Mode feature has been integrated into the core operating system. Running the command in step 3 above is therefore no longer required.

> **TIP**
> You can also add these features to an online image by using either DISM or Configuration Manager.

Enable virtualization-based security and Windows Defender Credential Guard

1. Open Registry Editor.
2. Enable virtualization-based security:
   a. Go to HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\DeviceGuard.
   b. Add a new DWORD value named **EnableVirtualizationBasedSecurity**. Set the value of this registry setting to 1 to enable virtualization-based security and set it to 0 to disable it.

c. Add a new DWORD value named **RequirePlatformSecurityFeatures**. Set the value of this registry setting to 1 to use **Secure Boot** only or set it to 3 to use **Secure Boot and DMA protection**.

3. Enable Windows Defender Credential Guard:

   a. Go to HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\LSA.

   b. Add a new DWORD value named **LsaCfgFlags**. Set the value of this registry setting to 1 to enable Windows Defender Credential Guard with UEFI lock, set it to 2 to enable Windows Defender Credential Guard without lock, and set it to 0 to disable it.

4. Close Registry Editor.

> **NOTE**
>
> You can also enable Windows Defender Credential Guard by setting the registry entries in the FirstLogonCommands unattend setting.

Enable Windows Defender Credential Guard by using the HVCI and Windows Defender Credential Guard hardware readiness tool

You can also enable Windows Defender Credential Guard by using the HVCI and Windows Defender Credential Guard hardware readiness tool.

```
DG_Readiness_Tool.ps1 -Enable -AutoReboot
```

> **IMPORTANT**
>
> When running the HVCI and Windows Defender Credential Guard hardware readiness tool on a non-English operating system, within the script, change `$OSArch = $(gwmi win32_operatingsystem).OSArchitecture` to be `$OSArch = $((gwmi win32_operatingsystem).OSArchitecture).tolower()` instead, in order for the tool to work.
>
> This is a known issue.

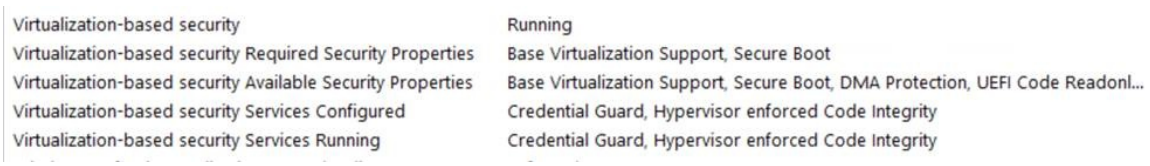
Review Windows Defender Credential Guard performance

Is Windows Defender Credential Guard running?

You can view System Information to check that Windows Defender Credential Guard is running on a PC.

1. Click **Start**, type **msinfo32.exe**, and then click **System Information**.
2. Click **System Summary**.
3. Confirm that **Credential Guard** is shown next to **Virtualization-based security Services Running**.

   Here's an example:



You can also check that Windows Defender Credential Guard is running by using the HVCI and Windows Defender Credential Guard hardware readiness tool.

```
DG_Readiness_Tool_v3.6.ps1 -Ready
```

IMPORTANT

When running the HVCI and Windows Defender Credential Guard hardware readiness tool on a non-English operating system, within the script, change `*$OSArch = $(gwmi win32_operatingsystem).OSArchitecture` to be `$OSArch = $((gwmi win32_operatingsystem).OSArchitecture).tolower()` instead, in order for the tool to work.

This is a known issue.

NOTE

For client machines that are running Windows 10 1703, LsaIso.exe is running whenever virtualization-based security is enabled for other features.

- We recommend enabling Windows Defender Credential Guard before a device is joined to a domain. If Windows Defender Credential Guard is enabled after domain join, the user and device secrets may already be compromised. In other words, enabling Credential Guard will not help to secure a device or identity that has already been compromised, which is why we recommend turning on Credential Guard as early as possible.
- You should perform regular reviews of the PCs that have Windows Defender Credential Guard enabled. This can be done with security audit policies or WMI queries. Here's a list of WinInit event IDs to look for:
  - **Event ID 13** Windows Defender Credential Guard (LsaIso.exe) was started and will protect LSA credentials.
  - **Event ID 14** Windows Defender Credential Guard (LsaIso.exe) configuration: [**0x0** | **0x1** | **0x2**], **0**
    - The first variable: **0x1** or **0x2** means that Windows Defender Credential Guard is configured to run. **0x0** means that it's not configured to run.
    - The second variable: **0** means that it's configured to run in protect mode. **1** means that it's configured to run in test mode. This variable should always be **0**.
  - **Event ID 15** Windows Defender Credential Guard (LsaIso.exe) is configured but the secure kernel is not running; continuing without Windows Defender Credential Guard.
  - **Event ID 16** Windows Defender Credential Guard (LsaIso.exe) failed to launch: [error code]
  - **Event ID 17** Error reading Windows Defender Credential Guard (LsaIso.exe) UEFI configuration: [error code]

    You can also verify that TPM is being used for key protection by checking Event ID 51 in the **Microsoft** -> **Windows** -> **Kernel-Boot** event source. If you are running with a TPM, the TPM PCR mask value will be something other than 0.
  - **Event ID 51** VSM Master Encryption Key Provisioning. Using cached copy status: **0x0**. Unsealing cached copy status: 0x1. New key generation status: 0x1. Sealing status: **0x1**. TPM PCR mask: **0x0**.
- You can use Windows PowerShell to determine whether credential guard is running on a client computer. On the computer in question, open an elevated PowerShell window and run the following command:

```
(Get-CimInstance -ClassName Win32_DeviceGuard -Namespace
root\Microsoft\Windows\DeviceGuard).SecurityServicesRunning
```

This command generates the following output:

- **0**: Windows Defender Credential Guard is disabled (not running)
- **1**: Windows Defender Credential Guard is enabled (running)

> **NOTE**
> Checking the task list or Task Manager to see if LSAISO.exe is running is not a recommended method for determining whether Windows Defender Credential Guard is running.

Disable Windows Defender Credential Guard

To disable Windows Defender Credential Guard, you can use the following set of procedures or the Device Guard and Credential Guard hardware readiness tool. If Credential Guard was enabled with UEFI Lock then you must use the following procedure as the settings are persisted in EFI (firmware) variables and it will require physical presence at the machine to press a function key to accept the change. If Credential Guard was enabled without UEFI Lock then you can turn it off by using Group Policy.

1. If you used Group Policy, disable the Group Policy setting that you used to enable Windows Defender Credential Guard (**Computer Configuration** -> **Administrative Templates** -> **System** -> **Device Guard** -> **Turn on Virtualization Based Security**).
2. Delete the following registry settings:
   - HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\LSA\LsaCfgFlags
   - HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\DeviceGuard\LsaCfgFlags
3. If you also wish to disable virtualization-based security delete the following registry settings:
   - HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\DeviceGuard\EnableVirtualizationBasedSecurity
   - HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\DeviceGuard\RequirePlatformSecurityFeatures

   > **IMPORTANT**
   > If you manually remove these registry settings, make sure to delete them all. If you don't remove them all, the device might go into BitLocker recovery.

4. Delete the Windows Defender Credential Guard EFI variables by using bcdedit. From an elevated command prompt, type the following commands:

```
mountvol X: /s
copy %WINDIR%\System32\SecConfig.efi X:\EFI\Microsoft\Boot\SecConfig.efi /Y
bcdedit /create {0cb3b571-2f2e-4343-a879-d86a476d7215} /d "DebugTool" /application osloader
bcdedit /set {0cb3b571-2f2e-4343-a879-d86a476d7215} path "\EFI\Microsoft\Boot\SecConfig.efi"
bcdedit /set {bootmgr} bootsequence {0cb3b571-2f2e-4343-a879-d86a476d7215}
bcdedit /set {0cb3b571-2f2e-4343-a879-d86a476d7215} loadoptions DISABLE-LSA-ISO
bcdedit /set {0cb3b571-2f2e-4343-a879-d86a476d7215} device partition=X:
mountvol X: /d
```

5. Restart the PC.
6. Accept the prompt to disable Windows Defender Credential Guard.
7. Alternatively, you can disable the virtualization-based security features to turn off Windows Defender

Credential Guard.

> **NOTE**
>
> The PC must have one-time access to a domain controller to decrypt content, such as files that were encrypted with EFS. If you want to turn off both Windows Defender Credential Guard and virtualization-based security, run the following bcdedit commands after turning off all virtualization-based security Group Policy and registry settings:
>
> ```
> bcdedit /set {0cb3b571-2f2e-4343-a879-d86a476d7215} loadoptions DISABLE-LSA-ISO,DISABLE-VBS
> bcdedit /set vsmlaunchtype off
> ```

For more info on virtualization-based security and HVCI, see Enable virtualization-based protection of code integrity.

> **NOTE**
>
> Credential Guard and Device Guard are not supported when using Azure Gen 1 VMs. These options are available with Gen 2 VMs only.

Disable Windows Defender Credential Guard by using the HVCI and Windows Defender Credential Guard hardware readiness tool

You can also disable Windows Defender Credential Guard by using the HVCI and Windows Defender Credential Guard hardware readiness tool.

```
DG_Readiness_Tool_v3.6.ps1 -Disable -AutoReboot
```

> **IMPORTANT**
>
> When running the HVCI and Windows Defender Credential Guard hardware readiness tool on a non-English operating system, within the script, change `*$OSArch = $(gwmi win32_operatingsystem).OSArchitecture` to be `$OSArch = $((gwmi win32_operatingsystem).OSArchitecture).tolower()` instead, in order for the tool to work.
>
> This is a known issue.

Disable Windows Defender Credential Guard for a virtual machine

From the host, you can disable Windows Defender Credential Guard for a virtual machine:

```
Set-VMSecurity -VMName <VMName> -VirtualizationBasedSecurityOptOut $true
```

Windows Defender Device Guard and Windows Defender Credential Guard hardware readiness tool

3/23/2021 • 19 minutes to read • Edit Online

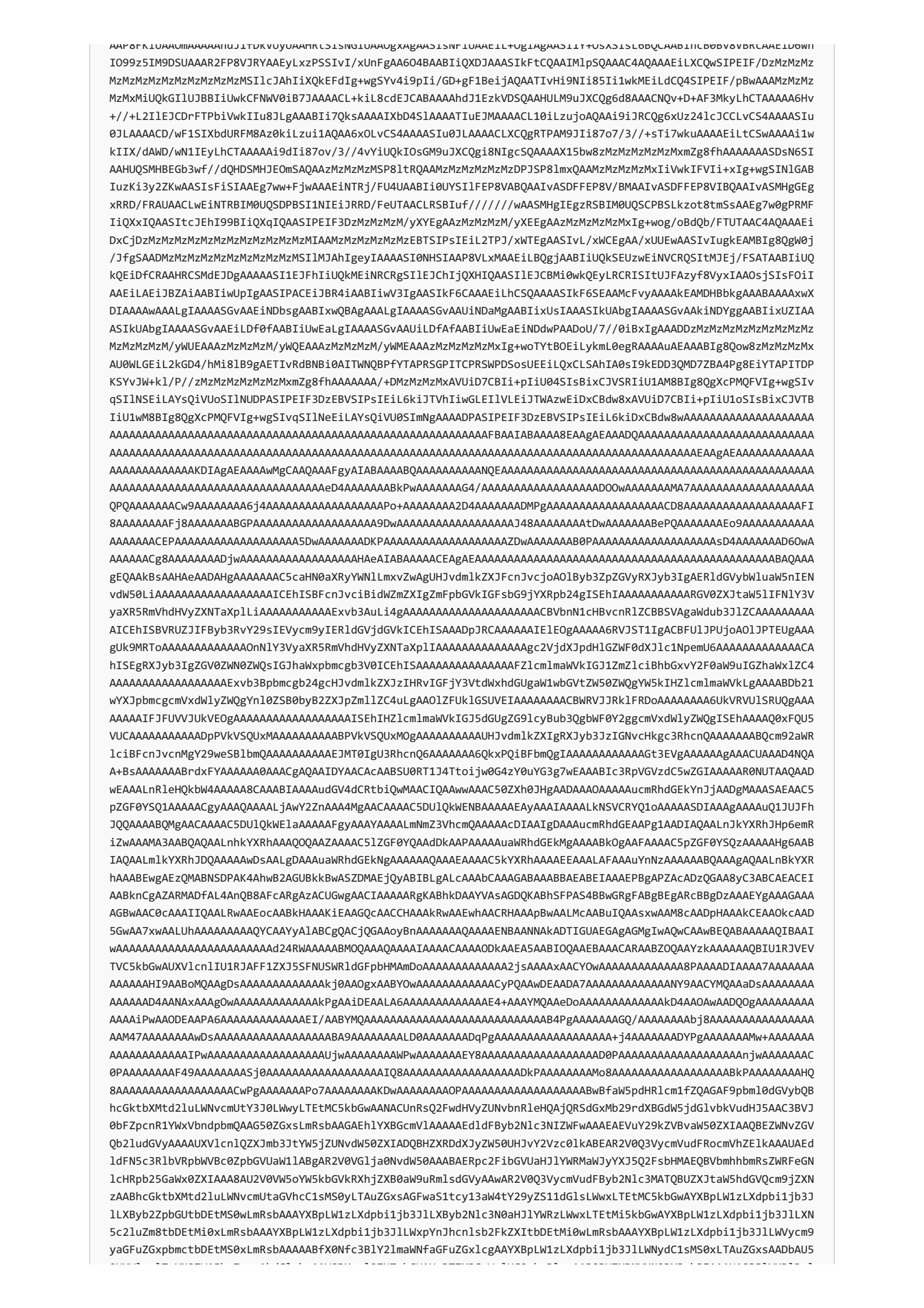
Applies to:

- Windows 10 Enterprise Edition

```
# Script to find out if a machine is Device Guard compliant.
# The script requires a driver verifier present on the system.

param([switch]$Capable, [switch]$Ready, [switch]$Enable, [switch]$Disable, $SIPolicyPath,
[switch]$AutoReboot, [switch]$DG, [switch]$CG, [switch]$HVCI, [switch]$HLK, [switch]$Clear,
[switch]$ResetVerifier)

$path = "C:\DGLogs\"
$LogFile = $path + "DeviceGuardCheckLog.txt"

$CompatibleModules = New-Object System.Text.StringBuilder
$FailingModules = New-Object System.Text.StringBuilder
$FailingExecuteWriteCheck = New-Object System.Text.StringBuilder

$DGVerifyCrit = New-Object System.Text.StringBuilder
$DGVerifyWarn = New-Object System.Text.StringBuilder
$DGVerifySuccess = New-Object System.Text.StringBuilder


$Sys32Path = "$env:windir\system32"
$DriverPath = "$env:windir\system32\drivers"

#generated by certutil -encode
$SIPolicy_Encoded = "BQAAAA43RKLJRAZMtVH2AW5WMHbk9wcuTBkgTbfJb0SmxaI0BACNkAgAAAAAAAAA
HQAAAAIAAAAAAAAAAAAKAEAAAAAMAAAAAQorBgEEAYI3CgMGDAAAAAEKKwYBBAGC
NwoDBQwAAAABCisGAQQBgjc9BAEMAAAAAQorBgEEAYI3PQUBDAAAAAEKKwYBBAGC
NwoDFQwAAAABCisGAQQBgjdMAwEMAAAAAQorBgEEAYI3TAUBDAAAAAEKKwYBBAGC
N0wLAQEAAAAGAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AQAAAAYAAAABAAAAAgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABAAAA
BgAAAAEAAAADAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAAAAGAAAA
AQAAAAEAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQAAAAUAAAABAAAA
AQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABAAAABAAAAAEAAAABAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAAAAGAAAAAQAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQAAAAYAAAABAAAAAgAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABAAAABgAAAAEAAAADAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAAAAGAAAAAQAAAAEAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAQAAAAUAAAABAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAABAAAADgAAAAEAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAEAAAAOAAAAAQAAAAIAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AQAAAA4AAAABAAAAAwAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABAAAA
DgAAAAEAAAABAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAAAAOAAAA
AQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQAAAA4AAAABAAAA
AgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABAAAADgAAAAEAAAADAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAAAAOAAAAAQAAAAEAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQAAAAQAAAABAAAAAQAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAIAAAAPye3j3MoJGGstO/m3OKIFDLGlVN
otyttV8/cu4XchN4AQAAAAUAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AQAAAAYAAAABAAAABAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABAAAA
DgAAAAEAAAAEAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAAAAHAAAA
AQAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQAAAAoAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAAAAKAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABAAAADAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAQAAAAYAAAABAAAABgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

AAAAAAAAAAABAAAABwAAAAEAAAAHAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAABAAAAFAAAAIMAAAAAAAAAADIAAAAsAAAAAAAAAAAAAAAEAAAAAAAAA
AgAAAAAAAAADAAAAAAAAAAQAAAAAAAAABQAAAAAAAAALAAAAAAAAAAwAAAAAAAAA
DQAAAAAAAAAOAAAAAAAAAABgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAMAAAAAAAAAAyAAAASAAAABgAAAAAAAAAAHAAAAAAAAAAgAAAAAAAAA
CQAAAAAAAAAKAAAAAAAAAABMAAAAAAAAAADwAAAAAAAAAAQAAAAAAAAAABEAAAAAAAAA
EgAAAAAAAAAUAAAAAAAAAABUAAAAAAAAAAGgAAAAAAAAAAbAAAAAAAAAABwAAAAAAAAA
FgAAAAAAAAAXAAAAAAAAAABkAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAgAAABQAAABQAG8AbABpAGMAeQBJAG4AZgBvAAAAAAAWAAAA
SQBuAGYAbwByAG0AYQB0AGkAbwBuAAAAAAAAAAAQAAABJAGQAAAAAAAMAAAAMAAAA
MAAzADEAMAAxADcAAAAAABQAAABQAG8AbABpAGMAeQBJAG4AZgBvAAAAAAAWAAAA
SQBuAGYAbwByAG0AYQB0AGkAbwBuAAAAAAAAAAAgAAABOAGEAbQBlAAAAAAADAAAA
JgAAAEQAZQBmAGEAdQBsAHQAVwBpAG4AZABvAHcAcwBBAHUAZABpAHQAAAAAAAAAAAA
AwAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAAAA
BQAAAAYAAAA="

$HSTITest_Encoded =
"TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA4AAAAA4fug4AtAnNIbgBTM0hVGh
pcyBwcm9ncmFtIGNhbm5vdCBiZSBydW4gaW4gRE9TIG1vZGUuDQ0KJAAAAAAAAAAADxXZfstTz5v7U8+b+1PPm/2GH4vrc8+b+8RGq/ojz5v9h
h+r63PPm/2GH9vr48+b+1PPi/qjz5v9hh+b60PPm/2GHwvrc8+b/YYfu+tDz5v1JpY2i1PPm/AAAAAAAAAAAAABQRQAAZIYFAGt3EVgAAAAAAAA
AAPAAIiALAg4AABIAAAAaAAAAAAAAkBsAAAAQAAAAAACAAQAAAAAQAAAAAgAACgAAAAoAAAAKAAAAAAAAAABwAAAABAAAxcwAAAMAYEEAAAQ
AAAAAAAAQAAAAAAAAAAAAQAAAAAAAEAAAAAAAAAAAAAAAQAAAAEDkAAGQAAAB0OQAABAEAAAAAAAAAAAAAAFAAACABAAAAAAAAAAAAAAABgAAA
YAAAAwDUAADgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQMAAA0AAAAAAAAAAAAAAAAAAAA4DAAAEgBAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAudGV4dAAAAMURAAAAEAAAABIAAAAEAAAAAAAAAAAAAAAAAAAAgAABgLnJkYXRhAAB4DwAAADAAAAAQAAAAFgAAAAAAAAAAAAAAAAAAQAA
AQC5kYXRhAAAAwAUAAABAAAAAAgAAACYAAAAAAAAAAAAAAAAAAEAAAMAucGRhdGEAACABAAAAUAAAAAIAAAAoAAAAAAAAAAAAAAAAAABAAAB
ALnJlbG9jAAAYAAAAGAAAAACAAAAKgAAAAAAAAAAAAAAAAAAQAAAQgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABIiVwkCFVWV0FWQVd
Ii+xIg+wwM/9IjUU4TIv5iX1ISI1NSIl9QEUzyYl9OEyNRUBIiUQkIDPS6AwJAACL2D1XAAeAD4WrAAAAi0VASGnYDCIAAP8V/yAAAI13CEy
Lw0iLyIvW/xX2IAAATIvwSIXAdQe7DgAHgOtxi104/xXWIAAARIvDi9ZIi8j/FdAgAABIi/BIhcB1B7sOAAeA6x5IjUU4TIvOTI1FQEiJRCQ
gSYvWSI1NSOiNCAAAi9j/FZUgAABNi8Yz0kiLyP8VlyAAAEiF9nQU/xV8IAAATIvGM9JIi8j/FX4gAAA5fUhAD5THQYk/i8NIi1wkYEiDxDB
BX0FeX15dw8zMzMzMzMzOkzCAAAzMzMzMzEiJXCQYSIl0JCBXSIHscAEAAEiLBbsuAABIM8RIiYQkYAEAAA8QBRkhAACL8kiL+TPSSI1
MJGBBuPQAAADzD39EJFDo6g4AAEiDZCQwAEiNTCRQg2QkQABFM8nHRCQogAAAALoAAABAx0QkIAMAAABFjUEB/xWSHwAASIvYSIP4/3RGQbk
CAAAARTPAM9JIi8j/FX0fAACD+P90HkiDZCQgAEyNTCRARIvGSIvXSIvL/xVmHwAAhcB1Bv8VPB8AAEiLy/8VYx8AAEiLjCRgAQAASDPM6As
LAABMjZwkcAEAAEmLWyBJi3MoSYvjX8PMzMzMzMxIg+woM9JMi8lIhcl0Hrr///9/M8BEi8I4AXQJSP/BSYPoAXXzTYXAdSEz0rhXAAeAM8m
FwEgPScp4C41RAUmLyejG/v//SIPEKMNJK9Dr4czMzMzMzMSIlcJAhIiXQkEFdIg+wgQYvZSYv4SIvy6Iv///+L00iLz+iN/v//SIvOSIt
cJDBIi3QkOEiDxCBf6Wr////MzMzMyJVCQQSIPsKAkRSI0Nsx8AAOhO////ugQAAABIjUwkOOhL/v//SI0NqB8AAOgz////SIPEKMPMzMz
MzMxAVVNWV0FUQVVBVkFXSI1sJOFIgeyYAAAASIsF6CwAAEgzxEiJRQ9FM/ZIiVXnM9JIiU3vRIl1p0GL3kiJXbdJi8BIiUXXTYvpRIl1r0G
L/kSJdfdFi+ZIiVX7RYv+SIlVA0yJdc9IhckPhBEFAABIhcAPhAgFAABNhckPhP8EAABBgzkBdBHHRaeAAAAAvwJAAIDp7QQAAEiNDQkfAAD
ohP7//0WLfQREiX2/SWnfDCIAAP8Vtx0AAEyLw7oIAAAASIvI/xWuHQAATIvgSIXAdShIjQ3vHgAA6Er+////FUwdAAAPt/iBzwAAB4CFwA9
O+EmL3umLBAAASI0N9x4AAOgi/v//RIl1s0WF/w+EiwIAAEmNXQhIiV3HSY20JAwCAABIjQ32HgAA6Pn9//+LQwiJhvT9//+FwHktPbsAAMB
1EUiNDe4eAADo2f3//+kaAgAASI0N/R4AAOjI/f//g02nQOkFAgAAixtJA92DOwN0Gw+6bacIugEAAABIjY78f//6Dv+///p4AEAAEyNhgD
+//+6BAAAAEmLwEiNSwgPEAFIjYmAAAAADxEASI2AgAAAAA8QSZAPEUiQDxBBoA8RQKAPEEmwDxFIsA8QQcAPEUDADxBJ0A8RSNAPEEHgDxF
A4A8QSfAPEUjwSIPqAXWuQbkAAgAASI0VgB4AAEiNDYEeAADodP3//4uLCAIAALoAEAAAQYv+TI0ES0iBwQwCAABMA8FIi85MK8ZIjYL+7/9
/SIXAdBdBD7cECGaFwHQNZokBSIPBAkiD6gF13UiF0nUJSIPpAr96AAeAZkSJMUiNFSYeAABIjQ0nHgAAQbkAIAAATIvG6AH9//9MjXMEQYs
OjUH/g/gDD4fDAAAA/0SN90iNFQMeAACJjvj9//9BuQQAAABIjQ34HQAATYvG6Mj8//9BiwaDfIX3AXZESI2O/P3//7oEAAAA6PH8//9Biw6
D6QF0JYPpAXQag+kBdA+D+QEPhaIAAACDTacI63eDTacE63GDTacC62uDTacB62WD+AF1YIuDCAIAAEyNRa9BuQQAAACJRa9IjRWTHQAASI0
NrB0AAOhP/P//RTP2RDl1r3UOD7ptpwlBjVYI6TX+//9IjYMMAgAASIlFz+sZD7ptpwlIjY78/f//ugIAAADoWfz//0Uz9otFs0iBxgwiAAB
Ig0XHDP/AiUWzQTvHcxdIi13H6ZP9//+/BUAAgEiLXbfp5wEAAEQ5dad0DkiNDU0dAADoePv//+vji12v/xW1GgAARIvDuggAAABIi8j/Faw
aAABIiUW3SIvYSIXAdRZIjQ1JHQAA6ET7//+/FwAA0OmXAQAASI0NYx0AAOgu+///i0WvRI2wBgEAAEaNNHBEiXWzRYX/D4TFAAAASY1cJAh
JjXUISI0N+xsAAOj++v//gXv4uwAAwHUOSI0N/hsAAOjp+v//63xEOXYEcxS6EAAAAA+6bacJSIvL6Gv7///rYosOSQPMVVUJ05TWNL0Wvd0Ae
6CAAAAOvaRTPATI0MQUyNFAhEOUWvdjpMi3W3Qw+2jBAMAgAA99FDhIwDIAIAAHQID7ptpwmDCyBDioQIDAIAAEMIBDBB/8BEO0Wvcs5Ei3W
zSIPGDEiBwwwiAABJg+8BD4VM////RIt9v0iLXbdFM/ZEO0Wnd8vLS9MhHAAA6Dn6///pkQAAAEGL9kQ5da8Phm4AAABMi3W3TItttz5NhkQ
cAADoE/r//4vGTI0FmcAAABCigwwSAACL1kGLDIDoLz4MvwAAegI8IRSaE/wgNC5lBAAAASI0VJx0AAEiNDYMbAADoSvr//0iNDa4bAADoRvr
DOEWrdBBIjQ2dHAAA6Lj5//+DZacAi0Wnv8qyEEQAAEkD1kGNQghNjQQWSYPHAYlFp02F7XQFTYX/dKOLRbNBiQdIhdt0QUiNDQgcAADoQ/n
F5HRdTIt150iLddNIi02fSIXJdAZIiwFJiQFIhcl0DUyNRcdIjVXvTIlF1+hE/f//SI2OlhsAAOjR+P//SIPF////pkQAAAEGL9kQ5da8Phm4A
AAE2LxDPSSIvI/xVwGAAASIXbdBT/FVUYAABMi8Mz0kiLyP8VVxgAAEiLRe9IhcB0B0iNDT8cAADoSgMAAEmLz+i6AwAAxwZXAAeASIXAdFF
dQVxfXltdw8zMzMzMxIi8RIiVgISIlwEEiJeBhMiWAgQVVBVkFXSIPsMEUz9kyL+kSL4UyNLV3SUDCAAEiLZCQwSIPEIF/DzMzMzMzMzEBVSI1
AdAkPuusfcxiAAAg6WvPr//0iL5SIvT6H4AgABIi9hIhcB0RkiL00iLzuhl/P//i/hIi8voHwIAAIX/dSpIi1QkSEiL1kiLzuhK/P//hcB1EEiF
JSIvQuaYAAAD/FQGYAACL2EiL1EiLzuhX/P//TI1EJEBIi9OLzugbAAAASI1MJEBIi9BIg8Qgi8PoHAMAAEmLzv8VyxcAAEiLXCRgSIt0JGh
YSIt8JHBIg8RAQV5fXsPMSIlcJAhXSIPsIEiL+TPbSIvTSIvP6Jf5//+FwHgJ/8OD+wRy7DPAi8NIi1wkMEiDxCBfw8zMzMzMSIlcJAhIiWw
0JGBIg8QgX8PMzMzMzMzMzMzMSIlcJAhXSIPsIEiL2jPSSIv56GoBAACLw0iLXCQwSIPEIF/DzMzMzMzMzMzMSIl
AAB8FKiUAAOmAAAAAhdJ1FEiJXCQIV0iD7CBIi9lIi/ro9gEAAEiLXCQwSIPEIF/DzMzMzMzMzMzMzMzMzMzMzMzMzMzMzMzMzMzMzEBTSIPsIEiL2v8VVRYAAEiL0zPJ/xVaFgAAiQNIg8QgW8PM



```
0UXVlcnlTeXN0ZW1JbmZvcm1hdGlvbgAAWQBXcml0ZUZpbGUAUwBTZXRGaWxlUG9pbnRlcgAABQBHZXRMYXN0RXJyb3IAAAUAQ3JlYXRlRml
sZUEAAABDbG9zZUhhbmRsZQACAEhlYXBBbGxvYwBhcGktbXMtd2luLWNvcmUtZmlsZS1sMS0yLTEuZGxsAGFwaS1tcy13aW4tY29yZS1oYW5
kbGUtbDEtMS0wLmRsbAAzAG1lbWNweQAANwBtZW1zZXQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAyot8tmSsAAM1dINJm1P//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABAQAAAXEQAAwDcAACwRAAAaEgAA1DcAACASAABwEgAA8DcAAHgSAAC2EgAA+Dc
AALwSAADyEgAACDgAAPgSAABRGQAAEDgAAFgZAACaGgAAMDgAAKAaAAB7GwAAyDgAAJAbAADNGwAA+DcAANQbAAD8HAAASDgAABAdAAAuHQA
A2DgAAFQdAAAkHgAA3DgAAEQeAABdHgAA8DcAAHweAACwHgAA6DgAAMAeAAAxIAAA8DgAAGwgAACJIAAA8DcAAJAgAADrIAAA/DgAAAhAAA
CIQAA+DgAAAghAAAqIQAAwDgAACohAABMIQAAwDgAAEwhAABuIQAAwDgAAG4hAACQIQAAwDgAAJAhAAC1IQAAwDgAALUhAADFIQAAwDgAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAADAAABgAAAAAoAigaKCAoIigkKA
oojCiAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAA="
```


```
function Log($message)
{
    $message | Out-File $LogFile -Append -Force
}

function LogAndConsole($message)
{
    Write-Host $message
    Log $message
}

function LogAndConsoleWarning($message)
{
    Write-Host $message -foregroundcolor "Yellow"
    Log $message
}

function LogAndConsoleSuccess($message)
{
    Write-Host $message -foregroundcolor "Green"
    Log $message
}

function LogAndConsoleError($message)
{
    Write-Host $message -foregroundcolor "Red"
    Log $message
}

function IsExempted([System.IO.FileInfo] $item)
{
    $cert = (Get-AuthenticodeSignature $item.FullName).SignerCertificate
    if($cert.ToString().Contains("CN=Microsoft Windows, O=Microsoft Corporation, L=Redmond, S=Washington, 
C=US"))
    {
        Log $item.FullName + "MS Exempted"
        return 1
    }
    else
    {
        Log $item.FullName + "Not-exempted"
        Log $cert.ToString()
        return 0
    }
```

```
}

function CheckExemption($_ModName)
{
    $mod1 = Get-ChildItem $Sys32Path $_ModName
    $mod2 = Get-ChildItem $DriverPath $_ModName
    if($mod1)
    {
        Log "NonDriver module" + $mod1.FullName
        return IsExempted($mod1)
    }
    elseif($mod2)
    {
        Log "Driver Module" + $mod2.FullName
        return IsExempted($mod2)
    }

}

function CheckFailedDriver($_ModName, $CIStats)
{
    Log "Module: " $_ModName.Trim()
    if(CheckExemption($_ModName.Trim()) - eq 1)
    {
        $CompatibleModules.AppendLine("Windows Signed: " + $_ModName.Trim()) | Out-Null
        return
    }
    $index = $CIStats.IndexOf("execute pool type count:".ToLower())
    if($index -eq -1)
    {
        return
    }
    $_tempStr = $CIStats.Substring($index)
    $Result = "PASS"
    $separator = "`r`n",""
    $option = [System.StringSplitOptions]::RemoveEmptyEntries
    $stats = $_tempStr.Split($separator,$option)
    Log $stats.Count

    $FailingStat = ""
    foreach( $stat in $stats)
    {
        $_t =$stat.Split(":")
        if($_t.Count -eq 2 -and $_t[1].trim() -ne "0")
        {
            $Result = "FAIL"
            $FailingStat = $stat
            break
        }
    }
    if($Result.Contains("PASS"))
    {
        $CompatibleModules.AppendLine($_ModName.Trim()) | Out-Null
    }
    elseif($FailingStat.Trim().Contains("execute-write"))
    {
        $FailingExecuteWriteCheck.AppendLine("Module: "+ $_ModName.Trim() + "`r`n`tReason: " +
$FailingStat.Trim() ) | Out-Null
    }
    else
    {
        $FailingModules.AppendLine("Module: "+ $_ModName.Trim() + "`r`n`tReason: " + $FailingStat.Trim() ) |
Out-Null
    }
    Log "Result: " $Result
}

function ListCIStats($_ModName, $str1)
{
```

```
    $i1 = $str1.IndexOf("Code Integrity Statistics:".ToLower())
    if($i1 -eq -1 )
    {
        Log "String := " $str1
        Log "Warning! CI Stats are missing for " $_ModName
        return
    }
    $temp_str1 = $str1.Substring($i1)
    $CIStats = $temp_str1.Substring(0).Trim()

    CheckFailedDriver $_ModName $CIStats
}

function ListDrivers($str)
{
    $_tempStr= $str

    $separator = "module:",""
    $option = [System.StringSplitOptions]::RemoveEmptyEntries
    $index1 = $_tempStr.IndexOf("MODULE:".ToLower())
    if($index1 -lt 0)
    {
        return
    }
    $_tempStr = $_tempStr.Substring($Index1)
    $_SplitStr = $_tempStr.Split($separator,$option)


    Log $_SplitStr.Count
    LogAndConsole "Verifying each module please wait ... "
    foreach($ModuleDetail in $_Splitstr)
    {
        #LogAndConsole $Module
        $Index2 = $ModuleDetail.IndexOf("(")
        if($Index2 -eq -1)
        {
            "Skipping .."
            continue
        }
        $ModName = $ModuleDetail.Substring(0,$Index2-1)
        Log "Driver: " $ModName
        Log "Processing module: " $ModName
        ListCIStats $ModName $ModuleDetail
    }

    $DriverScanCompletedMessage = "Completed scan. List of Compatible Modules can be found at " + $LogFile
    LogAndConsole $DriverScanCompletedMessage

    if($FailingModules.Length -gt 0 -or $FailingExecuteWriteCheck.Length -gt 0 )
    {
        $WarningMessage = "Incompatible HVCI Kernel Driver Modules found"
        if($HLK)
        {
            LogAndConsoleError $WarningMessage
        }
        else
        {
            LogAndConsoleWarning $WarningMessage
        }

        LogAndConsoleError $FailingExecuteWriteCheck.ToString()
        if($HLK)
        {
            LogAndConsoleError $FailingModules.ToString()
        }
        else
        {
            LogAndConsoleWarning $FailingModules.ToString()
        }
```

```
        if($FailingModules.Length -ne 0 -or $FailingExecuteWriteCheck.Length -ne 0 )
        {
            if($HLK)
            {
                $DGVerifyCrit.AppendLine($WarningMessage) | Out-Null
            }
            else
            {
                $DGVerifyWarn.AppendLine($WarningMessage) | Out-Null
            }
        }
    }
    else
    {
        LogAndConsoleSuccess "No Incompatible Drivers found"
    }
}

function ListSummary()
{
    if($DGVerifyCrit.Length -ne 0 )
    {
        LogAndConsoleError "Machine is not Device Guard / Credential Guard compatible because of the
following:"
        LogAndConsoleError $DGVerifyCrit.ToString()
        LogAndConsoleWarning $DGVerifyWarn.ToString()
        if(!$HVCI -and !$DG)
        {
            ExecuteCommandAndLog 'REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Capabilities\"
/v "CG_Capable" /t REG_DWORD /d 0 /f '
        }
        if(!$CG)
        {
            ExecuteCommandAndLog 'REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Capabilities\"
/v "DG_Capable" /t REG_DWORD /d 0 /f '
            ExecuteCommandAndLog 'REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Capabilities\"
/v "HVCI_Capable" /t REG_DWORD /d 0 /f '
        }

    }
    elseif ($DGVerifyWarn.Length -ne 0 )
    {
        LogAndConsoleSuccess "Device Guard / Credential Guard can be enabled on this machine.`n"
        LogAndConsoleWarning "The following additional qualifications, if present, can enhance the security
of Device Guard / Credential Guard on this system:"
        LogAndConsoleWarning $DGVerifyWarn.ToString()
        if(!$HVCI -and !$DG)
        {
            ExecuteCommandAndLog 'REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Capabilities\"
/v "CG_Capable" /t REG_DWORD /d 1 /f '
        }
        if(!$CG)
        {
            ExecuteCommandAndLog 'REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Capabilities\"
/v "DG_Capable" /t REG_DWORD /d 1 /f '
            ExecuteCommandAndLog 'REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Capabilities\"
/v "HVCI_Capable" /t REG_DWORD /d 1 /f '
        }
    }
    else
    {
        LogAndConsoleSuccess "Machine is Device Guard / Credential Guard Ready.`n"
        if(!$HVCI -and !$DG)
        {
            ExecuteCommandAndLog 'REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Capabilities\"
/v "CG_Capable" /t REG_DWORD /d 2 /f '
        }
        if(!$CG)
        {
```

```
        {
            ExecuteCommandAndLog 'REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Capabilities\"
/v "DG_Capable" /t REG_DWORD /d 2 /f '
            ExecuteCommandAndLog 'REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Capabilities\"
/v "HVCI_Capable" /t REG_DWORD /d 2 /f '
        }
    }
}


function Instantiate-Kernel32 {
    try
    {
        Add-Type -TypeDefinition @"
        using System;
        using System.Diagnostics;
        using System.Runtime.InteropServices;

        public static class Kernel32
        {
            [DllImport("kernel32", SetLastError=true, CharSet = CharSet.Ansi)]
                public static extern IntPtr LoadLibrary(
                    [MarshalAs(UnmanagedType.LPStr)]string lpFileName);

            [DllImport("kernel32", CharSet=CharSet.Ansi, ExactSpelling=true, SetLastError=true)]
                public static extern IntPtr GetProcAddress(
                    IntPtr hModule,
                    string procName);
        }

"@
    }
    catch
    {
        Log $_.Exception.Message
        LogAndConsole "Instantiate-Kernel32 failed"
    }
}

function Instantiate-HSTI {
    try
    {
        Add-Type -TypeDefinition @"
        using System;
        using System.Diagnostics;
        using System.Runtime.InteropServices;
        using System.Net;

        public static class HstiTest3
        {
            [DllImport("hstitest.dll", CharSet = CharSet.Unicode)]
            public static extern int QueryHSTIdetails(
                ref HstiOverallError pHstiOverallError,
                [In, Out] HstiProviderErrorDuple[] pHstiProviderErrors,
                ref uint pHstiProviderErrorsCount,
                byte[] hstiPlatformSecurityBlob,
                ref uint pHstiPlatformSecurityBlobBytes);

            [DllImport("hstitest.dll", CharSet = CharSet.Unicode)]
            public static extern int QueryHSTI(ref bool Pass);

            [StructLayout(LayoutKind.Sequential, CharSet = CharSet.Unicode)]
            public struct HstiProviderErrorDuple
            {
                internal uint protocolError;
                internal uint role;
                internal HstiProviderErrors providerError;
                [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 256)]
                internal string ID;
                [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 4096)]
```

```
            [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 4096)]
            internal string ErrorString;
        }

        [FlagsAttribute]
        public enum HstiProviderErrors : int
        {
            None              = 0x00000000,
            VersionMismatch   = 0x00000001,
            RoleUnknown       = 0x00000002,
            RoleDuplicated    = 0x00000004,
            SecurityFeatureSizeMismatch   = 0x00000008,
            SizeTooSmall      = 0x00000010,
            VerifiedMoreThanImplemented   = 0x00000020,
            VerifiedNotMatchImplemented   = 0x00000040
        }

        [FlagsAttribute]
        public enum HstiOverallError : int
        {
            None                                = 0x00000000,
            RoleTooManyPlatformReference        = 0x00000001,
            RoleTooManyIbv                      = 0x00000002,
            RoleTooManyOem                      = 0x00000004,
            RoleTooManyOdm                      = 0x00000008,
            RoleMissingPlatformReference        = 0x00000010,
            VerifiedIncomplete                  = 0x00000020,
            ProtocolErrors                      = 0x00000040,
            BlobVersionMismatch                 = 0x00000080,
            PlatformSecurityVersionMismatch     = 0x00000100,
            ProviderError                       = 0x00000200
        }

    }
"@

    $LibHandle = [Kernel32]::LoadLibrary("C:\Windows\System32\hstitest.dll")
    $FuncHandle = [Kernel32]::GetProcAddress($LibHandle, "QueryHSTIdetails")
    $FuncHandle2 = [Kernel32]::GetProcAddress($LibHandle, "QueryHSTI")

    if ([System.IntPtr]::Size -eq 8)
    {
        #assuming 64 bit
        Log "`nKernel32::LoadLibrary   64bit --> 0x$("{0:X16}" -f $LibHandle.ToInt64())"
        Log "HstiTest2::QueryHSTIdetails  64bit --> 0x$("{0:X16}" -f $FuncHandle.ToInt64())"
    }
    else
    {
        return
    }
    $overallError = New-Object HstiTest3+HstiOverallError
    $providerErrorDupleCount = New-Object int
    $blobByteSize = New-Object int
    $hr = [HstiTest3]::QueryHSTIdetails([ref] $overallError, $null, [ref] $providerErrorDupleCount,
$null, [ref] $blobByteSize)

    [byte[]]$blob = New-Object byte[] $blobByteSize
    [HstiTest3+HstiProviderErrorDuple[]]$providerErrors = New-Object HstiTest3+HstiProviderErrorDuple[]
$providerErrorDupleCount
    $hr = [HstiTest3]::QueryHSTIdetails([ref] $overallError, $providerErrors, [ref]
$providerErrorDupleCount, $blob, [ref] $blobByteSize)
    $string = $null
    $blob | foreach { $string = $string + $_.ToString("X2")+"," }

    $hstiStatus = New-Object bool
    $hr = [HstiTest3]::QueryHSTI([ref] $hstiStatus)

    LogAndConsole "HSTI Duple Count: $providerErrorDupleCount"
    LogAndConsole "HSTI Blob size: $blobByteSize"
    LogAndConsole "Strings: $string"
```

```
        LogAndConsole "String: $string"
        LogAndConsole "HSTIStatus: $hstiStatus"
        if(($blobByteSize -gt 512) -and ($providerErrorDupleCount -gt 0) -and $hstiStatus)
        {
            LogAndConsoleSuccess "HSTI validation successful"
        }
        elseif(($providerErrorDupleCount -eq 0) -or ($blobByteSize -le 512))
        {
            LogAndConsoleWarning "HSTI is absent"
            $DGVerifyWarn.AppendLine("HSTI is absent") | Out-Null
        }
        else
        {
            $ErrorMessage = "HSTI validation failed"
            if($HLK)
            {
                LogAndConsoleError $ErrorMessage
                $DGVerifyCrit.AppendLine($ErrorMessage) | Out-Null
            }
            else
            {
                LogAndConsoleWarning $ErrorMessage
                $DGVerifyWarn.AppendLine("HSTI is absent") | Out-Null
            }
        }

    }
    catch
    {
        LogAndConsoleError $_.Exception.Message
        LogAndConsoleError "Instantiate-HSTI failed"
    }
}


function CheckDGRunning($_val)
{
    $DGObj = Get-CimInstance -classname Win32_DeviceGuard -namespace root\Microsoft\Windows\DeviceGuard
    for($i=0; $i -lt $DGObj.SecurityServicesRunning.length; $i++)
    {
        if($DGObj.SecurityServicesRunning[$i] -eq $_val)
        {
            return 1
        }

    }
    return 0
}

function CheckDGFeatures($_val)
{
    $DGObj = Get-CimInstance -classname Win32_DeviceGuard -namespace root\Microsoft\Windows\DeviceGuard
    Log "DG_obj $DG_obj"
    Log "DG_obj.AvailableSecurityProperties.length $DG_obj.AvailableSecurityProperties.length"
    for($i=0; $i -lt $DGObj.AvailableSecurityProperties.length; $i++)
    {
        if($DGObj.AvailableSecurityProperties[$i] -eq $_val)
        {
            return 1
        }

    }
    return 0
}

function PrintConfigCIDetails($_ConfigCIState)
{
    $_ConfigCIRunning = "Config-CI is enabled and running."
    $_ConfigCIDisabled = "Config-CI is not running."
```

```
    $_ConfigCIMode = "Not Enabled"
    switch ($_ConfigCIState)
    {
        0 { $_ConfigCIMode = "Not Enabled" }
        1 { $_ConfigCIMode = "Audit mode" }
        2 { $_ConfigCIMode = "Enforced mode" }
        default { $_ConfigCIMode = "Not Enabled" }
    }

    if($_ConfigCIState -ge 1)
    {
        LogAndConsoleSuccess "$_ConfigCIRunning ($_ConfigCIMode)"
    }
    else
    {
        LogAndConsoleWarning "$_ConfigCIDisabled ($_ConfigCIMode)"
    }
}

function PrintHVCIDetails($_HVCIState)
{
    $_HvciRunning = "HVCI is enabled and running."
    $_HvciDisabled = "HVCI is not running."

    if($_HVCIState)
    {
        LogAndConsoleSuccess $_HvciRunning
    }
    else
    {
        LogAndConsoleWarning $_HvciDisabled
    }
}

function PrintCGDetails ($_CGState)
{
    $_CGRunning = "Credential-Guard is enabled and running."
    $_CGDisabled = "Credential-Guard is not running."

    if($_CGState)
    {
        LogAndConsoleSuccess $_CGRunning
    }
    else
    {
        LogAndConsoleWarning $_CGDisabled
    }
}

if(![IO.Directory]::Exists($path))
{
    New-Item -ItemType directory -Path $path
}
else
{
    #Do Nothing!!
}

function IsRedstone
{
    $_osVersion = [environment]::OSVersion.Version
    Log $_osVersion
    #Check if build Major is Windows 10
    if($_osVersion.Major -lt 10)
    {
        return 0
    }
    #Check if the build is post Threshold2 (1511 release) => Redstone
    if($_osVersion.Build -gt 10586)
```

```
    {
        return 1
    }
    #default return False
    return 0
}

function ExecuteCommandAndLog($_cmd)
{
    try
    {
        Log "Executing: $_cmd"
        $CmdOutput = Invoke-Expression $_cmd | Out-String
        Log "Output: $CmdOutput"
    }
    catch
    {
        Log "Exception while exectuing $_cmd"
        Log $_.Exception.Message
    }


}

function PrintRebootWarning
{
    LogAndConsoleWarning "Please reboot the machine, for settings to be applied."
}

function AutoRebootHelper
{
    if($AutoReboot)
    {
        LogAndConsole "PC will restart in 30 seconds"
        ExecuteCommandAndLog 'shutdown /r /t 30'
    }
    else
    {
        PrintRebootWarning
    }

}

function VerifierReset
{
    $verifier_state = verifier /query | Out-String
    if(!$verifier_state.ToString().Contains("No drivers are currently verified."))
    {
        ExecuteCommandAndLog 'verifier.exe /reset'
    }
    AutoRebootHelper
}

function PrintHardwareReq
{
    LogAndConsole "#############################################################################"
    LogAndConsole "OS and Hardware requirements for enabling Device Guard and Credential Guard"
    LogAndConsole " 1. OS SKUs: Available only on these OS Skus - Enterprise, Server, Education and
Enterprise IoT"
    LogAndConsole " 2. Hardware: Recent hardware that supports virtualization extension with SLAT"
    LogAndConsole "To learn more please visit: https://aka.ms/dgwhcr"
    LogAndConsole "############################################################################# `n"
}

function CheckDriverCompat
{
    $_HVCIState = CheckDGRunning(2)
    if($_HVCIState)
    {
```

```
        LogAndConsoleWarning "HVCI is already enabled on this machine, driver compat list might not be
complete."
        LogAndConsoleWarning "Please disable HVCI and run the script again..."
    }
    $verifier_state = verifier /query | Out-String
    if($verifier_state.ToString().Contains("No drivers are currently verified."))
    {
        LogAndConsole "Enabling Driver verifier"
        verifier.exe /flags 0x02000000 /all /bootmode oneboot /log.code_integrity

        LogAndConsole "Enabling Driver Verifier and Rebooting system"
        Log $verifier_state
        LogAndConsole "Please re-execute this script after reboot...."
        if($AutoReboot)
        {
            LogAndConsole "PC will restart in 30 seconds"
            ExecuteCommandAndLog 'shutdown /r /t 30'
        }
        else
        {
            LogAndConsole "Please reboot manually and run the script again...."
        }
        exit
    }
    else
    {
        LogAndConsole "Driver verifier already enabled"
        Log $verifier_state
        ListDrivers($verifier_state.Trim().ToLowerInvariant())
    }
}
function IsDomainController
{
    $_isDC = 0
    $CompConfig = Get-WmiObject Win32_ComputerSystem
    foreach ($ObjItem in $CompConfig)
    {
        $Role = $ObjItem.DomainRole
        Log "Role=$Role"
        Switch ($Role)
        {
            0 { Log "Standalone Workstation" }
            1 { Log "Member Workstation" }
            2 { Log "Standalone Server" }
            3 { Log "Member Server" }
            4
            {
                Log "Backup Domain Controller"
                $_isDC=1
                break
            }
            5
            {
                Log "Primary Domain Controller"
                $_isDC=1
                break
            }
            default { Log "Unknown Domain Role" }
        }
    }
    return $_isDC
}

function CheckOSSKU
{
    $osname = $((Get-ComputerInfo).WindowsProductName).ToLower()
    $_SKUSupported = 0
    Log "OSNAME:$osname"
    $SKUarray = @("Enterprise", "Education", "IoT", "Windows Server")
```

```
    $HLKAllowed = @("windows 10 pro")
    foreach ($SKUent in $SKUarray)
    {
        if($osname.ToString().Contains($SKUent.ToLower()))
        {
            $_SKUSupported = 1
            break
        }
    }

    # For running HLK tests only, professional SKU's are marked as supported.
    if($HLK)
    {
        if($osname.ToString().Contains($HLKAllowed.ToLower()))
        {
            $_SKUSupported = 1
        }
    }
    $_isDomainController = IsDomainController
    if($_SKUSupported)
    {
        LogAndConsoleSuccess "This PC edition is Supported for DeviceGuard";
        if(($_isDomainController -eq 1) -and !$HVCI -and !$DG)
        {
            LogAndConsoleError "This PC is configured as a Domain Controller, Credential Guard is not
supported on DC."
        }
        ExecuteCommandAndLog 'REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Capabilities\" /v
"OSSKU" /t REG_DWORD /d 2 /f '
    }
    else
    {
        LogAndConsoleError "This PC edition is Unsupported for Device Guard"
        $DGVerifyCrit.AppendLine("OS SKU unsupported") | Out-Null
        ExecuteCommandAndLog 'REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Capabilities\" /v
"OSSKU" /t REG_DWORD /d 0 /f '
    }
}

function CheckOSArchitecture
{
    $OSArch = $(gwmi win32_operatingsystem).OSArchitecture.ToLower()
    Log $OSArch
    if($OSArch -match ("^64\-?\s?bit"))
    {
        LogAndConsoleSuccess "64 bit architecture"
    }
    elseif($OSArch -match ("^32\-?\s?bit"))
    {
        LogAndConsoleError "32 bit architecture"
        $DGVerifyCrit.AppendLine("32 Bit OS, OS Architecture failure.") | Out-Null
    }
    else
    {
        LogAndConsoleError "Unknown architecture"
        $DGVerifyCrit.AppendLine("Unknown OS, OS Architecture failure.") | Out-Null
    }
}

function CheckSecureBootState
{
    $_secureBoot = Confirm-SecureBootUEFI
    Log $_secureBoot
    if($_secureBoot)
    {
        LogAndConsoleSuccess "Secure Boot is present"
        ExecuteCommandAndLog 'REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Capabilities\" /v
"SecureBoot" /t REG_DWORD /d 2 /f '
    }
```

```
    }
    else
    {
        LogAndConsoleError "Secure Boot is absent / not enabled."
        LogAndConsoleError "If Secure Boot is supported on the system, enable Secure Boot in the BIOS and
run the script again."
        ExecuteCommandAndLog 'REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Capabilities\" /v
"SecureBoot" /t REG_DWORD /d 0 /f '
        $DGVerifyCrit.AppendLine("Secure boot validation failed.") | Out-Null
    }
}

function CheckVirtualization
{
    $_vmmExtension = $(gwmi -Class Win32_processor).VMMonitorModeExtensions
    $_vmFirmwareExtension = $(gwmi -Class Win32_processor).VirtualizationFirmwareEnabled
    $_vmHyperVPresent =  (gcim -Class Win32_ComputerSystem).HypervisorPresent
    Log "VMMonitorModeExtensions $_vmmExtension"
    Log "VirtualizationFirmwareEnabled $_vmFirmwareExtension"
    Log "HyperVisorPresent $_vmHyperVPresent"

    #success if either processor supports and enabled or if hyper-v is present
    if(($_vmmExtension -and $_vmFirmwareExtension) -or $_vmHyperVPresent )
    {
        LogAndConsoleSuccess "Virtualization firmware check passed"
         ExecuteCommandAndLog 'REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Capabilities\" /v
"Virtualization" /t REG_DWORD /d 2 /f '
    }
    else
    {
        LogAndConsoleError "Virtualization firmware check failed."
        LogAndConsoleError "If Virtualization extensions are supported on the system, enable hardware
virtualization (Intel Virtualization Technology, Intel VT-x, Virtualization Extensions, or similar) in the
BIOS and run the script again."
        ExecuteCommandAndLog 'REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Capabilities\" /v
"Virtualization" /t REG_DWORD /d 0 /f '
        $DGVerifyCrit.AppendLine("Virtualization firmware check failed.") | Out-Null
    }
}

function CheckTPM
{
    $TPMLockout = $(get-tpm).LockoutCount

    if($TPMLockout)
    {

        if($TPMLockout.ToString().Contains("Not Supported for TPM 1.2"))
        {
            if($HLK)
            {
                LogAndConsoleSuccess "TPM 1.2 is present."
            }
            else
            {
                $WarningMsg = "TPM 1.2 is Present. TPM 2.0 is Preferred."
                LogAndConsoleWarning $WarningMsg
                $DGVerifyWarn.AppendLine($WarningMsg) | Out-Null
            }
        }
        else
        {
            LogAndConsoleSuccess "TPM 2.0 is present."
        }
        ExecuteCommandAndLog 'REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Capabilities\" /v
"TPM" /t REG_DWORD /d 2 /f '
    }
    else
    {
        $WarningMsg = "TPM is absent or not ready for use"
```

```
        $warningmsg = "TPM is absent or not ready for use"
        if($HLK)
        {
            LogAndConsoleError $WarningMsg
            $DGVerifyCrit.AppendLine($WarningMsg) | Out-Null
        }
        else
        {
            LogAndConsoleWarning $WarningMsg
            $DGVerifyWarn.AppendLine($WarningMsg) | Out-Null
        }
        ExecuteCommandAndLog 'REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Capabilities\" /v
"TPM" /t REG_DWORD /d 0 /f '
    }
}

function CheckSecureMOR
{
    $isSecureMOR = CheckDGFeatures(4)
    Log "isSecureMOR= $isSecureMOR "
    if($isSecureMOR -eq 1)
    {
        LogAndConsoleSuccess "Secure MOR is available"
        ExecuteCommandAndLog 'REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Capabilities\" /v
"SecureMOR" /t REG_DWORD /d 2 /f '
    }
    else
    {
        $WarningMsg = "Secure MOR is absent"
        if($HLK)
        {
            LogAndConsoleError $WarningMsg
            $DGVerifyCrit.AppendLine($WarningMsg) | Out-Null
        }
        else
        {
            LogAndConsoleWarning $WarningMsg
            $DGVerifyWarn.AppendLine($WarningMsg) | Out-Null
        }
        ExecuteCommandAndLog 'REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Capabilities\" /v
"SecureMOR" /t REG_DWORD /d 0 /f '
    }
}

function CheckNXProtection
{
    $isNXProtected = CheckDGFeatures(5)
    Log "isNXProtected= $isNXProtected "
    if($isNXProtected -eq 1)
    {
        LogAndConsoleSuccess "NX Protector is available"
        ExecuteCommandAndLog 'REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Capabilities\" /v
"UEFINX" /t REG_DWORD /d 2 /f '
    }
    else
    {
        LogAndConsoleWarning "NX Protector is absent"
        $DGVerifyWarn.AppendLine("NX Protector is absent") | Out-Null
        ExecuteCommandAndLog 'REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Capabilities\" /v
"UEFINX" /t REG_DWORD /d 0 /f '
    }
}

function CheckSMMProtection
{
    $isSMMMitigated = CheckDGFeatures(6)
    Log "isSMMMitigated= $isSMMMitigated "
    if($isSMMMitigated -eq 1)
    {
```

```
        LogAndConsoleSuccess "SMM Mitigation is available"
        ExecuteCommandAndLog 'REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Capabilities\" /v
"SMMProtections" /t REG_DWORD /d 2 /f '
    }
    else
    {
        LogAndConsoleWarning "SMM Mitigation is absent"
        $DGVerifyWarn.AppendLine("SMM Mitigation is absent") | Out-Null
        ExecuteCommandAndLog 'REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Capabilities\" /v
"SMMProtections" /t REG_DWORD /d 0 /f '
    }
}

function CheckHSTI
{
    LogAndConsole "Copying HSTITest.dll"
    try
    {
        $HSTITest_Decoded = [System.Convert]::FromBase64String($HSTITest_Encoded)
        [System.IO.File]::WriteAllBytes("$env:windir\System32\hstitest.dll",$HSTITest_Decoded)

    }
    catch
    {
        LogAndConsole $_.Exception.Message
        LogAndConsole "Copying and loading HSTITest.dll failed"
    }

    Instantiate-Kernel32
    Instantiate-HSTI
}

function PrintToolVersion
{
    LogAndConsole ""
    LogAndConsole "################################################################################"
    LogAndConsole ""
    LogAndConsole "Readiness Tool Version 3.7.2 Release. `nTool to check if your device is capable to run
Device Guard and Credential Guard."
    LogAndConsole ""
    LogAndConsole "################################################################################"
    LogAndConsole ""

}

PrintToolVersion

if(!($Ready) -and !($Capable) -and !($Enable) -and !($Disable) -and !($Clear) -and !($ResetVerifier))
{
    #Print Usage if none of the options are specified
    LogAndConsoleWarning "How to read the output:"
    LogAndConsoleWarning ""
    LogAndConsoleWarning " 1. Red Errors: Basic things are missing that will prevent enabling and using
DG/CG"
    LogAndConsoleWarning " 2. Yellow Warnings: This device can be used to enable and use DG/CG, but `n
additional security benefits will be absent. To learn more please go through: https://aka.ms/dgwhcr"
    LogAndConsoleWarning " 3. Green Messages: This device is fully compliant with DG/CG requirements`n"

    LogAndConsoleWarning "################################################################################"
    LogAndConsoleWarning ""
    LogAndConsoleWarning "Hardware requirements for enabling Device Guard and Credential Guard"
    LogAndConsoleWarning " 1. Hardware: Recent hardware that supports virtualization extension with SLAT"
    LogAndConsoleWarning ""
    LogAndConsoleWarning "################################################################################ `n"

    LogAndConsoleWarning "Usage: DG_Readiness.ps1 -[Capable/Ready/Enable/Disable/Clear] -[DG/CG/HVCI] -
[AutoReboot] -Path"
    LogAndConsoleWarning "Log file with details is found here: C:\DGLogs `n"
```

```
    LogAndConsoleWarning "To Enable DG/CG. If you have a custom SIPolicy.p7b then use the -Path parameter
else the hardcoded default policy is used"
    LogAndConsoleWarning "Usage: DG_Readiness.ps1 -Enable OR DG_Readiness.ps1 -Enable -Path <full path to
the SIPolicy.p7b> `n"

    LogAndConsoleWarning "To Enable only HVCI"
    LogAndConsoleWarning "Usage: DG_Readiness.ps1 -Enable -HVCI `n"

    LogAndConsoleWarning "To Enable only CG"
    LogAndConsoleWarning "Usage: DG_Readiness.ps1 -Enable -CG `n"

    LogAndConsoleWarning "To Verify if DG/CG is enabled"
    LogAndConsoleWarning "Usage: DG_Readiness.ps1 -Ready `n"

    LogAndConsoleWarning "To Disable DG/CG."
    LogAndConsoleWarning "Usage: DG_Readiness.ps1 -Disable `n"

    LogAndConsoleWarning "To Verify if DG/CG is disabled"
    LogAndConsoleWarning "Usage: DG_Readiness.ps1 -Ready `n"

    LogAndConsoleWarning "To Verify if this device is DG/CG Capable"
    LogAndConsoleWarning "Usage: DG_Readiness.ps1 -Capable`n"

    LogAndConsoleWarning "To Verify if this device is HVCI Capable"
    LogAndConsoleWarning "Usage: DG_Readiness.ps1 -Capable -HVCI`n"

    LogAndConsoleWarning "To Auto reboot with each option"
    LogAndConsoleWarning "Usage: DG_Readiness.ps1 -[Capable/Enable/Disable] -AutoReboot`n"
    LogAndConsoleWarning "################################################################################"
    LogAndConsoleWarning ""
    LogAndConsoleWarning "When the Readiness Tool with '-capable' is run the following RegKey values are
set:"
    LogAndConsoleWarning ""
    LogAndConsoleWarning "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\DeviceGuard\Capabilities"
    LogAndConsoleWarning "CG_Capable"
    LogAndConsoleWarning "DG_Capable"
    LogAndConsoleWarning "HVCI_Capable"
    LogAndConsoleWarning ""
    LogAndConsoleWarning "Value 0 = not possible to enable DG/CG/HVCI on this device"
    LogAndConsoleWarning "Value 1 = not fully compatible but has sufficient firmware/hardware/software
features to enable DG/CG/HVCI"
    LogAndConsoleWarning "Value 2 = fully compatible for DG/CG/HVCI"
    LogAndConsoleWarning ""
    LogAndConsoleWarning "################################################################################ `n"
}

$user = [Security.Principal.WindowsIdentity]::GetCurrent();
$TestForAdmin = (New-Object Security.Principal.WindowsPrincipal
$user).IsInRole([Security.Principal.WindowsBuiltinRole]::Administrator)

if(!$TestForAdmin)
{
    LogAndConsoleError "This script requires local administrator privileges. Please execute this script as a
local administrator."
    exit
}

$isRunningOnVM = (get-wmiobject win32_computersystem).model
if($isRunningOnVM.Contains("Virtual"))
{
    LogAndConsoleWarning "Running on a Virtual Machine. DG/CG is supported only if both guest VM and host
machine are running with Windows 10, version 1703 or later with English localization."
}


<# Check the DG status if enabled or disabled, meaning if the device is ready or not #>
if($Ready)
{
    PrintHardwareReq
```

```
    $DGRunning = $(Get-CimInstance -classname Win32_DeviceGuard -namespace
root\Microsoft\Windows\DeviceGuard).SecurityServicesRunning
    $_ConfigCIState = $(Get-CimInstance -classname Win32_DeviceGuard -namespace
root\Microsoft\Windows\DeviceGuard).CodeIntegrityPolicyEnforcementStatus
    Log "Current DGRunning = $DGRunning, ConfigCI= $_ConfigCIState"
    $_HVCIState = CheckDGRunning(2)
    $_CGState = CheckDGRunning(1)

    if($HVCI)
    {
        Log "_HVCIState: $_HVCIState"
        PrintHVCIDetails $_HVCIState
    }
    elseif($CG)
    {
        Log "_CGState: $_CGState"
        PrintCGDetails $_CGState

        if($_CGState)
        {
            ExecuteCommandAndLog 'REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Capabilities\"
/v "CG_Running" /t REG_DWORD /d 1 /f'
        }
        else
        {
            ExecuteCommandAndLog 'REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Capabilities\"
/v "CG_Running" /t REG_DWORD /d 0 /f'
        }
    }
    elseif($DG)
    {
        Log "_HVCIState: $_HVCIState, _ConfigCIState: $_ConfigCIState"

        PrintHVCIDetails $_HVCIState
        PrintConfigCIDetails $_ConfigCIState

        if($_ConfigCIState -and $_HVCIState)
        {
            LogAndConsoleSuccess "HVCI, and Config-CI are enabled and running."

            ExecuteCommandAndLog 'REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Capabilities\"
/v "DG_Running" /t REG_DWORD /d 1 /f'
        }
        else
        {
            LogAndConsoleWarning "Not all services are running."

            ExecuteCommandAndLog 'REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Capabilities\"
/v "DG_Running" /t REG_DWORD /d 0 /f'
        }
    }
    else
    {
        Log "_CGState: $_CGState, _HVCIState: $_HVCIState, _ConfigCIState: $_ConfigCIState"

        PrintCGDetails $_CGState
        PrintHVCIDetails $_HVCIState
        PrintConfigCIDetails $_ConfigCIState

        if(($DGRunning.Length -ge 2) -and ($_CGState) -and ($_HVCIState) -and ($_ConfigCIState -ge 1))
        {
            LogAndConsoleSuccess "HVCI, Credential Guard, and Config CI are enabled and running."
        }
        else
        {
            LogAndConsoleWarning "Not all services are running."
        }
    }
```

```
}

<# Enable and Disable #>
if($Enable)
{
    PrintHardwareReq

    LogAndConsole "Enabling Device Guard and Credential Guard"
    LogAndConsole "Setting RegKeys to enable DG/CG"

    ExecuteCommandAndLog 'REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard" /v
"EnableVirtualizationBasedSecurity" /t REG_DWORD /d 1 /f'
    #Only SecureBoot is required as part of RequirePlatformSecurityFeatures
    ExecuteCommandAndLog 'REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard" /v
"RequirePlatformSecurityFeatures" /t REG_DWORD /d 1 /f'

    $_isRedstone = IsRedstone
    if(!$_isRedstone)
    {
        ExecuteCommandAndLog 'REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard" /v "Unlocked" /t
REG_DWORD /d 1 /f'
    }
    else
    {
        ExecuteCommandAndLog 'REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard" /v "Locked" /t
REG_DWORD /d 0 /f'
    }

    if(!$HVCI -and !$DG)
    {
        # value is 2 for both Th2 and RS1
        ExecuteCommandAndLog 'REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" /v "LsaCfgFlags" /t
REG_DWORD /d 2 /f'
    }
    if(!$CG)
    {
        if(!$_isRedstone)
        {
            ExecuteCommandAndLog 'REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard" /v
"HypervisorEnforcedCodeIntegrity" /t REG_DWORD /d 1 /f'
        }
        else
        {
            ExecuteCommandAndLog 'REG ADD
"HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Scenarios\HypervisorEnforcedCodeIntegrity" /v "Enabled"
/t REG_DWORD /d 1 /f'
            ExecuteCommandAndLog 'REG ADD
"HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Scenarios\HypervisorEnforcedCodeIntegrity" /v "Locked" /t
REG_DWORD /d 0 /f'
        }
    }

    try
    {
        if(!$HVCI -and !$CG)
        {
            if(!$SIPolicyPath)
            {
                Log "Writing Decoded SIPolicy.p7b"
                $SIPolicy_Decoded = [System.Convert]::FromBase64String($SIPolicy_Encoded)

[System.IO.File]::WriteAllBytes("$env:windir\System32\CodeIntegrity\SIPolicy.p7b",$SIPolicy_Decoded)
            }
            else
            {
                LogAndConsole "Copying user provided SIpolicy.p7b"
                $CmdOutput = Copy-Item $SIPolicyPath "$env:windir\System32\CodeIntegrity\SIPolicy.p7b" |
Out-String
                Log $CmdOutput
```

```
                Log $CmdOutput
            }
        }
    }
    catch
    {
        LogAndConsole "Writing SIPolicy.p7b file failed"
    }

    LogAndConsole "Enabling Hyper-V and IOMMU"
    $_isRedstone = IsRedstone
    if(!$_isRedstone)
    {
        LogAndConsole "OS Not Redstone, enabling IsolatedUserMode separately"
        #Enable/Disable IOMMU separately
        ExecuteCommandAndLog 'DISM.EXE /Online /Enable-Feature:IsolatedUserMode /NoRestart'
    }
    $CmdOutput =  DISM.EXE /Online /Enable-Feature:Microsoft-Hyper-V-Hypervisor /All /NoRestart | Out-String
    if(!$CmdOutput.Contains("The operation completed successfully."))
    {
        $CmdOutput =  DISM.EXE /Online /Enable-Feature:Microsoft-Hyper-V-Online /All /NoRestart | Out-String
    }

    Log $CmdOutput
    if($CmdOutput.Contains("The operation completed successfully."))
    {
        LogAndConsoleSuccess "Enabling Hyper-V and IOMMU successful"
        #Reg key for HLK validation of DISM.EXE step
        ExecuteCommandAndLog 'REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Capabilities\" /v
"HyperVEnabled" /t REG_DWORD /d 1 /f'
    }
    else
    {
        LogAndConsoleWarning "Enabling Hyper-V failed please check the log file"
        #Reg key for HLK validation of DISM.EXE step
        ExecuteCommandAndLog 'REG ADD "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Capabilities\" /v
"HyperVEnabled" /t REG_DWORD /d 0 /f'
    }
    AutoRebootHelper
}

if($Disable)
{
    LogAndConsole "Disabling Device Guard and Credential Guard"
    LogAndConsole "Deleting RegKeys to disable DG/CG"

    ExecuteCommandAndLog 'REG DELETE "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard" /v
"EnableVirtualizationBasedSecurity" /f'
    ExecuteCommandAndLog 'REG DELETE "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard" /v
"RequirePlatformSecurityFeatures" /f'

    $_isRedstone = IsRedstone
    if(!$_isRedstone)
    {
        ExecuteCommandAndLog 'REG DELETE "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard" /v "NoLock" /f'
    }
    else
    {
        ExecuteCommandAndLog 'REG DELETE "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard" /v "Locked" /f'
    }

    if(!$CG)
    {
        ExecuteCommandAndLog 'REG DELETE "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard" /v
"HypervisorEnforcedCodeIntegrity" /f'
        if($_isRedstone)
        {
            ExecuteCommandAndLog 'REG DELETE
"HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Scenarios\HypervisorEnforcedCodeIntegrity" /f'
        }
```

```
        }
    }

    if(!$HVCI -and !$DG)
    {
        ExecuteCommandAndLog 'REG DELETE "HKLM\SYSTEM\CurrentControlSet\Control\Lsa" /v "LsaCfgFlags" /f'
    }

    if(!$HVCI -and !$CG)
    {
        ExecuteCommandAndLog 'del  "$env:windir\System32\CodeIntegrity\SIPolicy.p7b"'
    }

    if(!$HVCI -and !$DG -and !$CG)
    {
        LogAndConsole "Disabling Hyper-V and IOMMU"
        $_isRedstone = IsRedstone
        if(!$_isRedstone)
        {
            LogAndConsole "OS Not Redstone, disabling IsolatedUserMode separately"
            #Enable/Disable IOMMU separately
            ExecuteCommandAndLog 'DISM.EXE /Online /disable-Feature /FeatureName:IsolatedUserMode
/NoRestart'
        }
        $CmdOutput =  DISM.EXE /Online /disable-Feature /FeatureName:Microsoft-Hyper-V-Hypervisor /NoRestart
| Out-String
        if(!$CmdOutput.Contains("The operation completed successfully."))
        {
            $CmdOutput =  DISM.EXE /Online /disable-Feature /FeatureName:Microsoft-Hyper-V-Online /NoRestart
| Out-String
        }
        Log $CmdOutput
        if($CmdOutput.Contains("The operation completed successfully."))
        {
            LogAndConsoleSuccess "Disabling Hyper-V and IOMMU successful"
        }
        else
        {
            LogAndConsoleWarning "Disabling Hyper-V failed please check the log file"
        }

        #set of commands to run SecConfig.efi to delete UEFI variables if were set in pre OS
        #these steps can be performed even if the UEFI variables were not set - if not set it will lead to
No-Op but this can be run in general always
        #this requires a reboot and accepting the prompt in the Pre-OS which is self explanatory in the
message that is displayed in pre-OS
        $FreeDrive = ls function:[s-z]: -n | ?{ !(test-path $_) } | random
        Log "FreeDrive=$FreeDrive"
        ExecuteCommandAndLog 'mountvol $FreeDrive /s'
        $CmdOutput = Copy-Item "$env:windir\System32\SecConfig.efi"
$FreeDrive\EFI\Microsoft\Boot\SecConfig.efi -Force | Out-String
        LogAndConsole $CmdOutput
        ExecuteCommandAndLog 'bcdedit /create "{0cb3b571-2f2e-4343-a879-d86a476d7215}" /d DGOptOut
/application osloader'
        ExecuteCommandAndLog 'bcdedit /set "{0cb3b571-2f2e-4343-a879-d86a476d7215}" path
\EFI\Microsoft\Boot\SecConfig.efi'
        ExecuteCommandAndLog 'bcdedit /set "{bootmgr}" bootsequence "{0cb3b571-2f2e-4343-a879-
d86a476d7215}"'
        ExecuteCommandAndLog 'bcdedit /set "{0cb3b571-2f2e-4343-a879-d86a476d7215}" loadoptions DISABLE-LSA-
ISO,DISABLE-VBS'
        ExecuteCommandAndLog 'bcdedit /set "{0cb3b571-2f2e-4343-a879-d86a476d7215}" device
partition=$FreeDrive'
        ExecuteCommandAndLog 'mountvol $FreeDrive /d'
        #steps complete

    }
    AutoRebootHelper
}
```

```
if($Clear)
{
    ExecuteCommandAndLog 'REG DELETE "HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Capabilities" /f'
    VerifierReset
}

if($ResetVerifier)
{
    VerifierReset
}

<# Is machine Device Guard / Cred Guard Capable and Verify #>
if($Capable)
{
    PrintHardwareReq

    LogAndConsole "Checking if the device is DG/CG Capable"

    $_isRedstone = IsRedstone
    if(!$_isRedstone)
    {
        LogAndConsoleWarning "Capable is currently fully supported in Redstone only.."
    }
    $_StepCount = 1
    if(!$CG)
    {
        LogAndConsole " ====================== Step $_StepCount Driver Compat ====================== "
        $_StepCount++
        CheckDriverCompat
    }

    LogAndConsole " ====================== Step $_StepCount Secure boot present ====================== "
    $_StepCount++
    CheckSecureBootState

    if(!$HVCI -and !$DG -and !$CG)
    {
        #check only if sub-options are absent
        LogAndConsole " ====================== Step $_StepCount MS UEFI HSTI tests ====================== "
        $_StepCount++
        CheckHSTI
    }

    LogAndConsole " ====================== Step $_StepCount OS Architecture ====================== "
    $_StepCount++
    CheckOSArchitecture

    LogAndConsole " ====================== Step $_StepCount Supported OS SKU ====================== "
    $_StepCount++
    CheckOSSKU

    LogAndConsole " ====================== Step $_StepCount Virtualization Firmware ====================== "
    $_StepCount++
    CheckVirtualization

    if(!$HVCI -and !$DG)
    {
        LogAndConsole " ====================== Step $_StepCount TPM version ====================== "
        $_StepCount++
        CheckTPM

        LogAndConsole " ====================== Step $_StepCount Secure MOR ====================== "
        $_StepCount++
        CheckSecureMOR
    }

    LogAndConsole " ====================== Step $_StepCount NX Protector ====================== "
    $_StepCount++
    CheckNXProtection
```

```
    LogAndConsole " ====================== Step $_StepCount SMM Mitigation ====================== "
    $_StepCount++
    CheckSMMProtection

    LogAndConsole " ====================== End Check ====================== "

    LogAndConsole " ====================== Summary ====================== "
    ListSummary
    LogAndConsole "To learn more about required hardware and software please visit: https://aka.ms/dgwhcr"
}


# SIG # Begin signature block
## REPLACE
# SIG # End signature block
```

Windows Defender Credential Guard protection limits

3/5/2021 • 2 minutes to read • Edit Online

Applies to

- Windows 10
- Windows Server 2016

Some ways to store credentials are not protected by Windows Defender Credential Guard, including:

- Software that manages credentials outside of Windows feature protection
- Local accounts and Microsoft Accounts
- Windows Defender Credential Guard does not protect the Active Directory database running on Windows Server 2016 domain controllers. It also does not protect credential input pipelines, such as Windows Server 2016 servers running Remote Desktop Gateway. If you're using a Windows Server 2016 server as a client PC, it will get the same protection as it would when running Windows 10 Enterprise.
- Key loggers
- Physical attacks
- Does not prevent an attacker with malware on the PC from using the privileges associated with any credential. We recommend using dedicated PCs for high value accounts, such as IT Pros and users with access to high value assets in your organization.
- Third-party security packages
- Digest and CredSSP credentials
  - When Windows Defender Credential Guard is enabled, neither Digest nor CredSSP have access to users' logon credentials. This implies no Single Sign-On use for these protocols.
- Supplied credentials for NTLM authentication are not protected. If a user is prompted for and enters credentials for NTLM authentication, these credentials are vulnerable to be read from LSASS memory. Note that these same credentials are vulnerable to key loggers as well.-
- Kerberos service tickets are not protected by Credential Guard, but the Kerberos Ticket Granting Ticket (TGT) is.
- When Windows Defender Credential Guard is deployed on a VM, Windows Defender Credential Guard protects secrets from attacks inside the VM. However, it does not provide additional protection from privileged system attacks originating from the host.
- Windows logon cached password verifiers (commonly called "cached credentials") do not qualify as credentials because they cannot be presented to another computer for authentication, and can only be used locally to verify credentials. They are stored in the registry on the local computer and provide validation for credentials when a domain-joined computer cannot connect to AD DS during user logon. These "cached logons", or more specifically, cached domain account information, can be managed using the security policy setting **Interactive logon: Number of previous logons to cache** if a domain controller is not available.

See also

Deep Dive into Windows Defender Credential Guard: Related videos

Microsoft Cybersecurity Stack: Advanced Identity and Endpoint Protection: Manage Credential Guard

NOTE

- Note: Requires LinkedIn Learning subscription to view the full video

Considerations when using Windows Defender Credential Guard

3/26/2021 • 6 minutes to read • Edit Online

Applies to

- Windows 10
- Windows Server 2016

Passwords are still weak. We recommend that in addition to deploying Windows Defender Credential Guard, organizations move away from passwords to other authentication methods, such as physical smart cards, virtual smart cards, or Windows Hello for Business.

Windows Defender Credential Guard uses hardware security, so some features such as Windows To Go, are not supported.

Wi-fi and VPN Considerations

When you enable Windows Defender Credential Guard, you can no longer use NTLM classic authentication for Single Sign-On. You will be forced to enter your credentials to use these protocols and cannot save the credentials for future use. If you are using WiFi and VPN endpoints that are based on MS-CHAPv2, they are subject to similar attacks as for NTLMv1. For WiFi and VPN connections, Microsoft recommends that organizations move from MSCHAPv2-based connections such as PEAP-MSCHAPv2 and EAP-MSCHAPv2, to certificate-based authentication such as PEAP-TLS or EAP-TLS.

Kerberos Considerations

When you enable Windows Defender Credential Guard, you can no longer use Kerberos unconstrained delegation or DES encryption. Unconstrained delegation could allow attackers to extract Kerberos keys from the isolated LSA process. Use constrained or resource-based Kerberos delegation instead.

3rd Party Security Support Providers Considerations

Some 3rd party Security Support Providers (SSPs and APs) might not be compatible with Windows Defender Credential Guard because it does not allow third-party SSPs to ask for password hashes from LSA. However, SSPs and APs still get notified of the password when a user logs on and/or changes their password. Any use of undocumented APIs within custom SSPs and APs are not supported. We recommend that custom implementations of SSPs/APs are tested with Windows Defender Credential Guard. SSPs and APs that depend on any undocumented or unsupported behaviors fail. For example, using the KerbQuerySupplementalCredentialsMessage API is not supported. Replacing the NTLM or Kerberos SSPs with custom SSPs and APs. For more info, see Restrictions around Registering and Installing a Security Package on MSDN.

Upgrade Considerations

As the depth and breadth of protections provided by Windows Defender Credential Guard are increased, subsequent releases of Windows 10 with Windows Defender Credential Guard running may impact scenarios that were working in the past. For example, Windows Defender Credential Guard may block the use of a particular type of credential or a particular component to prevent malware from taking advantage of vulnerabilities. Test scenarios required for operations in an organization before upgrading a device using

Windows Defender Credential Guard.

Saved Windows Credentials Protected

Starting with Windows 10, version 1511, domain credentials that are stored with Credential Manager are protected with Windows Defender Credential Guard. Credential Manager allows you to store three types of credentials: Windows credentials, certificate-based credentials, and generic credentials. Generic credentials such as user names and passwords that you use to log on to websites are not protected since the applications require your cleartext password. If the application does not need a copy of the password, they can save domain credentials as Windows credentials that are protected. Windows credentials are used to connect to other computers on a network. The following considerations apply to the Windows Defender Credential Guard protections for Credential Manager:

- Windows credentials saved by Remote Desktop Client cannot be sent to a remote host. Attempts to use saved Windows credentials fail, displaying the error message "Logon attempt failed."
- Applications that extract Windows credentials fail.
- When credentials are backed up from a PC that has Windows Defender Credential Guard enabled, the Windows credentials cannot be restored. If you need to back up your credentials, you must do this before you enable Windows Defender Credential Guard. Otherwise, you cannot restore those credentials.

Clearing TPM Considerations

Virtualization-based Security (VBS) uses the TPM to protect its key. So when the TPM is cleared then the TPM protected key used to encrypt VBS secrets is lost.

> **WARNING**
> Clearing the TPM results in loss of protected data for all features that use VBS to protect data.
> When a TPM is cleared ALL features, which use VBS to protect data can no longer decrypt their protected data.

As a result Credential Guard can no longer decrypt protected data. VBS creates a new TPM protected key for Credential Guard. Credential Guard uses the new key to protect new data. However, the previously protected data is lost forever.

> **NOTE**
> Credential Guard obtains the key during initialization. So the data loss will only impact persistent data and occur after the next system startup.

Windows credentials saved to Credential Manager

Since Credential Manager cannot decrypt saved Windows Credentials, they are deleted. Applications should prompt for credentials that were previously saved. If saved again, then Windows credentials are protected Credential Guard.

Domain-joined device's automatically provisioned public key

Beginning with Windows 10 and Windows Server 2016, domain-devices automatically provision a bound public key, for more information about automatic public key provisioning, see Domain-joined Device Public Key Authentication.

Since Credential Guard cannot decrypt the protected private key, Windows uses the domain-joined computer's password for authentication to the domain. Unless additional policies are deployed, there should not be a loss of functionality. If a device is configured to only use public key, then it cannot authenticate with password until that policy is disabled. For more information on Configuring devices to only use public key, see Domain-joined Device Public Key Authentication.

Also if any access control checks including authentication policies require devices to have either the KEY TRUST IDENTITY (S-1-18-4) or FRESH PUBLIC KEY IDENTITY (S-1-18-3) well-known SIDs, then those access checks fail. For more information about authentication policies, see Authentication Policies and Authentication Policy Silos. For more information about well-known SIDs, see [MS-DTYP] Section 2.4.2.4 Well-known SID Structures.

Breaking DPAPI on domain-joined devices

On domain-joined devices, DPAPI can recover user keys using a domain controller from the user's domain. If a domain-joined device has no connectivity to a domain controller, then recovery is not possible.

> **IMPORTANT**
> Best practice when clearing a TPM on a domain-joined device is to be on a network with connectivity to domain controllers. This ensures DPAPI functions and the user does not experience strange behavior.
> Auto VPN configuration is protected with user DPAPI. User may not be able to use VPN to connect to domain controllers since the VPN configurations are lost.

If you must clear the TPM on a domain-joined device without connectivity to domain controllers, then you should consider the following.

Domain user sign-in on a domain-joined device after clearing a TPM for as long as there is no connectivity to a domain controller:

CREDENTIAL TYPE	WINDOWS 10 VERSION	BEHAVIOR
Certificate (smart card or Windows Hello for Business)	All	All data protected with user DPAPI is unusable and user DPAPI does not work at all.
Password	Windows 10 v1709 or later	If the user signed-in with a certificate or password prior to clearing the TPM, then they can sign-in with password and user DPAPI is unaffected.
Password	Windows 10 v1703	If the user signed-in with a password prior to clearing the TPM, then they can sign-in with that password and are unaffected.
Password	Windows 10 v1607 or earlier	Existing user DPAPI protected data is unusable. User DPAPI is able to protect new data.

Once the device has connectivity to the domain controllers, DPAPI recovers the user's key and data protected prior to clearing the TPM can be decrypted.

Impact of DPAPI failures on Windows Information Protection

When data protected with user DPAPI is unusable, then the user loses access to all work data protected by Windows Information Protection. The impact includes: Outlook 2016 is unable to start and work protected documents cannot be opened. If DPAPI is working, then newly created work data is protected and can be accessed.

Workaround: Users can resolve the problem by connecting their device to the domain and rebooting or using their Encrypting File System Data Recovery Agent certificate. For more information about Encrypting File System Data Recovery Agent certificate, see Create and verify an Encrypting File System (EFS) Data Recovery Agent (DRA) certificate.

See also

Related videos

What is virtualization-based security?

Additional mitigations

3/26/2021 • 18 minutes to read • Edit Online

Windows Defender Credential Guard can provide mitigation against attacks on derived credentials and prevent the use of stolen credentials elsewhere. However, PCs can still be vulnerable to certain attacks, even if the derived credentials are protected by Windows Defender Credential Guard. These attacks can include abusing privileges and use of derived credentials directly from a compromised device, re-using previously stolen credentials prior to Windows Defender Credential Guard, and abuse of management tools and weak application configurations. Because of this, additional mitigation also must be deployed to make the domain environment more robust.

Restricting domain users to specific domain-joined devices

Credential theft attacks allow the attacker to steal secrets from one device and use them from another device. If a user can sign on to multiple devices then any device could be used to steal credentials. How do you ensure that users only sign on using devices that have Windows Defender Credential Guard enabled? By deploying authentication policies that restrict them to specific domain-joined devices that have been configured with Windows Defender Credential Guard. For the domain controller to know what device a user is signing on from, Kerberos armoring must be used.

Kerberos armoring

Kerberos armoring is part of RFC 6113. When a device supports Kerberos armoring, its TGT is used to protect the user's proof of possession which can mitigate offline dictionary attacks. Kerberos armoring also provides the additional benefit of signed KDC errors this mitigates tampering which can result in things such as downgrade attacks.

To enable Kerberos armoring for restricting domain users to specific domain-joined devices

- Users need to be in domains that are running Windows Server 2012 R2 or higher
- All the domain controllers in these domains must be configured to support Kerberos armoring. Set the **KDC support for claims, compound authentication, and Kerberos armoring** Group Policy setting to either **Supported** or **Always provide claims**.
- All the devices with Windows Defender Credential Guard that the users will be restricted to must be configured to support Kerberos armoring. Enable the **Kerberos client support for claims, compound authentication and Kerberos armoring** Group Policy settings under **Computer Configuration** -> **Administrative Templates** -> **System** -> **Kerberos**.

Protecting domain-joined device secrets

Since domain-joined devices also use shared secrets for authentication, attackers can steal those secrets as well. By deploying device certificates with Windows Defender Credential Guard, the private key can be protected. Then authentication policies can require that users sign on devices that authenticate using those certificates. This prevents shared secrets stolen from the device to be used with stolen user credentials to sign on as the user.

Domain-joined device certificate authentication has the following requirements:

- Devices' accounts are in Windows Server 2012 domain functional level or higher.
- All domain controllers in those domains have KDC certificates which satisfy strict KDC validation certificate requirements:
  - KDC EKU present
  - DNS domain name matches the DNSName field of the SubjectAltName (SAN) extension

- Windows 10 devices have the CA issuing the domain controller certificates in the enterprise store.
- A process is established to ensure the identity and trustworthiness of the device in a similar manner as you would establish the identity and trustworthiness of a user before issuing them a smartcard.

Deploying domain-joined device certificates

To guarantee that certificates with the required issuance policy are only installed on the devices these users must use, they must be deployed manually on each device. The same security procedures used for issuing smart cards to users should be applied to device certificates.

For example, let's say you wanted to use the High Assurance policy only on these devices. Using a Windows Server Enterprise certificate authority, you would create a new template.

Creating a new certificate template

1. From the Certificate Manager console, right-click **Certificate Templates**, and then click **Manage.**
2. Right-click **Workstation Authentication**, and then click **Duplicate Template**.
3. Right-click the new template, and then click **Properties**.
4. On the **Extensions** tab, click **Application Policies**, and then click **Edit**.
5. Click **Client Authentication**, and then click **Remove**.
6. Add the ID-PKInit-KPClientAuth EKU. Click **Add**, click **New**, and then specify the following values:
   - Name: Kerberos Client Auth
   - Object Identifier: 1.3.6.1.5.2.3.4
7. On the **Extensions** tab, click **Issuance Policies**, and then click **Edit**.
8. Under **Issuance Policies**, click**High Assurance**.
9. On the **Subject name** tab, clear the **DNS name** check box, and then select the **User Principal Name (UPN)** check box.

Then on the devices that are running Windows Defender Credential Guard, enroll the devices using the certificate you just created.

Enrolling devices in a certificate

Run the following command:

```
CertReq -EnrollCredGuardCert MachineAuthentication
```

> **NOTE**
> You must restart the device after enrolling the machine authentication certificate.

How a certificate issuance policy can be used for access control

Beginning with the Windows Server 2008 R2 domain functional level, domain controllers support for authentication mechanism assurance provides a way to map certificate issuance policy OIDs to universal security groups. Windows Server 2012 domain controllers with claim support can map them to claims. To learn more about authentication mechanism assurance, see Authentication Mechanism Assurance for AD DS in Windows Server 2008 R2 Step-by-Step Guide on TechNet.

To see the issuance policies available

- The get-IssuancePolicy.ps1 shows all of the issuance policies that are available on the certificate authority. From a Windows PowerShell command prompt, run the following command:

  ```
  .\get-IssuancePolicy.ps1 –LinkedToGroup:All
  ```

To link an issuance policy to a universal security group

- The set-IssuancePolicyToGroupLink.ps1 creates a Universal security group, creates an organizational unit, and links the issuance policy to that Universal security group. From a Windows PowerShell command prompt, run the following command:

```
.\set-IssuancePolicyToGroupLink.ps1 –IssuancePolicyName:"<name of issuance policy>" –groupOU:"<Name
of OU to create>" –groupName:”<name of Universal security group to create>"
```

Restricting user sign on

So we now have completed the following:

- Created a special certificate issuance policy to identify devices that meet the deployment criteria required for the user to be able to sign on
- Mapped that policy to a universal security group or claim
- Provided a way for domain controllers to get the device authorization data during user sign on using Kerberos armoring. Now what is left to do is to configure the access check on the domain controllers. This is done using authentication policies.

Authentication policies have the following requirements:

- User accounts are in a Windows Server 2012 domain functional level or higher domain.

Creating an authentication policy restricting users to the specific universal security group

1. Open Active Directory Administrative Center.
2. Click **Authentication**, click **New**, and then click **Authentication Policy**.
3. In the **Display name** box, enter a name for this authentication policy.
4. Under the **Accounts** heading, click **Add**.
5. In the **Select Users, Computers, or Service Accounts** dialog box, type the name of the user account you wish to restrict, and then click **OK**.
6. Under the **User Sign On** heading, click the **Edit** button.
7. Click **Add a condition**.
8. In the **Edit Access Control Conditions** box, ensure that it reads **User** > **Group** > **Member of each** > **Value**, and then click **Add items**.
9. In the **Select Users, Computers, or Service Accounts** dialog box, type the name of the universal security group that you created with the set-IssuancePolicyToGroupLink script, and then click **OK**.
10. Click **OK** to close the **Edit Access Control Conditions** box.
11. Click **OK** to create the authentication policy.
12. Close Active Directory Administrative Center.

> **NOTE**
> When the authentication policy enforces policy restrictions, users will not be able to sign on using devices that do not have a certificate with the appropriate issuance policy deployed. This applies to both local and remote sign on scenarios. Therefore, it is strongly recommended to first only audit policy restrictions to ensure you don't have unexpected failures.

Discovering authentication failures due to authentication policies

To make tracking authentication failures due to authentication policies easier, an operational log exists with just those events. To enable the logs on the domain controllers, in Event Viewer, navigate to **Applications and Services Logs\Microsoft\Windows\Authentication, right-click AuthenticationPolicyFailures-DomainController**, and then click **Enable Log**.

To learn more about authentication policy events, see Authentication Policies and Authentication Policy Silos.

Appendix: Scripts

Here is a list of scripts mentioned in this topic.

Get the available issuance policies on the certificate authority

Save this script file as get-IssuancePolicy.ps1.

```
########################################
##     Parameters to be defined       ##
##     by the user                    ##
########################################
Param (
$Identity,
$LinkedToGroup
)
########################################
##     Strings definitions            ##
########################################
Data getIP_strings {
# culture="en-US"
ConvertFrom-StringData -stringdata @'
help1 = This command can be used to retrieve all available Issuance Policies in a forest. The forest of the
currently logged on user is targeted.
help2 = Usage:
help3 = The following parameter is mandatory:
help4 = -LinkedToGroup:<yes|no|all>
help5 = "yes" will return only Issuance Policies that are linked to groups. Checks that the linked Issuance
Policies are linked to valid groups.
help6 = "no" will return only Issuance Policies that are not currently linked to any group.
help7 = "all" will return all Issuance Policies defined in the forest. Checks that the linked Issuance
policies are linked to valid groups.
help8 = The following parameter is optional:
help9 = -Identity:<Name, Distinguished Name or Display Name of the Issuance Policy that you want to
retrieve>. If you specify an identity, the option specified in the "-LinkedToGroup" parameter is ignored.
help10 = Output: This script returns the Issuance Policy objects meeting the criteria defined by the above
parameters.
help11 = Examples:
errorIPNotFound = Error: no Issuance Policy could be found with Identity "{0}"
ErrorNotSecurity = Error: Issuance Policy "{0}" is linked to group "{1}" which is not of type "Security".
ErrorNotUniversal = Error: Issuance Policy "{0}" is linked to group "{1}" whose scope is not "Universal".
ErrorHasMembers = Error: Issuance Policy "{0}" is linked to group "{1}" which has a non-empty membership.
The group has the following members:
LinkedIPs = The following Issuance Policies are linked to groups:
displayName = displayName : {0}
Name = Name : {0}
dn = distinguishedName : {0}
        InfoName = Linked Group Name: {0}
        InfoDN = Linked Group DN: {0}
NonLinkedIPs = The following Issuance Policies are NOT linked to groups:
'@
}
##Import-LocalizedData getIP_strings
import-module ActiveDirectory
########################################
##           Help                     ##
########################################
function Display-Help {
    ""
    $getIP_strings.help1
    ""
$getIP_strings.help2
""
$getIP_strings.help3
"     " + $getIP_strings.help4
```

```
"             " + $getIP_strings.help5
    "             " + $getIP_strings.help6
    "             " + $getIP_strings.help7
""
$getIP_strings.help8
    "     " + $getIP_strings.help9
    ""
    $getIP_strings.help10
""
""
$getIP_strings.help11
    "     " + '$' + "myIPs = .\get-IssuancePolicy.ps1 -LinkedToGroup:All"
    "     " + '$' + "myLinkedIPs = .\get-IssuancePolicy.ps1 -LinkedToGroup:yes"
    "     " + '$' + "myIP = .\get-IssuancePolicy.ps1 -Identity:""Medium Assurance"""
""
}
$root = get-adrootdse
$domain = get-addomain -current loggedonuser
$configNCDN = [String]$root.configurationNamingContext
if ( !($Identity) -and !($LinkedToGroup) ) {
display-Help
break
}
if ($Identity) {
    $OIDs = get-adobject -Filter {(objectclass -eq "msPKI-Enterprise-Oid") -and ((name -eq $Identity) -or
(displayname -eq $Identity) -or (distinguishedName -like $Identity)) } -searchBase $configNCDN -properties *
    if ($OIDs -eq $null) {
$errormsg = $getIP_strings.ErrorIPNotFound -f $Identity
write-host $errormsg -ForegroundColor Red
    }
    foreach ($OID in $OIDs) {
        if ($OID."msDS-OIDToGroupLink") {
# In case the Issuance Policy is linked to a group, it is good to check whether there is any problem with
the mapping.
            $groupDN = $OID."msDS-OIDToGroupLink"
            $group = get-adgroup -Identity $groupDN
    $groupName = $group.Name
# Analyze the group
            if ($group.groupCategory -ne "Security") {
$errormsg = $getIP_strings.ErrorNotSecurity -f $Identity, $groupName
                write-host $errormsg -ForegroundColor Red
            }
            if ($group.groupScope -ne "Universal") {
                $errormsg = $getIP_strings.ErrorNotUniversal -f $Identity, $groupName
write-host $errormsg -ForegroundColor Red
            }
            $members = Get-ADGroupMember -Identity $group
            if ($members) {
                $errormsg = $getIP_strings.ErrorHasMembers -f $Identity, $groupName
write-host $errormsg -ForegroundColor Red
                foreach ($member in $members) {
                    write-host "          "  $member -ForeGroundColor Red
                }
            }
        }
    }
    return $OIDs
    break
}
if (($LinkedToGroup -eq "yes") -or ($LinkedToGroup -eq "all")) {
    $LDAPFilter = "(&(objectClass=msPKI-Enterprise-Oid)(msDS-OIDToGroupLink=*)(flags=2))"
    $LinkedOIDs = get-adobject -searchBase $configNCDN -LDAPFilter $LDAPFilter -properties *
    write-host ""
    write-host "**************************************************"
    write-host $getIP_strings.LinkedIPs
    write-host "**************************************************"
    write-host ""
    if ($LinkedOIDs -ne $null){
      foreach ($OID in $LinkedOIDs) {
```

```
# Display basic information about the Issuance Policies
          ""
  $getIP_strings.displayName -f $OID.displayName
  $getIP_strings.Name -f $OID.Name
  $getIP_strings.dn -f $OID.distinguishedName
# Get the linked group.
          $groupDN = $OID."msDS-OIDToGroupLink"
          $group = get-adgroup -Identity $groupDN
          $getIP_strings.InfoName -f $group.Name
          $getIP_strings.InfoDN -f $groupDN
# Analyze the group
          $OIDName = $OID.displayName
    $groupName = $group.Name
          if ($group.groupCategory -ne "Security") {
          $errormsg = $getIP_strings.ErrorNotSecurity -f $OIDName, $groupName
          write-host $errormsg -ForegroundColor Red
          }
          if ($group.groupScope -ne "Universal") {
          $errormsg = $getIP_strings.ErrorNotUniversal -f $OIDName, $groupName
          write-host $errormsg -ForegroundColor Red
          }
          $members = Get-ADGroupMember -Identity $group
          if ($members) {
          $errormsg = $getIP_strings.ErrorHasMembers -f $OIDName, $groupName
          write-host $errormsg -ForegroundColor Red
              foreach ($member in $members) {
                  write-host "          "  $member -ForeGroundColor Red
              }
          }
          write-host ""
      }
    }else{
write-host "There are no issuance policies that are mapped to a group"
    }
    if ($LinkedToGroup -eq "yes") {
        return $LinkedOIDs
        break
    }
}
if (($LinkedToGroup -eq "no") -or ($LinkedToGroup -eq "all")) {
    $LDAPFilter = "(&(objectClass=msPKI-Enterprise-Oid)(!(msDS-OIDToGroupLink=*))(flags=2))"
    $NonLinkedOIDs = get-adobject -searchBase $configNCDN -LDAPFilter $LDAPFilter -properties *
    write-host ""
    write-host "************************************************************"
    write-host $getIP_strings.NonLinkedIPs
    write-host "************************************************************"
    write-host ""
    if ($NonLinkedOIDs -ne $null) {
      foreach ($OID in $NonLinkedOIDs) {
# Display basic information about the Issuance Policies
write-host ""
$getIP_strings.displayName -f $OID.displayName
$getIP_strings.Name -f $OID.Name
$getIP_strings.dn -f $OID.distinguishedName
write-host ""
      }
    }else{
write-host "There are no issuance policies which are not mapped to groups"
    }
    if ($LinkedToGroup -eq "no") {
        return $NonLinkedOIDs
        break
    }
}
```

> **NOTE**
>
> If you're having trouble running this script, try replacing the single quote after the ConvertFrom-StringData parameter.

Link an issuance policy to a group

Save the script file as set-IssuancePolicyToGroupLink.ps1.

```
########################################
##     Parameters to be defined      ##
##     by the user                   ##
########################################
Param (
$IssuancePolicyName,
$groupOU,
$groupName
)
########################################
##     Strings definitions           ##
########################################
Data ErrorMsg {
# culture="en-US"
ConvertFrom-StringData -stringdata @'
help1 = This command can be used to set the link between a certificate issuance policy and a universal
security group.
help2 = Usage:
help3 = The following parameters are required:
help4 = -IssuancePolicyName:<name or display name of the issuance policy that you want to link to a group>
help5 = -groupName:<name of the group you want to link the issuance policy to>. If no name is specified, any
existing link to a group is removed from the Issuance Policy.
help6 = The following parameter is optional:
help7 = -groupOU:<Name of the Organizational Unit dedicated to the groups which are linked to issuance
policies>. If this parameter is not specified, the group is looked for or created in the Users container.
help8 = Examples:
help9 = This command will link the issuance policy whose display name is "High Assurance" to the group
"HighAssuranceGroup" in the Organizational Unit "OU_FOR_IPol_linked_groups". If the group or the
Organizational Unit do not exist, you will be prompted to create them.
help10 = This command will unlink the issuance policy whose name is "402.164959C40F4A5C12C6302E31D5476062"
from any group.
MultipleIPs = Error: Multiple Issuance Policies with name or display name "{0}" were found in the subtree of
"{1}"
NoIP = Error: no issuance policy with name or display name "{0}" could be found in the subtree of "{1}".
IPFound = An Issuance Policy with name or display name "{0}" was successfully found: {1}
MultipleOUs = Error: more than 1 Organizational Unit with name "{0}" could be found in the subtree of "{1}".
confirmOUcreation = Warning: The Organizational Unit that you specified does not exist. Do you want to
create it?
OUCreationSuccess = Organizational Unit "{0}" successfully created.
OUcreationError = Error: Organizational Unit "{0}" could not be created.
OUFoundSuccess = Organizational Unit "{0}" was successfully found.
multipleGroups = Error: More than one group with name "{0}" was found in Organizational Unit "{1}".
confirmGroupCreation = Warning: The group that you specified does not exist. Do you want to create it?
groupCreationSuccess = Univeral Security group "{0}" successfully created.
groupCreationError = Error: Univeral Security group "{0}" could not be created.
GroupFound = Group "{0}" was successfully found.
confirmLinkDeletion = Warning: The Issuance Policy "{0}" is currently linked to group "{1}". Do you really
want to remove the link?
UnlinkSuccess = Certificate issuance policy successfully unlinked from any group.
UnlinkError = Removing the link failed.
UnlinkExit = Exiting without removing the link from the issuance policy to the group.
IPNotLinked = The Certificate issuance policy is not currently linked to any group. If you want to link it
to a group, you should specify the -groupName option when starting this script.
ErrorNotSecurity = Error: You cannot link issuance Policy "{0}" to group "{1}" because this group is not of
type "Security".
ErrorNotUniversal = Error: You cannot link issuance Policy "{0}" to group "{1}" because the scope of this
group is not "Universal".
ErrorHasMembers = Error: You cannot link issuance Policy "{0}" to group "{1}" because it has a non-empty
```

```
membership. The group has the following members:
ConfirmLinkReplacement = Warning: The Issuance Policy "{0}" is currently linked to group "{1}". Do you
really want to update the link to point to group "{2}"?
LinkSuccess = The certificate issuance policy was successfully linked to the specified group.
LinkError = The certificate issuance policy could not be linked to the specified group.
ExitNoLinkReplacement = Exiting without setting the new link.
'@
}
# import-localizeddata ErrorMsg
function Display-Help {
""
write-host $ErrorMsg.help1
""
write-host $ErrorMsg.help2
""
write-host $ErrorMsg.help3
write-host "`t" $ErrorMsg.help4
write-host "`t" $ErrorMsg.help5
""
write-host $ErrorMsg.help6
write-host "`t" $ErrorMsg.help7
""
""
write-host $ErrorMsg.help8
""
write-host $ErrorMsg.help9
".\Set-IssuancePolicyToGroupMapping.ps1 -IssuancePolicyName ""High Assurance"" -groupOU
""OU_FOR_IPol_linked_groups"" -groupName ""HighAssuranceGroup"" "
""
write-host $ErrorMsg.help10
'.\Set-IssuancePolicyToGroupMapping.ps1 -IssuancePolicyName "402.164959C40F4A5C12C6302E31D5476062" -
groupName $null '
""
}
# Assumption:  The group to which the Issuance Policy is going
#              to be linked is (or is going to be created) in
#              the domain the user running this script is a member of.
import-module ActiveDirectory
$root = get-adrootdse
$domain = get-addomain -current loggedonuser
if ( !($IssuancePolicyName) ) {
display-Help
break
}
########################################
##     Find the OID object           ##
##     (aka Issuance Policy)         ##
########################################
$searchBase = [String]$root.configurationnamingcontext
$OID = get-adobject -searchBase $searchBase -Filter { ((displayname -eq $IssuancePolicyName) -or (name -eq
$IssuancePolicyName)) -and (objectClass -eq "msPKI-Enterprise-Oid")} -properties *
if ($OID -eq $null) {
$tmp = $ErrorMsg.NoIP -f $IssuancePolicyName, $searchBase
write-host $tmp -ForeGroundColor Red
break;
}
elseif ($OID.GetType().IsArray) {
$tmp = $ErrorMsg.MultipleIPs -f $IssuancePolicyName, $searchBase
write-host $tmp -ForeGroundColor Red
break;
}
else {
$tmp = $ErrorMsg.IPFound -f $IssuancePolicyName, $OID.distinguishedName
write-host $tmp -ForeGroundColor Green
}
########################################
##  Find the container of the group  ##
########################################
if ($groupOU -eq $null) {
```

```
# default to the Users container
$groupContainer = $domain.UsersContainer
}
else {
$searchBase = [string]$domain.DistinguishedName
$groupContainer = get-adobject -searchBase $searchBase -Filter { (Name -eq $groupOU) -and (objectClass -eq
"organizationalUnit")}
if ($groupContainer.count -gt 1) {
$tmp = $ErrorMsg.MultipleOUs -f $groupOU, $searchBase
write-host $tmp -ForegroundColor Red
break;
}
elseif ($groupContainer -eq $null) {
$tmp = $ErrorMsg.confirmOUcreation
write-host $tmp " ( (y)es / (n)o )" -ForegroundColor Yellow -nonewline
$userChoice = read-host
if ( ($userChoice -eq "y") -or ($userChoice -eq "yes") ) {
new-adobject -Name $groupOU -displayName $groupOU -Type "organizationalUnit" -
ProtectedFromAccidentalDeletion $true -path $domain.distinguishedName
if ($?){
$tmp = $ErrorMsg.OUCreationSuccess -f $groupOU
write-host $tmp -ForegroundColor Green
}
else{
$tmp = $ErrorMsg.OUCreationError -f $groupOU
write-host $tmp -ForeGroundColor Red
break;
}
$groupContainer = get-adobject -searchBase $searchBase -Filter { (Name -eq $groupOU) -and (objectClass -eq
"organizationalUnit")}
}
else {
break;
}
}
else {
$tmp = $ErrorMsg.OUFoundSuccess -f $groupContainer.name
write-host $tmp -ForegroundColor Green
}
}
########################################
##  Find the group               ##
########################################
if (($groupName -ne $null) -and ($groupName -ne "")){
##$searchBase = [String]$groupContainer.DistinguishedName
$searchBase = $groupContainer
$group = get-adgroup -Filter { (Name -eq $groupName) -and (objectClass -eq "group") } -searchBase
$searchBase
if ($group -ne $null -and $group.gettype().isarray) {
$tmp = $ErrorMsg.multipleGroups -f $groupName, $searchBase
write-host $tmp -ForeGroundColor Red
break;
}
elseif ($group -eq $null) {
$tmp = $ErrorMsg.confirmGroupCreation
write-host $tmp " ( (y)es / (n)o )" -ForegroundColor Yellow -nonewline
$userChoice = read-host
if ( ($userChoice -eq "y") -or ($userChoice -eq "yes") ) {
new-adgroup -samAccountName $groupName -path $groupContainer.distinguishedName -GroupScope "Universal" -
GroupCategory "Security"
if ($?){
$tmp = $ErrorMsg.GroupCreationSuccess -f $groupName
write-host $tmp -ForegroundColor Green
}else{
$tmp = $ErrorMsg.groupCreationError -f $groupName
write-host $tmp -ForeGroundColor Red
break
}
$group = get-adgroup -Filter { (Name -eq $groupName) -and (objectClass -eq "group") } -searchBase
```

```
$searchBase
}
else {
break;
}
}
else {
$tmp = $ErrorMsg.GroupFound -f $group.Name
write-host $tmp -ForegroundColor Green
}
}
else {
#####
## If the group is not specified, we should remove the link if any exists
#####
if ($OID."msDS-OIDToGroupLink" -ne $null) {
$tmp = $ErrorMsg.confirmLinkDeletion -f $IssuancePolicyName, $OID."msDS-OIDToGroupLink"
write-host $tmp " ( (y)es / (n)o )" -ForegroundColor Yellow -nonewline
$userChoice = read-host
if ( ($userChoice -eq "y") -or ($userChoice -eq "yes") ) {
set-adobject -Identity $OID -Clear "msDS-OIDToGroupLink"
if ($?) {
$tmp = $ErrorMsg.UnlinkSuccess
write-host $tmp -ForeGroundColor Green
}else{
$tmp = $ErrorMsg.UnlinkError
write-host $tmp -ForeGroundColor Red
}
}
else {
$tmp = $ErrorMsg.UnlinkExit
write-host $tmp
break
}
}
else {
$tmp = $ErrorMsg.IPNotLinked
write-host $tmp -ForeGroundColor Yellow
}
break;
}
########################################
##  Verify that the group is          ##
##  Universal, Security, and          ##
##  has no members                    ##
########################################
if ($group.GroupScope -ne "Universal") {
$tmp = $ErrorMsg.ErrorNotUniversal -f $IssuancePolicyName, $groupName
write-host $tmp -ForeGroundColor Red
break;
}
if ($group.GroupCategory -ne "Security") {
$tmp = $ErrorMsg.ErrorNotSecurity -f $IssuancePolicyName, $groupName
write-host $tmp -ForeGroundColor Red
break;
}
$members = Get-ADGroupMember -Identity $group
if ($members -ne $null) {
$tmp = $ErrorMsg.ErrorHasMembers -f $IssuancePolicyName, $groupName
write-host $tmp -ForeGroundColor Red
foreach ($member in $members) {write-host "   $member.name" -ForeGroundColor Red}
break;
}
########################################
##  We have verified everything. We   ##
##  can create the link from the      ##
##  Issuance Policy to the group.     ##
########################################
if ($OID."msDS-OIDToGroupLink" -ne $null) {
```

```
$tmp = $ErrorMsg.ConfirmLinkReplacement -f $IssuancePolicyName, $OID."msDS-OIDToGroupLink",
$group.distinguishedName
write-host $tmp  "( (y)es / (n)o )" -ForegroundColor Yellow -nonewline
$userChoice = read-host
if ( ($userChoice -eq "y") -or ($userChoice -eq "yes") ) {
$tmp = @{'msDS-OIDToGroupLink'= $group.DistinguishedName}
set-adobject -Identity $OID -Replace $tmp
if ($?) {
$tmp = $Errormsg.LinkSuccess
write-host $tmp -Foreground Green
}else{
$tmp = $ErrorMsg.LinkError
write-host $tmp -Foreground Red
}
} else {
$tmp = $Errormsg.ExitNoLinkReplacement
write-host $tmp
break
}
}
else {
$tmp = @{'msDS-OIDToGroupLink'= $group.DistinguishedName}
set-adobject -Identity $OID -Add $tmp
if ($?) {
$tmp = $Errormsg.LinkSuccess
write-host $tmp -Foreground Green
}else{
$tmp = $ErrorMsg.LinkError
write-host $tmp -Foreground Red
}
}
```

NOTE

If you're having trouble running this script, try replacing the single quote after the ConvertFrom-StringData parameter.

Windows Defender Credential Guard: Known issues

3/26/2021 • 3 minutes to read • Edit Online

Applies to

- Windows 10
- Windows Server 2016

Windows Defender Credential Guard has certain application requirements. Windows Defender Credential Guard blocks specific authentication capabilities. Therefore applications that require such capabilities will not function when it is enabled. For further information, see Application requirements.

The following known issue has been fixed in the Cumulative Security Update for November 2017:

- Scheduled tasks with stored credentials fail to run when Credential Guard is enabled. The task fails and reports Event ID 104 with the following message:
  "Task Scheduler failed to log on '\Test' .
  Failure occurred in 'LogonUserExEx' .
  User Action: Ensure the credentials for the task are correctly specified.
  Additional Data: Error Value: 2147943726. 2147943726 : ERROR_LOGON_FAILURE (The user name or password is incorrect)."

The following known issues have been fixed by servicing releases made available in the Cumulative Security Updates for April 2017:

- KB4015217 Windows Defender Credential Guard generates double bad password count on Active Directory domain-joined Windows 10 machines

  This issue can potentially lead to unexpected account lockouts. See also Microsoft® Knowledge Base articles KB4015219 and KB4015221

- KB4033236 Two incorrect logon attempts sent to Active Directory after Windows Defender Credential Guard installed on Windows 10

  This issue can potentially lead to unexpected account lockouts. The issue was fixed in servicing updates for each of the following operating systems:

  - Windows 10 Version 1607 and Windows Server 2016: KB4015217 (OS Build 14393.1066 and 14393.1083)
  - Windows 10 Version 1511: KB4015219 (OS Build 10586.873)
  - Windows 10 Version 1507: KB4015221 (OS Build 10240.17354)

Known issues involving third-party applications

The following issue affects the Java GSS API. See the following Oracle bug database article:

- JDK-8161921: Windows 10 Windows Defender Credential Guard does not allow sharing of TGT with Java

When Windows Defender Credential Guard is enabled on Windows 10, the Java GSS API will not authenticate. This is expected behavior because Windows Defender Credential Guard blocks specific application authentication capabilities and will not provide the TGT session key to applications regardless of registry key settings. For further information see Application requirements.

The following issue affects Cisco AnyConnect Secure Mobility Client:

- Blue screen on Windows 10 computers running Hypervisor-Protected Code Integrity and Windows Defender Credential Guard with Cisco Anyconnect 4.3.04027 *

*Registration required to access this article.

The following issue affects McAfee Application and Change Control (MACC):

- KB88869 Windows 10 machines exhibit high CPU usage with McAfee Application and Change Control (MACC) installed when Windows Defender Credential Guard is enabled [1]

The following issue affects AppSense Environment Manager. For further information, see the following Knowledge Base article:

- Installing AppSense Environment Manager on Windows 10 machines causes LSAISO.exe to exhibit high CPU usage when Windows Defender Credential Guard is enabled [1] **

The following issue affects Citrix applications:

- Windows 10 machines exhibit high CPU usage with Citrix applications installed when Windows Defender Credential Guard is enabled. [1]

[1] Products that connect to Virtualization Based Security (VBS) protected processes can cause Windows Defender Credential Guard-enabled Windows 10 or Windows Server 2016 machines to exhibit high CPU usage. For technical and troubleshooting information, see the following Microsoft Knowledge Base article:

- KB4032786 High CPU usage in the LSAISO process on Windows 10 or Windows Server 2016

For further technical information on LSAISO.exe, see the MSDN article: Isolated User Mode (IUM) Processes

** Registration is required to access this article.

Vendor support

See the following article on Citrix support for Secure Boot:

- Citrix Support for Secure Boot

Windows Defender Credential Guard is not supported by either these products, products versions, computer systems, or Windows 10 versions:

- For Windows Defender Credential Guard on Windows 10 with McAfee Encryption products, see: Support for Hypervisor-Protected Code Integrity and Windows Defender Credential Guard on Windows 10 with McAfee encryption products
- For Windows Defender Credential Guard on Windows 10 with Check Point Endpoint Security Client, see: Check Point Endpoint Security Client support for Microsoft Windows 10 Windows Defender Credential Guard and Hypervisor-Protected Code Integrity features
- For Windows Defender Credential Guard on Windows 10 with VMWare Workstation Windows 10 host fails when running VMWare Workstation when Windows Defender Credential Guard is enabled
- For Windows Defender Credential Guard on Windows 10 with specific versions of the Lenovo ThinkPad ThinkPad support for Hypervisor-Protected Code Integrity and Windows Defender Credential Guard in Microsoft Windows 10 – ThinkPad
- For Windows Defender Credential Guard on Windows 10 with Symantec Endpoint Protection Windows 10 with Windows Defender Credential Guard and Symantec Endpoint Protection 12.1

  This is not a comprehensive list. Check whether your product vendor, product version, or computer system, supports Windows Defender Credential Guard on systems that run Windows 10 or specific

versions of Windows 10. Specific computer system models may be incompatible with Windows Defender Credential Guard.

Microsoft encourages third-party vendors to contribute to this page by providing relevant product support information and by adding links to their own product support statements.

Protect Remote Desktop credentials with Windows Defender Remote Credential Guard

3/26/2021 • 8 minutes to read • Edit Online

Applies to

- Windows 10
- Windows Server 2016

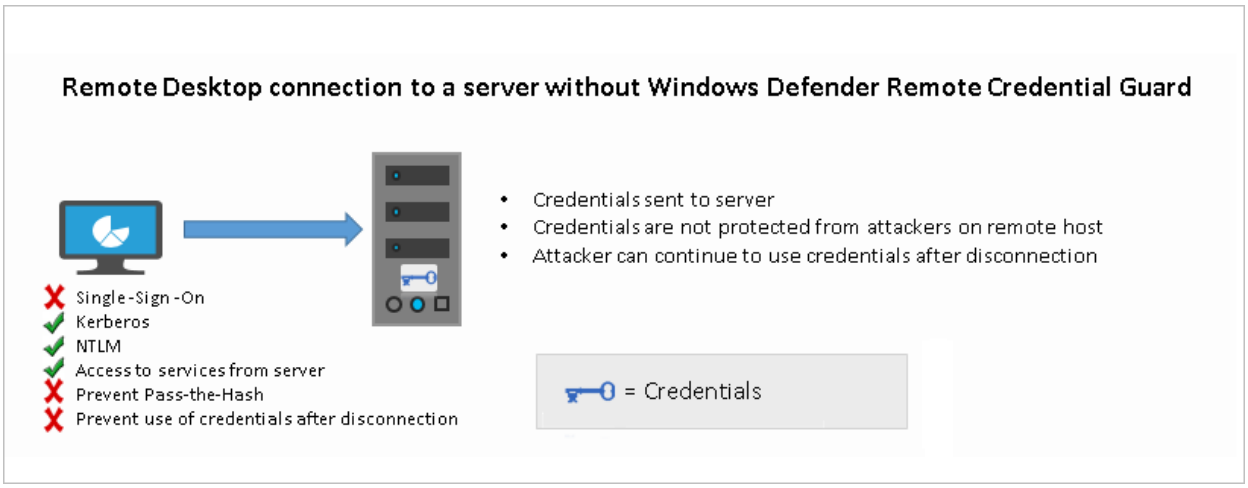
Introduced in Windows 10, version 1607, Windows Defender Remote Credential Guard helps you protect your credentials over a Remote Desktop connection by redirecting Kerberos requests back to the device that's requesting the connection. It also provides single sign-on experiences for Remote Desktop sessions.

Administrator credentials are highly privileged and must be protected. By using Windows Defender Remote Credential Guard to connect during Remote Desktop sessions, if the target device is compromised, your credentials are not exposed because both credential and credential derivatives are never passed over the network to the target device.

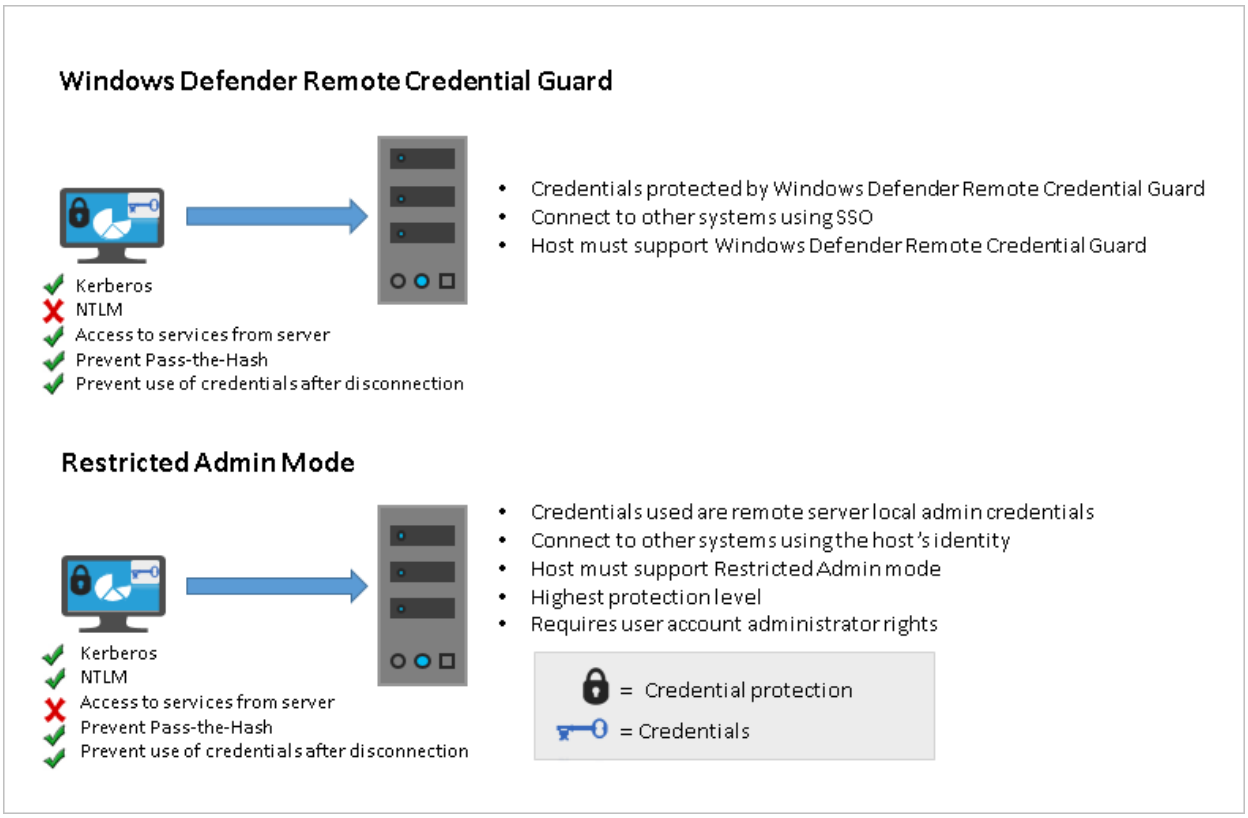
> **IMPORTANT**
> For information on Remote Desktop connection scenarios involving helpdesk support, see Remote Desktop connections and helpdesk support scenarios in this article.

Comparing Windows Defender Remote Credential Guard with other Remote Desktop connection options

The following diagram helps you to understand how a standard Remote Desktop session to a server without Windows Defender Remote Credential Guard works:



The following diagram helps you to understand how Windows Defender Remote Credential Guard works, what it helps to protect against, and compares it with the Restricted Admin mode option:



As illustrated, Windows Defender Remote Credential Guard blocks NTLM (allowing only Kerberos), prevents Pass-the-Hash (PtH) attacks, and also prevents use of credentials after disconnection.

Use the following table to compare different Remote Desktop connection security options:

FEATURE	REMOTE DESKTOP	WINDOWS DEFENDER REMOTE CREDENTIAL GUARD	RESTRICTED ADMIN MODE
Protection benefits	Credentials on the server are not protected from Pass-the-Hash attacks.	User credentials remain on the client. An attacker can act on behalf of the user *only* when the session is ongoing	User logs on to the server as local administrator, so an attacker cannot act on behalf of the "domain user". Any attack is local to the server
Version support	The remote computer can run any Windows operating system	Both the client and the remote computer must be running **at least Windows 10, version 1607, or Windows Server 2016**.	The remote computer must be running **at least patched Windows 7 or patched Windows Server 2008 R2**. For more information about patches (software updates) related to Restricted Admin mode, see Microsoft Security Advisory 2871997.
Helps prevent	N/A	• Pass-the-Hash • Use of a credential after disconnection	• Pass-the-Hash • Use of domain identity during connection

FEATURE	REMOTE DESKTOP	WINDOWS DEFENDER REMOTE CREDENTIAL GUARD	RESTRICTED ADMIN MODE
Credentials supported from the remote desktop client device	• **Signed on** credentials • **Supplied** credentials • **Saved** credentials	• **Signed on** credentials only	• **Signed on** credentials • **Supplied** credentials • **Saved** credentials
Access	**Users allowed**, that is, members of Remote Desktop Users group of remote host.	**Users allowed**, that is, members of Remote Desktop Users of remote host.	**Administrators only**, that is, only members of Administrators group of remote host.
Network identity	Remote Desktop session **connects to other resources as signed-in user**.	Remote Desktop session **connects to other resources as signed-in user**.	Remote Desktop session **connects to other resources as remote host's identity**.
Multi-hop	From the remote desktop, **you can connect through Remote Desktop to another computer**	From the remote desktop, you **can connect through Remote Desktop to another computer**.	Not allowed for user as the session is running as a local host account
Supported authentication	Any negotiable protocol.	Kerberos only.	Any negotiable protocol

For further technical information, see Remote Desktop Protocol and How Kerberos works.

Remote Desktop connections and helpdesk support scenarios

For helpdesk support scenarios in which personnel require administrative access to provide remote assistance to computer users via Remote Desktop sessions, Microsoft recommends that Windows Defender Remote Credential Guard should not be used in that context. This is because if an RDP session is initiated to a compromised client that an attacker already controls, the attacker could use that open channel to create sessions on the user's behalf (without compromising credentials) to access any of the user's resources for a limited time (a few hours) after the session disconnects.

Therefore, we recommend instead that you use the Restricted Admin mode option. For helpdesk support scenarios, RDP connections should only be initiated using the /RestrictedAdmin switch. This helps ensure that credentials and other user resources are not exposed to compromised remote hosts. For more information, see Mitigating Pass-the-Hash and Other Credential Theft v2.

To further harden security, we also recommend that you implement Local Administrator Password Solution (LAPS), a Group Policy client-side extension (CSE) introduced in Windows 8.1 that automates local administrator password management. LAPS mitigates the risk of lateral escalation and other cyberattacks facilitated when customers use the same administrative local account and password combination on all their computers. You can download and install LAPS here.

For further information on LAPS, see Microsoft Security Advisory 3062591.

Remote Credential Guard requirements

To use Windows Defender Remote Credential Guard, the Remote Desktop client and remote host must meet the following requirements:

The Remote Desktop client device:

- Must be running at least Windows 10, version 1703 to be able to supply credentials, which is sent to the remote device. This allows users to run as different users without having to send credentials to the remote machine.
- Must be running at least Windows 10, version 1607 or Windows Server 2016 to use the user's signed-in credentials. This requires the user's account be able to sign in to both the client device and the remote host.
- Must be running the Remote Desktop Classic Windows application. The Remote Desktop Universal Windows Platform application doesn't support Windows Defender Remote Credential Guard.
- Must use Kerberos authentication to connect to the remote host. If the client cannot connect to a domain controller, then RDP attempts to fall back to NTLM. Windows Defender Remote Credential Guard does not allow NTLM fallback because this would expose credentials to risk.

The Remote Desktop remote host:

- Must be running at least Windows 10, version 1607 or Windows Server 2016.
- Must allow Restricted Admin connections.
- Must allow the client's domain user to access Remote Desktop connections.
- Must allow delegation of non-exportable credentials.

There are no hardware requirements for Windows Defender Remote Credential Guard.

> **NOTE**
>
> Remote Desktop client devices running earlier versions, at minimum Windows 10 version 1607, only support signed-in credentials, so the client device must also be joined to an Active Directory domain. Both Remote Desktop client and server must either be joined to the same domain, or the Remote Desktop server can be joined to a domain that has a trust relationship to the client device's domain.
>
> GPO Remote host allows delegation of non-exportable credentials should be enabled for delegation of non-exportable credentials.

- For Windows Defender Remote Credential Guard to be supported, the user must authenticate to the remote host using Kerberos authentication.
- The remote host must be running at least Windows 10 version 1607, or Windows Server 2016.
- The Remote Desktop classic Windows app is required. The Remote Desktop Universal Windows Platform app doesn't support Windows Defender Remote Credential Guard.

Enable Windows Defender Remote Credential Guard

You must enable Restricted Admin or Windows Defender Remote Credential Guard on the remote host by using the Registry.

1. Open Registry Editor on the remote host.
2. Enable Restricted Admin and Windows Defender Remote Credential Guard:

- Go to HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Lsa.
- Add a new DWORD value named **DisableRestrictedAdmin**.
- To turn on Restricted Admin and Windows Defender Remote Credential Guard, set the value of this registry setting to 0 to turn on Windows Defender Remote Credential Guard.

3. Close Registry Editor.

You can add this by running the following command from an elevated command prompt:

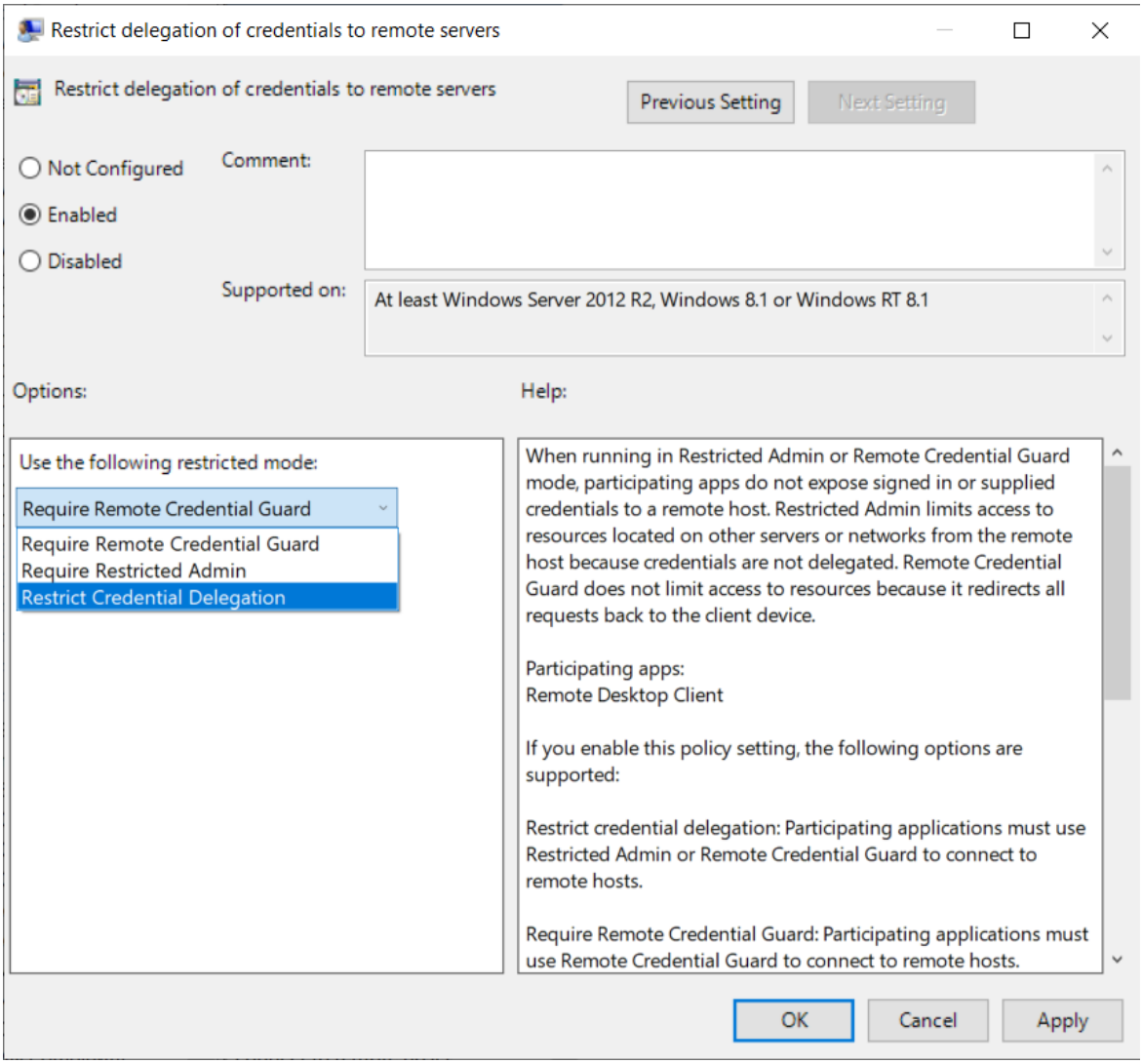
```
reg add HKLM\SYSTEM\CurrentControlSet\Control\Lsa /v DisableRestrictedAdmin /d 0 /t REG_DWORD
```

Using Windows Defender Remote Credential Guard

Beginning with Windows 10 version 1703, you can enable Windows Defender Remote Credential Guard on the client device either by using Group Policy or by using a parameter with the Remote Desktop Connection.

Turn on Windows Defender Remote Credential Guard by using Group Policy

1. From the Group Policy Management Console, go to **Computer Configuration** -> **Administrative Templates** -> **System** -> **Credentials Delegation**.
2. Double-click **Restrict delegation of credentials to remote servers**.



3. Under **Use the following restricted mode**:

- If you want to require either Restricted Admin mode or Windows Defender Remote Credential

Guard, choose **Restrict Credential Delegation**. In this configuration, Windows Defender Remote Credential Guard is preferred, but it will use Restricted Admin mode (if supported) when Windows Defender Remote Credential Guard cannot be used.

> **NOTE**
> Neither Windows Defender Remote Credential Guard nor Restricted Admin mode will send credentials in clear text to the Remote Desktop server.

- If you want to require Windows Defender Remote Credential Guard, choose **Require Remote Credential Guard**. With this setting, a Remote Desktop connection will succeed only if the remote computer meets the requirements listed earlier in this topic.
- If you want to require Restricted Admin mode, choose **Require Restricted Admin**. For information about Restricted Admin mode, see the table in Comparing Windows Defender Remote Credential Guard with other Remote Desktop connection options, earlier in this topic.

4. Click **OK**.
5. Close the Group Policy Management Console.
6. From a command prompt, run **gpupdate.exe /force** to ensure that the Group Policy object is applied.

Use Windows Defender Remote Credential Guard with a parameter to Remote Desktop Connection

If you don't use Group Policy in your organization, or if not all your remote hosts support Remote Credential Guard, you can add the remoteGuard parameter when you start Remote Desktop Connection to turn on Windows Defender Remote Credential Guard for that connection.

```
mstsc.exe /remoteGuard
```

> **NOTE**
> The user must be authorized to connect to the remote server using Remote Desktop Protocol, for example by being a member of the Remote Desktop Users local group on the remote computer.

Considerations when using Windows Defender Remote Credential Guard

- Windows Defender Remote Credential Guard does not support compound authentication. For example, if you're trying to access a file server from a remote host that requires a device claim, access will be denied.
- Windows Defender Remote Credential Guard can be used only when connecting to a device that is joined to a Windows Server Active Directory domain, including AD domain-joined servers that run as Azure virtual machines (VMs). Windows Defender Remote Credential Guard cannot be used when connecting to remote devices joined to Azure Active Directory.
- Remote Desktop Credential Guard only works with the RDP protocol.
- No credentials are sent to the target device, but the target device still acquires Kerberos Service Tickets on its own.
- The server and client must authenticate using Kerberos.

Smart Card Technical Reference

3/5/2021 • 2 minutes to read • Edit Online

Applies To: Windows 10, Windows Server 2016

The Smart Card Technical Reference describes the Windows smart card infrastructure for physical smart cards and how smart card-related components work in Windows. This document also contains information about tools that information technology (IT) developers and administrators can use to troubleshoot, debug, and deploy smart card-based strong authentication in the enterprise.

Audience

This document explains how the Windows smart card infrastructure works. To understand this information, you should have basic knowledge of public key infrastructure (PKI) and smart card concepts. This document is intended for:

- Enterprise IT developers, managers, and staff who are planning to deploy or are using smart cards in their organization.
- Smart card vendors who write smart card minidrivers or credential providers.

What are smart cards?

Smart cards are tamper-resistant portable storage devices that can enhance the security of tasks such as authenticating clients, signing code, securing e-mail, and signing in with a Windows domain account.

Smart cards provide:

- Tamper-resistant storage for protecting private keys and other forms of personal information.
- Isolation of security-critical computations that involve authentication, digital signatures, and key exchange from other parts of the computer. These computations are performed on the smart card.
- Portability of credentials and other private information between computers at work, home, or on the road.

Smart cards can be used to sign in to domain accounts only, not local accounts. When you use a password to sign in interactively to a domain account, Windows uses the Kerberos version 5 (v5) protocol for authentication. If you use a smart card, the operating system uses Kerberos v5 authentication with X.509 v3 certificates.

Virtual smart cards were introduced in Windows Server 2012 and Windows 8 to alleviate the need for a physical smart card, the smart card reader, and the associated administration of that hardware. For information about virtual smart card technology, see Virtual Smart Card Overview.

In this technical reference

This reference contains the following topics.

How Smart Card Sign-in Works in Windows

3/5/2021 • 2 minutes to read • Edit Online

Applies To: Windows 10, Windows Server 2016

This topic for IT professional provides links to resources about the implementation of smart card technologies in the Windows operating system. It includes the following resources about the architecture, certificate management, and services that are related to smart card use:

- Smart Card Architecture: Learn about enabling communications with smart cards and smart card readers, which can be different according to the vendor that supplies them.
- Certificate Requirements and Enumeration: Learn about requirements for smart card certificates based on the operating system, and about the operations that are performed by the operating system when a smart card is inserted into the computer.
- Smart Card and Remote Desktop Services: Learn about using smart cards for remote desktop connections.
- Smart Cards for Windows Service: Learn about how the Smart Cards for Windows service is implemented.
- Certificate Propagation Service: Learn about how the certificate propagation service works when a smart card is inserted into a computer.
- Smart Card Removal Policy Service: Learn about using Group Policy to control what happens when a user removes a smart card.

Smart Card Architecture

3/26/2021 • 19 minutes to read • Edit Online

Applies To: Windows 10, Windows Server 2016

This topic for the IT professional describes the system architecture that supports smart cards in the Windows operating system, including credential provider architecture and the smart card subsystem architecture.

Authentication is a process for verifying the identity of an object or person. When you authenticate an object, such as a smart card, the goal is to verify that the object is genuine. When you authenticate a person, the goal is to verify that you are not dealing with an imposter.

In a networking context, authentication is the act of proving identity to a network application or resource. Typically, identity is proven by a cryptographic operation that uses a key only the user knows (such as with public key cryptography), or a shared key. The server side of the authentication exchange compares the signed data with a known cryptographic key to validate the authentication attempt. Storing the cryptographic keys in a secure central location makes the authentication process scalable and maintainable.

For smart cards, Windows supports a provider architecture that meets the secure authentication requirements and is extensible so that you can include custom credential providers. This topic includes information about:

- Credential provider architecture
- Smart card subsystem architecture

Credential provider architecture

The following table lists the components that are included in the interactive sign-in architecture of the Windows Server and Windows operating systems.

COMPONENT	DESCRIPTION
Winlogon	Provides an interactive sign-in infrastructure.
Logon UI	Provides interactive UI rendering.
Credential providers (password and smart card)	Describes credential information and serializing credentials.
Local Security Authority (LSA)	Processes sign-in credentials.
Authentication packages	Includes NTLM and the Kerberos protocol. Communicates with server authentication packages to authenticate users.

Interactive sign-in in Windows begins when the user presses CTRL+ALT+DEL. The CTRL+ALT+DEL key combination is called a secure attention sequence (SAS). To keep other programs and processes from using it, Winlogon registers this sequence during the boot process.

After receiving the SAS, the UI then generates the sign-in tile from the information received from the registered credential providers. The following graphic shows the architecture for credential providers in the Windows operating system.

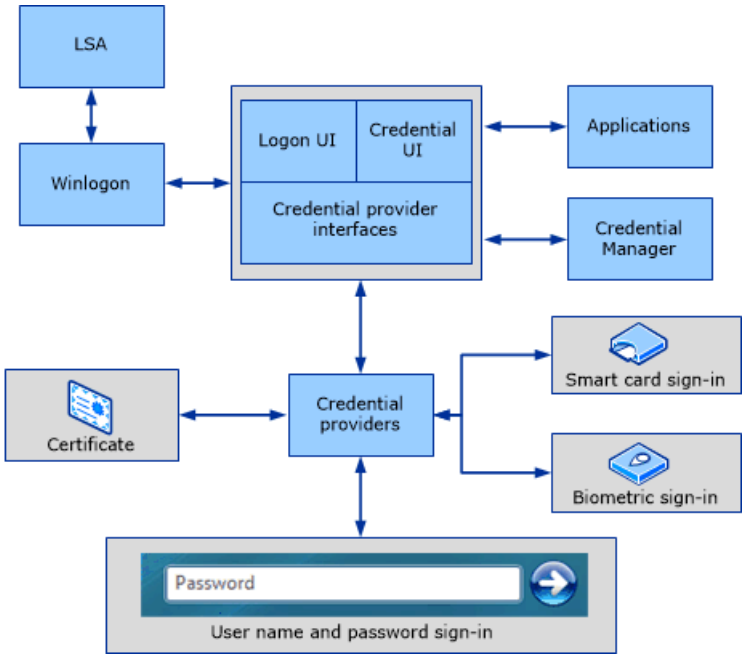


Figure 1 Credential provider architecture

Typically, a user who signs in to a computer by using a local account or a domain account must enter a user name and password. These credentials are used to verify the user's identity. For smart card sign-in, a user's credentials are contained on the smart card's security chip. A smart card reader lets the computer interact with the security chip on the smart card. When users sign in with a smart card, they enter a personal identification number (PIN) instead of a user name and password.

Credential providers are in-process COM objects that run on the local system and are used to collect credentials. The Logon UI provides interactive UI rendering, Winlogon provides interactive sign-in infrastructure, and credential providers work with both of these components to help gather and process credentials.

Winlogon instructs the Logon UI to display credential provider tiles after it receives an SAS event. The Logon UI queries each credential provider for the number of credentials it wants to enumerate. Credential providers have the option of specifying one of these tiles as the default. After all providers have enumerated their tiles, the Logon UI displays them to the user. The user interacts with a tile to supply the proper credentials. The Logon UI submits these credentials for authentication.

Combined with supporting hardware, credential providers can extend the Windows operating system to enable users to sign in by using biometrics (for example, fingerprint, retinal, or voice recognition), password, PIN, smart card certificate, or any custom authentication package. Enterprises and IT professionals can develop and deploy custom authentication mechanisms for all domain users, and they may explicitly require users to use this custom sign-in mechanism.

> **Note** Credential providers are not enforcement mechanisms. They are used to gather and serialize credentials. The LSA and authentication packages enforce security.

Credential providers can be designed to support single sign-in (SSO). In this process, they authenticate users to a secure network access point (by using RADIUS and other technologies) for signing in to the computer. Credential providers are also designed to support application-specific credential gathering, and they can be used for authentication to network resources, joining computers to a domain, or to provide administrator consent for User Account Control (UAC).

Multiple credential providers can coexist on a computer.

Credential providers must be registered on a computer running Windows, and they are responsible for:

- Describing the credential information that is required for authentication.

- Handling communication and logic with external authentication authorities.
- Packaging credentials for interactive and network sign-in.

> **Note** The Credential Provider API does not render the UI. It describes what needs to be rendered.
> Only the password credential provider is available in safe mode.
> The smart card credential provider is available in safe mode during networking.

Smart card subsystem architecture

Vendors provide smart cards and smart card readers, and in many cases the vendors are different for the smart card and the smart card reader. Drivers for smart card readers are written to the Personal Computer/Smart Card (PC/SC) standard. Each smart card must have a Cryptographic Service Provider (CSP) that uses the CryptoAPI interfaces to enable cryptographic operations, and the WinSCard APIs to enable communications with smart card hardware.

Base CSP and smart card minidriver architecture

Figure 2 illustrates the relationship between the CryptoAPI, CSPs, the Smart Card Base Cryptographic Service Provider (Base CSP), and smart card minidrivers.

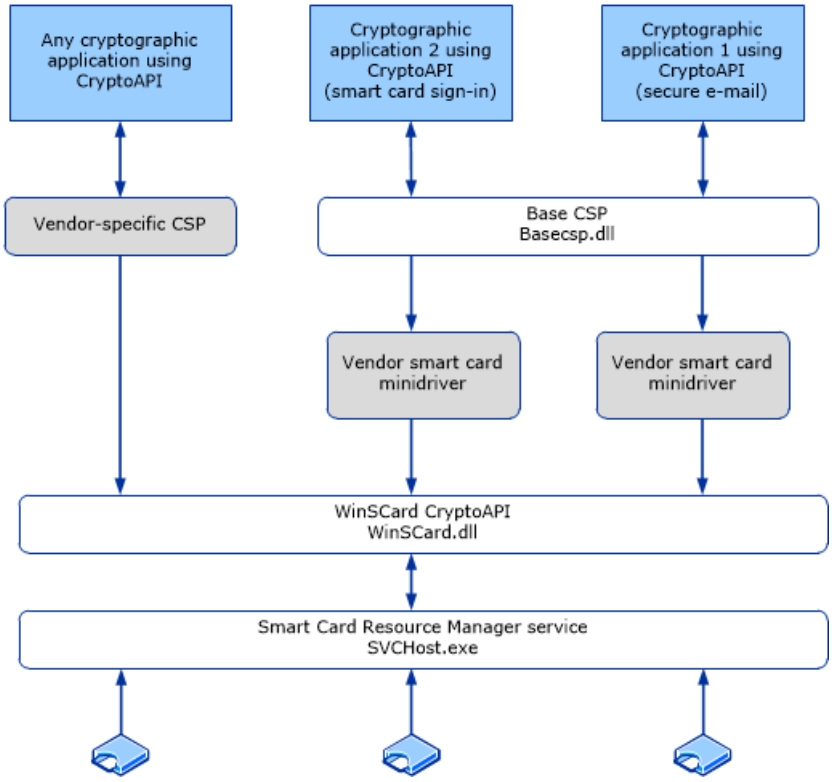


Figure 2 Base CSP and smart card minidriver architecture

Caching with Base CSP and smart card KSP

Smart card architecture uses caching mechanisms to assist in streamlining operations and to improve a user's access to a PIN.

- Data caching: The data cache provides for a single process to minimize smart card I/O operations.
- PIN caching: The PIN cache helps the user from having to reenter a PIN each time the smart card is unauthenticated.

Data caching

Each CSP implements the current smart card data cache separately. The Base CSP implements a robust caching mechanism that allows a single process to minimize smart card I/O operations.

The existing global cache works as follows:

1. The application requests a cryptographic operation. For example, a user certificate is to be read from the smart card.
2. The CSP checks its cache for the item.
3. If the item is not found in the cache, or if the item is cached but is not up-to-date, the item is read from the smart card.
4. After any item has been read from the smart card, it is added to the cache. Any existing out-of-date copy of that item is replaced.

Three types of objects or data are cached by the CSP: pins (for more information, see PIN caching), certificates, and files. If any of the cached data changes, the corresponding object is read from the smart card in successive operations. For example, if a file is written to the smart card, the CSP cache becomes out-of-date for the files, and other processes read the smart card at least once to refresh their CSP cache.

The global data cache is hosted in the Smart Cards for Windows service. Windows includes two public smart card API calls, SCardWriteCache and SCardReadCache. These API calls make global data caching functionality available to applications. Every smart card that conforms to the smart card minidriver specification has a 16-byte card identifier. This value is used to uniquely identify cached data that pertains to a given smart card. The standard Windows GUID type is used. These APIs allow an application to add data to and read data from the global cache.

PIN caching

The PIN cache protects the user from entering a PIN every time the smart card is unauthenticated. After a smart card is authenticated, it will not differentiate among host-side applications—any application can access private data on the smart card.

To mitigate this, the smart card enters an exclusive state when an application authenticates to the smart card. However, this means that other applications cannot communicate with the smart card and will be blocked. Therefore, such exclusive connections are minimized. The issue is that a protocol (such as the Kerberos protocol) requires multiple signing operations. Therefore, the protocol requires exclusive access to the smart card over an extended period, or it require multiple authentication operations. This is where the PIN cache is used to minimize exclusive use of the smart card without forcing the user to enter a PIN multiple times.

The following example illustrates how this works. In this scenario, there are two applications: Outlook and Internet Explorer. The applications use smart cards for different purposes.

1. The user starts Outlook and tries to send a signed e-mail. The private key is on the smart card.
2. Outlook prompts the user for the smart card PIN. The user enters the correct PIN.
3. E-mail data is sent to the smart card for the signature operation. The Outlook client formats the response and sends the e-mail.
4. The user opens Internet Explorer and tries to access a protected site that requires Transport Layer Security (TLS) authentication for the client.
5. Internet Explorer prompts the user for the smart card PIN. The user enters the correct PIN.
6. The TLS-related private key operation occurs on the smart card, and the user is authenticated and signed in.
7. The user returns to Outlook to send another signed e-mail. This time, the user is not prompted for a PIN because the PIN is cached from the previous operation. Similarly, if the user uses Internet Explorer again for another operation, Internet Explorer will not prompt the user for a PIN.

The Base CSP internally maintains a per-process cache of the PIN. The PIN is encrypted and stored in memory. The functions that are used to secure the PIN are RtlEncryptMemory, RtlDecryptMemory, and RtlSecureZeroMemory, which will empty buffers that contained the PIN.

Smart card selection

The following sections in this topic describe how Windows leverages the smart card architecture to select the correct smart card reader software, provider, and credentials for a successful smart card sign-in:

- Container specification levels
- Container operations
- Context flags
- Create a new container in silent context
- Smart card selection behavior
- Make a smart card reader match
- Make a smart card match
- Open an existing default container (no reader specified)
- Open an existing GUID-named container (no reader specified)
- Create a new container (no reader specified)
- Delete a container

Container specification levels

In response to a CryptAcquireContext call in CryptoAPI, the Base CSP tries to match the container that the caller specifies to a specific smart card and reader. The caller can provide a container name with varying levels of specificity, as shown in the following table, and sorted from most-specific to least-specific requests.

Similarly, in response to a NCryptOpenKey call in CNG, the smart card KSP tries to match the container the same way, and it takes the same container format, as shown in the following table.

> **Note** Before opening a key by using the smart card KSP, a call to NCryptOpenStorageProvider (MS_SMART_CARD_KEY_STORAGE_PROVIDER) must be made.

TYPE	NAME	FORMAT
I	Reader Name and Container Name	\\.\<Reader Name>\<Container Name>
II	Reader Name and Container Name (NULL)	\\.\<Reader Name>
III	Container Name Only	<Container Name>
IV	Default Container (NULL) Only	NULL

The Base CSP and smart card KSP cache smart card handle information about the calling process and about the smart cards the process has accessed. When searching for a smart card container, the Base CSP or smart card KSP first checks its cache for the process. If the cached handle is invalid or no match is found, the SCardUIDlg API is called to get the card handle.

Container operations

The following three container operations can be requested by using CryptAcquireContext:

1. Create a new container. (The CNG equivalent of CryptAcquireContext with dwFlags set to CRYPT_NEWKEYSET is NCryptCreatePersistedKey.)
2. Open an existing container. (The CNG equivalent of CryptAcquireContext to open the container is NCryptOpenKey.)
3. Delete a container. (The CNG equivalent of CryptAcquireContext with dwFlags set to CRYPT_DELETEKEYSET is NCryptDeleteKey.)

The heuristics that are used to associate a cryptographic handle with a particular smart card and reader are based on the container operation requested and the level of container specification used.

The following table shows the restrictions for the container creation operation.

SPECIFICATION	RESTRICTION
No silent context	Key container creation must always be able to show UI, such as the PIN prompt.
No overwriting existing containers	If the specified container already exists on the chosen smart card, choose another smart card or cancel the operation.

Context flags

The following table shows the context flags used as restrictions for the container creation operation.

FLAG	DESCRIPTION
CRYPT_SILENT	No UI can be displayed during this operation.
CRYPT_MACHINE_KEYSET	No cached data should be used during this operation.
CRYPT_VERIFYCONTEXT	Only public data can be accessed on the smart card.

In addition to container operations and container specifications, you must consider other user options, such as the CryptAcquireContext flags, during smart card selection.

> **Important** The CRYPT_SILENT flag cannot be used to create a new container.

Create a new container in silent context

Applications can call the Base CSP with CRYPT_DEFAULT_CONTAINER_OPTIONAL, set the PIN in silent context, and then create a new container in silent context. This operation occurs as follows:

1. Call CryptAcquireContext by passing the smart card reader name in as a type II container specification level, and specifying the CRYPT_DEFAULT_CONTAINER_OPTIONAL flag.
2. Call CryptSetProvParam by specifying PP_KEYEXCHANGE_PIN or PP_SIGNATURE_PIN and a null-terminated ASCII PIN.
3. Release the context acquired in Step 1.
4. Call CryptAcquireContext with CRYPT_NEWKEYSET, and specify the type I container specification level.
5. Call CryptGenKey to create the key.

Smart card selection behavior

In some of the following scenarios, the user can be prompted to insert a smart card. If the user context is silent,

this operation fails and no UI is displayed. Otherwise, in response to the UI, the user can insert a smart card or click **Cancel**. If the user cancels the operation, the operation fails. The flow chart in Figure 3 shows the selection steps performed by the Windows operating system.

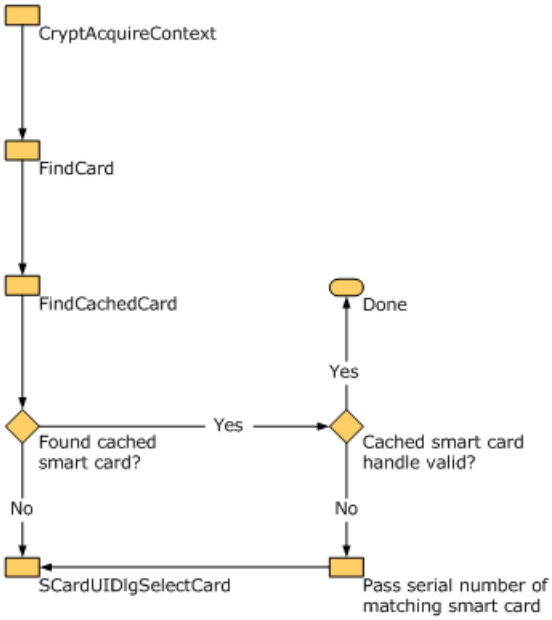


Figure 3 Smart card selection behavior

In general, smart card selection behavior is handled by the SCardUIDlgSelectCard API. The Base CSP interacts with this API by calling it directly. The Base CSP also sends callback functions that have the purpose of filtering and matching candidate smart cards. Callers of CryptAcquireContext provide smart card matching information. Internally, the Base CSP uses a combination of smart card serial numbers, reader names, and container names to find specific smart cards.

Each call to SCardUI * may result in additional information read from a candidate smart card. The Base CSP smart card selection callbacks cache this information.

Make a smart card reader match

For type I and type II container specification levels, the smart card selection process is less complex because only the smart card in the named reader can be considered a match. The process for matching a smart card with a smart card reader is:

1. Find the requested smart card reader. If it cannot be found, the process fails. (This requires a cache search by reader name.)
2. If no smart card is in the reader, the user is prompted to insert a smart card. (This is only in non-silent mode; if the call is made in silent mode, it will fail.)
3. For container specification level II only, the name of the default container on the chosen smart card is determined.
4. To open an existing container or delete an existing container, find the specified container. If the specified container cannot be found on this smart card, the user is prompted to insert a smart card.
5. If the system attempts to create a new container, if the specified container already exists on this smart card, the process fails.

Make a smart card match

For container specification levels III and IV, a broader method is used to match an appropriate smart card with a user context, because multiple cached smart cards might meet the criteria provided.

Open an existing default container (no reader specified)

> **Note** This operation requires that you use the smart card with the Base CSP.

1. For each smart card that has been accessed by the Base CSP and the handle and container information are cached, the Base CSP looks for a valid default container. An operation is attempted on the cached SCARDHANDLE to verify its validity. If the smart card handle is not valid, the Base CSP continues to search for a new smart card.
2. If a matching smart card is not found in the Base CSP cache, the Base CSP calls to the smart card subsystem. SCardUIDlgSelectCard() is used with an appropriate callback filter to find a matching smart card with a valid default container.

Open an existing GUID-named container (no reader specified)

> **Note** This operation requires that you use the smart card with the Base CSP.

1. For each smart card that is already registered with the Base CSP, search for the requested container. Attempt an operation on the cached SCARDHANDLE to verify its validity. If the smart card handle is not valid, the smart card's serial number is passed to the SCardUI * API to continue searching for this specific smart card (rather than only a general match for the container name).
2. If a matching smart card is not found in the Base CSP cache, a call is made to the smart card subsystem. SCardUIDlgSelectCard() is used with an appropriate callback filter to find a matching smart card with the requested container. Or, if a smart card serial number resulted from the search in Step 1, the callback filter attempts to match the serial number, not the container name.

Create a new container (no reader specified)

> **Note** This operation requires that you use the smart card with the Base CSP.

If the PIN is not cached, no CRYPT_SILENT is allowed for the container creation because the user must be prompted for a PIN, at a minimum.

For other operations, the caller may be able to acquire a "verify" context against the default container (CRYPT_DEFAULT_CONTAINER_OPTIONAL) and then make a call with CryptSetProvParam to cache the user PIN for subsequent operations.

1. For each smart card already known by the CSP, refresh the stored SCARDHANDLE and make the following checks:
    a. If the smart card has been removed, continue the search.
    b. If the smart card is present, but it already has the named container, continue the search.
    c. If the smart card is available, but a call to CardQueryFreeSpace indicates that the smart card has insufficient storage for an additional key container, continue the search.
    d. Otherwise, use the first available smart card that meets the above criteria for the container creation.
2. If a matching smart card is not found in the CSP cache, make a call to the smart card subsystem. The callback that is used to filter enumerated smart cards verifies that a candidate smart card does not already have the named container, and that CardQueryFreeSpace indicates the smart card has sufficient space for an additional container. If no suitable smart card is found, the user is prompted to insert a smart card.

Delete a container

1. If the specified container name is NULL, the default container is deleted. Deleting the default container

causes a new default container to be selected arbitrarily. For this reason, this operation is not recommended.

2. For each smart card already known by the CSP, refresh the stored SCARDHANDLE and make the following checks:

   a. If the smart card does not have the named container, continue the search.

   b. If the smart card has the named container, but the smart card handle is no longer valid, store the serial number of the matching smart card and pass it to SCardUI *.

3. If a matching smart card is not found in the CSP cache, make a call to the smart card subsystem. The callback that is used to filter enumerated smart cards should verify that a candidate smart card has the named container. If a serial number was provided as a result of the previous cache search, the callback should filter enumerated smart cards on serial number rather than on container matches. If the context is non-silent and no suitable smart card is found, display UI that prompts the user to insert a smart card.

Base CSP and KSP-based architecture in Windows

Figure 4 shows the Cryptography architecture that is used by the Windows operating system.

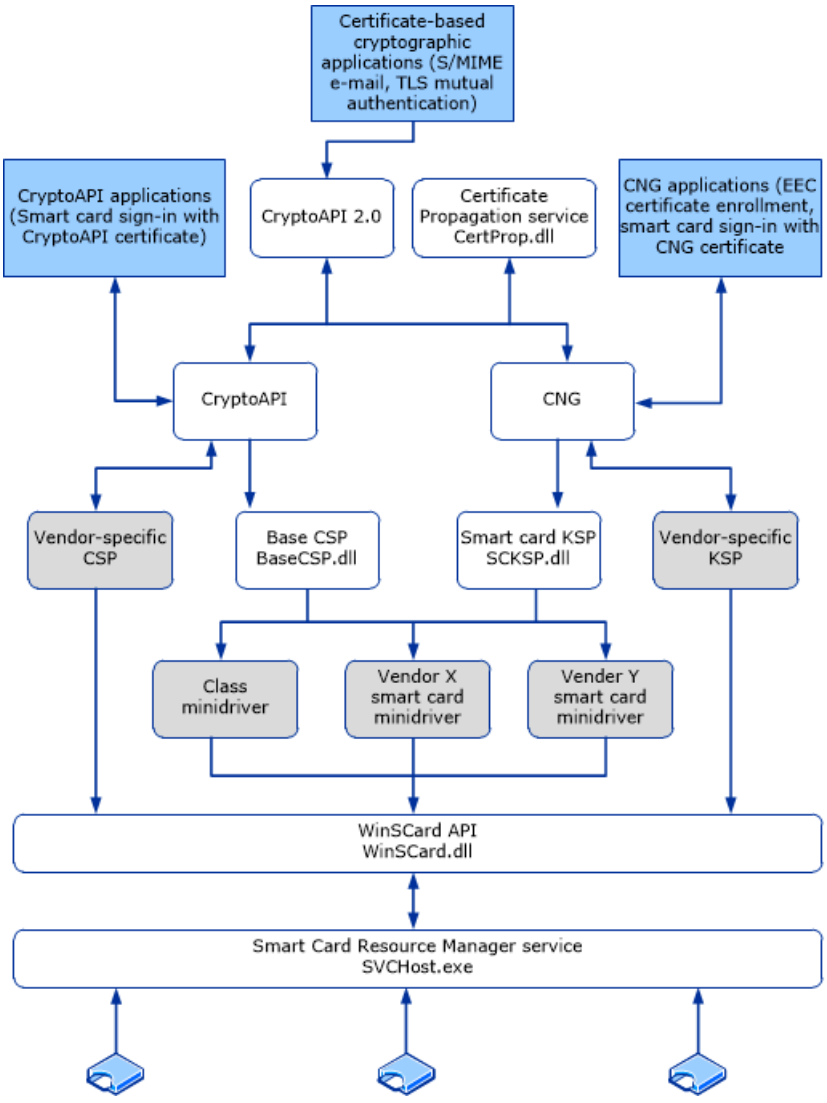


Figure 4 Cryptography architecture

Base CSP and smart card KSP properties in Windows

The following properties are supported in versions of Windows designated in the **Applies To** list at the beginning of this topic.

> **Note** The API definitions are located in WinCrypt.h and WinSCard.h.

PROPERTY	DESCRIPTION
PP_USER_CERTSTORE	- Used to return an HCERTSTORE that contains all user certificates on the smart card - Read-only (used only by CryptGetProvParam) - Caller responsible for closing the certificate store - Certificate encoded using PKCS_7_ASN_ENCODING or X509_ASN_ENCODING - CSP should set KEY_PROV_INFO on certificates - Certificate store should be assumed to be an in-memory store - Certificates should have a valid CRYPT_KEY_PROV_INFO as a property
PP_ROOT_CERTSTORE	- Read and Write (used by CryptGetProvParam and CryptSetProvParam) - Used to write a collection of root certificates to the smart card or return HCERTSTORE, which contains root certificates from the smart card - Used primarily for joining a domain by using a smart card - Caller responsible for closing the certificate store
PP_SMARTCARD_READER	- Read-only (used only by CryptGetProvParam) - Returns the smart card reader name as an ANSI string that is used to construct a fully qualified container name (that is, a smart card reader plus a container)
PP_SMARTCARD_GUID	- Return smart card GUID (also known as a serial number), which should be unique for each smart card - Used by the certificate propagation service to track the source of a root certificate
PP_UI_PROMPT	- Used to set the search string for the SCardUIDlgSelectCard card insertion dialog box - Persistent for the entire process when it is set - Write-only (used only by CryptSetProvParam)

Implications for CSPs in Windows

Cryptographic Service Providers (CSPs), including custom smart card CSPs, continue to be supported but this approach is not recommended. Using the existing Base CSP and smart card KSP with the smart card minidriver model for smart cards provides significant benefits in terms of performance, and PIN and data caching. One minidriver can be configured to work under CryptoAPI and CNG layers. This provides benefits from enhanced cryptographic support, including elliptic curve cryptography and AES.

If a smart card is registered by a CSP and a smart card minidriver, the one that was installed most recently will be used to communicate with the smart card.

Write a smart card minidriver, CSP, or KSP

CSPs and KSPs are meant to be written only if specific functionality is not available in the current smart card minidriver architecture. For example, the smart card minidriver architecture supports hardware security modules, so a minidriver could be written for a hardware security module, and a CSP or KSP may not be required unless it is needed to support algorithms that are not implemented in the Base CSP or smart card KSP.

For more information about how to write a smart card minidriver, CSP, or KSP, see Smart Card Minidrivers.

Certificate Requirements and Enumeration

3/26/2021 • 19 minutes to read • Edit Online

Applies To: Windows 10, Windows Server 2016

This topic for the IT professional and smart card developers describes how certificates are managed and used for smart card sign-in.

When a smart card is inserted, the following steps are performed.

> **Note** Unless otherwise mentioned, all operations are performed silently (CRYPT_SILENT is passed to CryptAcquireContext).

1. The smart card resource manager database searches for the smart card's cryptographic service provider (CSP).
2. A qualified container name is constructed by using the smart card reader name, and it is passed to the CSP. The format is \\.\<*Reader name*>\
3. CryptAcquireContext is called to retrieve a context to the default container. If a failure occurs, the smart card will be unusable for smart card sign-in.
4. The name of the container is retrieved by using the PP_CONTAINER parameter with CryptGetProvParam.
5. Using the context acquired in Step 3, the CSP is queried for the PP_USER_CERTSTORE parameter (added in Windows Vista). For more information, see Smart Card Architecture. If the operation is successful, the name of a certificate store is returned, and the program flow skips to Step 8.
6. If the operation in Step 5 fails, the default container context from Step 3 is queried for the AT_KEYEXCHANGE key.
7. The certificate is then queried from the key context by using KP_CERTIFICATE. The certificate is added to an in-memory certificate store.
8. For each certificate in the certificate store from Step 5 or Step 7, the following checks are performed:
   a. The certificate must be valid, based on the computer system clock (not expired or valid with a future date).
   b. The certificate must not be in the AT_SIGNATURE part of a container.
   c. The certificate must have a valid user principal name (UPN).
   d. The certificate must have the digital signature key usage.
   e. The certificate must have the smart card logon EKU.

   Any certificate that meets these requirements is displayed to the user with the certificate's UPN (or e-mail address or subject, depending on the presence of the certificate extensions).

   > **Note** These requirements are the same as those in Windows Server 2003, but they are performed before the user enters the PIN. You can override many of them by using Group Policy settings.

9. The process then chooses a certificate, and the PIN is entered.

10. LogonUI.exe packages the information and sends it to Lsass.exe to process the sign-in attempt.
11. If successful, LogonUI.exe closes. This causes the context acquired in Step 3 to be released.

About Certificate support for compatibility

Although versions of Windows earlier than Windows Vista include support for smart cards, the types of certificates that smart cards can contain are limited. The limitations are:

- Each certificate must have a user principal name (UPN) and the smart card sign-in object identifier (also known as OID) in the enhanced key usage (EKU) attribute field. There is a Group Policy setting, Allow ECC certificates to be used for logon and authentication, to make the EKU optional.
- Each certificate must be stored in the AT_KEYEXCHANGE portion of the default CryptoAPI container, and non-default CryptoAPI containers are not supported.

The following table lists the certificate support in older Windows operating system versions.

OPERATING SYSTEM	CERTIFICATE SUPPORT
Windows Server 2008 R2 and Windows 7	Support for smart card sign-in with ECC-based certificates. ECC smart card sign-in is enabled through Group Policy. ECDH_P256 ECDH Curve P-256 from FIPS 186-2 ECDSA_P256 ECDSA Curve P-256 from FIPS 186-2 ECDH_P384 ECDH Curve P-384 from FIPS 186-2 ECDH_P521 ECDH Curve P-521 from FIPS 186-2 ECDSA_P256 ECDH Curve P-256 from FIPS 186-2 ECDSA_P384 ECDSA Curve P-384 from FIPS 186-2 ECDSA_P521 ECDSA Curve P-384 from FIPS 186-2
Windows Server 2008 and Windows Vista	Valid certificates are enumerated and displayed from all smart cards and presented to the user. Keys are no longer restricted to the default container, and certificates in different containers can be chosen. Elliptic curve cryptography (ECC)-based certificates are not supported for smart card sign-in

Smart card sign-in flow in Windows

Most issues during authentication occur because of session behavior changes. When changes occur, the Local

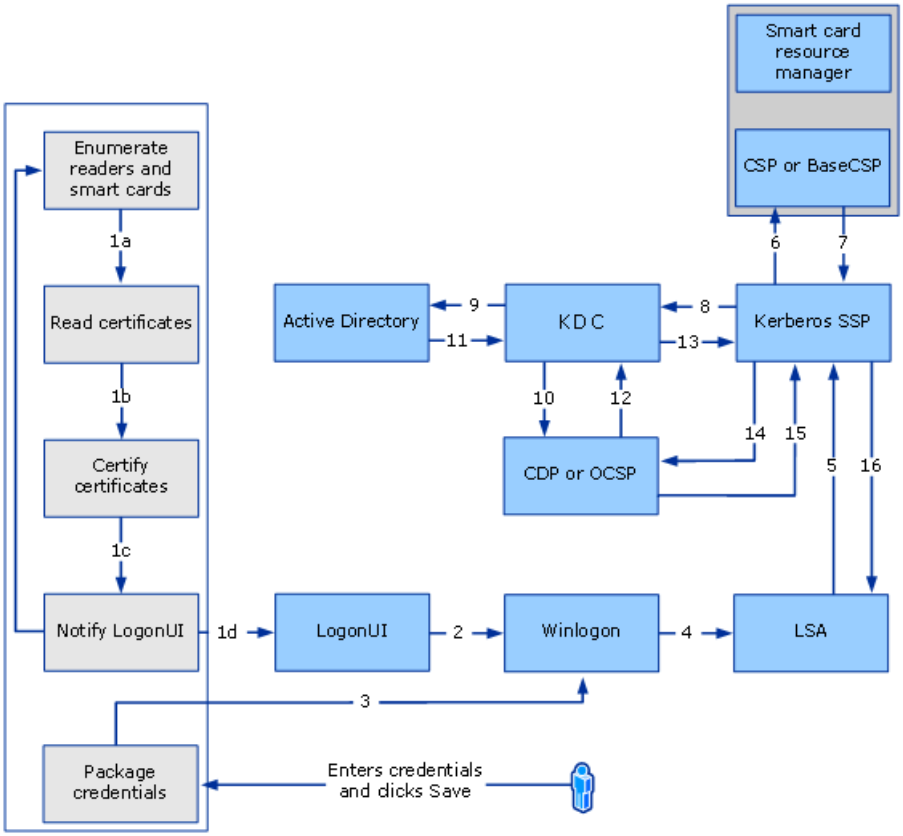
Security Authority (LSA) does not reacquire the session context; it relies instead on the Cryptographic Service Provider to handle the session change.

In the supported versions of Windows designated in the **Applies To** list at the beginning of this topic, client certificates that do not contain a UPN in the **subjectAltName** (SAN) field of the certificate can be enabled for sign-in, which supports a wider variety of certificates and supports multiple sign-in certificates on the same card.

Support for multiple certificates on the same card is enabled by default. New certificate types must be enabled through Group Policy.

If you enable the **Allow signature keys valid for Logon** credential provider policy, any certificates that are available on the smart card with a signature-only key are listed on the sign-in screen. This allows users to select their sign-in experience. If the policy is disabled or not configured, smart card signature-key-based certificates are not listed on the sign-in screen.

The following diagram illustrates how smart card sign-in works in the supported versions of Windows.



Smart card sign-in flow

Following are the steps that are performed during a smart card sign-in:

1. Winlogon requests the sign-in UI credential information.
2. Asynchronously, smart card resource manager starts, and the smart card credential provider does the following:
    a. Gets credential information (a list of known credentials, or if no credentials exist, the smart card reader information that Windows detected).
    b. Gets a list of smart card readers (by using the WinSCard API) and the list of smart cards inserted in each of them.
    c. Enumerates each card to verify that a sign-in certificate that is controlled by Group Policy is

present. If the certificate is present, the smart card credential provider copies it into a temporary, secure cache on the computer or terminal.

> **Note** Smartcard cache entries are created for certificates with a subject name or with a subject key identifier. If the certificate has a subject name, it is stored with an index that is based on the subject name and certificate issuer. If another certificate with the same subject name and certificate issuer is used, it will replace the existing cached entry. A change in this behavior after Windows Vista, allows for the condition when the certificate does not have a subject name, the cache is created with an index that is based on the subject key identifier and certificate issuer. If another certificate has the same the subject key identifier and certificate issuer, the cache entry is replaced. When certificates have neither a subject name nor subject key identifier, a cached entry is not created.

d. Notifies the sign-in UI that it has new credentials.

3. The sign-in UI requests the new credentials from the smart card credential provider. As a response, the smart card credential provider provides each sign-in certificate to the sign-in UI, and corresponding sign-in tiles are displayed. The user selects a smart card-based sign-in certificate tile, and Windows displays a PIN dialog box.
4. The user enters the PIN, and then presses ENTER. The smart card credential provider encrypts the PIN.
5. The credential provider that resides in the LogonUI system collects the PIN. As part of packaging credentials in the smart card credential provider, the data is packaged in a KERB_CERTIFICATE_LOGON structure. The main contents of the KERB_CERTIFICATE_LOGON structure are the smart card PIN, CSP data (such as reader name and container name), user name, and domain name. User name is required if the sign-in domain is not in the same forest because it enables a certificate to be mapped to multiple user accounts.
6. The credential provider wraps the data (such as the encrypted PIN, container name, reader name, and card key specification) and sends it back to LogonUI.
7. Winlogon presents the data from LogonUI to the LSA with the user information in LSALogonUser.
8. LSA calls the Kerberos authentication package (Kerberos SSP) to create a Kerberos authentication service request (KRB_AS_REQ), which containing a preauthenticator (as specified in RFC 4556: Public Key Cryptography for Initial Authentication in Kerberos (PKINIT)).

   If the authentication is performed by using a certificate that uses a digital signature, the preauthentication data consists of the user's public certificate and the certificate that is digitally signed with the corresponding private key.
   If the authentication is performed by using a certificate that uses key encipherment, the preauthentication data consists of the user's public certificate and the certificate that is encrypted with the corresponding private key.
9. To sign the request digitally (as per RFC 4556), a call is made to the corresponding CSP for a private key operation. Because the private key in this case is stored in a smart card, the smart card subsystem is called, and the necessary operation is completed. The result is sent back to the Kerberos security support provider (SSP).
10. The Kerberos SSP sends an authentication request for a ticket-granting-ticket (TGT) (per RFC 4556) to the Key Distribution Center (KDC) service that runs on a domain controller.
11. The KDC finds the user's account object in Active Directory Domain Services (AD DS), as detailed in Client certificate requirements and mappings, and uses the user's certificate to verify the signature.
12. The KDC validates the user's certificate (time, path, and revocation status) to ensure that the certificate is from a trusted source. The KDC uses CryptoAPI to build a certification path from the user's certificate to a

root certification authority (CA) certificate that resides in the root store on the domain controller. The KDC then uses CryptoAPI to verify the digital signature on the signed authenticator that was included in the preauthentication data fields. The domain controller verifies the signature and uses the public key from the user's certificate to prove that the request originated from the owner of the private key that corresponds to the public key. The KDC also verifies that the issuer is trusted and appears in the NTAUTH certificate store.

13. The KDC service retrieves user account information from AD DS. The KDC constructs a TGT, which is based on the user account information that it retrieves from AD DS. The TGT's authorization data fields include the user's security identifier (SID), the SIDs for universal and global domain groups to which the user belongs, and (in a multidomain environment) the SIDs for any universal groups of which the user is a member.

14. The domain controller returns the TGT to the client as part of the KRB_AS_REP response.

    > **Note** The KRB_AS_REP packet consists of:
    >
    > - Privilege attribute certificate (PAC)
    > - User's SID
    > - SIDs of any groups of which the user is a member
    > - A request for ticket-granting service (TGS)
    > - Preauthentication data

    TGT is encrypted with the master key of the KDC, and the session key is encrypted with a temporary key. This temporary key is derived based on RFC 4556. Using CryptoAPI, the temporary key is decrypted. As part of the decryption process, if the private key is on a smart card, a call is made to the smart card subsystem by using the specified CSP to extract the certificate corresponding to the user's public key. (Programmatic calls for the certificate include CryptAcquireContext, CryptSetProvParam with the PIN, CryptgetUserKey, and CryptGetKeyParam.) After the temporary key is obtained, the Kerberos SSP decrypts the session key.

15. The client validates the reply from the KDC (time, path, and revocation status). It first verifies the KDC's signature by the construction of a certification path from the KDC's certificate to a trusted root CA, and then it uses the KDC's public key to verify the reply signature.

16. Now that a TGT has been obtained, the client obtains a service ticket, which is used to sign in to the local computer.

17. With success, LSA stores the tickets and returns a success message to LSALogonUser. After this success message is issued, user profile for the device is selected and set, Group Policy refresh is instantiated, and other actions are performed.

18. After the user profile is loaded, the Certification Propagation Service (CertPropSvc) detects this event, reads the certificates from the smart card (including the root certificates), and then populates them into the user's certificate store (MYSTORE).

19. CSP to smart card resource manager communication happens on the LRPC Channel.

20. On successful authentication, certificates are propagated to the user's store asynchronously by the Certificate Propagation Service (CertPropSvc).

21. When the card is removed, certificates in the temporary secure cache store are removed. The Certificates are no longer available for sign-in, but they remain in the user's certificate store.

> **Note** A SID is created for each user or group at the time a user account or a group account is created within the local security accounts database or within AD DS. The SID never changes, even if the user or group

account is renamed.

For more information about the Kerberos protocol, see Microsoft Kerberos.

By default, the KDC verifies that the client's certificate contains the smart card client authentication EKU szOID_KP_SMARTCARD_LOGON. However, if enabled, the **Allow certificates with no extended key usage certificate attribute** Group Policy setting allows the KDC to not require the SC-LOGON EKU. SC-LOGON EKU is not required for account mappings that are based on the public key.

KDC certificate

Active Directory Certificate Services provides three kinds of certificate templates:

- Domain controller
- Domain controller authentication
- Kerberos authentication

Depending on the configuration of the domain controller, one of these types of certificates is sent as a part of the AS_REP packet.

Client certificate requirements and mappings

Certificate requirements are listed by versions of the Windows operating system. Certificate mapping describes how information from the certificate is mapped to the user account.

Certificate requirements

The smart card certificate has specific format requirements when it is used with Windows XP and earlier operating systems. You can enable any certificate to be visible for the smart card credential provider.

COMPONENT	REQUIREMENTS FOR WINDOWS 8.1, WINDOWS 8, WINDOWS 7, WINDOWS VISTA, AND WINDOWS 10	REQUIREMENTS FOR WINDOWS XP
CRL distribution point location	Not required	The location must be specified, online, and available, for example: [1]CRL Distribution Point Distribution Point Name: Full Name: URL=http://server1.contoso.com/CertEnroll/caname.crl
Key usage	Digital signature	Digital signature
Basic constraints	Not required	[Subject Type=End Entity, Path Length Constraint=None] (Optional)
Enhanced key usage (EKU)	The smart card sign-in object identifier is not required. **Note** If an EKU is present, it must contain the smart card sign-in EKU. Certificates with no EKU can be used for sign-in.	- Client Authentication (1.3.6.1.5.5.7.3.2) The client authentication object identifier is required only if a certificate is used for SSL authentication. - Smart Card Sign-in (1.3.6.1.4.1.311.20.2.2)

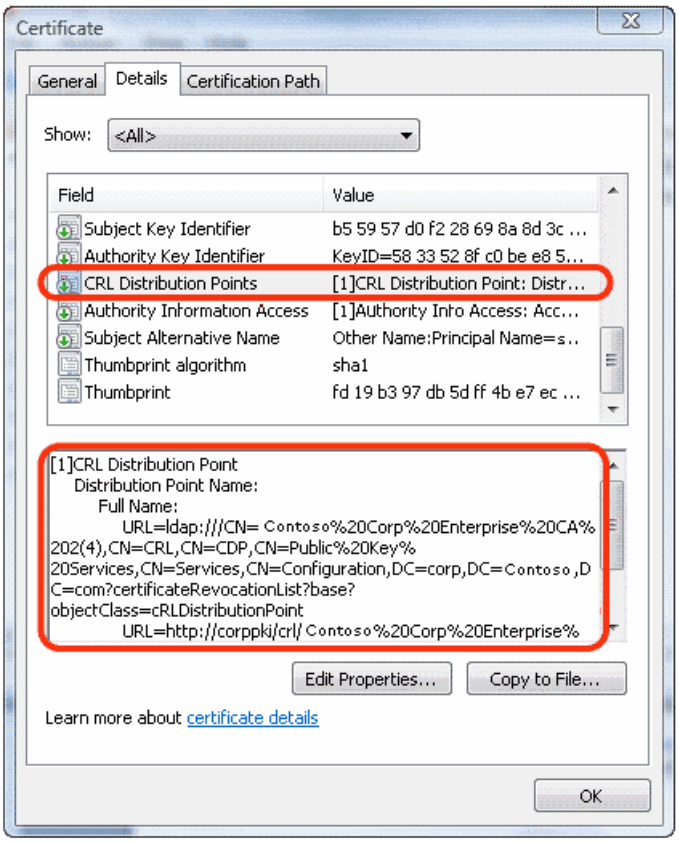
COMPONENT	REQUIREMENTS FOR WINDOWS 8.1, WINDOWS 8, WINDOWS 7, WINDOWS VISTA, AND WINDOWS 10	REQUIREMENTS FOR WINDOWS XP
Subject alternative name	E-mail ID is not required for smart card sign-in.	Other Name: Principal Name=(UPN), for example: UPN=user1@contoso.com The UPN OtherName object identifier is 1.3.6.1.4.1.311.20.2.3. The UPN OtherName value must be an ASN1-encoded UTF8 string.
Subject	Not required	Distinguished name of user. This field is a mandatory extension, but the population of this field is optional.
Key exchange (AT_KEYEXCHANGE field)	Not required for smart card sign-in certificates if a Group Policy setting is enabled. (By default, Group Policy settings are not enabled.)	Not required
CRL	Not required	Not required
UPN	Not required	Not required
Notes	You can enable any certificate to be visible for the smart card credential provider.	There are two predefined types of private keys. These keys are Signature Only (AT_SIGNATURE) and Key Exchange (AT_KEYEXCHANGE). Smart card sign-in certificates must have a Key Exchange (AT_KEYEXCHANGE) private key type.

Client certificate mappings

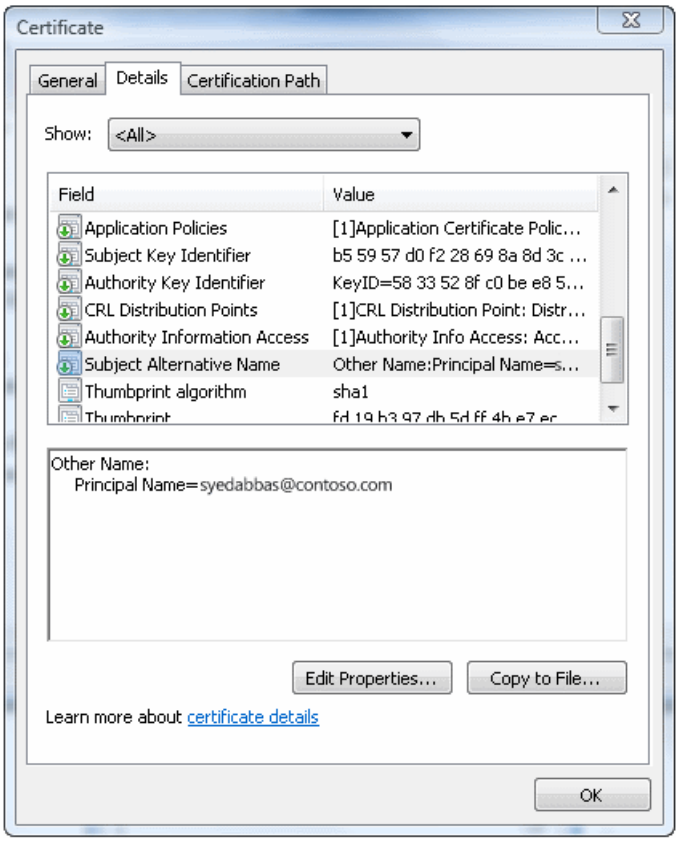
Certificate mapping is based on the UPN that is contained in the subjectAltName (SAN) field of the certificate. Client certificates that do not contain information in the SAN field are also supported.

SSL/TLS can map certificates that do not have SAN, and the mapping is done by using the AltSecID attributes on client accounts. The X509 AltSecID, which is used by SSL/TLS client authentication is of the form "X509: <I>"*<Issuer Name>*"<S>"*<Subject Name>*. The *<Issuer Name>* and *<Subject Name>* are taken from the client certificate, with '\r' and '\n' replaced with ','.

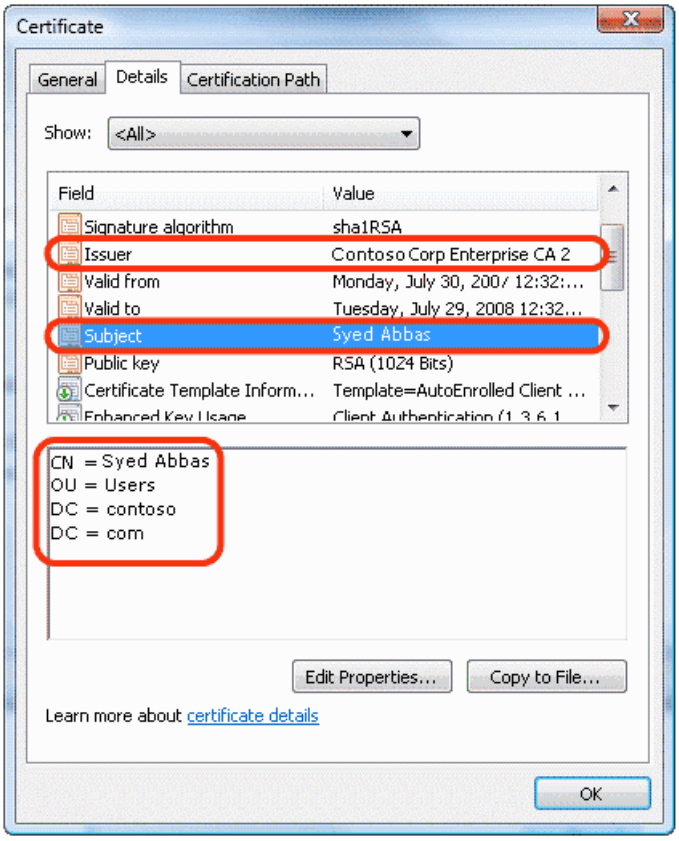
Certificate revocation list distribution points



UPN in Subject Alternative Name field

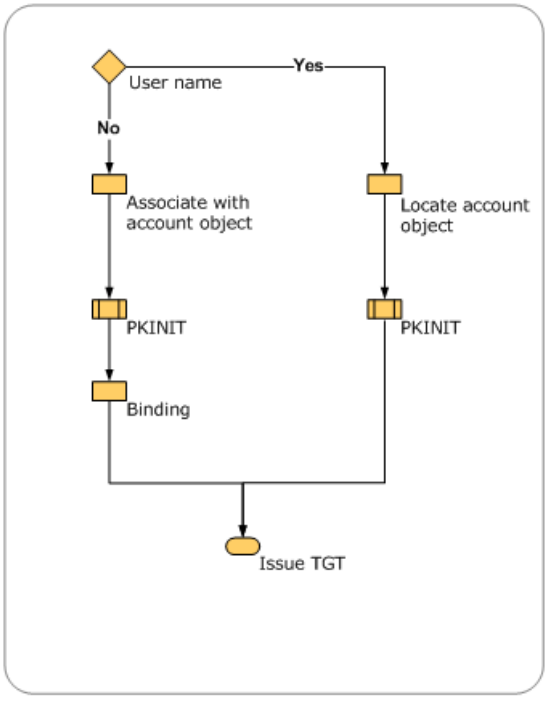


Subject and Issuer fields



This account mapping is supported by the KDC in addition to six other mapping methods. The following figure demonstrates a flow of user account mapping logic that is used by the KDC.

High-level flow of certificate processing for sign-in



The certificate object is parsed to look for content to perform user account mapping.

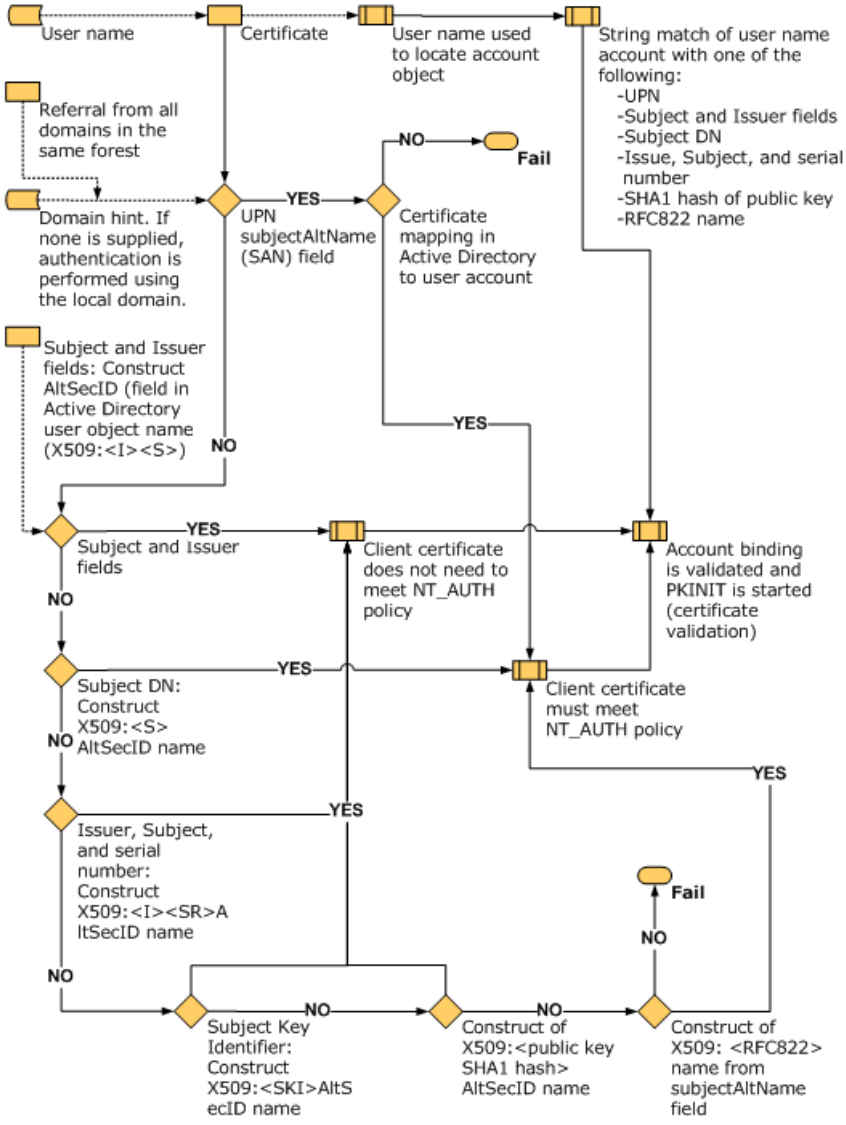
- When a user name is provided with the certificate, the user name is used to locate the account object. This operation is the fastest, because string matching occurs.
- When only the certificate object is provided, a series of operations are performed to locate the user name to map the user name to an account object.
- When no domain information is available for authentication, the local domain is used by default. If any

other domain is to be used for lookup, a domain name hint should be provided to perform the mapping and binding.

Mapping based on generic attributes is not possible because there is no generic API to retrieve attributes from a certificate. Currently, the first method that locates an account successfully stops the search. But a configuration error occurs if two methods map the same certificate to different user accounts when the client does not supply the client name through the mapping hints.

The following figure illustrates the process of mapping user accounts for sign-in in the directory by viewing various entries in the certificate.

Certificate processing logic



NT_AUTH policy is best described in the CERT_CHAIN_POLICY_NT_AUTH parameter section of the CertVerifyCertificateChainPolicy function. For more information, see CertVerifyCertificateChainPolicy.

Smart card sign-in for a single user with one certificate into multiple accounts

A single user certificate can be mapped to multiple accounts. For example, a user might be able to sign in to a user account and also to sign in as a domain administrator. The mapping is done by using the constructed AltSecID based on attributes from client accounts. For information about how this mapping is evaluated, see Client certificate requirements and mappings.

> **Note** Because each account has a different user name, we recommend that you enable the **Allow user**

> **name hint** Group Policy setting (**X509HintsNeeded** registry key) to provide the optional fields that allow users to enter their user names and domain information to sign in.

Based on the information that is available in the certificate, the sign-in conditions are:

1. If no UPN is present in the certificate:
    a. Sign-in can occur in the local forest or in another forest if a single user with one certificate needs to sign in to different accounts.
    b. A hint must be supplied if mapping is not unique (for example, if multiple users are mapped to the same certificate).
2. If a UPN is present in the certificate:
    a. The certificate cannot be mapped to multiple users in the same forest.
    b. The certificate can be mapped to multiple users in different forests. For a user to sign in to other forests, an X509 hint must be supplied to the user.

Smart card sign-in for multiple users into a single account

A group of users might sign in to a single account (for example, an administrator account). For that account, user certificates are mapped so that they are enabled for sign-in.

Several distinct certificates can be mapped to a single account. For this to work properly, the certificate cannot have UPNs.

For example, if Certificate1 has CN=CNName1, Certificate2 has CN=User1, and Certificate3 has CN=User2, the AltSecID of these certificates can be mapped to a single account by using the Active Directory Users and Computers name mapping.

Smart card sign-in across forests

For account mapping to work across forests, particularly in cases where there is not enough information available on the certificate, the user might enter a hint in the form of a user name, such as *domain**user*, or a fully qualified UPN such as *user@contoso.com*.

> **Note** For the hint field to appear during smart card sign-in, the **Allow user name hint** Group Policy setting (**X509HintsNeeded** registry key) must be enabled on the client.

OCSP support for PKINIT

Online Certificate Status Protocol (OCSP), which is defined in RFC 2560, enables applications to obtain timely information about the revocation status of a certificate. Because OCSP responses are small and well bound, constrained clients might want to use OCSP to check the validity of the certificates for Kerberos on the KDC, to avoid transmission of large CRLs, and to save bandwidth on constrained networks. For information about CRL registry keys, see Smart Card Group Policy and Registry Settings.

The KDCs in Windows attempt to get OCSP responses and use them when available. This behavior cannot be disabled. CryptoAPI for OCSP caches OCSP responses and the status of the responses. The KDC supports only OCSP responses for the signer certificate.

Windows client computers attempt to request the OCSP responses and use them in the reply when they are available. This behavior cannot be disabled.

Smart card root certificate requirements for use with domain sign-in

For sign-in to work in a smart card-based domain, the smart card certificate must meet the following conditions:

- The KDC root certificate on the smart card must have an HTTP CRL distribution point listed in its certificate.
- The smart card sign-in certificate must have the HTTP CRL distribution point listed in its certificate.
- The CRL distribution point must have a valid CRL published and a delta CRL, if applicable, even if the CRL distribution point is empty.
- The smart card certificate must contain one of the following:
    - A subject field that contains the DNS domain name in the distinguished name. If it does not, resolution to an appropriate domain fails, so Remote Desktop Services and the domain sign-in with the smart card fail.
    - A UPN where the domain name resolves to the actual domain. For example, if the domain name is Engineering.Corp.Contoso, the UPN is username@engineering.corp.contoso.com. If any part of the domain name is omitted, the Kerberos client cannot find the appropriate domain.

Although the HTTP CRL distribution points are on by default in Windows Server 2008, subsequent versions of the Windows Server operating system do not include HTTP CRL distribution points. To allow smart card sign-in to a domain in these versions, do the following:

1. Enable HTTP CRL distribution points on the CA.
2. Restart the CA.
3. Reissue the KDC certificate.
4. Issue or reissue the smart card sign-in certificate.
5. Propagate the updated root certificate to the smart card that you want to use for the domain sign-in.

The workaround is to enable the **Allow user name hint** Group Policy setting (**X509HintsNeeded** registry key), which allows the user to supply a hint in the credentials user interface for domain sign-in.

If the client computer is not joined to the domain or if it is joined to a different domain, the client computer can resolve the server domain only by looking at the distinguished name on the certificate, not the UPN. For this scenario to work, the certificate requires a full subject, including DC=*<DomainControllerName>*, for domain name resolution.

To deploy root certificates on a smart card for the currently joined domain, you can use the following command:

certutil -scroots update

For more information about this option for the command-line tool, see -SCRoots.

See also

How Smart Card Sign-in Works in Windows

Smart Card and Remote Desktop Services

3/26/2021 • 5 minutes to read • Edit Online

Applies To: Windows 10, Windows Server 2016

This topic for the IT professional describes the behavior of Remote Desktop Services when you implement smart card sign-in.

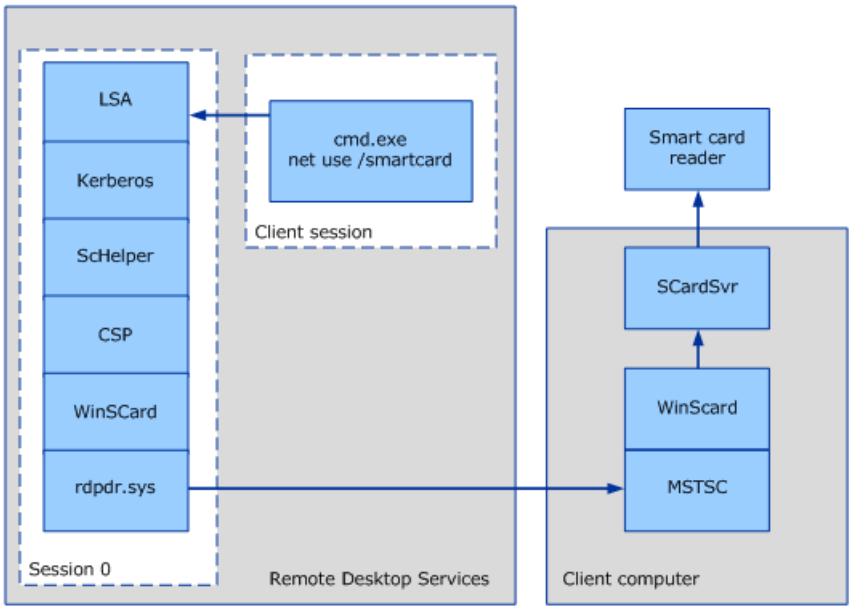
The content in this topic applies to the versions of Windows that are designated in the **Applies To** list at the beginning of this topic. In these versions, smart card redirection logic and **WinSCard** API are combined to support multiple redirected sessions into a single process.

Smart card support is required to enable many Remote Desktop Services scenarios. These include:

- Using Fast User Switching or Remote Desktop Services. A user is not able to establish a redirected smart card-based remote desktop connection. That is, the connect attempt is not successful in Fast User Switching or from a Remote Desktop Services session.
- Enabling Encrypting File System (EFS) to locate the user's smart card reader from the Local Security Authority (LSA) process in Fast User Switching or in a Remote Desktop Services session. If EFS is not able to locate the smart card reader or certificate, EFS cannot decrypt user files.

Remote Desktop Services redirection

In a Remote Desktop scenario, a user is using a remote server for running services, and the smart card is local to the computer that the user is using. In a smart card sign-in scenario, the smart card service on the remote server redirects to the smart card reader that is connected to the local computer where the user is trying to sign in.



Remote Desktop redirection

Notes about the redirection model:

1. This scenario is a remote sign-in session on a computer with Remote Desktop Services. In the remote session (labeled as "Client session"), the user runs **net use /smartcard**.

2. Arrows represent the flow of the PIN after the user types the PIN at the command prompt until it reaches the user's smart card in a smart card reader that is connected to the Remote Desktop Connection (RDC) client computer.
3. The authentication is performed by the LSA in session 0.
4. The CryptoAPI processing is performed in the LSA (Lsass.exe). This is possible because RDP redirector (rdpdr.sys) allows per-session, rather than per-process, context.
5. The WinScard and SCRedir components, which were separate modules in operating systems earlier than Windows Vista, are now included in one module. The ScHelper library is a CryptoAPI wrapper that is specific to the Kerberos protocol.
6. The redirection decision is made on a per smart card context basis, based on the session of the thread that performs the SCardEstablishContext call.
7. Changes to WinSCard.dll implementation were made in Windows Vista to improve smart card redirection.

RD Session Host server single sign-in experience

As a part of the Common Criteria compliance, the RDC client must be configurable to use Credential Manager to acquire and save the user's password or smart card PIN. Common Criteria compliance requires that applications not have direct access to the user's password or PIN.

Common Criteria compliance requires specifically that the password or PIN never leave the LSA unencrypted. A distributed scenario should allow the password or PIN to travel between one trusted LSA and another, and it cannot be unencrypted during transit.

When smart card-enabled single sign-in (SSO) is used for Remote Desktop Services sessions, users still need to sign in for every new Remote Desktop Services session. However, the user is not prompted for a PIN more than once to establish a Remote Desktop Services session. For example, after the user double-clicks a Microsoft Word document icon that resides on a remote computer, the user is prompted to enter a PIN. This PIN is sent by using a secure channel that the credential SSP has established. The PIN is routed back to the RDC client over the secure channel and sent to Winlogon. The user does not receive any additional prompts for the PIN, unless the PIN is incorrect or there are smart card-related failures.

Remote Desktop Services and smart card sign-in

Remote Desktop Services enable users to sign in with a smart card by entering a PIN on the RDC client computer and sending it to the RD Session Host server in a manner similar to authentication that is based on user name and password.

In addition, Group Policy settings that are specific to Remote Desktop Services need to be enabled for smart card-based sign-in.

To enable smart card sign-in to a Remote Desktop Session Host (RD Session Host) server, the Key Distribution Center (KDC) certificate must be present on the RDC client computer. If the computer is not in the same domain or workgroup, the following command can be used to deploy the certificate:

certutil -dspublish NTAuthCA "*DSCDPContainer*"

The *DSCDPContainer* Common Name (CN) is usually the name of the certification authority.

Example:

certutil -dspublish NTAuthCA <*CertFile*> **"CN=NTAuthCertificates,CN=Public Key Services,CN=Services,CN=Configuration,DC=engineering,DC=contoso,DC=com"**

For information about this option for the command-line tool, see -dsPublish.

Remote Desktop Services and smart card sign-in across domains

To enable remote access to resources in an enterprise, the root certificate for the domain must be provisioned on the smart card. From a computer that is joined to a domain, run the following command at the command line:

certutil -scroots update

For information about this option for the command-line tool, see -SCRoots.

For Remote Desktop Services across domains, the KDC certificate of the RD Session Host server must also be present in the client computer's NTAUTH store. To add the store, run the following command at the command line:

certutil -addstore -enterprise NTAUTH <*CertFile*>

Where <*CertFile*> is the root certificate of the KDC certificate issuer.

For information about this option for the command-line tool, see -addstore.

> **Note** If you use the credential SSP on computers running the supported versions of the operating system that are designated in the **Applies To** list at the beginning of this topic: To sign in with a smart card from a computer that is not joined to a domain, the smart card must contain the root certification of the domain controller. A public key infrastructure (PKI) secure channel cannot be established without the root certification of the domain controller.

Sign-in to Remote Desktop Services across a domain works only if the UPN in the certificate uses the following form: <*ClientName*>@<*DomainDNSName*>

The UPN in the certificate must include a domain that can be resolved. Otherwise, the Kerberos protocol cannot determine which domain to contact. You can resolve this issue by enabling GPO X509 domain hints. For more information about this setting, see Smart Card Group Policy and Registry Settings.

See also

How Smart Card Sign-in Works in Windows

Smart Cards for Windows Service

3/5/2021 • 2 minutes to read • Edit Online

Applies To: Windows 10, Windows Server 2016

This topic for the IT professional and smart card developers describes how the Smart Cards for Windows service (formerly called Smart Card Resource Manager) manages readers and application interactions.

The Smart Cards for Windows service provides the basic infrastructure for all other smart card components as it manages smart card readers and application interactions on the computer. It is fully compliant with the specifications set by the PC/SC Workgroup. For information about these specifications, see the PC/SC Workgroup Specifications website.

The Smart Cards for Windows service runs in the context of a local service, and it is implemented as a shared service of the services host (svchost) process. The Smart Cards for Windows service, Scardsvr, has the following service description:

```
<serviceData
    dependOnService="PlugPlay"
    description="@%SystemRoot%\System32\SCardSvr.dll,-5"
    displayName="@%SystemRoot%\System32\SCardSvr.dll,-1"
    errorControl="normal"
    group="SmartCardGroup"
    imagePath="%SystemRoot%\system32\svchost.exe -k LocalServiceAndNoImpersonation"
    name="SCardSvr"
    objectName="NT AUTHORITY\LocalService"
    requiredPrivileges="SeCreateGlobalPrivilege,SeChangeNotifyPrivilege"
    sidType="unrestricted"
    start="demand"
    type="win32ShareProcess"
    >
  <failureActions resetPeriod="900">
       <actions>
          <action
              delay="120000"
              type="restartService"
          />
          <action
              delay="300000"
              type="restartService"
          />
          <action
               delay="0"
              type="none"
          />
      </actions>
  </failureActions>
  <securityDescriptor name="ServiceXSecurity"/>
</serviceData>

  <registryKeys buildFilter="">
      <registryKey keyName="HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SCardSvr\Parameters">
      <registryValue
          name="ServiceDll"
          value="%SystemRoot%\System32\SCardSvr.dll"
          valueType="REG_EXPAND_SZ"
          />
      <registryValue
          name="ServiceMain"
          value="CalaisMain"
          valueType="REG_SZ"
          />
      <registryValue
          name="ServiceDllUnloadOnStop"
          value="1"
          valueType="REG_DWORD"
          />
      </registryKey>
  </registryKeys>
```

Note For winscard.dll to be invoked as the proper class installer, the INF file for a smart card reader must specify the following for **Class** and **ClassGUID**:

```
Class=SmartCardReader
ClassGuid={50DD5230-BA8A-11D1-BF5D-0000F805F530}
```

By default, the service is configured for manual mode. Creators of smart card reader drivers must configure their INFs so that they start the service automatically and winscard.dll files call a predefined entry point to start the service during installation. The entry point is defined as part of the **SmartCardReader** class, and it is not called directly. If a device advertises itself as part of this class, the entry point is automatically invoked to start the service when the device is inserted. Using this method ensures that the service is enabled when it is needed,

but it is also disabled for users who do not use smart cards.

When the service is started, it performs several functions:

1. It registers itself for service notifications.
2. It registers itself for Plug and Play (PnP) notifications related to device removal and additions.
3. It initializes its data cache and a global event that signals that the service has started.

> **Note** For smart card implementations, consider sending all communications in Windows operating systems with smart card readers through the Smart Cards for Windows service. This provides an interface to track, select, and communicate with all drivers that declare themselves members of the smart card reader device group.

The Smart Cards for Windows service categorizes each smart card reader slot as a unique reader, and each slot is also managed separately, regardless of the device's physical characteristics. The Smart Cards for Windows service handles the following high-level actions:

- Device introduction
- Reader initialization
- Notifying clients of new readers
- Serializing access to readers
- Smart card access
- Tunneling of reader-specific commands

See also

How Smart Card Sign-in Works in Windows

Certificate Propagation Service

3/5/2021 • 2 minutes to read • Edit Online

Applies To: Windows 10, Windows Server 2016

This topic for the IT professional describes the certificate propagation service (CertPropSvc), which is used in smart card implementation.

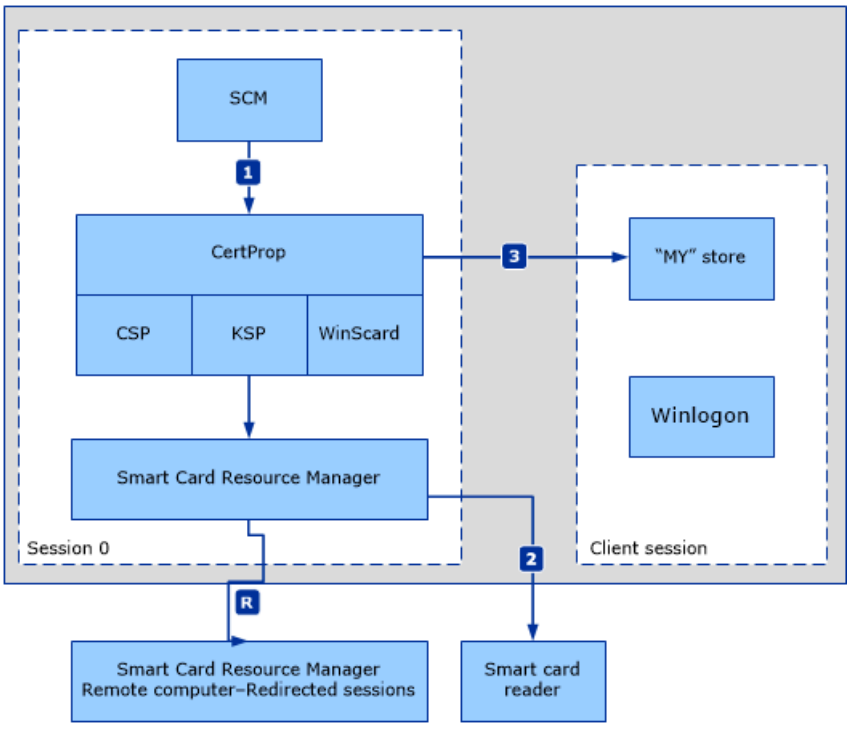
The certificate propagation service activates when a signed-in user inserts a smart card in a reader that is attached to the computer. This action causes the certificate to be read from the smart card. The certificates are then added to the user's Personal store. Certificate propagation service actions are controlled by using Group Policy. For more information, see Smart Card Group Policy and Registry Settings.

> **Note** The certificate propagation service must be running for smart card Plug and Play to work.

The following figure shows the flow of the certificate propagation service. The action begins when a signed-in user inserts a smart card.

1. The arrow labeled **1** indicates that the Service Control Manager (SCM) notifies the certificate propagation service (CertPropSvc) when a user signs in, and CertPropSvc begins to monitor the smart cards in the user session.
2. The arrow labeled **R** represents the possibility of a remote session and the use of smart card redirection.
3. The arrow labeled **2** indicates the certification to the reader.
4. The arrow labeled **3** indicates the access to the certificate store during the client session.

Certificate propagation service



1. A signed-in user inserts a smart card.
2. CertPropSvc is notified that a smart card was inserted.

3. CertPropSvc reads all certificates from all inserted smart cards. The certificates are written to the user's personal certificate store.

> **Note** The certificate propagation service is started as a Remote Desktop Services dependency.

Properties of the certificate propagation service include:

- CERT_STORE_ADD_REPLACE_EXISTING_INHERIT_PROPERTIES adds certificates to a user's Personal store.
- If the certificate has the CERT_ENROLLMENT_PROP_ID property (as defined by wincrypt.h), it filters empty requests and places them in the current user's request store, but it does not propagate them to the user's Personal store.
- The service does not propagate any computer certificates to a user's Personal store or propagate user certificates to a computer store.
- The service propagates certificates according to Group Policy options that are set, which may include:
  - **Turn on certificate propagation from the smart card** specifies whether a user's certificate should be propagated.
  - **Turn on root certificate propagation from smart card** specifies whether root certificates should be propagated.
  - **Configure root certificate cleanup** specifies how root certificates are removed.

Root certificate propagation service

Root certificate propagation is responsible for the following smart card deployment scenarios when public key infrastructure (PKI) trust has not yet been established:

- Joining the domain
- Accessing a network remotely

In both cases, the computer is not joined to a domain, and therefore, trust is not being managed by Group Policy. However, the objective is to authenticate to a remote server, such as the domain controller. Root certificate propagation provides the ability to use the smart card to include the missing trust chain.

When the smart card is inserted, the certificate propagation service propagates any root certificates on the card to the trusted smart card root computer certificate stores. This process establishes a trust relationship with the enterprise resources. You may also use a subsequent cleanup action when the user's smart card is removed from the reader, or when the user signs out. This is configurable with Group Policy. For more information, see Smart Card Group Policy and Registry Settings.

For more information about root certificate requirements, see Smart card root certificate requirements for use with domain sign-in.

See also

How Smart Card Sign-in Works in Windows

Smart Card Removal Policy Service

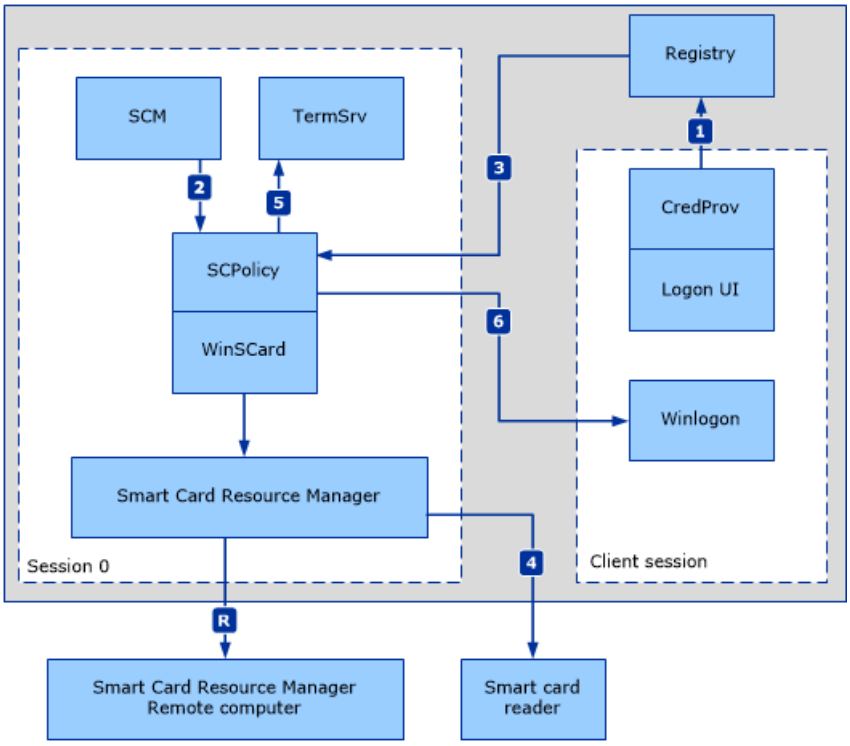
3/5/2021 • 2 minutes to read • Edit Online

Applies To: Windows 10, Windows Server 2016

This topic for the IT professional describes the role of the removal policy service (ScPolicySvc) in smart card implementation.

The smart card removal policy service is applicable when a user has signed in with a smart card and subsequently removes that smart card from the reader. The action that is performed when the smart card is removed is controlled by Group Policy settings. For more information, see Smart Card Group Policy and Registry Settings.

Smart card removal policy service



The numbers in the previous figure represent the following actions:

1. Winlogon is not directly involved in monitoring for smart card removal events. The sequence of steps that are involved when a smart card is removed begins with the smart card credential provider in the sign-in UI process. When a user successfully signs in with a smart card, the smart card credential provider captures the reader name. This information is then stored in the registry with the session identifier where the sign in was initiated.

2. The smart card resource manager service notifies the smart card removal policy service that a sign-in has occurred.

3. ScPolicySvc retrieves the smart card information that the smart card credential provider stored in the registry. This call is redirected if the user is in a remote session. If the smart card is removed, ScPolicySvc is notified.

4. ScPolicySvc calls Remote Desktop Services to take the appropriate action if the request is to sign out the user or to disconnect the user's session, which might result in data loss. If the setting is configured to lock the computer when the smart card is removed, ScPolicySvc sends a message to Winlogon to lock the

computer.

See also

How Smart Card Sign-in Works in Windows

Smart Card Tools and Settings

3/5/2021 • 2 minutes to read • Edit Online

Applies To: Windows 10, Windows Server 2016

This topic for the IT professional and smart card developer links to information about smart card debugging, settings, and events.

This section of the Smart Card Technical Reference contains information about the following:

- Smart Cards Debugging Information: Learn about tools and services in supported versions of Windows to help identify certificate issues.
- Smart Card Group Policy and Registry Settings: Learn about smart card-related Group Policy settings and registry keys that can be set on a per-computer basis, including how to edit and apply Group Policy settings to local or domain computers.
- Smart Card Events: Learn about events that can be used to manage smart cards in an organization, including how to monitor installation, use, and errors.

See also

Smart Card Technical Reference

Smart Card Troubleshooting

5/7/2021 • 5 minutes to read • Edit Online

Applies To: Windows 10, Windows Server 2016

This article explains tools and services that smart card developers can use to help identify certificate issues with the smart card deployment.

Debugging and tracing smart card issues requires a variety of tools and approaches. The following sections provide guidance about tools and approaches you can use.

- Certutil
- Debugging and tracing using Windows software trace preprocessor (WPP)
- Kerberos protocol, Key Distribution Center (KDC), and NTLM debugging and tracing
- Smart Card service
- Smart card readers
- CryptoAPI 2.0 Diagnostics

Certutil

For a complete description of Certutil including examples that show how to use it, see Certutil [W2012].

List certificates available on the smart card

To list certificates that are available on the smart card, type `certutil -scinfo`.

> **NOTE**
> Entering a PIN is not required for this operation. You can press ESC if you are prompted for a PIN.

Delete certificates on the smart card

Each certificate is enclosed in a container. When you delete a certificate on the smart card, you're deleting the container for the certificate.

To find the container value, type `certutil -scinfo`.

To delete a container, type **certutil -delkey -csp "Microsoft Base Smart Card Crypto Provider"** "<*ContainerValue*>".

Debugging and tracing using WPP

WPP simplifies tracing the operation of the trace provider. It provides a mechanism for the trace provider to log real-time binary messages. Logged messages can be converted to a human-readable trace of the operation. For more information, see Diagnostics with WPP - The NDIS blog.

Enable the trace

Using WPP, use one of the following commands to enable tracing:

- **tracelog.exe -kd -rt -start** <*FriendlyName*> **-guid #**<*GUID*> **-f .**<*LogFileName*>**.etl -flags** <*flags*> **-ft 1**

- **logman start** <*FriendlyName*> **-ets -p {**<*GUID*>**} -**<*Flags*> **-ft 1 -rt -o .**<*LogFileName*>***.etl -mode 0x00080000***

You can use the parameters in the following table.

FRIENDLY NAME	GUID	FLAGS
`scardsvr`	13038e47-ffec-425d-bc69-5707708075fe	0xffff
`winscard`	3fce7c5f-fb3b-4bce-a9d8-55cc0ce1cf01	0xffff
`basecsp`	133a980d-035d-4e2d-b250-94577ad8fced	0x7
`scksp`	133a980d-035d-4e2d-b250-94577ad8fced	0x7
`msclmd`	fb36caf4-582b-4604-8841-9263574c4f2c	0x7
`credprov`	dba0e0e0-505a-4ab6-aa3f-22f6f743b480	0xffff
`certprop`	30eae751-411f-414c-988b-a8bfa8913f49	0xffff
`scfilter`	eed7f3c9-62ba-400e-a001-658869df9a91	0xffff
`wudfusbccid`	a3c09ba3-2f62-4be5-a50f-8278a646ac9d	0xffff

Examples

To enable tracing for the SCardSvr service:

- **tracelog.exe -kd -rt -start scardsvr -guid #13038e47-ffec-425d-bc69-5707708075fe -f .\scardsvr.etl -flags 0xffff -ft 1**
- **logman start scardsvr -ets -p {13038e47-ffec-425d-bc69-5707708075fe} 0xffff -ft 1 -rt -o .\scardsvr.etl -mode 0x00080000**

To enable tracing for scfilter.sys:

- **tracelog.exe -kd -rt -start scfilter -guid #eed7f3c9-62ba-400e-a001-658869df9a91 -f .\scfilter.etl -flags 0xffff -ft 1**

Stop the trace

Using WPP, use one of the following commands to stop the tracing:

- **tracelog.exe -stop** <*FriendlyName*>
- **logman -stop** <*FriendlyName*> **-ets**

Examples

To stop a trace:

- **tracelog.exe -stop scardsvr**
- **logman -stop scardsvr -ets**

Kerberos protocol, KDC, and NTLM debugging and tracing

You can use these resources to troubleshoot these protocols and the KDC:

- Kerberos and LDAP Troubleshooting Tips.
- Windows Driver Kit (WDK) and Debugging Tools for Windows (WinDbg). You can use the trace log tool in this SDK to debug Kerberos authentication failures.

To begin tracing, you can use `Tracelog`. Different components use different control GUIDs as explained in these examples. For more information, see `Tracelog`.

NTLM

To enable tracing for NTLM authentication, run the following command on the command line:

- **tracelog.exe -kd -rt -start ntlm -guid #5BBB6C18-AA45-49b1-A15F-085F7ED0AA90 -f .\ntlm.etl -flags 0x15003 -ft 1**

To stop tracing for NTLM authentication, run this command:

- **tracelog -stop ntlm**

Kerberos authentication

To enable tracing for Kerberos authentication, run this command:

- **tracelog.exe -kd -rt -start kerb -guid #6B510852-3583-4e2d-AFFE-A67F9F223438 -f .\kerb.etl -flags 0x43 -ft 1**

To stop tracing for Kerberos authentication, run this command:

- **tracelog.exe -stop kerb**

KDC

To enable tracing for the KDC, run the following command on the command line:

- **tracelog.exe -kd -rt -start kdc -guid #1BBA8B19-7F31-43c0-9643-6E911F79A06B -f .\kdc.etl -flags 0x803 -ft 1**

To stop tracing for the KDC, run the following command on the command line:

- **tracelog.exe -stop kdc**

To stop tracing from a remote computer, run this command: logman.exe -s *<ComputerName>*.

> **NOTE**
> The default location for logman.exe is %systemroot%system32\. Use the **-s** option to supply a computer name.

Configure tracing with the registry

You can also configure tracing by editing the Kerberos registry values shown in the following table.

ELEMENT	REGISTRY KEY SETTING

ELEMENT	REGISTRY KEY SETTING
NTLM	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\MSV1_0 Value name: NtLmInfoLevel Value type: DWORD Value data: c0015003
Kerberos	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\Kerberos Value name: LogToFile Value type: DWORD Value data: 00000001 HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\Kerberos\Parameters Value name: KerbDebugLevel Value type: DWORD Value data: c0000043 HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\Kerberos\Parameters Value name: LogToFile Value type: DWORD Value data: 00000001
KDC	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Kdc Value name: KdcDebugLevel Value type: DWORD Value data: c0000803

If you used `Tracelog`, look for the following log file in your current directory: kerb.etl/kdc.etl/ntlm.etl.

If you used the registry key settings shown in the previous table, look for the trace log files in the following locations:

- NTLM: %systemroot%\tracing\msv1_0
- Kerberos: %systemroot%\tracing\kerberos
- KDC: %systemroot%\tracing\kdcsvc

To decode event trace files, you can use `Tracefmt` (tracefmt.exe). `Tracefmt` is a command-line tool that formats and displays trace messages from an event trace log file (.etl) or a real-time trace session. `Tracefmt` can display the messages in the Command Prompt window or save them in a text file. It is located in the \tools\tracing subdirectory of the Windows Driver Kit (WDK). For more information, see `Tracefmt`.

Smart Card service

The smart card resource manager service runs in the context of a local service. It's implemented as a shared service of the services host (svchost) process.

To check if Smart Card service is running

1. Press CTRL+ALT+DEL, and then select **Start Task Manager**.
2. In the **Windows Task Manager** dialog box, select the **Services** tab.
3. Select the **Name** column to sort the list alphabetically, and then type **s**.

4. In the **Name** column, look for **SCardSvr**, and then look under the **Status** column to see if the service is running or stopped.

To restart Smart Card service

1. Run as administrator at the command prompt.
2. If the **User Account Control** dialog box appears, confirm that the action it displays is what you want, and then select **Yes**.
3. At the command prompt, type `net stop SCardSvr`.
4. At the command prompt, type `net start SCardSvr`.

You can use the following command at the command prompt to check whether the service is running: `sc queryex scardsvr`.

The following code sample is an example output from this command:

```
SERVICE_NAME: scardsvr
    TYPE        : 20 WIN32_SHARE_PROCESS
    STATE       : 4 RUNNING
                  (STOPPABLE, NOT_PAUSABLE, ACCEPTS_SHUTDOWN)
    WIN32_EXIT_CODE  : 0 (0x0)
    SERVICE_EXIT_CODE : 0 (0x0)
    CHECKPOINT     : 0x0
    WAIT_HINT     : 0x0
    PID        : 1320
    FLAGS       :
C:\>
```

Smart card readers

As with any device connected to a computer, Device Manager can be used to view properties and begin the debug process.

To check if smart card reader is working

1. Navigate to **Computer**.
2. Right-click **Computer**, and then select **Properties**.
3. Under **Tasks**, select **Device Manager**.
4. In Device Manager, expand **Smart card readers**, select the name of the smart card reader you want to check, and then select **Properties**.

> **NOTE**
> If the smart card reader is not listed in Device Manager, in the **Action** menu, select **Scan for hardware changes**.

CryptoAPI 2.0 Diagnostics

CryptoAPI 2.0 Diagnostics is available in Windows versions that support CryptoAPI 2.0 and can help you troubleshoot public key infrastructure (PKI) issues.

CryptoAPI 2.0 Diagnostics logs events in the Windows event log. The logs contain detailed information about certificate chain validation, certificate store operations, and signature verification. This information makes it easier to identify the causes of issues and reduces the time required for diagnosis.

For more information about CryptoAPI 2.0 Diagnostics, see Troubleshooting an Enterprise PKI.

See also

Smart Card Technical Reference

Smart Card Group Policy and Registry Settings

3/5/2021 • 20 minutes to read • Edit Online

Applies to: Windows 10, Windows Server 2016

This article for IT professionals and smart card developers describes the Group Policy settings, registry key settings, local security policy settings, and credential delegation policy settings that are available for configuring smart cards.

The following sections and tables list the smart card-related Group Policy settings and registry keys that can be set on a per-computer basis. If you use domain Group Policy Objects (GPOs), you can edit and apply Group Policy settings to local or domain computers.

Primary Group Policy settings for smart cards

The following smart card Group Policy settings are in Computer Configuration\Administrative Templates\Windows Components\Smart Card.

The registry keys are in the following locations:

- HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\ScPnP\EnableScPnP
- HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\SmartCardCredentialProvider
- HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\CertProp

> **NOTE**
>
> Smart card reader registry information is in HKEY_LOCAL_MACHINE\Software\Microsoft\Cryptography\Calais\Readers.
> Smart card registry information is in HKEY_LOCAL_MACHINE\Software\Microsoft\Cryptography\Calais\SmartCards.

The following table lists the default values for these GPO settings. Variations are documented under the policy descriptions in this article.

SERVER TYPE OR GPO	DEFAULT VALUE
Default Domain Policy	Not configured
Default Domain Controller Policy	Not configured
Stand-Alone Server Default Settings	Not configured
Domain Controller Effective Default Settings	Disabled
Member Server Effective Default Settings	Disabled
Client Computer Effective Default Settings	Disabled

Allow certificates with no extended key usage certificate attribute

You can use this policy setting to allow certificates without an enhanced key usage (EKU) set to be used for sign in.

> **NOTE**
>
> Enhanced key usage certificate attribute is also known as extended key usage.
>
> In versions of Windows before Windows Vista, smart card certificates that are used to sign in require an EKU extension with a smart card logon object identifier. This policy setting can be used to modify that restriction.

When this policy setting is turned on, certificates with the following attributes can also be used to sign in with a smart card:

- Certificates with no EKU
- Certificates with an All Purpose EKU
- Certificates with a Client Authentication EKU

When this policy setting isn't turned on, only certificates that contain the smart card logon object identifier can be used to sign in with a smart card.

ITEM	DESCRIPTION
Registry key	AllowCertificatesWithNoEKU
Default values	No changes per operating system versions Disabled and not configured are equivalent
Policy management	Restart requirement: None Sign off requirement: None Policy conflicts: None
Notes and resources	

Allow ECC certificates to be used for logon and authentication

You can use this policy setting to control whether elliptic curve cryptography (ECC) certificates on a smart card can be used to sign in to a domain.

When this setting is turned on, ECC certificates on a smart card can be used to sign in to a domain.

When this setting isn't turned on, ECC certificates on a smart card can't be used to sign in to a domain.

ITEM	DESCRIPTION
Registry key	**EnumerateECCCerts**
Default values	No changes per operating system versions Disabled and not configured are equivalent
Policy management	Restart requirement: None Sign off requirement: None Policy conflicts: None
Notes and resources	This policy setting only affects a user's ability to sign in to a domain. ECC certificates on a smart card that are used for other applications, such as document signing, aren't affected by this policy setting. If you use an ECDSA key to sign in, you must also have an associated ECDH key to permit sign in when you're not connected to the network.

Allow Integrated Unblock screen to be displayed at the time of logon

You can use this policy setting to determine whether the integrated unblock feature is available in the sign-in user interface (UI). The feature was introduced as a standard feature in the Credential Security Support Provider in Windows Vista.

When this setting is turned on, the integrated unblock feature is available.

When this setting isn't turned on, the feature is not available.

ITEM	DESCRIPTION
Registry key	**AllowIntegratedUnblock**
Default values	No changes per operating system versions Disabled and not configured are equivalent

ITEM	DESCRIPTION
Policy management	Restart requirement: None Sign off requirement: None Policy conflicts: None
Notes and resources	To use the integrated unblock feature, the smart card must support it. Check with the hardware manufacturer to verify that the smart card supports this feature. You can create a custom message that the user sees when the smart card is blocked by configuring the policy setting Display string when smart card is blocked.

Allow signature keys valid for Logon

You can use this policy setting to allow signature key–based certificates to be enumerated and available for sign in.

When this setting is turned on, any certificates that are available on the smart card with a signature-only key are listed on the sign-in screen.

When this setting isn't turned on, certificates available on the smart card with a signature-only key aren't listed on the sign-in screen.

ITEM	DESCRIPTION
Registry key	**AllowSignatureOnlyKeys**
Default values	No changes per operating system versions Disabled and not configured are equivalent
Policy management	Restart requirement: None Sign off requirement: None Policy conflicts: None
Notes and resources	

Allow time invalid certificates

You can use this policy setting to permit certificates that are expired or not yet valid to be displayed for sign in.

> **NOTE**
> Before Windows Vista, certificates were required to contain a valid time and to not expire. For a certificate to be used, it must be accepted by the domain controller. This policy setting only controls which certificates are displayed on the client computer.

When this setting is turned on, certificates are listed on the sign-in screen whether they have an invalid time, or their time validity has expired.

When this policy setting isn't turned on, certificates that are expired or not yet valid aren't listed on the sign-in screen.

ITEM	DESCRIPTION
Registry key	**AllowTimeInvalidCertificates**

ITEM	DESCRIPTION
Default values	No changes per operating system versions Disabled and not configured are equivalent
Policy management	Restart requirement: None Sign off requirement: None Policy conflicts: None
Notes and resources	

Allow user name hint

You can use this policy setting to determine whether an optional field appears during sign in and provides a subsequent elevation process where users can enter their username or username and domain, which associates a certificate with the user.

When this policy setting is turned on, users see an optional field where they can enter their username or username and domain.

When this policy setting isn't turned on, users don't see this optional field.

ITEM	DESCRIPTION
Registry key	**X509HintsNeeded**
Default values	No changes per operating system versions Disabled and not configured are equivalent
Policy management	Restart requirement: None Sign off requirement: None Policy conflicts: None
Notes and resources	

Configure root certificate clean up

You can use this policy setting to manage the cleanup behavior of root certificates. Certificates are verified by using a trust chain, and the trust anchor for the digital certificate is the Root Certification Authority (CA). A CA can issue multiple certificates with the root certificate as the top certificate of the tree structure. A private key is used to sign other certificates. This creates an inherited trustworthiness for all certificates immediately under the root certificate.

When this policy setting is turned on, you can set the following cleanup options:

- **No cleanup**. When the user signs out or removes the smart card, the root certificates used during their session persist on the computer.
- **Clean up certificates on smart card removal**. When the smart card is removed, the root certificates are removed.
- **Clean up certificates on log off**. When the user signs out of Windows, the root certificates are removed.

When this policy setting isn't turned on, root certificates are automatically removed when the user signs out of Windows.

ITEM	DESCRIPTION
Registry key	**RootCertificateCleanupOption**
Default values	No changes per operating system versions Disabled and not configured are equivalent
Policy management	Restart requirement: None Sign off requirement: None Policy conflicts: None
Notes and resources	

Display string when smart card is blocked

You can use this policy setting to change the default message that a user sees if their smart card is blocked.

When this policy setting is turned on, you can create and manage the displayed message that the user sees when a smart card is blocked.

When this policy setting isn't turned on (and the integrated unblock feature is also enabled), the user sees the system's default message when the smart card is blocked.

ITEM	DESCRIPTION
Registry key	**IntegratedUnblockPromptString**
Default values	No changes per operating system versions Disabled and not configured are equivalent
Policy management	Restart requirement: None Sign off requirement: None Policy conflicts: This policy setting is only effective when the Allow Integrated Unblock screen to be displayed at the time of logon policy is enabled.
Notes and resources	

Filter duplicate logon certificates

You can use this policy setting to configure which valid sign-in certificates are displayed.

> **NOTE**
>
> During the certificate renewal period, a user's smart card can have multiple valid sign-in certificates issued from the same certificate template, which can cause confusion about which certificate to select. This behavior can occur when a certificate is renewed and the old certificate has not expired yet.
>
> If two certificates are issued from the same template with the same major version and they are for the same user (this is determined by their UPN), they are determined to be the same.

When this policy setting is turned on, filtering occurs so that the user can select from only the most current valid certificates.

If this policy setting isn't turned on, all the certificates are displayed to the user.

This policy setting is applied to the computer after the Allow time invalid certificates policy setting is applied.

ITEM	DESCRIPTION
Registry key	**FilterDuplicateCerts**
Default values	No changes per operating system versions Disabled and not configured are equivalent
Policy management	Restart requirement: None Sign off requirement: None Policy conflicts: None
Notes and resources	If there are two or more of the same certificates on a smart card and this policy setting is enabled, the certificate that is used to sign in to computers running Windows 2000, Windows XP, or Windows Server 2003 will be displayed. Otherwise, the certificate with the most distant expiration time will be displayed.

Force the reading of all certificates from the smart card

You can use this policy setting to manage how Windows reads all certificates from the smart card for sign in. During sign in, Windows reads only the default certificate from the smart card unless it supports retrieval of all certificates in a single call. This policy setting forces Windows to read all the certificates from the smart card.

When this policy setting is turned on, Windows attempts to read all certificates from the smart card, regardless of the CSP feature set.

When this policy isn't turned on, Windows attempts to read only the default certificate from smart cards that don't support retrieval of all certificates in a single call. Certificates other than the default aren't available for sign in.

ITEM	DESCRIPTION
Registry key	**ForceReadingAllCertificates**
Default values	No changes per operating system versions Disabled and not configured are equivalent
Policy management	Restart requirement: None Sign off requirement: None Policy conflicts: None **Important**: Enabling this policy setting can adversely impact performance during the sign in process in certain situations.
Notes and resources	Contact the smart card vendor to determine if your smart card and associated CSP support the required behavior.

Notify user of successful smart card driver installation

You can use this policy setting to control whether the user sees a confirmation message when a smart card device driver is installed.

When this policy setting is turned on, the user sees a confirmation message when a smart card device driver is installed.

When this setting isn't turned on, the user doesn't see a smart card device driver installation message.

ITEM	DESCRIPTION
Registry key	**ScPnPNotification**
Default values	No changes per operating system versions Disabled and not configured are equivalent
Policy management	Restart requirement: None Sign off requirement: None Policy conflicts: None
Notes and resources	This policy setting applies only to smart card drivers that have passed the Windows Hardware Quality Labs (WHQL) testing process.

Prevent plaintext PINs from being returned by Credential Manager

You can use this policy setting to prevent Credential Manager from returning plaintext PINs.

> **NOTE**
> Credential Manager is controlled by the user on the local computer, and it stores credentials from supported browsers and Windows applications. Credentials are saved in special encrypted folders on the computer under the user's profile.

When this policy setting is turned on, Credential Manager doesn't return a plaintext PIN.

When this setting isn't turned on, Credential Manager can return plaintext PINs.

ITEM	DESCRIPTION
Registry key	**DisallowPlaintextPin**
Default values	No changes per operating system versions Disabled and not configured are equivalent
Policy management	Restart requirement: None Sign off requirement: None Policy conflicts: None
Notes and resources	If this policy setting is enabled, some smart cards might not work in computers running Windows. Consult the smart card manufacturer to determine whether this policy setting should be enabled.

Reverse the subject name stored in a certificate when displaying

You can use this policy setting to control the way the subject name appears during sign in.

> **NOTE**
> To help users distinguish one certificate from another, the user principal name (UPN) and the common name are displayed by default. For example, when this setting is enabled, if the certificate subject is CN=User1, OU=Users, DN=example, DN=com and the UPN is user1@example.com, "User1" is displayed with "user1@example.com." If the UPN is not present, the entire subject name is displayed. This setting controls the appearance of that subject name, and it might need to be adjusted for your organization.

When this policy setting is turned on, the subject name during sign in appears reversed from the way that it's stored in the certificate.

When this policy setting isn't turned on, the subject name appears the same as it's stored in the certificate.

ITEM	DESCRIPTION
Registry key	**ReverseSubject**
Default values	No changes per operating system versions Disabled and not configured are equivalent
Policy management	Restart requirement: None Sign off requirement: None Policy conflicts: None
Notes and resources	

Turn on certificate propagation from smart card

You can use this policy setting to manage the certificate propagation that occurs when a smart card is inserted.

> **NOTE**
> The certificate propagation service applies when a signed-in user inserts a smart card in a reader that is attached to the computer. This action causes the certificate to be read from the smart card. The certificates are then added to the user's Personal store.

When this policy setting is turned on, certificate propagation occurs when the user inserts the smart card.

When this policy setting is turned off, certificate propagation doesn't occur, and the certificates aren't available to applications, like Outlook.

ITEM	DESCRIPTION
Registry key	**CertPropEnabled**
Default values	No changes per operating system versions Enabled and not configured are equivalent
Policy management	Restart requirement: None Sign off requirement: None Policy conflicts: This policy setting must be enabled to allow the Turn on root certificate propagation from smart card setting to work when it is enabled.
Notes and resources	

Turn on root certificate propagation from smart card

You can use this policy setting to manage the root certificate propagation that occurs when a smart card is inserted.

> **NOTE**
> The certificate propagation service applies when a signed-in user inserts a smart card in a reader that is attached to the computer. This action causes the certificate to be read from the smart card. The certificates are then added to the user's Personal store.

When this policy setting is turned on, root certificate propagation occurs when the user inserts the smart card.

When this policy setting isn't turned on, root certificate propagation doesn't occur when the user inserts the smart card.

ITEM	DESCRIPTION
Registry key	**EnableRootCertificate Propagation**
Default values	No changes per operating system versions Enabled and not configured are equivalent
Policy management	Restart requirement: None Sign off requirement: None Policy conflicts: For this policy setting to work, the Turn on certificate propagation from smart card policy setting must also be enabled.
Notes and resources	

Turn on Smart Card Plug and Play service

You can use this policy setting to control whether Smart Card Plug and Play is enabled.

> **NOTE**
> Your users can use smart cards from vendors who have published their drivers through Windows Update without needing special middleware. These drivers will be downloaded in the same way as drivers for other devices in Windows. If an appropriate driver isn't available from Windows Update, a PIV-compliant mini driver that's included with any of the supported versions of Windows is used for these cards.

When this policy setting is turned on, the system attempts to install a smart card device driver the first time a smart card is inserted in a smart card reader.

When this policy setting isn't turned on, a device driver isn't installed when a smart card is inserted in a smart card reader.

ITEM	DESCRIPTION
Registry key	**EnableScPnP**
Default values	No changes per operating system versions Enabled and not configured are equivalent
Policy management	Restart requirement: None Sign off requirement: None Policy conflicts: None

ITEM	DESCRIPTION
Notes and resources	This policy setting applies only to smart card drivers that have passed the Windows Hardware Quality Labs (WHQL) testing process.

Base CSP and Smart Card KSP registry keys

The following registry keys can be configured for the base cryptography service provider (CSP) and the smart card key storage provider (KSP). The following tables list the keys. All keys use the DWORD type.

The registry keys for the Base CSP are in the registry in **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography\Defaults\Provider\Microsoft Base Smart Card Crypto Provider**.

The registry keys for the smart card KSP are in **HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Cryptography\Providers\Microsoft Smart Card Key Storage Provider**.

Registry keys for the base CSP and smart card KSP

REGISTRY KEY	DESCRIPTION
AllowPrivateExchangeKeyImport	A non-zero value allows RSA exchange (for example, encryption) private keys to be imported for use in key archival scenarios. Default value: 00000000
AllowPrivateSignatureKeyImport	A non-zero value allows RSA signature private keys to be imported for use in key archival scenarios. Default value: 00000000
DefaultPrivateKeyLenBits	Defines the default length for private keys, if desired. Default value: 00000400 Default key generation parameter: 1024-bit keys
RequireOnCardPrivateKeyGen	This key sets the flag that requires on-card private key generation (default). If this value is set, a key generated on a host can be imported into the smart card. This is used for smart cards that don't support on-card key generation or where key escrow is required. Default value: 00000000
TransactionTimeoutMilliseconds	Default timeout values allow you to specify whether transactions that take an excessive amount of time will fail. Default value: 000005dc1500 The default timeout for holding transactions to the smart card is 1.5 seconds.

Additional registry keys for the smart card KSP

REGISTRY KEY	DESCRIPTION
AllowPrivateECDHEKeyImport	This value allows Ephemeral Elliptic Curve Diffie-Hellman (ECDHE) private keys to be imported for use in key archival scenarios. Default value: 00000000

REGISTRY KEY	DESCRIPTION
AllowPrivateECDSAKeyImport	This value allows Elliptic Curve Digital Signature Algorithm (ECDSA) private keys to be imported for use in key archival scenarios. Default value: 00000000

CRL checking registry keys

The following table lists the keys and the corresponding values to turn off certificate revocation list (CRL) checking at the Key Distribution Center (KDC) or client. To manage CRL checking, you must configure settings for both the KDC and the client.

CRL checking registry keys

REGISTRY KEY	DETAILS
HKEY_LOCAL_MACHINE\SYSTEM\CCS\Services\Kdc\UseCachedCRLOnlyAndIgnoreRevocationUnknownErrors	Type = DWORD Value = 1
HKEY_LOCAL_MACHINE\SYSTEM\CCS\Control\LSA\Kerberos\Parameters\UseCachedCRLOnlyAndIgnoreRevocationUnknownErrors	Type = DWORD Value = 1

Additional smart card Group Policy settings and registry keys

In a smart card deployment, additional Group Policy settings can be used to enhance ease-of-use or security. Two of these policy settings that can complement a smart card deployment are:

- Turning off delegation for computers
- Interactive logon: Do not require CTRL+ALT+DEL (not recommended)

The following smart card-related Group Policy settings are in Computer Configuration\Windows Settings\Security Settings\Local Policies\Security Options.

Local security policy settings

GROUP POLICY SETTING AND REGISTRY KEY	DEFAULT	DESCRIPTION
Interactive logon: Require smart card **scforceoption**	Disabled	This security policy setting requires users to sign in to a computer by using a smart card. **Enabled** Users can sign in to the computer only by using a smart card. **Disabled** Users can sign in to the computer by using any method.

GROUP POLICY SETTING AND REGISTRY KEY	DEFAULT	DESCRIPTION
Interactive logon: Smart card removal behavior **scremoveoption**	This policy setting isn't defined, which means that the system treats it as **No Action**.	This setting determines what happens when the smart card for a signed-in user is removed from the smart card reader. The options are: **No Action** **Lock Workstation**: The workstation is locked when the smart card is removed, so users can leave the area, take their smart card with them, and still maintain a protected session. **Force Logoff**: The user is automatically signed out when the smart card is removed. **Disconnect if a Remote Desktop Services session**: Removal of the smart card disconnects the session without signing out the user. The user can reinsert the smart card and resume the session later, or at another computer that's equipped with a smart card reader, without having to sign in again. If the session is local, this policy setting functions identically to the **Lock Workstation** option. **Note**: In earlier versions of Windows Server, Remote Desktop Services was called Terminal Services.

From the Local Security Policy Editor (secpol.msc), you can edit and apply system policies to manage credential delegation for local or domain computers.

The following smart card-related Group Policy settings are in Computer Configuration\Administrative Templates\System\Credentials Delegation.

Registry keys are in **HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Lsa\Credssp\PolicyDefaults**.

> **NOTE**
> In the following table, fresh credentials are those that you are prompted for when running an application.

Credential delegation policy settings

GROUP POLICY SETTING AND REGISTRY KEY	DEFAULT	DESCRIPTION

GROUP POLICY SETTING AND REGISTRY KEY	DEFAULT	DESCRIPTION
Allow Delegating Fresh Credentials **AllowFreshCredentials**	Not configured	This policy setting applies: When server authentication was achieved through a trusted X509 certificate or Kerberos protocol. To applications that use the CredSSP component (for example, Remote Desktop Services). **Enabled**: You can specify the servers where the user's fresh credentials can be delegated. **Not configured**: After proper mutual authentication, delegation of fresh credentials is permitted to Remote Desktop Services running on any computer. **Disabled**: Delegation of fresh credentials to any computer isn't permitted. **Note**: This policy setting can be set to one or more service principal names (SPNs). The SPN represents the target server where the user credentials can be delegated. A single wildcard character is permitted when specifying the SPN, for example: Use *TERMSRV/** for Remote Desktop Session Host (RD Session Host) running on any computer. Use *TERMSRV/host.humanresources.fabrikam.com* for RD Session Host running on the host.humanresources.fabrikam.com computer. Use *TERMSRV/*.humanresources.fabrikam.com* for RD Session Host running on all computers in .humanresources.fabrikam.com

GROUP POLICY SETTING AND REGISTRY KEY	DEFAULT	DESCRIPTION
Allow Delegating Fresh Credentials with NTLM-only Server Authentication **AllowFreshCredentialsWhenNTLMOnly**	Not configured	This policy setting applies: When server authentication was achieved by using NTLM. To applications that use the CredSSP component (for example, Remote Desktop). **Enabled**: You can specify the servers where the user's fresh credentials can be delegated. **Not configured**: After proper mutual authentication, delegation of fresh credentials is permitted to RD Session Host running on any computer (TERMSRV/*). **Disabled**: Delegation of fresh credentials isn't permitted to any computer. **Note**: This policy setting can be set to one or more SPNs. The SPN represents the target server where the user credentials can be delegated. A single wildcard character (*) is permitted when specifying the SPN. See the **Allow Delegating Fresh Credentials** policy setting description for examples.
Deny Delegating Fresh Credentials **DenyFreshCredentials**	Not configured	This policy setting applies to applications that use the CredSSP component (for example, Remote Desktop). **Enabled**: You can specify the servers where the user's fresh credentials can't be delegated. **Disabled** or **Not configured**: A server is not specified. **Note**: This policy setting can be set to one or more SPNs. The SPN represents the target server where the user credentials can't be delegated. A single wildcard character (*) is permitted when specifying the SPN. For examples, see the "Allow delegating fresh credentials" policy setting.

If you're using Remote Desktop Services with smart card logon, you can't delegate default and saved credentials. The registry keys in the following table, which are at **HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Lsa\Credssp\PolicyDefaults**, and the corresponding Group Policy settings are ignored.

REGISTRY KEY	CORRESPONDING GROUP POLICY SETTING
AllowDefaultCredentials	Allow Delegating Default Credentials

REGISTRY KEY	CORRESPONDING GROUP POLICY SETTING
AllowDefaultCredentialsWhenNTLMOnly	Allow Delegating Default Credentials with NTLM-only Server Authentication
AllowSavedCredentials	Allow Delegating Saved Credentials
AllowSavedCredentialsWhenNTLMOnly	Allow Delegating Saved Credentials with NTLM-only Server Authentication

See also

Smart Card Technical Reference

Smart Card Events

5/19/2021 • 13 minutes to read • Edit Online

Applies To: Windows 10, Windows Server 2016

This topic for the IT professional and smart card developer describes events that are related to smart card deployment and development.

A number of events can be used to monitor smart card activities on a computer, including installation, use, and errors. The following sections describe the events and information that can be used to manage smart cards in an organization.

- Smart card reader name
- Smart card warning events
- Smart card error events
- Smart card Plug and Play events

Smart card reader name

The Smart Card resource manager does not use the device name from Device Manager to describe a smart card reader. Instead, the name is constructed from three device attributes that are queried directly from the smart card reader driver.

The following three attributes are used to construct the smart card reader name:

- Vendor name
- Interface device type
- Device unit

The smart card reader device name is constructed in the form <*VendorName*> <*Type*> <*DeviceUnit*>. For example 'Contoso Smart Card Reader 0' is constructed from the following information:

- Vendor name: Contoso
- Interface device type: Smart Card Reader
- Device unit: 0

Smart card warning events

> **Note** IOCTL in the following table refers to input and output control.

EVENT ID	WARNING MESSAGE	DESCRIPTION

EVENT ID	WARNING MESSAGE	DESCRIPTION
620	Smart Card Resource Manager was unable to cancel IOCTL %3 for reader '%2': %1. The reader may no longer be responding. If this error persists, your smart card or reader may not be functioning correctly. %n%nCommand Header: %4	This occurs if the resource manager attempts to cancel a command to the smart card reader when the smart card service is shutting down or after a smart card is removed from the smart card reader and the command could not to be canceled. This can leave the smart card reader in an unusable state until it is removed from the computer or the computer is restarted. %1 = Windows error code %2 = Smart card reader name %3 = IOCTL being canceled %4 = First 4 bytes of the command that was sent to the smart card
619	Smart Card Reader '%2' has not responded to IOCTL %3 in %1 seconds. If this error persists, your smart card or reader may not be functioning correctly. %n%nCommand Header: %4	This occurs when a reader has not responded to an IOCTL after an unusually long period of time. Currently, this error is sent after a reader does not respond for 150 seconds. This can leave the smart card reader in an unusable state until it is removed from the computer or the computer is restarted. %1 = Number of seconds the IOCTL has been waiting %2 = Smart card reader name %3 = IOCTL sent %4 = First 4 bytes of the command that was sent to the smart card

Smart card error events

EVENT ID	ERROR MESSAGE	DESCRIPTION
202	Failed to initialize Server Application	An error occurred, and the service cannot initialize properly. Restarting the computer may resolve the issue.
203	Server Control has no memory for reader reference object.	This is an internal, unrecoverable error that indicates a failure in the smart card service. The most common cause is limited computer resources. Restarting the computer may resolve the issue.
204	Server Control failed to create shutdown event: %1	This is an internal, unrecoverable error that indicates a failure in the smart card service. The most common cause is limited computer resources. Restarting the computer may resolve the issue. %1 = Windows error code

EVENT ID	ERROR MESSAGE	DESCRIPTION
205	Reader object has duplicate name: %1	There are two smart card readers that have the same name. Remove the smart card reader that is causing this error message. %1 = Name of the smart card reader that is duplicated
206	Failed to create global reader change event.	This is an internal, unrecoverable error that indicates a failure in the smart card service. The most common cause is limited computer resources. Restarting the computer may resolve the issue.
401	Reader shutdown exception from eject smart card command	A smart card reader could not eject a smart card while the smart card reader was shutting down.
406	Reader object cannot Identify Device	A smart card reader did not properly respond to a request for information about the device, which is required for constructing the smart card reader name. The smart card reader will not be recognized by the service until it is removed from the computer and reinserted or until the computer is restarted.
502	Initialization of Service Status Critical Section failed	This is an internal, unrecoverable error that indicates a failure in the smart card service. The most common cause is limited computer resources. Restarting the computer may resolve the issue.
504	Resource Manager cannot create shutdown event flag: %1	This is an internal, unrecoverable error that indicates a failure in the smart card service. The most common cause is limited computer resources. Restarting the computer may resolve the issue. %1 = Windows error code
506	Smart Card Resource Manager failed to register service: %1	This is an internal, unrecoverable error that indicates a failure in the smart card service. The most common cause is limited computer resources. Restarting the computer may resolve the issue. %1 = Windows error code
506	Smart Card Resource Manager received unexpected exception from PnP event %1	An attempt to add a Plug and Play reader failed. The device may already be in use or may be defective. To resolve this error message, try to add the device again or restart the computer. %1 = The affected handle name

EVENT ID	ERROR MESSAGE	DESCRIPTION
507	No memory available for Service Status Critical Section	There is not enough system memory available. This prevents the service from managing the status. Restarting the computer may resolve the issue.
508	Smart Card Resource Manager received unexpected exception from PnP event %1	An attempt to add a Plug and Play reader failed. The device may already be in use or may be defective. To resolve this error message, try to add the device again or restart the computer. %1 = The affected handle name
509	Smart Card Resource Manager received unexpected exception from PnP event %1	An attempt to add a Plug and Play reader failed. The device may already be in use or may be defective. To resolve this error message, try to add the device again or restart the computer. %1 = The affected handle name
510	Smart Card Resource Manager received NULL handle from PnP event %1	An attempt to add a Plug and Play smart card reader failed. The device may already be in use or may be defective. To resolve this error message, try to add the device again or restart the computer. %1 = The affected handle name
511	Smart Card Resource Manager received unexpected exception from PnP event %1	An attempt to add a Plug and Play reader failed. The device may already be in use or may be defective. To resolve this error message, try to add the device again or restart the computer. %1 = The affected handle name
512	Smart Card Resource Manager received NULL handle from PnP event %1	An attempt to add a Plug and Play smart card reader failed. The device may already be in use or may be defective. To resolve this error message, try to add the device again or restart the computer. %1 = The affected handle name
513	Smart Card Resource Manager received unexpected exception from PnP event %1	An attempt to add a Plug and Play reader failed. The device may already be in use or may be defective. To resolve this error message, try to add the device again or restart the computer. %1 = The affected handle name

EVENT ID	ERROR MESSAGE	DESCRIPTION
514	Smart Card Resource Manager failed to add reader %2: %1	This is an internal, unrecoverable error that indicates a failure in the smart card service. The most common cause is limited computer resources. Restarting the computer may resolve the issue. %1 = Windows error code %2 = Smart card reader name
515	Smart Card Resource Manager failed to declare state: %1	This is an internal unrecoverable error that indicates a failure in the smart card service. The smart card service may not operate properly. Restarting the service or computer may resolve this issue. %1 = Windows error code
516	Smart Card Resource Manager Failed to declare shutdown: %1	This is an internal, unrecoverable error that indicates a failure in the smart card service. The smart card service may not be able to stop. Restarting the computer may resolve this issue. %1 = Windows error code
517	Smart Card Resource Manager received unexpected exception attempting to add reader %1	This is an internal, unrecoverable error that indicates a failure in the smart card service. The most common cause is limited computer resources. Restarting the computer may resolve the issue. %1 = Smart card reader name
521	Smart Card Resource Manager received NULL handle from PnP event %1	An attempt to add a Plug and Play smart card reader failed. The device may already be in use or may be defective. To resolve this error message, try to add the device again or restart the computer. %1 = The affected handle name
523	Smart Card Resource Manager received NULL handle from PnP event %1	An attempt to add a Plug and Play smart card reader failed. The device may already be in use or may be defective. To resolve this error message, try to add the device again or restart the computer. %1 = The affected handle name
602	WDM Reader driver initialization cannot open reader device: %1	The service cannot open a communication channel with the smart card reader. You cannot use the smart card reader until the issue is resolved. %1 = Windows error code

EVENT ID	ERROR MESSAGE	DESCRIPTION
603	WDM Reader driver initialization has no memory available to control device %1	There is not enough system memory available. This prevents the service from managing the smart card reader that was added. Restarting the computer may resolve the issue. %1 = Name of affected reader
604	Server control cannot set reader removal event: %1	This is an internal, unrecoverable error that indicates a failure in the smart card service. The most common cause is limited computer resources. Restarting the computer may resolve the issue. %1 = Windows error code
605	Reader object failed to create overlapped event: %1	This is an internal, unrecoverable error that indicates a failure in the smart card service. The most common cause is limited computer resources. Restarting the computer may resolve the issue. %1 = Windows error code
606	Reader object failed to create removal event: %1	This is an internal, unrecoverable error that indicates a failure in the smart card service. The most common cause is limited computer resources. Restarting the computer may resolve the issue. %1 = Windows error code
607	Reader object failed to start monitor thread: %1	This is an internal, unrecoverable error that indicates a failure in the smart card service. The most common cause is limited computer resources. Restarting the computer may resolve the issue. %1 = Windows error code
608	Reader monitor failed to create power down timer: %1	This is an internal, unrecoverable error that indicates a failure in the smart card service. The most common cause is limited computer resources. Restarting the computer may resolve the issue. %1 = Windows error code
609	Reader monitor failed to create overlapped event: %1	This is an internal, unrecoverable error that indicates a failure in the smart card service. The most common cause is limited computer resources. Restarting the computer may resolve the issue. %1 = Windows error code

EVENT ID	ERROR MESSAGE	DESCRIPTION
610	Smart Card Reader '%2' rejected IOCTL %3: %1 If this error persists, your smart card or reader may not be functioning correctly.%n%nCommand Header: %4	The reader cannot successfully transmit the indicated IOCTL to the smart card. This can indicate hardware failure, but this error can also occur if a smart card or smart card reader is removed from the system while an operation is in progress. %1 = Windows error code %2 = Name of the smart card reader %3 = IOCTL that was sent %4 = First 4 bytes of the command sent to the smart card These events are caused by legacy functionality in the smart card stack. It can be ignored if there is no noticeable failure in the smart card usage scenarios. You might also see this error if your eSIM is recognized as a smartcard controller.
611	Smart Card Reader initialization failed	This is an internal, unrecoverable error that indicates a failure in the smart card service. The most common cause is limited computer resources. Restarting the computer may resolve this issue.
612	Reader insertion monitor error retry threshold reached: %1	This occurs when a smart card reader fails several times to respond properly to the IOCTL, which indicates whether a smart card is present in the reader. The smart card reader is marked as defective, and it is not recognized by the service until it is removed from the computer and reinserted or until the computer is restarted. %1 = Windows error code
615	Reader removal monitor error retry threshold reached: %1	This occurs when a smart card reader fails several times to respond properly to the IOCTL, which indicates whether a smart card is present in the reader. The smart card reader is marked as defective, and it is not recognized by the service until it is removed from the computer and reinserted or until the computer is restarted. %1 = Windows error code

EVENT ID	ERROR MESSAGE	DESCRIPTION
616	Reader monitor '%2' received uncaught error code: %1	This occurs when a smart card reader fails several times to respond properly to the IOCTL, which indicates whether a smart card is present in the reader. The smart card reader is marked as defective, and it is not recognized by the service until it is removed from the computer and reinserted or until the computer is restarted. %1 = Windows error code %2 = Reader name
617	Reader monitor '%1' exception -- exiting thread	An unknown error occurred while monitoring a smart card reader for smart card insertions and removals. The smart card reader is marked as defective, and it is not recognized by the service until it is removed from the computer and reinserted or until the computer is restarted. %1 = Smart card reader name
618	Smart Card Resource Manager encountered an unrecoverable internal error.	This is an internal, unrecoverable error that indicates a failure in the smart card service. The most common cause is limited computer resources. Restarting the computer may resolve the issue.
621	Server Control failed to access start event: %1	This is an internal, unrecoverable error that indicates a failure in the smart card service. The most common cause is limited computer resources. Restarting the computer may resolve the issue. %1 = Windows error code These events are caused by legacy functionality in the smart card stack. It can be ignored if there is no noticeable failure in the smart card usage scenarios.
622	Server Control failed to access stop event: %1	This is an internal, unrecoverable error that indicates a failure in the smart card service. The most common cause is limited computer resources. Restarting the computer may resolve the issue. %1 = Windows error code

Smart card Plug and Play events

EVENT ID	EVENT TYPE	EVENT MESSAGE	DESCRIPTION

EVENT ID	EVENT TYPE	EVENT MESSAGE	DESCRIPTION
1000	Error	Could not get device ID for smart card in reader %1. The return code is %2.	Smart card Plug and Play could not obtain the device ID for the smart card. This information is required to determine the correct driver. The smart card may be defective. %1 = Smart card reader name %2 = Windows error code
1001	Information	Software successfully installed for smart card in reader %1. The smart card name is %2.	Smart card Plug and Play successfully installed a minidriver for the inserted card. %1 = Smart card reader name %2 = Name of new smart card device

See also

Smart Card Technical Reference

Virtual Smart Card Overview

3/26/2021 • 8 minutes to read • Edit Online

Applies To: Windows 10, Windows Server 2016

This topic for IT professional provides an overview of the virtual smart card technology that was developed by Microsoft and includes links to additional topics to help you evaluate, plan, provision, and administer virtual smart cards.

Did you mean…

- Smart Cards

> **NOTE**
> Windows Hello for Business is the modern, two-factor authentication for Windows 10. Microsoft will be deprecating virtual smart cards in the future, but no date has been set at this time. Customers using Windows 10 and virtual smart cards should move to Windows Hello for Business. Microsoft will publish the date early to ensure customers have adequate lead time to move to Windows Hello for Business. We recommend that new Windows 10 deployments use Windows Hello for Business. Virtual smart cards remain supported for Windows 7 and Windows 8.

Feature description

Virtual smart card technology from Microsoft offers comparable security benefits to physical smart cards by using two-factor authentication. Virtual smart cards emulate the functionality of physical smart cards, but they use the Trusted Platform Module (TPM) chip that is available on computers in many organizations, rather than requiring the use of a separate physical smart card and reader. Virtual smart cards are created in the TPM, where the keys that are used for authentication are stored in cryptographically secured hardware.

By utilizing TPM devices that provide the same cryptographic capabilities as physical smart cards, virtual smart cards accomplish the three key properties that are desired for smart cards: non-exportability, isolated cryptography, and anti-hammering.

Practical applications

Virtual smart cards are functionally similar to physical smart cards and appear in Windows as smart cards that are always-inserted. Virtual smart cards can be used for authentication to external resources, protection of data by secure encryption, and integrity through reliable signing. They are easily deployed by using in-house methods or a purchased solution, and they can become a full replacement for other methods of strong authentication in a corporate setting of any scale.

Authentication use cases

Two-factor authentication–based remote access

After a user has a fully functional TPM virtual smart card, provisioned with a sign-in certificate, the certificate is used to gain strongly authenticated access to corporate resources. When the proper certificate is provisioned to the virtual card, the user need only provide the PIN for the virtual smart card, as if it was a physical smart card, to sign in to the domain.

In practice, this is as easy as entering a password to access the system. Technically, it is far more secure. Using the virtual smart card to access the system proves to the domain that the user who is requesting authentication

has possession of the personal computer upon which the card has been provisioned and knows the virtual smart card PIN. Because this request could not have possibly originated from a system other than the system certified by the domain for this user's access, and the user could not have initiated the request without knowing the PIN, a strong two-factor authentication is established.

Client authentication

Virtual smart cards can also be used for client authentication by using Secure Socket Layer (SSL) or a similar technology. Similar to domain access with a virtual smart card, an authentication certificate can be provisioned for the virtual smart card, provided to a remote service, as requested in the client authentication process. This adheres to the principles of two-factor authentication because the certificate is only accessible from the computer that hosts the virtual smart card, and the user is required to enter the PIN for initial access to the card.

Virtual smart card redirection for remote desktop connections

The concept of two-factor authentication associated with virtual smart cards relies on the proximity of users to the computers that they access domain resources through. Therefore, when a user remotely connects to a computer that is hosting virtual smart cards, the virtual smart cards that are located on the remote computer cannot be used during the remote session. However, the virtual smart cards that are stored on the connecting computer (which is under physical control of the user) are loaded onto the remote computer, and they can be used as if they were installed by using the remote computer's TPM. This extends a user's privileges to the remote computer, while maintaining the principles of two-factor authentication.

Windows To Go and virtual smart cards

Virtual smart cards work well with Windows To Go, where a user can boot into a supported version of Windows from a compatible removable storage device. A virtual smart card can be created for the user, and it is tied to the TPM on the physical host computer to which the removable storage device is connected. When the user boots the operating system from a different physical computer, the virtual smart card will not be available. This can be used for scenarios when a single physical computer is shared by many users. Each user can be given a removable storage device for Windows To Go, which has a virtual smart card provisioned for the user. This way, users are only able to access their personal virtual smart card.

Confidentiality use cases

S/MIME email encryption

Physical smart cards are designed to hold private keys that can be used for email encryption and decryption. This functionality also exists in virtual smart cards. By using S/MIME with a user's public key to encrypt email, the sender of an email can be assured that only the person with the corresponding private key will be able to decrypt the email. This assurance is a result of the non-exportability of the private key. It never exists within reach of malicious software, and it remains protected by the TPM—even during decryption.

BitLocker for data volumes

sBitLocker Drive Encryption technology makes use of symmetric-key encryption to protect the content of a user's hard drive. This ensures that if the physical ownership of a hard drive is compromised, an adversary will not be able to read data off the drive. The key used to encrypt the drive can be stored in a virtual smart card, which necessitates knowledge of the virtual smart card PIN to access the drive and possession of the computer that is hosting the TPM virtual smart card. If the drive is obtained without access to the TPM that hosts the virtual smart card, any brute force attack will be very difficult.

BitLocker can also be used to encrypt portable drives, which involves storing keys in virtual smart cards. In this scenario (unlike using BitLocker with a physical smart card), the encrypted drive can be used only when it is connected to the host for the virtual smart card that is used to encrypt the drive, because the BitLocker key is only accessible from this computer. However, this method can be useful to ensure the security of backup drives and personal storage uses outside the main hard drive.

Data integrity use case

Signing data

To verify authorship of data, a user can sign it by using a private key that is stored in the virtual smart card. Digital signatures confirm the integrity and origin of the data. If the key is stored in an operating system that is accessible, a malicious user could access it and use it to modify already signed data or to spoof the key owner's identity. However, if this key is stored in a virtual smart card, it can be used only to sign data on the host computer. It cannot be exported to other systems (intentionally or unintentionally, such as with malware theft). This makes digital signatures far more secure than other methods for private key storage.

New and changed functionality as of Windows 8.1

Enhancements in Windows 8.1 enabled developers to build Microsoft Store apps to create and manage virtual smart cards.

The DCOM Interfaces for Trusted Platform Module (TPM) Virtual Smart Card device management protocol provides a Distributed Component Object Model (DCOM) Remote Protocol interface used for creating and destroying virtual smart cards. A virtual smart card is a device that presents a device interface complying with the PC/SC specification for PC-connected interface devices to its host operating system (OS) platform. This protocol does not assume anything about the underlying implementation of virtual smart card devices. In particular, while it is primarily intended for the management of virtual smart cards based on TPMs, it can also be used to manage other types of virtual smart cards.

What value does this change add?

Starting with Windows 8.1, application developers can build into their apps the following virtual smart card maintenance capabilities to relieve some of your administrative burdens.

- Create a new virtual smart card or select a virtual smart card from the list of available virtual smart cards on the system. Identify the one that the application is supposed to work with.
- Personalize the virtual smart card.
- Change the admin key.
- Diversify the admin key which allows the user to unblock the PIN in a PIN-blocked scenario.
- Change the PIN.
- Reset or Unblock the PIN.
- Destroy the virtual smart card.

What works differently?

Starting with Windows 8.1, Microsoft Store app developers are able to build apps that have the capability to prompt the user to reset or unblock and change a virtual smart card PIN. This places more responsibility on the user to maintain their virtual smart card but it can also provide a more consistent user experience and administration experience in your organization.

For more information about developing Microsoft Store apps with these capabilities, see Trusted Platform Module Virtual Smart Card Management Protocol.

For more information about managing these capabilities in virtual smart cards, see Understanding and Evaluating Virtual Smart Cards.

Hardware requirements

To use the virtual smart card technology, TPM 1.2 is the minimum required for computers running Windows 10

or Windows Server 2016.

Software requirements

To use the virtual smart card technology, computers must be running one of the following operating systems:

- Windows Server 2016
- Windows Server 2012 R2
- Windows Server 2012
- Windows 10
- Windows 8.1
- Windows 8

See also

- Understanding and Evaluating Virtual Smart Cards
- Get Started with Virtual Smart Cards: Walkthrough Guide
- Use Virtual Smart Cards
- Deploy Virtual Smart Cards
- Evaluate Virtual Smart Card Security
- Tpmvscmgr

Understanding and Evaluating Virtual Smart Cards

3/5/2021 • 13 minutes to read • Edit Online

Applies To: Windows 10, Windows Server 2016

This topic for the IT professional describes the virtual smart card technology that was developed by Microsoft; suggests how it can fit into your authentication design; and provides links to additional resources that you can use to design, deploy, and troubleshoot virtual smart cards.

Virtual smart card technology uses cryptographic keys that are stored on computers that have the Trusted Platform Module (TPM) installed. Virtual smart cards offer comparable security benefits to conventional smart cards by using two-factor authentication. The technology also offers more convenience for users and has a lower cost to deploy. By utilizing TPM devices that provide the same cryptographic capabilities as conventional smart cards, virtual smart cards accomplish the three key properties that are desired for smart cards: non-exportability, isolated cryptography, and anti-hammering.

Virtual smart cards are functionally similar to physical smart cards. They appear as always-inserted smart cards, and they can be used for authentication to external resources, protection of data by secure encryption, and integrity through reliable signing. Because TPM-enabled hardware is readily available and virtual smart cards can be easily deployed by using existing certificate enrollment methods, virtual smart cards can become a full replacement for other methods of strong authentication in a corporate setting of any scale.

This topic contains the following sections:

- Comparing virtual smart cards with physical smart cards: Compares properties, functional aspects, security, and cost.
- Authentication design options: Describes how passwords, smart cards, and virtual smart cards can be used to reach authentication goals in your organization.
- See also: Links to other topics that can help you design, deploy, and troubleshoot virtual smart cards.

Comparing virtual smart cards with physical smart cards

Virtual smart cards function much like physical smart cards, but they differ in that they protect private keys by using the TPM of the computer instead of smart card media.

A virtual smart card appears to applications as a conventional smart card. Private keys in the virtual smart card are protected, not by isolation of physical memory, but by the cryptographic capabilities of the TPM. All sensitive information is encrypted by using the TPM and then stored on the hard drive in its encrypted form.

All cryptographic operations occur in the secure, isolated environment of the TPM, and the unencrypted private keys are never used outside this environment. So like physical smart cards, virtual smart cards remain secure from any malware on the host. Additionally, if the hard drive is compromised in some way, a malicious user will not be able to access keys that are stored in the virtual smart card because they are securely encrypted by using the TPM. Keys can also be protected by BitLocker Drive Encryption.

Virtual smart cards maintain the three key properties of physical smart cards:

- **Non-exportability**: Because all private information on the virtual smart card is encrypted by using the TPM on the host computer, it cannot be used on a different computer with a different TPM. Additionally, TPMs are designed to be tamper-resistant and non-exportable, so a malicious user cannot reverse engineer an identical TPM or install the same TPM on a different computer. For more information, see

Evaluate Virtual Smart Card Security.

- **Isolated cryptography**: TPMs provide the same properties of isolated cryptography that are offered by physical smart cards, and this is utilized by virtual smart cards. Unencrypted copies of private keys are loaded only within the TPM and never into memory that is accessible by the operating system. All cryptographic operations with these private keys occur inside the TPM.
- **Anti-hammering**: If a user enters a PIN incorrectly, the virtual smart card responds by using the anti-hammering logic of the TPM, which rejects further attempts for a period of time instead of blocking the card. This is also known as lockout. For more information, see Evaluate Virtual Smart Card Security.

The following subsections compare the functionality, security, and cost of virtual smart cards and physical smart cards.

Functionality

The virtual smart card system that was designed by Microsoft closely mimics the functionality of conventional smart cards. The most striking difference to the end user is that the virtual smart card is essentially a smart card that is always inserted into the computer. There is no method to export the user's virtual smart card for use on other computers, which adds to the security of virtual smart cards. If a user requires access to network resources on multiple computers, multiple virtual smart cards can be issued for that user. Additionally, a computer that is shared among multiple users can host multiple virtual smart cards for different users.

The basic user experience for a virtual smart card is as simple as using a password to access a network. Because the smart card is loaded by default, the user must simply enter the PIN that is tied to the card to gain access. Users are no longer required to carry cards and readers or to take physical action to use the card.

Additionally, although the anti-hammering functionality of the virtual smart card is equally secure to that of a physical smart card, virtual smart card users are never required to contact an administrator to unblock the card. Instead, they simply wait a period of time (depending on the TPM specifications) before they reattempt to enter the PIN. Alternatively, the administrator can reset the lockout by providing owner authentication data to the TPM on the host computer.

Security

Physical smart cards and virtual smart cards offer comparable levels of security. They both implement two-factor authentication for using network resources. However, they differ in certain aspects, including physical security and the practicality of an attack. Due to their compact and portable design, conventional smart cards are most frequently kept close to their intended user. They offer little opportunity for acquisition by a potential adversary, so any sort of interaction with the card is difficult without committing some variety of theft.

TPM virtual smart cards, however, reside on a user's computer that may frequently be left unattended, which provides an opportunity for a malicious user to hammer the TPM. Although virtual smart cards are fully protected from hammering (as are physical smart cards), this accessibility makes the logistics of an attack somewhat simpler. Additionally, the anti-hammering behavior of a TPM smart card differs in that it only presents a time delay in response to repeated PIN failures, as opposed to fully blocking the user.

However, there are several advantages provided by virtual smart cards to mitigate these slight security deficits. Most importantly, a virtual smart card is much less likely to be lost. Virtual smart cards are integrated into computers and devices that the user already owns for other purposes and has incentive to keep safe. If the computer or device that hosts the virtual smart card is lost or stolen, a user will more immediately notice its loss than the loss of a physical smart card. When a computer or device is identified as lost, the user can notify the administrator of the system, who can revoke the certificate that is associated with the virtual smart card on that device. This precludes any future unauthorized access on that computer or device if the PIN for the virtual smart card is compromised.

Cost

If a company wants to deploy physical smart cards, they need to purchase smart cards and smart card readers for all employees. Although relatively inexpensive options can be found, options that ensure the three key properties of smart card security (most notably, non-exportability) are more expensive. If employees have computers with a built-in TPM, virtual smart cards can be deployed with no additional material costs. These computers and devices are relatively common in the market.

Additionally, the maintenance cost of virtual smart cards is less than that for physical smart cards, which are easily lost, stolen, or broken from normal wear. TPM virtual smart cards are only lost or broken if the host computer or device is lost or broken, which in most cases is much less frequently.

Comparison summary

PHYSICAL SMART CARDS	TPM VIRTUAL SMART CARDS
Protects private keys by using the built-in cryptographic functionality of the card.	Protects private keys by using the cryptographic functionality of the TPM.
Stores private keys in isolated non-volatile memory on the card, which means that access to private keys is only from the card, and access is never allowed to the operating system.	Stores encrypted private keys on the hard drive. The encryption ensures that these keys can only be decrypted and used in the TPM, not in the accessible memory of the operating system.
Guarantees non-exportability through the card manufacturer, which includes isolating private information from operating system access.	Guarantees non-exportability through the TPM manufacturer, which includes the inability of an adversary to replicate or remove the TPM.
Performs and isolates cryptographic operations within the built-in capabilities of the card.	Performs and isolates cryptographic operations in the TPM of the user's computer or device.
Provides anti-hammering through the card. After a certain number of failed PIN entry attempts, the card blocks further access until administrative action is taken.	Provides anti-hammering through the TPM. Successive failed attempts increase the device lockout time (the time the user has to wait before trying again). This can be reset by an administrator.
Requires that users carry their smart card and smart card reader with them to access network resources.	Allows users to access their TPM-enabled computers or devices, and potentially access the network, without additional equipment.
Enables credential portability by inserting the smart card into smart card readers that are attached to other computers.	Prevents exporting credentials from a given computer or device. However, virtual smart cards can be issued for the same user on multiple computers or devices by using additional certificates.
Enables multiple users to access network resources through the same computer by inserting their personal smart cards.	Enables multiple users to access network resources through the same computer or device by issuing a virtual smart card for each user on that computer or device.
Requires the user to carry the card, making it more difficult for an attacker to access the device and launch a hammering attempt.	Stores virtual smart card on the user's computer, which may be left unattended and allow a greater risk window for hammering attempts.
Provides a generally single-purpose device that is carried explicitly for the purpose of authentication. The smart card can be easily misplaced or forgotten.	Installs the virtual smart card on a device that has other purposes for the user, so the user has greater incentive to be responsible for the computer or device.

PHYSICAL SMART CARDS	TPM VIRTUAL SMART CARDS
Alerts users that their card is lost or stolen only when they need to sign in and notice it is missing.	Installs the virtual smart card on a device that the user likely needs for other purposes, so users will notice its loss much more quickly. This reduces the associated risk window.
Requires companies to invest in smart cards and smart card readers for all employees.	Requires that companies ensure all employees have TPM-enabled computers, which are relatively common.
Enables using a smart card removal policy to affect system behavior when the smart card is removed. For example, the policy can dictate if the user's sign-in session is locked or terminated when the user removes the card.	Eliminates the necessity for a smart card removal policy because a TPM virtual smart card is always present and cannot be removed from the computer.

Authentication design options

The following section presents several commonly used options and their respective strengths and weaknesses, which organizations can consider for authentication.

Passwords

A password is a secret string of characters that is tied to the identification credentials for a user's account. This establishes the user's identity. Although passwords are the most commonly used form of authentication, they are also the weakest. In a system where passwords are used as the sole method of user authentication, only individuals who know their passwords are considered valid users.

Password authentication places a great deal of responsibility on the user. Passwords must be sufficiently complex so they cannot be easily guessed, but they must be simple enough to be committed to memory and not stored in a physical location. Even if this balance is successfully achieved, a wide variety of attacks exist (such as brute force attacks, eavesdropping, and social engineering tactics) where a malicious user can acquire a user's password and impersonate that person's identity. A user often will not realize that the password is compromised, which makes it is easy for a malicious user to maintain access to a system if a valid password has been obtained.

One-time passwords

A one-time password (OTP) is similar to a traditional password, but it is more secure in that it can be used only once to authenticate a user. The method for determining each new password varies by implementation. However, assuming a secure deployment of each new password, OTPs have several advantages over the classic password model of authentication. Most importantly, if a given OTP token is intercepted in transmission between the user and the system, the interceptor cannot use it for any future transactions. Similarly, if a malicious user obtains a valid user's OTP, the interceptor will have limited access to the system (only one session).

Smart cards

Smart cards are physical authentication devices, which improve on the concept of a password by requiring that users actually have their smart card device with them to access the system, in addition to knowing the PIN that provides access to the smart card. Smart cards have three key properties that help maintain their security:

- **Non-exportability**: Information stored on the card, such as the user's private keys, cannot be extracted from one device and used in another medium.
- **Isolated cryptography**: Any cryptographic operations that are related to the card (such as secure encryption and decryption of data) occur in a cryptographic processor on the card, so malicious software on the host computer cannot observe the transactions.

- **Anti-hammering**: To prevent access to the card by a brute-force attack, a set number of consecutive unsuccessful PIN entry attempts blocks the card until administrative action is taken.

Smart cards provide greatly enhanced security over passwords alone, because it is much more difficult for a malicious user to gain and maintain access to a system. Most importantly, access to a smart card system requires that users have a valid card and that they know the PIN that provides access to that card. It is extremely difficult for a thief to acquire the card and the PIN.

Additional security is achieved by the singular nature of the card because only one copy of the card exists, only one individual can use the sign-in credentials, and users will quickly notice if the card has been lost or stolen. This greatly reduces the risk window of credential theft when compared to using a password alone.

Unfortunately, this additional security comes with added material and support costs. Traditional smart cards are expensive to purchase (cards and card readers must be supplied to employees), and they also can be easily misplaced or stolen.

Virtual smart cards

To address these issues, virtual smart cards emulate the functionality of traditional smart cards, but instead of requiring the purchase of additional hardware, they utilize technology that users already own and are more likely to have with them at all times. Theoretically, any device that can provide the three key properties of smart cards (non-exportability, isolated cryptography, and anti-hammering) can be commissioned as a virtual smart card. However, the virtual smart card platform developed by Microsoft is currently limited to the use of the Trusted Platform Module (TPM) chip, which is installed on most modern computers.

Virtual smart cards that utilize a TPM provide the three main security principles of traditional smart cards (non-exportability, isolated cryptography, and anti-hammering). They are also less expensive to implement and more convenient for users. Because many corporate computers already have a built-in TPM, there is no cost associated with purchasing new hardware. The user's possession of a computer or device is equivalent to the possession of a smart card, and a user's identity cannot be assumed from any other computer or device without administrative provisioning of further credentials. Thus, two-factor authentication is achieved because the user must have a computer that is set up with a virtual smart card and know the PIN to use the virtual smart card.

See also

- Get Started with Virtual Smart Cards: Walkthrough Guide
- Use Virtual Smart Cards
- Deploy Virtual Smart Cards
- Evaluate Virtual Smart Card Security

Get Started with Virtual Smart Cards: Walkthrough Guide

3/5/2021 • 5 minutes to read • Edit Online

Applies To: Windows 10, Windows Server 2016

This topic for the IT professional describes how to set up a basic test environment for using TPM virtual smart cards.

Virtual smart cards are a technology from Microsoft, which offer comparable security benefits in two-factor authentication to physical smart cards. They also offer more convenience for users and lower cost for organizations to deploy. By utilizing Trusted Platform Module (TPM) devices that provide the same cryptographic capabilities as physical smart cards, virtual smart cards accomplish the three key properties that are desired by smart cards: non-exportability, isolated cryptography, and anti-hammering.

This step-by-step walkthrough shows you how to set up a basic test environment for using TPM virtual smart cards. After you complete this walkthrough, you will have a functional virtual smart card installed on the Windows computer.

Time requirements

You should be able to complete this walkthrough in less than one hour, excluding installing software and setting up the test domain.

Walkthrough steps

- Prerequisites
- Step 1: Create the certificate template
- Step 2: Create the TPM virtual smart card
- Step 3: Enroll for the certificate on the TPM Virtual Smart Card

> **Important** This basic configuration is for test purposes only. It is not intended for use in a production environment.

Prerequisites

You will need:

- A computer running Windows 10 with an installed and fully functional TPM (version 1.2 or version 2.0).
- A test domain to which the computer listed above can be joined.
- Access to a server in that domain with a fully installed and running certification authority (CA).

Step 1: Create the certificate template

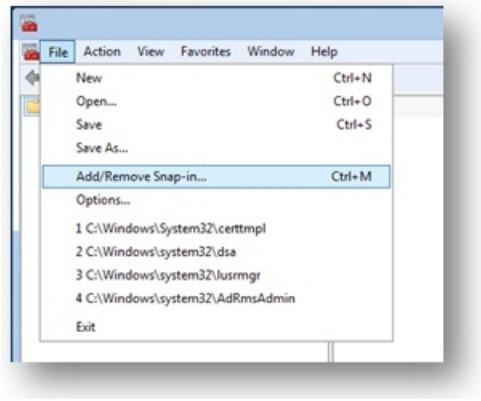
On your domain server, you need to create a template for the certificate that you will request for the virtual smart card.

To create the certificate template

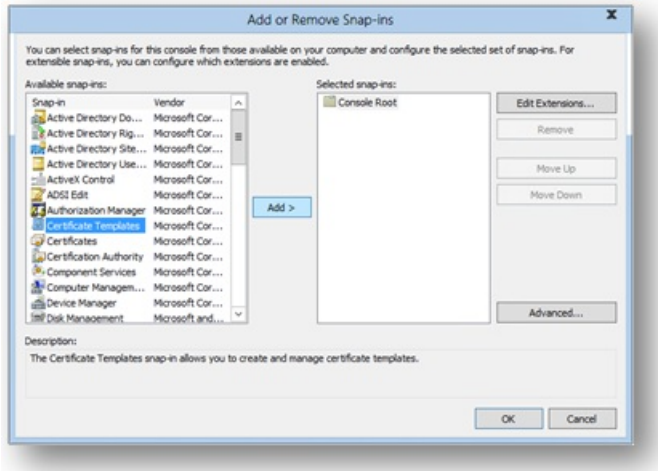
1. On your server, open the Microsoft Management Console (MMC). One way to do this is to type **mmc.exe**

from the **Start** menu, right-click **mmc.exe**, and click **Run as administrator**.

2. Click **File**, and then click **Add/Remove Snap-in**.

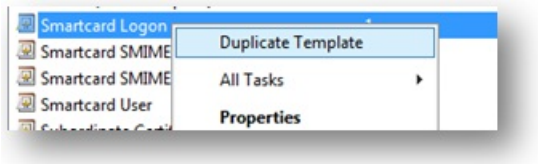


3. In the available snap-ins list, click **Certificate Templates**, and then click **Add**.

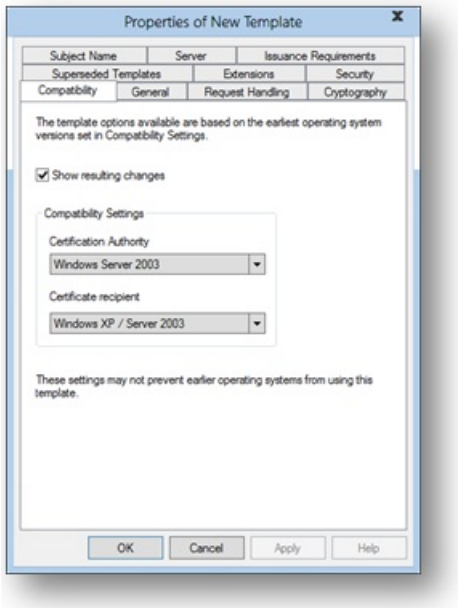


4. Certificate Templates is now located under **Console Root** in the MMC. Double-click it to view all the available certificate templates.

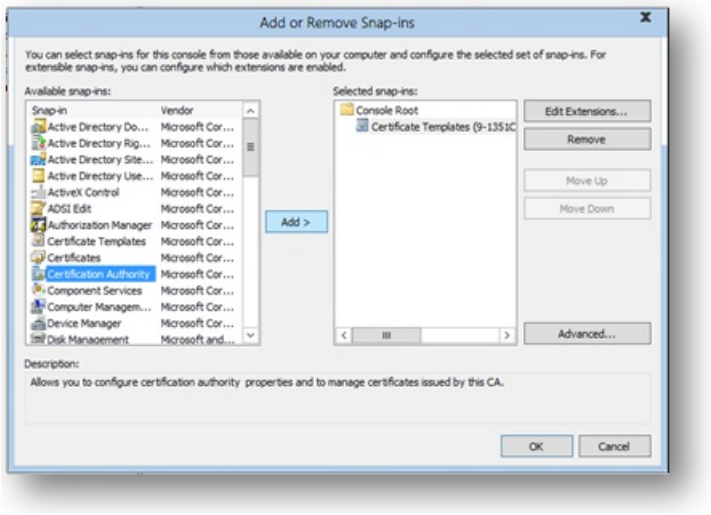
5. Right-click the **Smartcard Logon** template, and click **Duplicate Template**.



6. On the **Compatibility** tab, under **Certification Authority**, review the selection, and change it if needed.

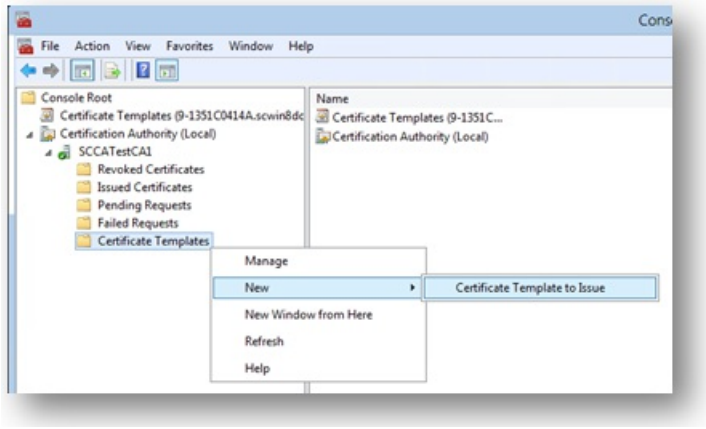


7. On the **General** tab:
    a. Specify a name, such as **TPM Virtual Smart Card Logon**.
    b. Set the validity period to the desired value.
8. On the **Request Handling** tab:
    a. Set the **Purpose** to **Signature and smartcard logon**.
    b. Click **Prompt the user during enrollment**.
9. On the **Cryptography** tab:
    a. Set the minimum key size to 2048.
    b. Click **Requests must use one of the following providers**, and then select **Microsoft Base Smart Card Crypto Provider**.
10. On the **Security** tab, add the security group that you want to give **Enroll** access to. For example, if you want to give access to all users, select the **Authenticated users** group, and then select **Enroll** permissions for them.
11. Click **OK** to finalize your changes and create the new template. Your new template should now appear in the list of Certificate Templates.
12. Select **File**, then click **Add/Remove Snap-in** to add the Certification Authority snap-in to your MMC console. When asked which computer you want to manage, select the computer on which the CA is located, probably **Local Computer**.



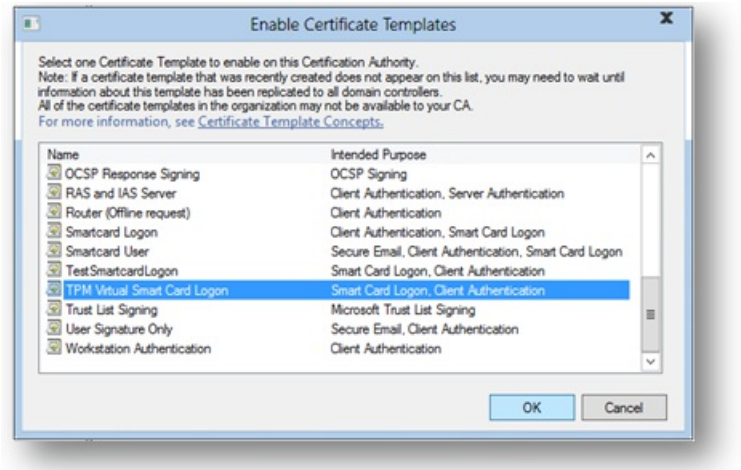
13. In the left pane of the MMC, expand **Certification Authority (Local)**, and then expand your CA within the Certification Authority list.

14. Right-click **Certificate Templates**, click **New**, and then click **Certificate Template to Issue**.

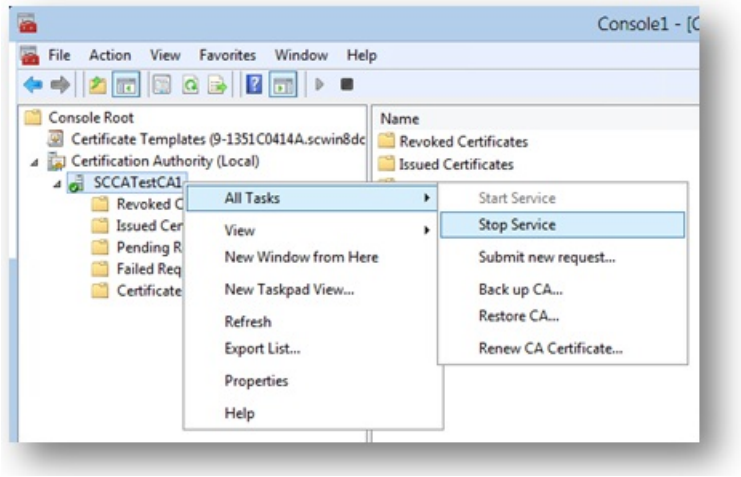


15. From the list, select the new template that you just created (**TPM Virtual Smart Card Logon**), and then click **OK**.

> **Note** It can take some time for your template to replicate to all servers and become available in this list.



16. After the template replicates, in the MMC, right-click in the Certification Authority list, click **All Tasks**, and then click **Stop Service**. Then, right-click the name of the CA again, click **All Tasks**, and then click **Start Service**.

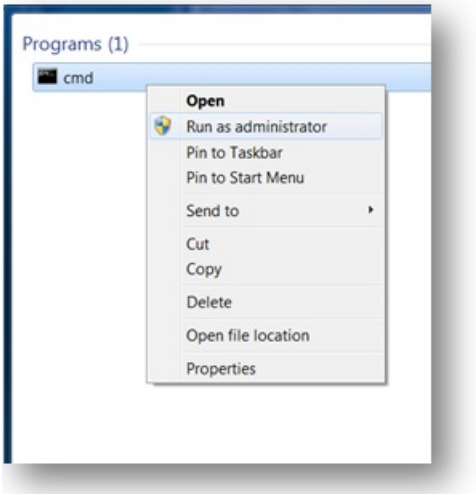


Step 2: Create the TPM virtual smart card

In this step, you will create the virtual smart card on the client computer by using the command-line tool, Tpmvscmgr.exe.

To create the TPM virtual smart card

1. On a domain-joined computer, open a Command Prompt window with Administrative credentials.



2. At the command prompt, type the following, and then press ENTER:

```
tpmvscmgr.exe create /name TestVSC /pin default /adminkey random /generate
```

   This will create a virtual smart card with the name **TestVSC**, omit the unlock key, and generate the file system on the card. The PIN will be set to the default, 12345678. To be prompted for a PIN, instead of **/pin default** you can type **/pin prompt**.

   For more information about the Tpmvscmgr command-line tool, see Use Virtual Smart Cards and Tpmvscmgr.

3. Wait several seconds for the process to finish. Upon completion, Tpmvscmgr.exe will provide you with the device instance ID for the TPM Virtual Smart Card. Store this ID for later reference because you will need it to manage or remove the virtual smart card.

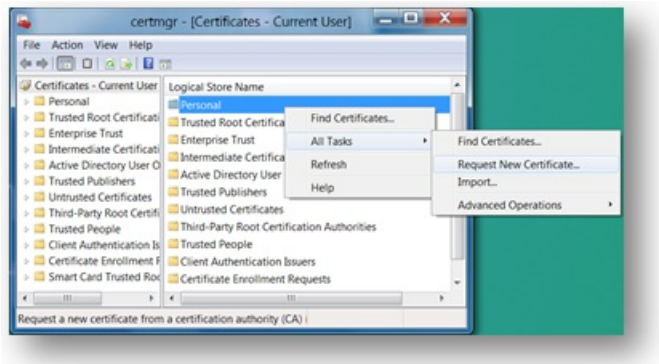
Step 3: Enroll for the certificate on the TPM Virtual Smart Card

The virtual smart card must be provisioned with a sign-in certificate for it to be fully functional.

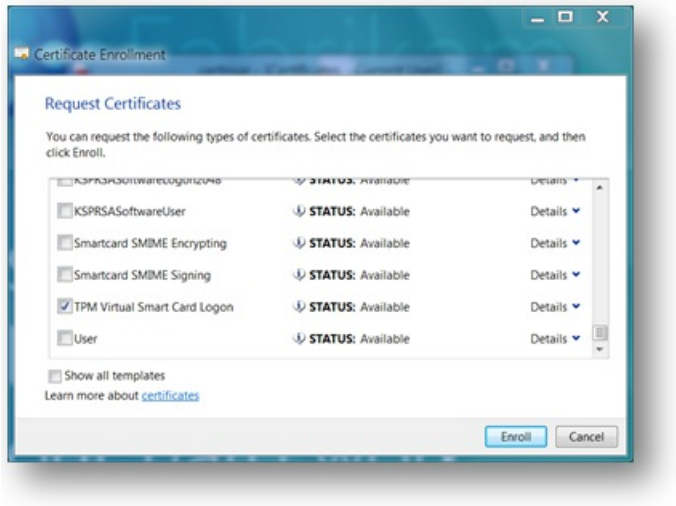
To enroll the certificate

1. Open the Certificates console by typing **certmgr.msc** on the **Start** menu.

2. Right-click **Personal**, click **All Tasks**, and then click **Request New Certificate**.



3. Follow the prompts and when offered a list of templates, select the **TPM Virtual Smart Card Logon** check box (or whatever you named the template in Step 1).



4. If prompted for a device, select the Microsoft virtual smart card that corresponds to the one you created in the previous section. It displays as **Identity Device (Microsoft Profile)**.
5. Enter the PIN that was established when you created the TPM virtual smart card, and then click **OK**.
6. Wait for the enrollment to finish, and then click **Finish**.

The virtual smart card can now be used as an alternative credential to sign in to your domain. To verify that your virtual smart card configuration and certificate enrollment were successful, sign out of your current session, and then sign in. When you sign in, you will see the icon for the new TPM virtual smart card on the Secure Desktop (sign in) screen or you will be automatically directed to the TPM smart card sign-in dialog box. Click the icon, enter your PIN (if necessary), and then click **OK**. You should be signed in to your domain account.

See also

- Understanding and Evaluating Virtual Smart Cards
- Use Virtual Smart Cards
- Deploy Virtual Smart Cards

Use Virtual Smart Cards

6/9/2021 • 5 minutes to read • Edit Online

Applies To: Windows 10, Windows Server 2016

This topic for the IT professional describes requirements for virtual smart cards, how to use virtual smart cards, and tools that are available to help you create and manage them.

Requirements, restrictions, and limitations

AREA	REQUIREMENTS AND DETAILS
Supported operating systems	Windows Server 2016 Windows Server 2012 R2 Windows Server 2012 Windows 10 Windows 8.1 Windows 8
Supported Trusted Platform Module (TPM)	Any TPM that adheres to the TPM main specifications for version 1.2 or version 2.0 (as set by the Trusted Computing Group) is supported for use as a virtual smart card. For more information, see the TPM Main Specification.
Supported virtual smart cards per computer	Ten smart cards can be connected to a computer or device at one time. This includes physical and virtual smart cards combined. **Note** You can create more than one virtual smart card; however, after creating more than four virtual smart cards, you may start to notice performance degradation. Because all smart cards appear as if they are always inserted, if more than one person shares a computer or device, each person can see all the virtual smart cards that are created on that computer or device. If the user knows the PIN values for all the virtual smart cards, the user will also be able to use them.
Supported number of certificates on a virtual smart card	A single TPM virtual smart card can contain 30 distinct certificates with the corresponding private keys. Users can continue to renew certificates on the card until the total number of certificates on a card exceeds 90. The reason that the total number of certificates is different from the total number of private keys is that sometimes the renewal can be done with the same private key—in which case a new private key is not generated.
PIN, PIN Unlock Key (PUK), and Administrative key requirements	The PIN and the PUK must be a minimum of eight characters that can include numerals, alphabetic characters, and special characters. The Administrative key must be entered as 48 hexadecimal characters. It is a 3-key triple DES with ISO/IEC 9797 padding method 2 in CBC chaining mode.

Using Tpmvscmgr.exe

To create and delete TPM virtual smart cards for end users, the Tpmvscmgr command-line tool is included as a command-line tool with the operating system. You can use the **Create** and **Delete** parameters to manage virtual smart cards on local or remote computers. For information about using this tool, see Tpmvscmgr.

Create and delete virtual smart cards programmatically

Virtual smart cards can also be created and deleted by using APIs. For more information, see the following classes and interfaces:

- TpmVirtualSmartCardManager
- RemoteTpmVirtualSmartCardManager
- ITpmVirtualSmartCardManager
- ITPMVirtualSmartCardManagerStatusCallBack

You can use APIs that were introduced in the Windows.Device.SmartCards namespace in Windows Server 2012 R2 and Windows 8.1 to build Microsoft Store apps to manage the full lifecycle of virtual smart cards. For information about how to build an app to do this, see Strong Authentication: Building Apps That Leverage Virtual Smart Cards in Enterprise, BYOD, and Consumer Environments | Build 2013 | Channel 9.

The following table describes the features that can be developed in a Microsoft Store app:

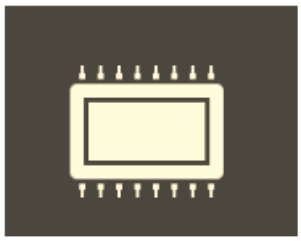
FEATURE	PHYSICAL SMART CARD	VIRTUAL SMART CARD
Query and monitor smart card readers	Yes	Yes
List available smart cards in a reader, and retrieve the card name and card ID	Yes	Yes
Verify if the administrative key of a card is correct	Yes	Yes
Provision (or reformat) a card with a given card ID	Yes	Yes
Change the PIN by entering the old PIN and specifying a new PIN	Yes	Yes
Change the administrative key, reset the PIN, or unblock the smart card by using a challenge/response method	Yes	Yes
Create a virtual smart card	Not applicable	Yes
Delete a virtual smart card	Not applicable	Yes
Set PIN policies	No	Yes

For more information about these Windows APIs, see:

- Windows.Devices.SmartCards namespace (Windows)
- Windows.Security.Cryptography.Certificates namespace (Windows)

Distinguishing TPM-based virtual smart cards from physical smart cards

To help users visually distinguish a Trusted Platform Module (TPM)-based virtual smart card from physical smart cards, the virtual smart card has a different icon. The following icon is displayed during sign in, and on other screens that require the user to enter the PIN for a virtual smart card.



A TPM-based virtual smart card is labeled **Security Device** in the user interface.

Changing the PIN

The PIN for a virtual smart card can be changed by following these steps:

- Sign in with the old PIN or password.
- Press Ctrl+Alt+Del and choose **Change a password**.
- Select **Sign-in Options**.
- Select the virtual smart card icon.
- Enter and confirm the new PIN.

Resolving issues

TPM not provisioned

For a TPM-based virtual smart card to function properly, a provisioned TPM must be available on the computer. If the TPM is disabled in the BIOS, or it is not provisioned with full ownership and the storage root key, the TPM virtual smart card creation will fail.

If the TPM is initialized after creating a virtual smart card, the card will no longer function, and it will need to be re-created.

If the TPM ownership was established on a Windows Vista installation, the TPM will not be ready to use virtual smart cards. The system administrator needs to clear and initialize the TPM for it to be suitable for creating TPM virtual smart cards.

If the operating system is reinstalled, prior TPM virtual smart cards are no longer available and need to be re-created. If the operating system is upgraded, prior TPM virtual smart cards will be available to use in the upgraded operating system.

TPM in lockout state

Sometimes, due to frequent incorrect PIN attempts from a user, the TPM may enter the lockout state. To resume using the TPM virtual smart card, it is necessary to reset the lockout on the TPM by using the owner's password or to wait for the lockout to expire. Unblocking the user PIN does not reset the lockout in the TPM. When the TPM is in lockout, the TPM virtual smart card appears as if it is blocked. When the TPM enters the lockout state because the user entered an incorrect PIN too many times, it may be necessary to reset the user PIN by using the virtual smart card management tools, such as Tpmvscmgr command-line tool.

See also

For information about authentication, confidentiality, and data integrity use cases, see Virtual Smart Card Overview.

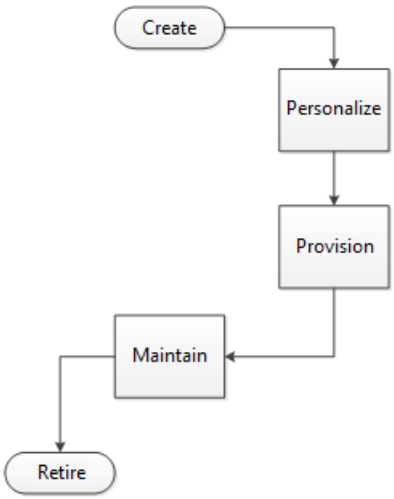
Deploy Virtual Smart Cards

3/26/2021 • 24 minutes to read • Edit Online

Applies To: Windows 10, Windows Server 2016

This topic for the IT professional discusses the factors to consider when you deploy a virtual smart card authentication solution.

Traditional identity devices, such as physical smart cards, follow a predictable lifecycle in any deployment, as shown in the following diagram.



Physical devices are created by a dedicated manufacturer and then purchased by the corporation that will ultimately deploy it. The device passes through the personalization stage, where its unique properties are set. In smart cards, these properties are the administrator key, Personal Identification Number (PIN), PIN Unlock Key (PUK), and its physical appearance. To provision the device, it is loaded with the required certificates, such as a sign-in certificate. After you provision the device, it is ready for use. The device must simply be maintained. For example, you must replace cards when they are lost or stolen and reset PINs when users forget them. Finally, you'll retire devices when they exceed their intended lifetime or when employees leave the company.

This topic contains information about the following phases in a virtual smart card lifecycle:

- Create and personalize virtual smart cards
- Provision virtual smart cards
- Maintain virtual smart cards

Create and personalize virtual smart cards

A corporation purchases the devices to deploy then. The device passes through the personalization stage, where its unique properties are set. In smart cards, these properties are the administrator key, Personal Identification Number (PIN), PIN Unlock Key (PUK), and its physical appearance. The security that is provided for a TPM virtual smart card is fully provisioned in the host TPM.

Trusted Platform Module readiness

The TPM Provisioning Wizard, which is launched from the **TPM Management Console**, takes the user through all the steps to prepare the TPM for use.

When you create virtual smart cards, consider the following actions in the TPM:

- **Enable and Activate**: TPMs are built in to many industry ready computers, but they often are not enabled and activated by default. In some cases, the TPM must be enabled and activated through the BIOS. For more information, see Initialize and Configure Ownership of the TPM.
- **Take ownership**: When you provision the TPM, you set an owner password for managing the TPM in the future, and you establish the storage root key. To provide anti-hammering protection for virtual smart cards, the user or a domain administrator must be able to reset the TPM owner password. For corporate use of TPM virtual smart cards, we recommend that the corporate domain administrator restrict access to the TPM owner password by storing it in Active Directory, not in the local registry. When TPM ownership is set in Windows Vista, the TPM needs to be cleared and reinitialized. For more information, see Trusted Platform Module Technology Overview.
- **Manage**: You can manage ownership of a virtual smart card by changing the owner password, and you can manage anti-hammering logic by resetting the lockout time. For more information, see Manage TPM Lockout.

A TPM might operate in reduced functionality mode. This could occur, for example, if the operating system cannot determine if the owner password is available to the user. In those cases, the TPM can be used to create a virtual smart card, but it is strongly recommended to bring the TPM to a fully ready state so that any unexpected circumstances will not leave the user blocked from using the computer.

Those smart card deployment management tools that require a status check of a TPM before attempting to create a TPM virtual smart card can do so using the TPM WMI interface.

Depending on the setup of the computer that is designated for installing TPM virtual smart cards, it might be necessary to provision the TPM before continuing with the virtual smart card deployment. For more information about provisioning, see Use Virtual Smart Cards.

For more information about managing TPMs by using built-in tools, see Trusted Platform Module Services Group Policy Settings.

Creation

A TPM virtual smart card simulates a physical smart card, and it uses the TPM to provide the same functionality as physical smart card hardware. A virtual smart card appears within the operating system as a physical smart card that is always inserted. Supported versions of the Windows operating system present a virtual smart card reader and virtual smart card to applications with the same interface as physical smart cards, but messages to and from the virtual smart card are translated to TPM commands. This process ensures the integrity of the virtual smart card through the three properties of smart card security:

- **Non-exportability**: Because all private information on the virtual smart card is encrypted by using the TPM on the host computer, it cannot be used on a different computer with a different TPM. Additionally, TPMs are designed to be tamper-resistant and non-exportable, so a malicious user cannot reverse engineer an identical TPM or install the same TPM on a different computer. For more information, see Evaluate Virtual Smart Card Security.
- **Isolated cryptography**: TPMs provide the same properties of isolated cryptography that is offered by physical smart cards, and this is utilized by virtual smart cards. Unencrypted copies of private keys are loaded only within the TPM and never into memory that is accessible by the operating system. All cryptographic operations with these private keys occur inside the TPM.
- **Anti-hammering**: If a user enters a PIN incorrectly, the virtual smart card responds by using the anti-hammering logic of the TPM, which rejects further attempts for a period of time instead of blocking the card. This is also known as lockout. For more information, see Blocked virtual smart card and Evaluate Virtual Smart Card Security.

There are several options for creating virtual smart cards, depending on the size of the deployment and budget

of the organization. The lowest cost option is using Tpmvscmgr.exe to create cards individually on users' computers. Alternatively, a virtual smart card management solution can be purchased to more easily accomplish virtual smart card creation on a larger scale and aid in further phases of deployment. Virtual smart cards can be created on computers that are to be provisioned for an employee or on those that are already in an employee's possession. In either approach, there should be some central control over personalization and provisioning. If a computer is intended for use by multiple employees, multiple virtual smart cards can be created on a computer.

For information about the TPM Virtual Smart Card command-line tool, see Tpmvscmgr.

Personalization

During virtual smart card personalization, the values for the administrator key, PIN, and PUK are assigned. As with a physical card, knowing the administrator key is important for resetting the PIN or for deleting the card in the future. (If a PUK is set, the administrator key can no longer be used to reset the PIN.)

Because the administrator key is critical to the security of the card, it is important to consider the deployment environment and decide on the proper administrator key setting strategy. Options for these strategies include:

- **Uniform**: Administrator keys for all the virtual smart cards that are deployed in the organization are the same. Although this makes the maintenance infrastructure easy (only one key needs to be stored), it is highly insecure. This strategy might be sufficient for very small organizations, but if the administrator key is compromised, all virtual smart cards that use this key must be reissued.
- **Random, not stored**: Administrator keys are assigned randomly for all virtual smart cards, and they are not recorded. This is a valid option if the deployment administrators do not require the ability to reset PINs, and instead prefer to delete and reissue virtual smart cards. This could also be a viable strategy if the administrator prefers to set PUK values for the virtual smart cards and then use this value to reset PINs, if necessary.
- **Random, stored**: Administrator keys are assigned randomly and stored in a central location. Each card's security is independent of the others. This is secure on a large scale unless the administrator key database is compromised.
- **Deterministic**: Administrator keys are the result of some function or known information. For example, the user ID could be used to randomly generate data that can be further processed through a symmetric encryption algorithm by using a secret. This administrator key can be similarly regenerated when needed, and it does not need to be stored. The security of this method relies on the security of the secret used.

Although the PUK and the administrator key methodologies provide unlocking and resetting functionality, they do so in different ways. The PUK is a PIN that is simply entered on the computer to enable a user PIN reset.

The administrator key methodology takes a challenge-response approach. The card provides a set of random data after users verify their identity to the deployment administrator. The administrator then encrypts the data with the administrator key and gives the encrypted data back to the user. If the encrypted data matches that produced by the card during verification, the card will allow PIN reset. Because the administrator key is never accessible by anyone other than the deployment administrator, it cannot be intercepted or recorded by any other party (including employees). This provides significant security benefits beyond using a PUK, an important consideration during the personalization process.

TPM virtual smart cards can be personalized on an individual basis when they are created with the Tpmvscmgr command-line tool. Or organizations can purchase a management solution that can incorporate personalization into an automated routine. An additional advantage of such a solution is the automated creation of administrator keys. Tpmvscmgr.exe allows users to create their own administrator keys, which can be detrimental to the security of the virtual smart cards.

Provision virtual smart cards

Provisioning is the process of loading specific credentials onto a TPM virtual smart card. These credentials consist of certificates that are created to give users access to a specific service, such as domain sign in. A maximum of 30 certificates is allowed on each virtual smart card. As with physical smart cards, several decisions must be made regarding the provisioning strategy, based on the environment of the deployment and the desired level of security.

A high-assurance level of secure provisioning requires absolute certainty about the identity of the individual who is receiving the certificate. Therefore, one method of high-assurance provisioning is utilizing previously provisioned strong credentials, such as a physical smart card, to validate identity during provisioning. In-person proofing at enrollment stations is another option, because an individual can easily and securely prove his or her identity with a passport or driver's license, although this can become infeasible on a larger scale. To achieve a similar level of assurance, a large organization can implement an "enroll-on-behalf-of" strategy, in which employees are enrolled with their credentials by a superior who can personally verify their identities. This creates a chain of trust that ensures individuals are checked in person against their proposed identities, but without the administrative strain of provisioning all virtual smart cards from a single central enrollment station.

For deployments in which a high-assurance level is not a primary concern, you can use self-service solutions. These can include using an online portal to obtain credentials or simply enrolling for certificates by using Certificate Manager, depending on the deployment. Consider that virtual smart card authentication is only as strong as the method of provisioning. For example, if weak domain credentials (such as a password alone) are used to request the authentication certificate, virtual smart card authentication will be equivalent to using only the password, and the benefits of two-factor authentication are lost.

For information about using Certificate Manager to configure virtual smart cards, see Get Started with Virtual Smart Cards: Walkthrough Guide.

High-assurance and self-service solutions approach virtual smart card provisioning by assuming that the user's computer has been issued prior to the virtual smart card deployment, but this is not always the case. If virtual smart cards are being deployed with new computers, they can be created, personalized, and provisioned on the computer before the user has contact with that computer.

In this situation, provisioning becomes relatively simple, but identity checks must be put in place to ensure that the recipient of the computer is the individual who was expected during provisioning. This can be accomplished by requiring the employee to set the initial PIN under the supervision of the deployment administrator or manager.

When you are provisioning your computers, you should also consider the longevity of credentials that are supplied for virtual smart cards. This choice must be based on the risk threshold of the organization. Although longer lived credentials are more convenient, they are also more likely to become compromised during their lifetime. To decide on the appropriate lifetime for credentials, the deployment strategy must take into account the vulnerability of their cryptography (how long it could take to crack the credentials), and the likelihood of attack.

If a virtual smart card is compromised, administrators should be able to revoke the associated credentials, like they would with a lost or stolen laptop. This requires a record of which credentials match which user and computer, which is functionality that does not exist natively in Windows. Deployment administrators might want to consider add-on solutions to maintain such a record.

Virtual smart cards on consumer devices used for corporate access

There are techniques that allow employees to provision virtual smart cards and enroll for certificates that can be used to authenticate the users. This is useful when employees attempt to access corporate resources from devices that are not joined to the corporate domain. Those devices can be further defined to not allow users to download and run applications from sources other than the Windows Store (for example, devices running Windows RT).

You can use APIs that were introduced in Windows Server 2012 R2 and Windows 8.1 to build Windows Store

apps that you can use to manage the full lifecycle of virtual smart cards. For more information, see Create and delete virtual smart cards programmatically.

TPM ownerAuth in the registry

When a device or computer is not joined to a domain, the TPM ownerAuth is stored in the registry under HKEY_LOCAL_MACHINE. This exposes some threats. Most of the threat vectors are protected by BitLocker, but threats that are not protected include:

- A malicious user possesses a device that has an active local sign-in session before the device locks. The malicious user could attempt a brute-force attack on the virtual smart card PIN, and then access the corporate secrets.
- A malicious user possesses a device that has an active virtual private network (VPN) session. The device is then compromised.

The proposed mitigation for the previous scenarios is to use Exchange ActiveSync (EAS) policies to reduce the automatic lockout time from five minutes to 30 seconds of inactivity. Policies for automatic lockout can be set while provisioning virtual smart cards. If an organization wants more security, they can also configure a setting to remove the ownerAuth from the local device.

For configuration information about the TPM ownerAuth registry key, see the Group Policy setting Configure the level of TPM owner authorization information available to the operating system.

For information about EAS policies, see Exchange ActiveSync Policy Engine Overview.

Managed and unmanaged cards

The following table describes the important differences between managed and unmanaged virtual smart cards that exist on consumer devices:

OPERATION	MANAGED AND UNMANAGED CARDS	UNMANAGED CARDS
Reset PIN when the user forgets the PIN	Yes	No, the card has to be deleted and created again.
Allow user to change the PIN	Yes	No, the card has to be deleted and created again.

Managed cards

A managed virtual smart card can be serviced by the IT administrator or another person in that designated role. It allows the IT administrator to have influence or complete control over specific aspects of the virtual smart card from its creation to deletion. To manage these cards, a virtual smart card deployment management tool is often required.

Managed card creation

A user can create blank virtual smart card by using the Tpmvscmgr command-line tool, which is a built-in tool that is run with administrative credentials through an elevated command prompt. This virtual smart card needs to be created with well-known parameters (such as default values), and it should be left unformatted (specifically, the **/generate** option should not be specified).

The following command creates a virtual smart card that can later be managed by a smart card management tool launched from another computer (as explained in the next section):

```
tpmvscmgr.exe create /name "VirtualSmartCardForCorpAccess" /AdminKey DEFAULT /PIN PROMPT
```

Alternatively, instead of using a default administrator key, a user can enter an administrator key at the command line:

```
tpmvscmgr.exe create /name "VirtualSmartCardForCorpAccess" /AdminKey PROMPT /PIN PROMPT
```

In either case, the card management system needs to be aware of the initial administrator key that is used so that it can take ownership of the virtual smart card and change the administrator key to a value that is only accessible through the card management tool operated by the IT administrator. For example, when the default value is used, the administrator key is set to:

```
102030405060708010203040506070801020304050607080
```

For information about using this command-line tool, see Tpmvscmgr.

Managed card management

After the virtual smart card is created, the user needs to open a remote desktop connection to an enrollment station, for example, in a computer that is joined to the domain. Virtual smart cards that are associated with a client computer are available for use in the remote desktop connection. The user can open a card management tool inside the remote session that can take ownership of the card and provision it for use by the user. This requires that a user is allowed to establish a remote desktop connection from a non-domain-joined computer to a domain-joined computer. This might require a specific network configuration, such as through IPsec policies.

When users need to reset or change a PIN, they need to use the remote desktop connection to complete these operations. They can use the built-in tools for PIN unlock and PIN change or the smart card management tool.

Certificate management for managed cards

Similar to physical smart cards, virtual smart cards require certificate enrollment.

Certificate issuance

Users can enroll for certificates from within a remote desktop session that is established to provision the card. This process can also be managed by the smart card management tool that the user runs through the remote desktop connection. This model works for deployments that require the user to sign a request for enrollment by using a physical smart card. The driver for the physical smart card does not need to be installed on the client computer if it is installed on the remote computer. This is made possible by smart card redirection functionality that was introduced in Windows Server 2003, which ensures that smart cards that are connected to the client computer are available for use during a remote session.

Alternatively, without establishing a remote desktop connection, users can enroll for certificates from the Certificate Management console (certmgr.msc) on a client computer. Users can also create a request and submit it to a server from within a custom certificate enrollment application (for example, a registration authority) that has controlled access to the certification authority (CA). This requires specific enterprise configuration and deployments for Certificate Enrollment Policies (CEP) and Certificate Enrollment Services (CES).

Certificate lifecycle management

You can renew certificates through remote desktop connections, certificate enrollment policies, or certificate enrollment services. Renewal requirements could be different from the initial issuance requirements, based on the renewal policy.

Certificate revocation requires careful planning. When information about the certificate to be revoked is reliably available, the specific certificate can be easily revoked. When information about the certificate to be revoked is not easy to determine, all certificates that are issued to the user under the policy that was used to issue the certificate might need to be revoked. For example, this could occur if an employee reports a lost or compromised device, and information that associates the device with a certificate is not available.

Unmanaged cards

Unmanaged virtual smart cards are not serviceable by an IT administrator. Unmanaged cards might be suitable if an organization does not have an elaborate smart card deployment management tool and using remote desktop connections to manage the card is not desirable. Because unmanaged cards are not serviceable by the

IT administrator, when a user needs help with a virtual smart card (for example, resetting or unlocking a PIN), the only option available to the user is to delete the card and create it again. This results in loss of the user's credentials and he or she must re-enroll.

Unmanaged card creation

A user can create a virtual smart card by using the Tpmvscmgr command-line tool, which is run with administrative credentials through an elevated command prompt. The following command creates an unmanaged card that can be used to enroll certificates:

```
tpmvscmgr.exe create /name "VirtualSmartCardForCorpAccess" /AdminKey RANDOM /PIN PROMPT /generate
```

This command creates a card with a randomized administrator key. The key is automatically discarded after the creation of the card. If users forget or want to change their PIN, they need to delete the card and create it again. To delete the card, a user can run the following command:

```
tpmvscmgr.exe destroy /instance <instance ID>
```

where <instance ID> is the value that is printed on the screen when the user creates the card. Specifically, for the first card created, the instance ID is ROOT\SMARTCARDREADER\0000).

Certificate management for unmanaged cards

Depending on the security requirements that are unique to an organization, users can initially enroll for certificates from the certificate management console (certmgr.msc) or from within custom certificate enrollment applications. The latter method can create a request and submit it to a server that has access to the Certification Authority. This requires specific organizational configurations and deployments for certificate enrollment policies and certificate enrollment services. Windows has built-in tools, specifically Certreq.exe and Certutil.exe, which can be used by scripts to perform the enrollment from the command line.

Requesting the certificate by providing domain credentials only

The simplest way for users to request certificates is to provide their domain credentials through a script that can perform the enrollment through built-in components you have in place for certificate requests.

Alternatively, an application (such as a line-of-business app) can be installed on the computer to perform enrollment by generating a request on the client. The request is submitted to an HTTP server, which can forward it to a registration authority.

Another option is to have the user access an enrollment portal that is available through Internet Explorer. The webpage can use the scripting APIs to perform certificate enrollment.

Signing the request with another certificate

You can provide users with a short-term certificate through a Personal Information Exchange (.pfx) file. You can generate the .pfx file by initiating a request from a domain-joined computer. Additional policy constraints can be enforced on the .pfx file to assert the identity of the user.

The user can import the certificate into the **MY** store (which is the user's certificate store). And your organization can present the user with a script that can be used to sign the request for the short-term certificate and to request a virtual smart card.

For deployments that require users to use a physical smart card to sign the certificate request, you can use the procedure:

1. Users initiate a request on a domain-joined computer.
2. Users complete the request by using a physical smart card to sign the request.
3. Users download the request to the virtual smart card on their client computer.

Using one-time password for enrollment

Another option to ensure that users are strongly authenticated before virtual smart card certificates are issued

is to send a user a one-time password through SMS, email, or phone. The user then types the one-time password during the certificate enrollment from an application or a script on a desktop that invokes built-in command-line tools.

Certificate lifecycle management

Certificate renewal can be done from the same tools that are used for the initial certificate enrollment. Certificate enrollment policies and certificate enrollment services can also be used to perform automatic renewal.

Certificate revocation requires careful planning. When information about the certificate to be revoked is reliably available, the specific certificate can be easily revoked. When information about the certificate to be revoked is not easy to determine, all certificates that are issued to the user under the policy that was used to issue the certificate might need to be revoked. For example, this could occur if an employee reports a lost or compromised device, and information that associates the device with a certificate is not available.

Maintain virtual smart cards

Maintenance is a significant portion of the virtual smart card lifecycle and one of the most important considerations from a management perspective. After virtual smart cards are created, personalized, and provisioned, they can be used for convenient two-factor authentication. Deployment administrators must be aware of several common administrative scenarios, which can be approached by using a purchased virtual smart card solution or on a case-by-case basis with in-house methods.

Renewal: Renewing virtual smart card credentials is a regular task that is necessary to preserve the security of a virtual smart card deployment. Renewal is the result of a signed request from a user who specifies the key pair desired for the new credentials. Depending on user's choice or deployment specification, the user can request credentials with the same key pair as previously used, or choose a newly generated key pair.

When renewing with a previously used key, no extra steps are required because a strong certificate with this key was issued during the initial provisioning. However, when the user requests a new key pair, you must take the same steps that were used during provisioning to assure the strength of the credentials. Renewal with new keys should occur periodically to counter sophisticated long-term attempts by malicious users to infiltrate the system. When new keys are assigned, you must ensure that the new keys are being used by the expected individuals on the same virtual smart cards.

Resetting PINs: Resetting virtual smart card PINs is also a frequent necessity, because employees forget their PINs. There are two ways to accomplish this, depending on choices made earlier in the deployment: Use a PUK (if the PUK is set), or use a challenge-response approach with the administration key. Before resetting the PIN, the user's identity must be verified by using some means other than the card—most likely the verification method that you used during initial provisioning (for example, in-person proofing). This is necessary in user-error scenarios when users forget their PINs. However, you should never reset a PIN if it has been compromised because the level of vulnerability after the PIN is exposed is difficult to identify. The entire card should be reissued.

Lockout reset: A frequent precursor to resetting a PIN is the necessity of resetting the TPM lockout time because the TPM anti-hammering logic will be engaged with multiple PIN entry failures for a virtual smart card. This is currently device specific.

Retiring cards: The final aspect of virtual smart card management is retiring cards when they are no longer needed. When an employee leaves the company, it is desirable to revoke domain access. Revoking sign-in credentials from the certification authority (CA) accomplishes this goal.

The card should be reissued if the same computer is used by other employees without reinstalling the operating system. Reusing the former card can allow the former employee to change the PIN after leaving the organization, and then hijack certificates that belong to the new user to obtain unauthorized domain access. However, if the employee takes the virtual smart card-enabled computer, it is only necessary to revoke the

certificates that are stored on the virtual smart card.

Emergency preparedness

Card reissuance

The most common scenario in an organization is reissuing virtual smart cards, which can be necessary if the operating system is reinstalled or if the virtual smart card is compromised in some manner. Reissuance is essentially the recreation of the card, which involves establishing a new PIN and administrator key and provisioning a new set of associated certificates. This is an immediate necessity when a card is compromised, for example, if the virtual smart card-protected computer is exposed to an adversary who might have access to the correct PIN. Reissuance is the most secure response to an unknown exposure of a card's privacy. Additionally, reissuance is necessary after an operating system is reinstalled because the virtual smart card device profile is removed with all other user data when the operating system is reinstalled.

Blocked virtual smart card

The anti-hammering behavior of a TPM virtual smart card is different from that of a physical smart card. A physical smart card blocks itself after the user enters the wrong PIN a few times. A TPM virtual smart card enters a timed delay after the user enters the wrong PIN a few times. If the TPM is in the timed-delay mode, when the user attempts to use the TPM virtual smart card, the user is notified that the card is blocked. Furthermore, if you enable the integrated unlock functionality, the user can see the user interface to unlock the virtual smart card and change the PIN. Unlocking the virtual smart card does not reset the TPM lockout. The user needs to perform an extra step to reset the TPM lockout or wait for the timed delay to expire.

For more information about setting the Allow Integrated Unblock policy, see Allow Integrated Unblock screen to be displayed at the time of logon.

See also

Understanding and Evaluating Virtual Smart Cards

Get Started with Virtual Smart Cards: Walkthrough Guide

Use Virtual Smart Cards

Evaluate Virtual Smart Card Security

Tpmvscmgr

Evaluate Virtual Smart Card Security

3/5/2021 • 3 minutes to read • Edit Online

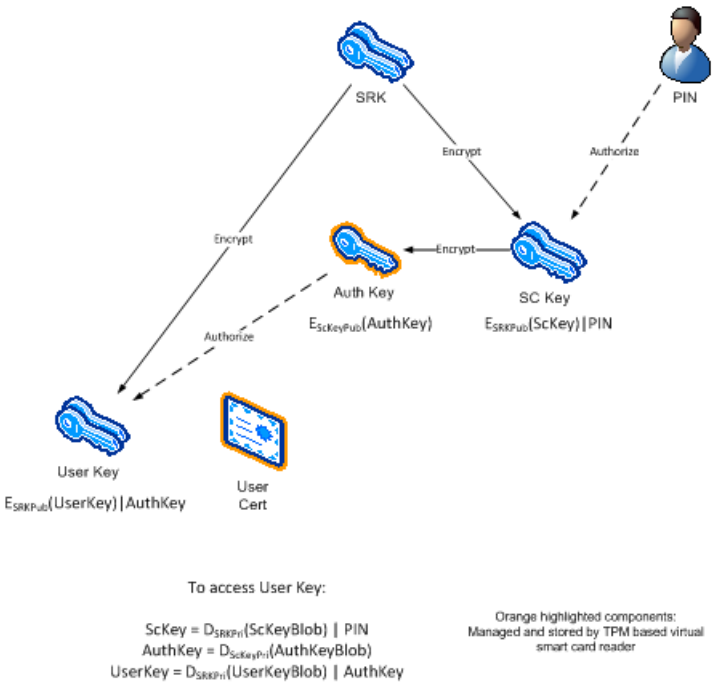
Applies To: Windows 10, Windows Server 2016

This topic for the IT professional describes security characteristics and considerations when deploying TPM virtual smart cards.

Virtual smart card non-exportability details

A crucial aspect of TPM virtual smart cards is their ability to securely store and use secret data, specifically that the secured data is non-exportable. Data can be accessed and used within the virtual smart card system, but it is meaningless outside of its intended environment. In TPM virtual smart cards, security is ensured with a secure key hierarchy, which includes several chains of encryption. This originates with the TPM storage root key, which is generated and stored within the TPM and never exposed outside the chip. The TPM key hierarchy is designed to allow encryption of user data with the storage root key, but it authorizes decryption with the user PIN in such a way that changing the PIN doesn't require re-encryption of the data.

The following diagram illustrates the secure key hierarchy and the process of accessing the user key.



The following keys are stored on the hard disk:

- User key
- Smart card key, which is encrypted by the storage root key
- Authorization key for the user key decryption, which is encrypted by the public portion of the smart card key

When the user enters a PIN, the use of the decrypted smart card key is authorized with this PIN. If this authorization succeeds, the decrypted smart card key is used to decrypt the auth key. The auth key is then provided to the TPM to authorize the decryption and use of the user's key that is stored on the virtual smart card.

The auth key is the only sensitive data that is used as plaintext outside the TPM, but its presence in memory is protected by Microsoft Data Protection API (DPAPI), such that before being stored in any way, it is encrypted. All data other than the auth key is processed only as plaintext within the TPM, which is completely isolated from external access.

Virtual smart card anti-hammering details

The anti-hammering functionality of virtual smart cards relies on the anti-hammering functionality of the TPM that is enabling the virtual smart card. However, the TPM version 1.2 and subsequent specifications (as designed by the Trusted Computing Group) provide very flexible guidelines for responding to hammering. The spec requires only that the TPM implement protection against trial-and-error attacks on the user PIN, PUK, and challenge/response mechanism.

The Trusted Computing Group also specifies that if the response to attacks involves suspending proper function of the TPM for some period of time or until administrative action is taken, the TPM must prevent running the authorized TPM commands. The TPM can prevent running any TPM commands until the termination of the attack response. Beyond using a time delay or requiring administrative action, a TPM could also force a reboot when an attack is detected. The Trusted Computing Group allows manufacturers a level of creativity in their choice of implementation. Whatever methodology is chosen by TPM manufacturers determines the anti-hammering response of TPM virtual smart cards. Some typical aspects of protection from attacks include:

1. Allow only a limited number of wrong PIN attempts before enabling a lockout that enforces a time delay before any further commands are accepted by the TPM.

   > **Note** Introduced in Windows Server 2012 R2 and Windows 8.1, if the user enters the wrong PIN five consecutive times for a virtual smart card (which works in conjunction with the TPM), the card is blocked. When the card is blocked, it has to be unblocked by using the administrative key or the PUK.

2. Increase the time delay exponentially as the user enters the wrong PIN so that an excessive number of wrong PIN attempts quickly trigger long delays in accepting commands.
3. Have a failure leakage mechanism to allow the TPM to reset the timed delays over a period of time. This is useful in cases where a valid user has entered the wrong PIN occasionally, for example, due to complexity of the PIN.

As an example, it will take 14 years to guess an 8-character PIN for a TPM that implements the following protection:

1. Number of wrong PINs allowed before entering lockout (threshold): 9
2. Time the TPM is in lockout after the threshold is reached: 10 seconds
3. Timed delay doubles for each wrong PIN after the threshold is reached

See also

Understanding and Evaluating Virtual Smart Cards

Tpmvscmgr

3/26/2021 • 5 minutes to read • Edit Online

Applies To: Windows 10, Windows Server 2016

The Tpmvscmgr command-line tool allows users with Administrative credentials to create and delete TPM virtual smart cards on a computer. For examples of how this command can be used, see Examples.

Syntax

```
Tpmvscmgr create [/quiet] /name <name> /AdminKey {DEFAULT | PROMPT | RANDOM} [/PIN {DEFAULT | PROMPT}] [/PUK {DEFAULT | PROMPT}] [/generate] [/machine <machine name>] [/pinpolicy [policy options]] [/attestation {AIK_AND_CERT | AIK_ONLY}] [/?]
```

```
Tpmvscmgr destroy [/quiet] [/instance <device instance ID>] [/machine <machine name>] [/?]
```

Parameters for Create command

The Create command sets up new virtual smart cards on the user's system. It returns the instance ID of the newly created card for later reference if deletion is required. The instance ID is in the format ROOT\SMARTCARDREADER\000n where n starts from 0 and is increased by 1 each time you create a new virtual smart card.

PARAMETER	DESCRIPTION
/name	Required. Indicates the name of the new virtual smart card.
/AdminKey	Indicates the desired administrator key that can be used to reset the PIN of the card if the user forgets the PIN. **DEFAULT** Specifies the default value of 010203040506070801020304050607080102030405060708. **PROMPT** Prompts the user to enter a value for the administrator key. **RANDOM** Results in a random setting for the administrator key for a card that is not returned to the user. This creates a card that might not be manageable by using smart card management tools. When generated with RANDOM, the administrator key is set as 48 hexadecimal characters.
/PIN	Indicates desired user PIN value. **DEFAULT** Specifies the default PIN of 12345678. **PROMPT** Prompts the user to enter a PIN at the command line. The PIN must be a minimum of eight characters, and it can contain numerals, characters, and special characters.
/PUK	Indicates the desired PIN Unlock Key (PUK) value. The PUK value must be a minimum of eight characters, and it can contain numerals, characters, and special characters. If the parameter is omitted, the card is created without a PUK. **DEFAULT** Specifies the default PUK of 12345678. **PROMPT** Prompts the user to enter a PUK at the command line.

PARAMETER	DESCRIPTION
/generate	Generates the files in storage that are necessary for the virtual smart card to function. If the /generate parameter is omitted, it is equivalent to creating a card without this file system. A card without a file system can be managed only by a smart card management system such as Microsoft Endpoint Configuration Manager.
/machine	Allows you to specify the name of a remote computer on which the virtual smart card can be created. This can be used in a domain environment only, and it relies on DCOM. For the command to succeed in creating a virtual smart card on a different computer, the user running this command must be a member in the local administrators group on the remote computer.
/pinpolicy	If **/pin prompt** is used, **/pinpolicy** allows you to specify the following PIN policy options: **minlen** <minimum PIN length> If not specified, defaults to 8. The lower bound is 4. **maxlen** <maximum PIN length> If not specified, defaults to 127. The upper bound is 127. **uppercase** Can be **ALLOWED**, **DISALLOWED**, or **REQUIRED.** Default is **ALLOWED.** **lowercase** Can be **ALLOWED**, **DISALLOWED**, or **REQUIRED.** Default is **ALLOWED.** **digits** Can be **ALLOWED**, **DISALLOWED**, or **REQUIRED.** Default is **ALLOWED.** **specialchars** Can be **ALLOWED**, **DISALLOWED**, or **REQUIRED.** Default is **ALLOWED.** When using **/pinpolicy**, PIN characters must be printable ASCII characters.
/attestation	Configures attestation (subject only). This attestation uses an Attestation Identity Key (AIK) certificate as a trust anchor to vouch that the virtual smart card keys and certificates are truly hardware bound. The attestation methods are: **AIK_AND_CERT** Creates an AIK and obtains an AIK certificate from the Microsoft cloud certification authority (CA). This requires the device to have a TPM with an EK certificate. If this option is specified and there is no network connectivity, it is possible that creation of the virtual smart card will fail. **AIK_ONLY** Creates an AIK but does not obtain an AIK certificate.
/?	Displays Help for this command.

Parameters for Destroy command

The Destroy command securely deletes a virtual smart card from a computer.

> **WARNING**
> When a virtual smart card is deleted, it cannot be recovered.

PARAMETER	DESCRIPTION
/instance	Specifies the instance ID of the virtual smart card to be removed. The instanceID was generated as output by Tpmvscmgr.exe when the card was created. The **/instance** parameter is a required field for the Destroy command.
/machine	Allows you to specify the name of a remote computer on which the virtual smart card will be deleted. This can be used in a domain environment only, and it relies on DCOM. For the command to succeed in deleting a virtual smart card on a different computer, the user running this command must be a member in the local administrators group on the remote computer.
/?	Displays Help for this command.

Remarks

Membership in the Administrators group (or equivalent) on the target computer is the minimum required to run all the parameters of this command.

For alphanumeric inputs, the full 127 character ASCII set is allowed.

Examples

The following command shows how to create a virtual smart card that can be later managed by a smart card management tool launched from another computer.

```
tpmvscmgr.exe create /name "VirtualSmartCardForCorpAccess" /AdminKey DEFAULT /PIN PROMPT
```

Alternatively, instead of using a default administrator key, you can create an administrator key at the command line. The following command shows how to create an administrator key.

```
tpmvscmgr.exe create /name "VirtualSmartCardForCorpAccess" /AdminKey PROMPT /PIN PROMPT
```

The following command will create the unmanaged virtual smart card that can be used to enroll certificates.

```
tpmvscmgr.exe create /name "VirtualSmartCardForCorpAccess" /AdminKey RANDOM /PIN PROMPT /generate
```

The preceding command will create a virtual smart card with a randomized administrator key. The key is automatically discarded after the card is created. This means that if the user forgets the PIN or wants to the change the PIN, the user needs to delete the card and create it again. To delete the card, the user can run the following command.

```
tpmvscmgr.exe destroy /instance <instance ID>
```

where <instance ID> is the value printed on the screen when the user created the card. Specifically, for the first card created, the instance ID is ROOT\SMARTCARDREADER\0000.

The following command will create a TPM virtual smart card with the default value for the administrator key and a specified PIN policy and attestation method:

```
tpmvscmgr.exe create /name "VirtualSmartCardForCorpAccess" /PIN PROMPT /pinpolicy minlen 4 maxlen 8
/AdminKey DEFAULT /attestation AIK_AND_CERT /generate
```

Additional references

- Virtual Smart Card Overview

Enterprise Certificate Pinning

11/2/2020 • 12 minutes to read • Edit Online

Applies to

- Windows 10

Enterprise certificate pinning is a Windows feature for remembering, or "pinning," a root issuing certificate authority or end entity certificate to a given domain name. Enterprise certificate pinning helps reduce man-in-the-middle attacks by enabling you to protect your internal domain names from chaining to unwanted certificates or to fraudulently issued certificates.

> **NOTE**
> External domain names, where the certificate issued to these domains is issued by a public certificate authority, are not ideal for enterprise certificate pinning.

Windows Certificate APIs (CertVerifyCertificateChainPolicy and WinVerifyTrust) are updated to check if the site's server authentication certificate chain matches a restricted set of certificates. These restrictions are encapsulated in a Pin Rules Certificate Trust List (CTL) that is configured and deployed to Windows 10 computers. Any site certificate triggering a name mismatch causes Windows to write an event to the CAPI2 event log and prevents the user from navigating to the web site using Microsoft Edge or Internet Explorer.

> **NOTE**
> Enterprise Certificate Pinning feature triggering doesn't cause clients other than Microsoft Edge or Internet Explorer to block the connection.

Deployment

To deploy enterprise certificate pinning, you need to:

- Create a well-formatted certificate pinning rule XML file
- Create a pin rules certificate trust list file from the XML file
- Apply the pin rules certificate trust list file to a reference administrative computer
- Deploy the registry configuration on the reference computer using Group Policy Management Console (GPMC), which is included in the Remote Server Administration Tools (RSAT).

Create a Pin Rules XML file

The XML-based pin rules file consists of a sequence of PinRule elements. Each PinRule element contains a sequence of one or more Site elements and a sequence of zero or more Certificate elements.

```
<PinRules ListIdentifier="PinRulesExample" Duration="P28D">

  <PinRule Name="AllCertificateAttributes" Error="None" Log="true">
    <Certificate File="Single.cer"/>
    <Certificate File="Multiple.p7b"/>
    <Certificate File="Multiple.sst"/>
    <Certificate Directory="Multiple"/>
    <Certificate Base64="MIIBy … QFzuM"/>
    <Certificate File="WillExpire.cer" EndDate="2015-05-12T00:00:00Z"/>
    <Site Domain="xyz.com"/>
  </PinRule>

  <PinRule Name="MultipleSites" Log="false">
    <Certificate File="Root.cer"/>
    <Site Domain="xyz.com"/>
    <Site Domain=".xyz.com"/>
    <Site Domain="*.abc.xyz.com" AllSubdomains="true"/>
    <Site Domain="WillNormalize.com"/>
  </PinRule>

</PinRules>
```

PinRules Element

The PinRules element can have the following attributes. For help with formatting Pin Rules, see Representing a Date in XML or Representing a Duration in XML.

ATTRIBUTE	DESCRIPTION	REQUIRED
Duration or **NextUpdate**	Specifies when the Pin Rules will expire. Either is required. **NextUpdate** takes precedence if both are specified. **Duration**, represented as an XML TimeSpan data type, does not allow years and months. You represent the **NextUpdate** attribute as a XML DateTime data type in UTC.	**Required?** Yes. At least one is required.
LogDuration or **LogEndDate**	Configures auditing only to extend beyond the expiration of enforcing the Pin Rules. **LogEndDate**, represented as an XML DateTime data type in UTC, takes precedence if both are specified. You represent **LogDuration** as an XML TimeSpan data type, which does not allow years and months. If neither attribute is specified, auditing expiration uses **Duration** or **NextUpdate** attributes.	No.
ListIdentifier	Provides a friendly name for the list of pin rules. Windows does not use this attribute for certificate pinning enforcement, however it is included when the pin rules are converted to a certificate trust list (CTL).	No.

PinRule Element

The **PinRule** element can have the following attributes.

ATTRIBUTE	DESCRIPTION	REQUIRED
Name	Uniquely identifies the **PinRule**. Windows uses this attribute to identify the element for a parsing error or for verbose output. The attribute is not included in the generated certificate trust list (CTL).	Yes.
Error	Describes the action Windows performs when it encounters a PIN mismatch. You can choose from the following string values: - **Revoked** - Windows reports the certificate protecting the site as if it was revoked. This typically prevents the user from accessing the site. - **InvalidName** - Windows reports the certificate protecting the site as if the name on the certificate does not match the name of the site. This typically results in prompting the user before accessing the site. - **None** - The default value. No error is returned. You can use this setting to audit the pin rules without introducing any user friction.	No.
Log	A Boolean value represent as string that equals **true** or **false**. By default, logging is enabled (**true**).	No.

Certificate element

The **Certificate** element can have the following attributes.

ATTRIBUTE	DESCRIPTION	REQUIRED
File	Path to a file containing one or more certificates. Where the certificate(s) can be encoded as: - single certificate - p7b - sst These files can also be Base64 formatted. All **Site** elements included in the same **PinRule** element can match any of these certificates.	Yes (File, Directory or Base64 must be present).
Directory	Path to a directory containing one or more of the above certificate files. Skips any files not containing any certificates.	Yes (File, Directory or Base64 must be present).

ATTRIBUTE	DESCRIPTION	REQUIRED
Base64	Base64 encoded certificate(s). Where the certificate(s) can be encoded as: - single certificate - p7b - sst This allows the certificates to be included in the XML file without a file directory dependency. Note: You can use **certutil -encode** to convert a .cer file into base64. You can then use Notepad to copy and paste the base64 encoded certificate into the pin rule.	Yes (File, Directory or Base64 must be present).
EndDate	Enables you to configure an expiration date for when the certificate is no longer valid in the pin rule. If you are in the process of switching to a new root or CA, you can set the **EndDate** to allow matching of this element's certificates. If the current time is past the **EndDate**, then, when creating the certificate trust list (CTL), the parser outputs a warning message and exclude the certificate(s) from the Pin Rule in the generated CTL. For help with formatting Pin Rules, see Representing a Date in XML.	No.

Site element

The **Site** element can have the following attributes.

ATTRIBUTE	DESCRIPTION	REQUIRED
Domain	Contains the DNS name to be matched for this pin rule. When creating the certificate trust list, the parser normalizes the input name string value as follows: - If the DNS name has a leading "*" it is removed. - Non-ASCII DNS name are converted to ASCII Puny Code. - Upper case ASCII characters are converted to lower case. If the normalized name has a leading ".", then, wildcard left hand label matching is enabled. For example, ".xyz.com" would match "abc.xyz.com".	Yes.

ATTRIBUTE	DESCRIPTION	REQUIRED
AllSubdomains	By default, wildcard left hand label matching is restricted to a single left hand label. This attribute can be set to "true" to enable wildcard matching of all of the left-hand labels. For example, setting this attribute would also match "123.abc.xyz.com" for the ".xyz.com" domain value.	No.

Create a Pin Rules Certificate Trust List

The command line utility, **Certutil.exe**, includes the **generatePinRulesCTL** argument to parse the XML file and generate the encoded certificate trust list (CTL) that you add to your reference Windows 10 version 1703 computer and subsequently deploy. The usage syntax is:

```
CertUtil [Options] -generatePinRulesCTL XMLFile CTLFile [SSTFile]
  Generate Pin Rules CTL
    XMLFile -- input XML file to be parsed.
    CTLFile -- output CTL file to be generated.
    SSTFile -- optional .sst file to be created.
         The .sst file contains all of the certificates
         used for pinning.

Options:
  -f                -- Force overwrite
  -v                -- Verbose operation
```

The same certificate(s) can occur in multiple **PinRule** elements. The same domain can occur in multiple **PinRule** elements. Certutil coalesces these in the resultant pin rules certificate trust list.

Certutil.exe does not strictly enforce the XML schema definition. It does perform the following to enable other tools to add/consume their own specific elements and attributes:

- Skips elements before and after the **PinRules** element.
- Skips any element not matching **Certificate** or **Site** within the **PinRules** element.
- Skips any attributes not matching the above names for each element type.

Use the **certutil** command with the **generatePinRulesCTL** argument along with your XML file that contains your certificate pinning rules. Lastly, provide the name of an output file that will include your certificate pinning rules in the form of a certificate trust list.

```
certutil -generatePinRulesCTL certPinRules.xml pinrules.stl
```

Applying Certificate Pinning Rules to a Reference Computer

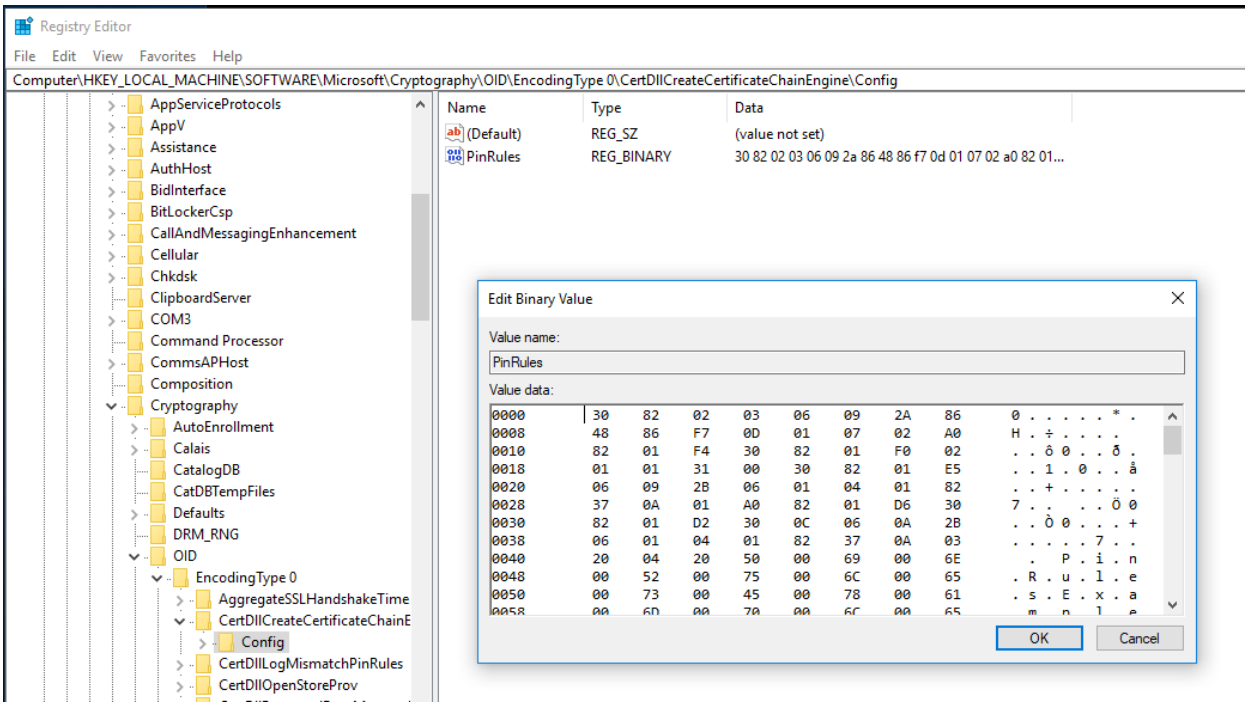
Now that your certificate pinning rules are in the certificate trust list format, you need to apply the settings to a reference computer as a prerequisite to deploying the setting to your enterprise. To simplify the deployment configuration, it is best to apply your certificate pinning rules to a computer that has the Group Policy Management Console (GPMC) that is include in the Remote Server Administration Tools (RSAT).

Use **certutil.exe** to apply your certificate pinning rules to your reference computer using the **setreg** argument. The **setreg** argument takes a secondary argument that determines the location of where certutil writes the certificate pining rules. This secondary argument is **chain\PinRules**. The last argument you provide is the name of file that contains your certificate pinning rules in certificate trust list format (.stl). You'll pass the name of the file as the last argument; however, you need to prefix the file name with the '@' symbol as shown in the following example. You need to perform this command from an elevated command prompt.

```
Certutil -setreg chain\PinRules @pinrules.stl
```

Certutil writes the binary information to the following registration location:

NAME	VALUE
Key	HKLM\SOFTWARE\Microsoft\Cryptography\OID\EncodingType0\CertDllCreateCertificateChainEngine\Config
Name	PinRules
Value	Binary contents from the certificate pin rules certificate trust list file
Data type	REG_BINARY



Deploying Enterprise Pin Rule Settings using Group Policy

You've successfully created a certificate pinning rules XML file. From the XML file you have created a certificate pinning trust list file, and you have applied the contents of that file to your reference computer from which you can run the Group Policy Management Console. Now you need to configure a Group Policy object to include the applied certificate pin rule settings and deploy it to your environment.

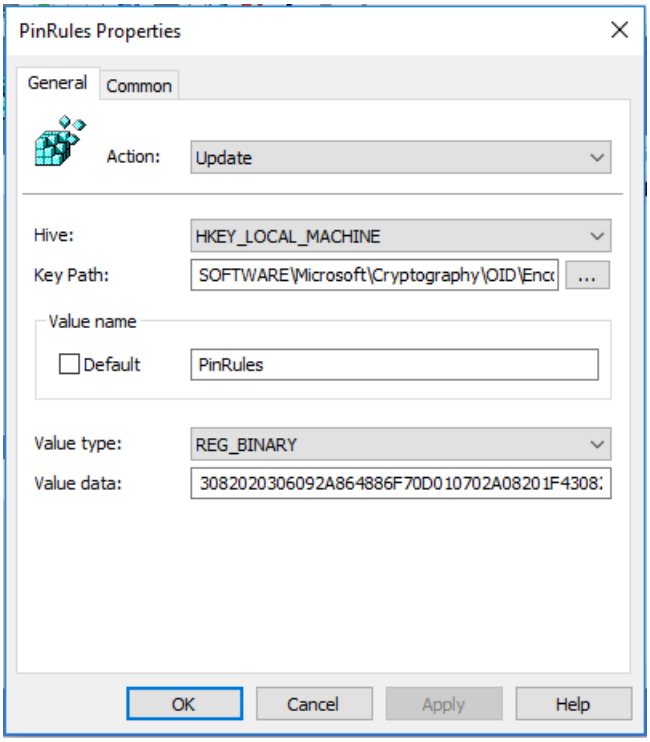
Sign-in to the reference computer using domain administrator equivalent credentials.

1. Start the **Group Policy Management Console** (gpmc.msc)
2. In the navigation pane, expand the forest node and then expand the domain node.
3. Expand the node that has contains your Active Directory's domain name
4. Select the **Group Policy objects** node. Right-click the **Group Policy objects** node and click **New**.
5. In the **New GPO** dialog box, type *Enterprise Certificate Pinning Rules* in the **Name** text box and click **OK**.
6. In the content pane, right-click the **Enterprise Certificate Pinning Rules** Group Policy object and click **Edit**.

7. In the **Group Policy Management Editor**, in the navigation pane, expand the **Preferences** node under **Computer Configuration**. Expand **Windows Settings**.
8. Right-click the **Registry** node and click **New**.
9. In the **New Registry Properties** dialog box, select **Update** from the **Action** list. Select **HKEY_LOCAL_MACHINE** from the **Hive** list.
10. For the **Key Path**, click **...** to launch the **Registry Item Browser**. Navigate to the following registry key and select the **PinRules** registry value name:

    HKLM\SOFTWARE\Microsoft\Cryptography\OID\EncodingType0\CertDllCreateCertificateChainEngine\Config

    Click **Select** to close the **Registry Item Browser**.
11. The **Key Path** should contain the selected registry key. The **Value name** configuration should contain the registry value name ***PinRules***. **Value type** should read ***REG_BINARY*** and **Value data** should contain a long series of numbers from 0-9 and letters ranging from A-F (hexadecimal). Click **OK** to save your settings and close the dialog box.



12. Close the **Group Policy Management Editor** to save your settings.
13. Link the **Enterprise Certificate Pinning Rules** Group Policy object to apply to computers that run Windows 10, version 1703 in your enterprise. When these domain-joined computers apply Group Policy, the registry information configured in the Group Policy object is applied to the computer.

Additional Pin Rules Logging

To assist in constructing certificate pinning rules, you can configure the **PinRulesLogDir** setting under the certificate chain configuration registry key to include a parent directory to log pin rules.

NAME	VALUE
Key	HKLM\SOFTWARE\Microsoft\Cryptography\OID\EncodingType0\CertDllCreateCertificateChainEngine\Config

NAME	VALUE
Name	PinRulesLogDir
Value	The Parent directory where Windows should write the additional pin rule logs
Data type	REG_SZ

Permission for the Pin Rule Log Folder

The folder in which Windows writes the additional pin rule logs must have permissions so that all users and applications have full access. You can run the following commands from an elevated command prompt to achieved the proper permissions.

```
set PinRulesLogDir=c:\PinRulesLog
mkdir %PinRulesLogDir%
icacls %PinRulesLogDir% /grant *S-1-15-2-1:(OI)(CI)(F)
icacls %PinRulesLogDir% /grant *S-1-1-0:(OI)(CI)(F)
icacls %PinRulesLogDir% /grant *S-1-5-12:(OI)(CI)(F)
icacls %PinRulesLogDir% /inheritance:e /setintegritylevel (OI)(CI)L
```

Whenever an application verifies a TLS/SSL certificate chain that contains a server name matching a DNS name in the server certificate, Windows writes a .p7b file consisting of all the certificates in the server's chain to one of three child folders:

- AdminPinRules Matched a site in the enterprise certificate pinning rules.
- AutoUpdatePinRules Matched a site in the certificate pinning rules managed by Microsoft.
- NoPinRules Didn't match any site in the certificate pin rules.

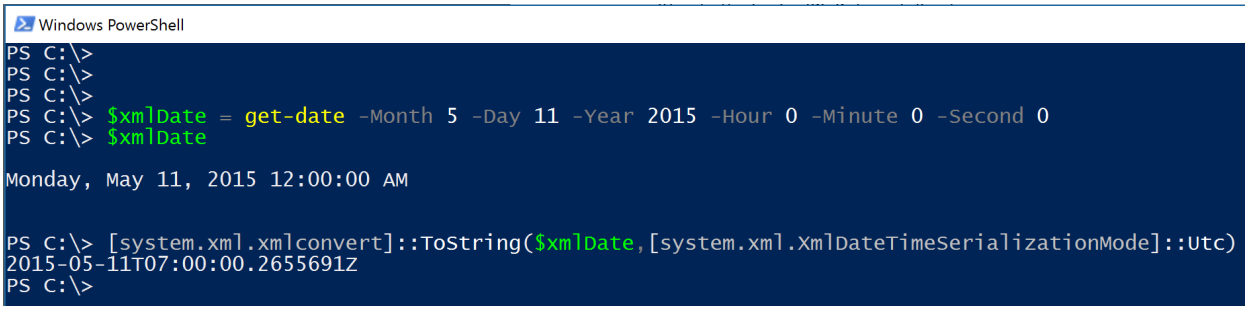
The output file name consists of the leading 8 ASCII hex digits of the root's SHA1 thumbprint followed by the server name. For example:

- D4DE20D0_xsi.outlook.com.p7b
- DE28F4A4_www.yammer.com.p7b

If there is either an enterprise certificate pin rule or Microsoft certificate pin rule mismatch, then Windows writes the .p7b file to the **MismatchPinRules** child folder. If the pin rules have expired, then Windows writes the .p7b to the **ExpiredPinRules** child folder.

Representing a Date in XML

Many attributes within the pin rules xml file are dates.
These dates must be properly formatted and represented in UTC.
You can use Windows PowerShell to format these dates.
You can then copy and paste the output of the cmdlet into the XML file.



For simplicity, you can truncate decimal point (.) and the numbers after it. However, be certain to append the uppercase "Z" to the end of the XML date string.

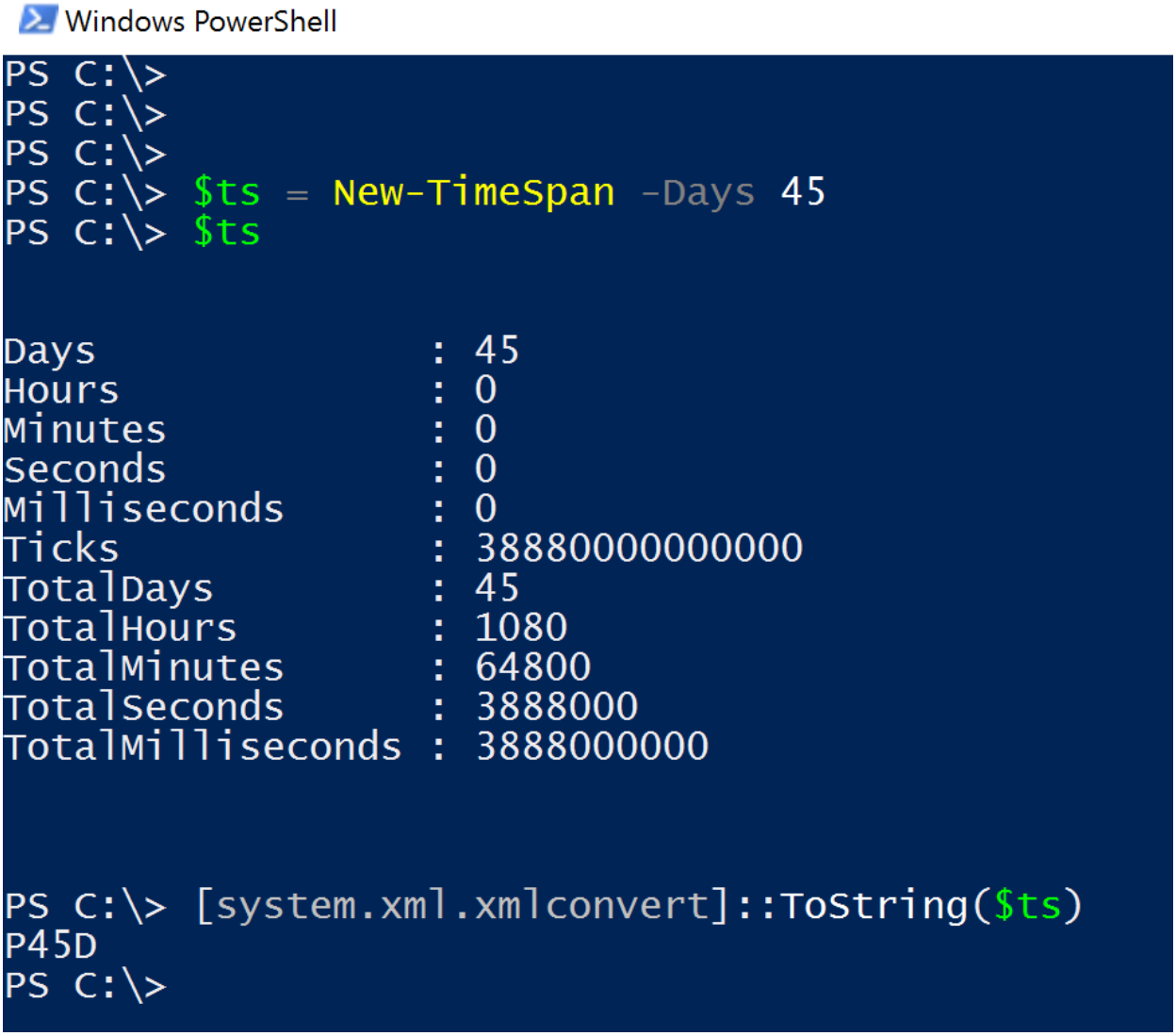
```
2015-05-11T07:00:00.2655691Z
2015-05-11T07:00:00Z
```

Converting an XML Date

You can also use Windows PowerShell to validate convert an XML date into a human readable date to validate it's the correct date.

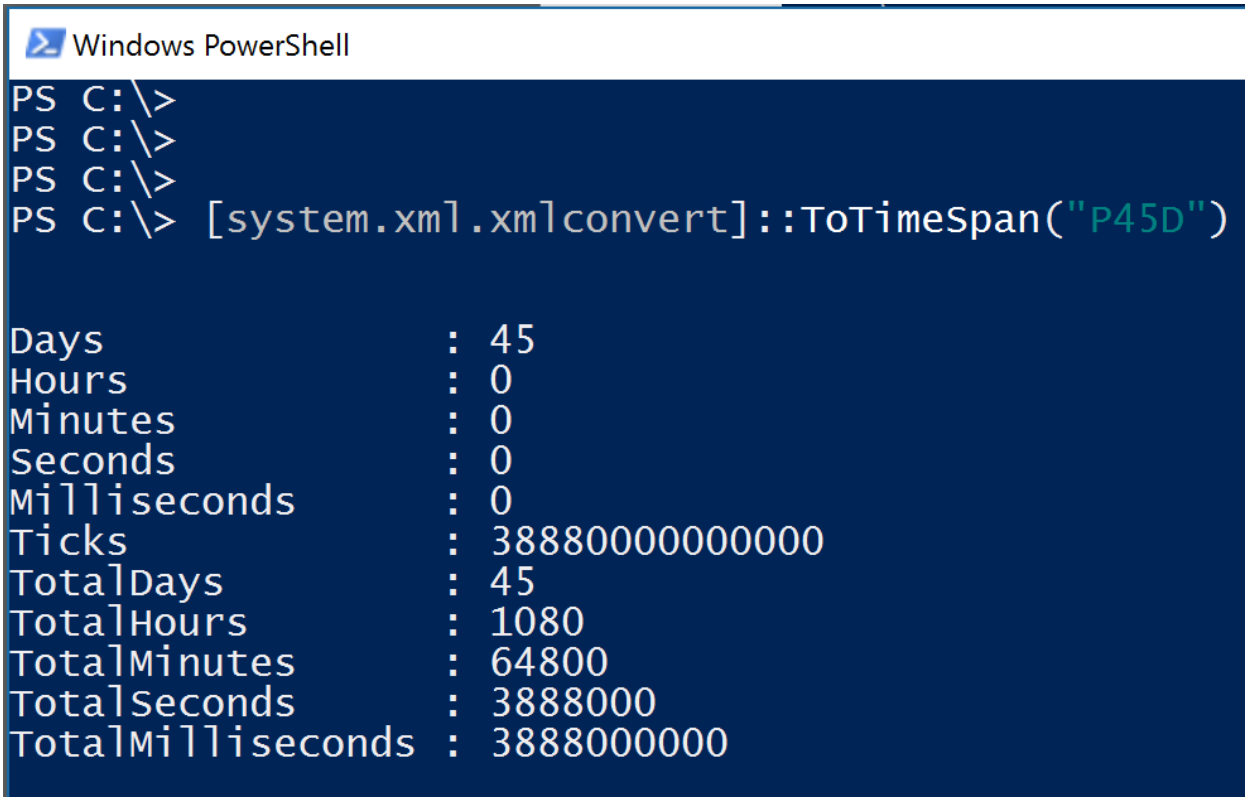
Representing a Duration in XML

Some elements may be configured to use a duration rather than a date. You must represent the duration as an XML timespan data type. You can use Windows PowerShell to properly format and validate durations (timespans) and copy and paste them into your XML file.



Converting an XML Duration

You can convert a XML formatted timespan into a timespan variable that you can read.



Certificate Trust List XML Schema Definition (XSD)

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="PinRules">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="PinRule" maxOccurs="unbounded" minOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Certificate" maxOccurs="unbounded" minOccurs="0">
                <xs:complexType>
                  <xs:simpleContent>
                    <xs:extension base="xs:string">
                      <xs:attribute type="xs:dateTime" name="EndDate" use="optional"/>
                      <xs:attribute type="xs:string" name="File" use="optional"/>
                      <xs:attribute type="xs:string" name="Directory" use="optional"/>
                      <xs:attribute type="xs:base64Binary" name="Base64" use="optional"/>
                    </xs:extension>
                  </xs:simpleContent>
                </xs:complexType>
              </xs:element>
              <xs:element name="Site" maxOccurs="unbounded" minOccurs="1">
                <xs:complexType>
                  <xs:simpleContent>
                    <xs:extension base="xs:string">
                      <xs:attribute type="xs:string" name="Domain"/>
                      <xs:attribute type="xs:boolean" name="AllSubdomains" use="optional" default="false"/>
                    </xs:extension>
                  </xs:simpleContent>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
            <xs:attribute type="xs:string" name="Name"/>
            <xs:attribute name="Error" use="optional" default="None">
              <xs:simpleType>
                <xs:restriction base="xs:string">
                  <xs:enumeration value ="Revoked"/>
                  <xs:enumeration value ="InvalidName"/>
                  <xs:enumeration value ="None"/>
                </xs:restriction>
              </xs:simpleType>
            </xs:attribute>
            <xs:attribute type="xs:boolean" name="Log" use="optional" default="true"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute type="xs:duration" name="Duration" use="optional"/>
      <xs:attribute type="xs:duration" name="LogDuration" use="optional"/>
      <xs:attribute type="xs:dateTime" name="NextUpdate" use="optional"/>
      <xs:attribute type="xs:dateTime" name="LogEndDate" use="optional"/>
      <xs:attribute type="xs:string" name="ListIdentifier" use="optional"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Install digital certificates on Windows 10 Mobile

3/26/2021 • 3 minutes to read • Edit Online

Applies to

- Windows 10 Mobile

Digital certificates bind the identity of a user or computer to a pair of keys that can be used to encrypt and sign digital information. Certificates are issued by a certification authority (CA) that vouches for the identity of the certificate holder, and they enable secure client communications with websites and services.

Certificates in Windows 10 Mobile are primarily used for the following purposes:

- To create a secure channel using Secure Sockets Layer (SSL) between a phone and a web server or service.
- To authenticate a user to a reverse proxy server that is used to enable Microsoft Exchange ActiveSync (EAS) for email.
- For installation and licensing of applications (from the Windows Phone Store or a custom company distribution site).

WARNING

In Windows 10, Version 1607, if you have multiple certificates provisioned on the device and the Wi-Fi profile provisioned does not have a strict filtering criteria, you may see connection failures when connecting to Wi-Fi. Learn more about this known issue in Version 1607

Install certificates using Microsoft Edge

A certificate can be posted on a website and made available to users through a device-accessible URL that they can use to download the certificate. When a user accesses the page and taps the certificate, it opens on the device. The user can inspect the certificate, and if they choose to continue, the certificate is installed on the Windows 10 Mobile device.

Install certificates using email

The Windows 10 Mobile certificate installer supports .cer, .p7b, .pem, and .pfx files. Some email programs block .cer files for security reasons. If this is the case in your organization, use an alternative method to deploy the certificate. Certificates that are sent via email appear as message attachments. When a certificate is received, a user can tap to review the contents and then tap to install the certificate. Typically, when an identity certificate is installed, the user is prompted for the password (or passphrase) that protects it.

Install certificates using mobile device management (MDM)

Windows 10 Mobile supports root, CA, and client certificate to be configured via MDM. Using MDM, an administrator can directly add, delete, or query root and CA certificates, and configure the device to enroll a client certificate with a certificate enrollment server that supports Simple Certificate Enrollment Protocol (SCEP). SCEP enrolled client certificates are used by Wi-Fi, VPN, email, and browser for certificate-based client authentication. An MDM server can also query and delete SCEP enrolled client certificate (including user installed certificates), or trigger a new enrollment request before the current certificate is expired.

WARNING

Do not use SCEP for encryption certificates for S/MIME. You must use a PFX certificate profile to support S/MIME on Windows 10 Mobile. For instructions on creating a PFX certificate profile in Microsoft Intune, see Enable access to company resources using certificate profiles with Microsoft Intune.

Process of installing certificates using MDM

1. The MDM server generates the initial cert enroll request including challenge password, SCEP server URL, and other enrollment related parameters.
2. The policy is converted to the OMA DM request and sent to the device.
3. The trusted CA certificate is installed directly during MDM request.
4. The device accepts certificate enrollment request.
5. The device generates private/public key pair.
6. The device connects to Internet-facing point exposed by MDM server.
7. MDM server creates a certificate that is signed with proper CA certificate and returns it to device.

   NOTE

   The device supports the pending function to allow server side to do additional verification before issuing the cert. In this case, a pending status is sent back to the device. The device will periodically contact the server, based on preconfigured retry count and retry period parameters. Retrying ends when either:

   - A certificate is successfully received from the server
   - The server returns an error
   - The number of retries reaches the preconfigured limit

8. The cert is installed in the device. Browser, Wi-Fi, VPN, email, and other first party applications have access to this certificate.

   NOTE

   If MDM requested private key stored in Trusted Process Module (TPM) (configured during enrollment request), the private key will be saved in TPM. Note that SCEP enrolled cert protected by TPM isn't guarded by a PIN. However, if the certificate is imported to the Windows Hello for Business Key Storage Provider (KSP), it is guarded by the Hello PIN.

Related topics

Configure S/MIME

Windows 10 Credential Theft Mitigation Guide Abstract

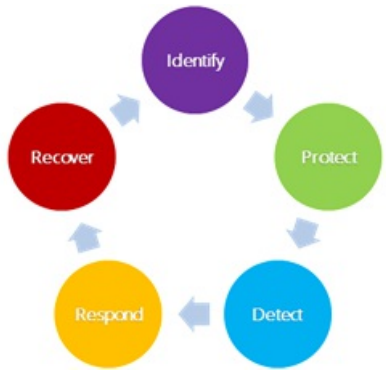
3/5/2021 • 2 minutes to read • Edit Online

Applies to

- Windows 10

This topic provides a summary of the Windows 10 credential theft mitigation guide, which can be downloaded from the Microsoft Download Center. This guide explains how credential theft attacks occur and the strategies and countermeasures you can implement to mitigate them, following these security stages:

- Identify high-value assets
- Protect against known and unknown threats
- Detect pass-the-hash and related attacks
- Respond to suspicious activity
- Recover from a breach



Attacks that steal credentials

Learn about the different types of attacks that are used to steal credentials, and the factors that can place your organization at risk. The types of attacks that are covered include:

- Pass the hash
- Kerberos pass the ticket
- Kerberos golden ticket and silver ticket
- Key loggers
- Shoulder surfing

Credential protection strategies

This part of the guide helps you consider the mindset of the attacker, with prescriptive guidance about how to prioritize high-value accounts and computers. You'll learn how to architect a defense against credential theft:

- Establish a containment model for account privileges
- Harden and restrict administrative hosts
- Ensure that security configurations and best practices are implemented

Technical countermeasures for credential theft

Objectives and expected outcomes are covered for each of these countermeasures:

- Use Windows 10 with Credential Guard
- Restrict and protect high-privilege domain accounts
- Restrict and protect local accounts with administrative privileges
- Restrict inbound network traffic

Many other countermeasures are also covered, such as using Microsoft Passport and Windows Hello, or multifactor authentication.

Detecting credential attacks

This sections covers how to detect the use of stolen credentials and how to collect computer events to help you detect credential theft.

Responding to suspicious activity

Learn Microsoft's recommendations for responding to incidents, including how to recover control of compromised accounts, how to investigate attacks, and how to recover from a breach.

Configure S/MIME for Windows 10 and Windows 10 Mobile

3/26/2021 • 2 minutes to read • Edit Online

Applies to

- Windows 10
- Windows 10 Mobile

S/MIME stands for Secure/Multipurpose Internet Mail Extensions, and provides an added layer of security for email sent to and from an Exchange ActiveSync (EAS) account. In Windows 10, S/MIME lets users encrypt outgoing messages and attachments so that only intended recipients who have a digital identification (ID), also known as a certificate, can read them. Users can digitally sign a message, which provides the recipients with a way to verify the identity of the sender and that the message hasn't been tampered with.

About message encryption

Users can send encrypted message to people in their organization and people outside their organization if they have their encryption certificates. However, users using Windows 10 Mail app can only read encrypted messages if the message is received on their Exchange account and they have corresponding decryption keys.

Encrypted messages can be read only by recipients who have a certificate. If you try to send an encrypted message to recipient(s) whose encryption certificate are not available, the app will prompt you to remove these recipients before sending the email.

About digital signatures

A digitally signed message reassures the recipient that the message hasn't been tampered with and verifies the identity of the sender. Recipients can only verify the digital signature if they're using an email client that supports S/MIME.

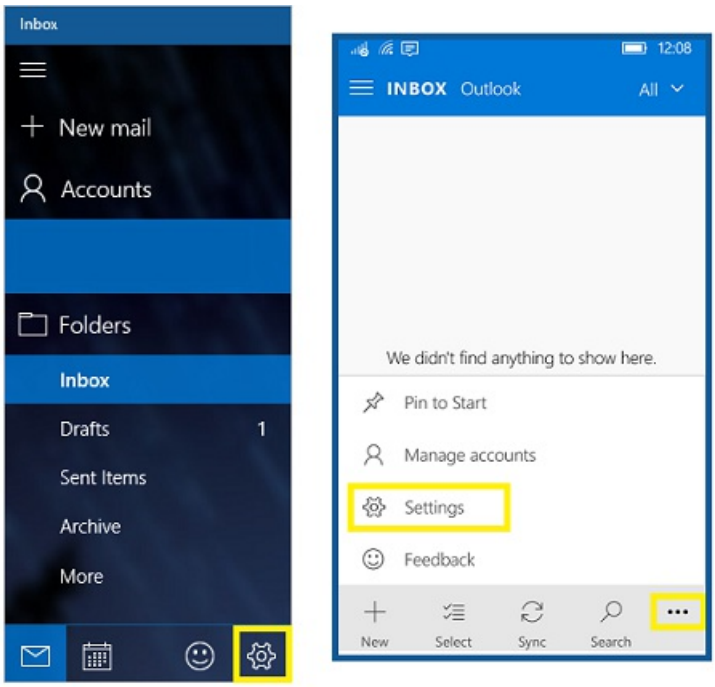
Prerequisites

- S/MIME is enabled for Exchange accounts (on-premises and Office 365). Users can't use S/MIME signing and encryption with a personal account such as Outlook.com.
- Valid Personal Information Exchange (PFX) certificates are installed on the device.
  - How to Create PFX Certificate Profiles in Configuration Manager
  - Enable access to company resources using certificate profiles with Microsoft Intune
  - Install digital certificates on Windows 10 Mobile

Choose S/MIME settings

On the device, perform the following steps: (add select certificate)

1. Open the Mail app. (In Windows 10 Mobile, the app is Outlook Mail.)
2. Open **Settings** by tapping the gear icon on a PC, or the ellipsis (...) and then the gear icon on a phone.



Desktop app Phone app

3. Tap **Email security**.

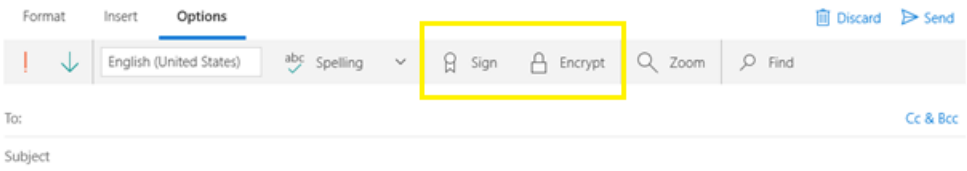
4. In **Select an account**, select the account for which you want to configure S/MIME options.
5. Make a certificate selection for digital signature and encryption.
   - Select **Automatically** to let the app choose the certificate.
   - Select **Manually** to specify the certificate yourself from the list of valid certificates on the device.
6. (Optional) Select **Always sign with S/MIME**, **Always encrypt with S/MIME**, or both, to automatically digitally sign or encrypt all outgoing messages.

   > **Note:** The option to sign or encrypt can be changed for individual messages, unless EAS policies prevent it.

7. Tap the back arrow.

Encrypt or sign individual messages

1. While composing a message, choose **Options** from the ribbon. On phone, **Options** can be accessed by tapping the ellipsis (...).
2. Use **Sign** and **Encrypt** icons to turn on digital signature and encryption for this message.



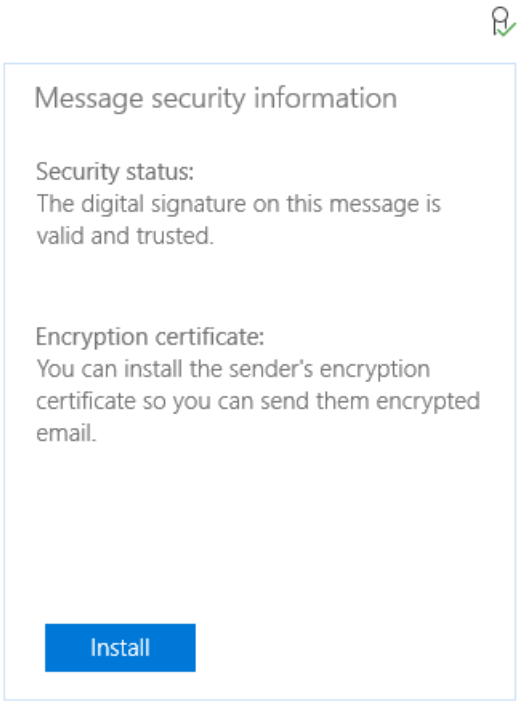
Read signed or encrypted messages

When you receive an encrypted message, the mail app will check whether there is a certificate available on your computer. If there is a certificate available, the message will be decrypted when you open it. If your certificate is stored on a smartcard, you will be prompted to insert the smartcard to read the message. Your smartcard may also require a PIN to access the certificate.

Install certificates from a received message

When you receive a signed email, the app provide feature to install corresponding encryption certificate on your device if the certificate is available. This certificate can then be used to send encrypted email to this person.

1. Open a signed email.
2. Tap or click the digital signature icon in the reading pane.
3. Tap **Install.**



Windows 10 VPN technical guide

3/26/2021 • 2 minutes to read • Edit Online

Applies to

- Windows 10
- Windows 10 Mobile

This guide will walk you through the decisions you will make for Windows 10 clients in your enterprise VPN solution and how to configure your deployment. This guide references the VPNv2 Configuration Service Provider (CSP) and provides mobile device management (MDM) configuration instructions using Microsoft Intune and the VPN Profile template for Windows 10.

To create a Windows 10 VPN device configuration profile see: Windows 10 and Windows Holographic device settings to add VPN connections using Intune.

> **NOTE**
> This guide does not explain server deployment.

In this guide

TOPIC	DESCRIPTION
VPN connection types	Select a VPN client and tunneling protocol
VPN routing decisions	Choose between split tunnel and force tunnel configuration
VPN authentication options	Select a method for Extensible Authentication Protocol (EAP) authentication.
VPN and conditional access	Use Azure Active Directory policy evaluation to set access policies for VPN connections.
VPN name resolution	Decide how name resolution should work
VPN auto-triggered profile options	Set a VPN profile to connect automatically by app or by name, to be "always on", and to not trigger VPN on trusted networks
VPN security features	Set a LockDown VPN profile, configure traffic filtering, and connect VPN profile to Windows Information Protection (WIP)
VPN profile options	Combine settings into single VPN profile using XML

Learn more

- Create VPN profiles to connect to VPN servers in Intune

VPN connection types

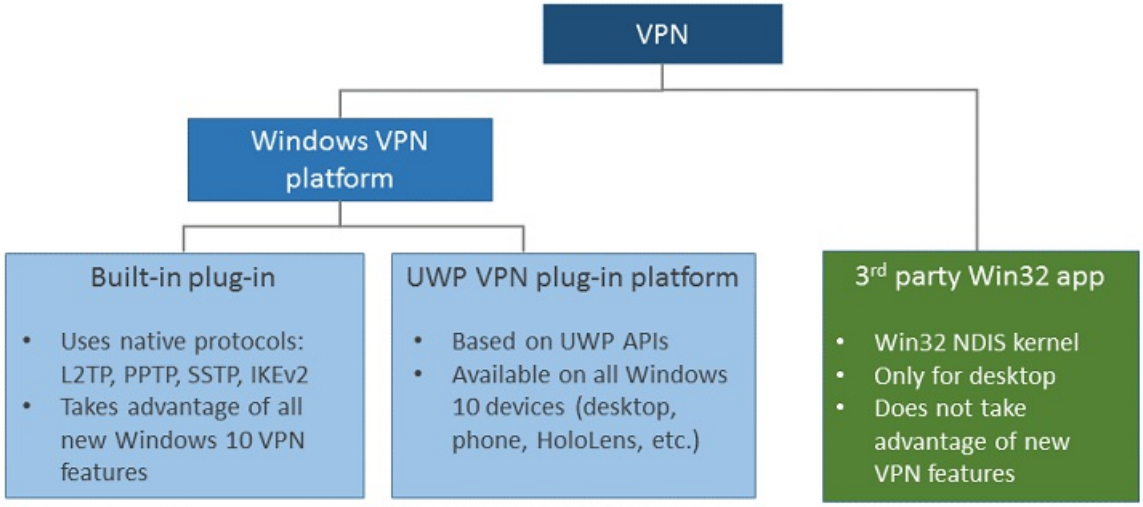
3/26/2021 • 2 minutes to read • Edit Online

Applies to

- Windows 10
- Windows 10 Mobile

Virtual private networks (VPNs) are point-to-point connections across a private or public network, such as the Internet. A VPN client uses special TCP/IP or UDP-based protocols, called *tunneling protocols*, to make a virtual call to a virtual port on a VPN server. In a typical VPN deployment, a client initiates a virtual point-to-point connection to a remote access server over the Internet. The remote access server answers the call, authenticates the caller, and transfers data between the VPN client and the organization's private network.

There are many options for VPN clients. In Windows 10, the built-in plug-in and the Universal Windows Platform (UWP) VPN plug-in platform are built on top of the Windows VPN platform. This guide focuses on the Windows VPN platform clients and the features that can be configured.



Built-in VPN client

- Tunneling protocols
  - Internet Key Exchange version 2 (IKEv2)

    Configure the IPsec/IKE tunnel cryptographic properties using the **Cryptography Suite** setting in the VPNv2 Configuration Service Provider (CSP).
  - L2TP

    L2TP with pre-shared key (PSK) authentication can be configured using the **L2tpPsk** setting in the VPNv2 CSP.
  - PPTP
  - SSTP

    SSTP is supported for Windows desktop editions only. SSTP cannot be configured using mobile device management (MDM), but it is one of the protocols attempted in the **Automatic** option.

> **NOTE**
>
> When a VPN plug-in is used, the adapter will be listed as an SSTP adapter, even though the VPN protocol used is the plug-in's protocol.

- Automatic

  The **Automatic** option means that the device will try each of the built-in tunneling protocols until one succeeds. It will attempt from most secure to least secure.

  Configure **Automatic** for the **NativeProtocolType** setting in the VPNv2 CSP.

Universal Windows Platform VPN plug-in

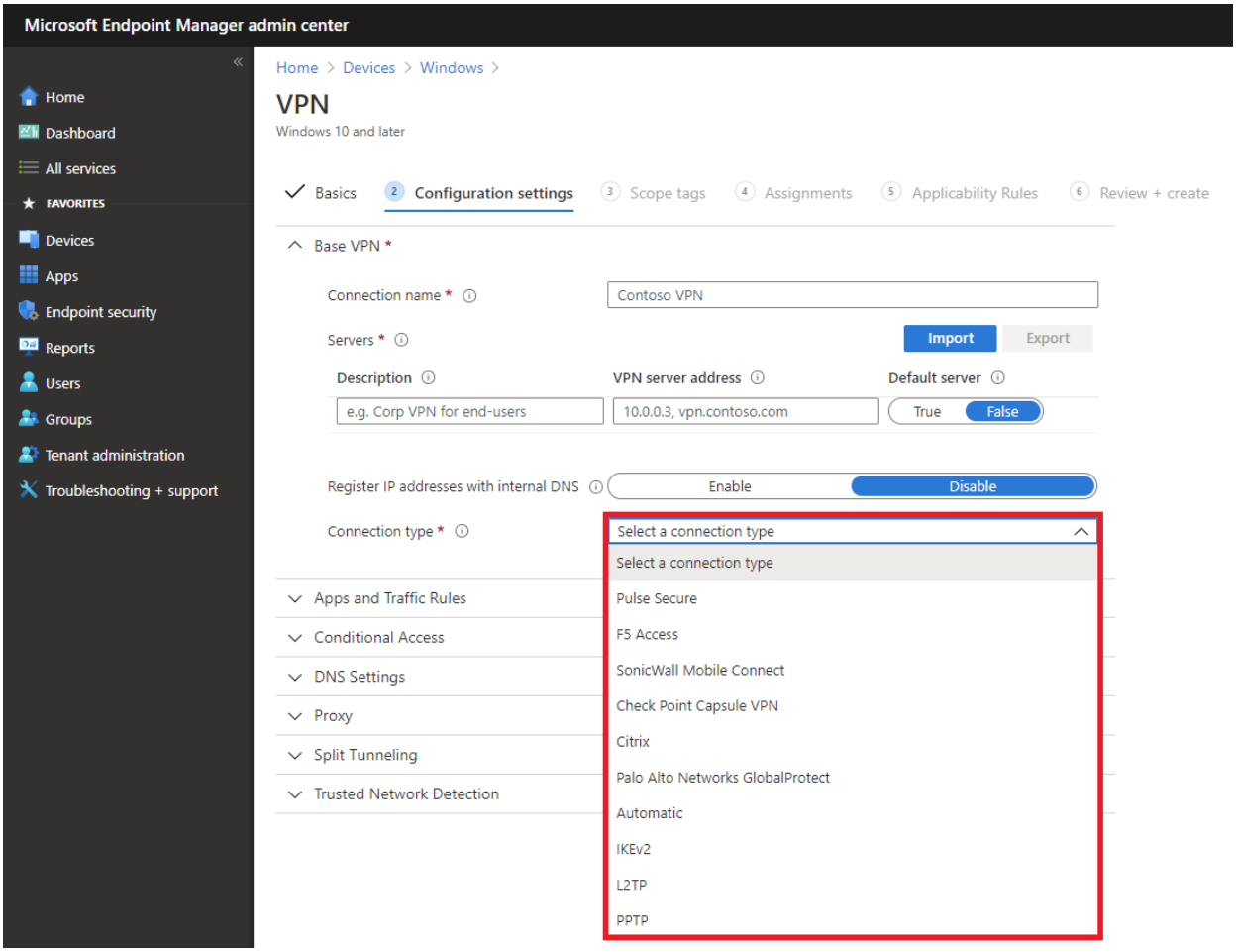
The Universal Windows Platform (UWP) VPN plug-ins were introduced in Windows 10, although there were originally separate versions available for the Windows 8.1 Mobile and Windows 8.1 PC platforms. Using the UWP platform, third-party VPN providers can create app-containerized plug-ins using WinRT APIs, eliminating the complexity and problems often associated with writing to system-level drivers.

There are a number of Universal Windows Platform VPN applications, such as Pulse Secure, Cisco AnyConnect, F5 Access, Sonicwall Mobile Connect, and Check Point Capsule. If you want to use a UWP VPN plug-in, work with your vendor for any custom settings needed to configure your VPN solution.

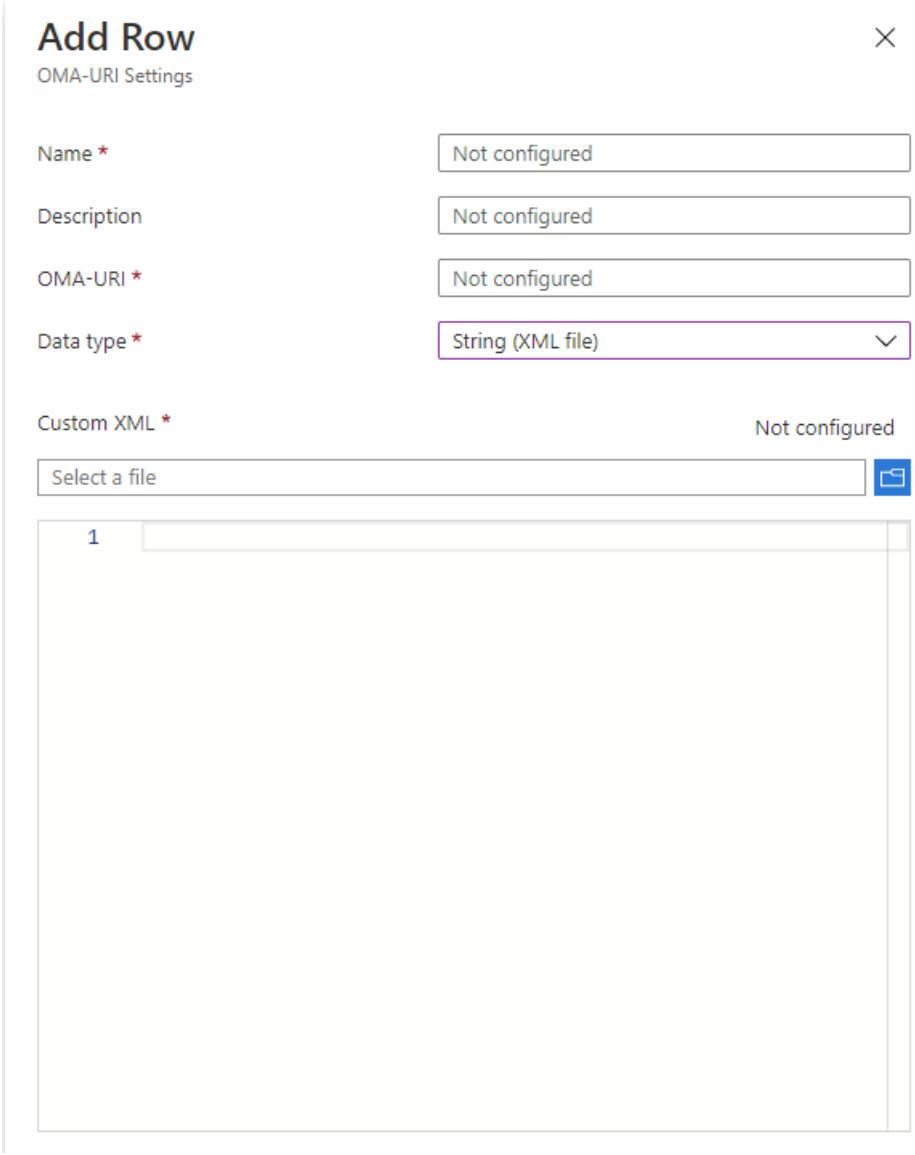
Configure connection type

See VPN profile options and VPNv2 CSP for XML configuration.

The following image shows connection options in a VPN Profile configuration policy using Microsoft Intune:



In Intune, you can also include custom XML for third-party plug-in profiles:



Related topics

- VPN technical guide
- VPN routing decisions
- VPN authentication options
- VPN and conditional access
- VPN name resolution
- VPN auto-triggered profile options
- VPN security features
- VPN profile options

VPN routing decisions

3/26/2021 • 2 minutes to read • Edit Online

Applies to

- Windows 10
- Windows 10 Mobile

Network routes are required for the stack to understand which interface to use for outbound traffic. One of the most important decision points for VPN configuration is whether you want to send all the data through VPN (*force tunnel*) or only some data through the VPN (*split tunnel*). This decision impacts the configuration and the capacity planning, as well as security expectations from the connection.

Split tunnel configuration

In a split tunnel configuration, routes can be specified to go over VPN and all other traffic will go over the physical interface.

Routes can be configured using the VPNv2/*ProfileName*/RouteList setting in the VPNv2 Configuration Service Provider (CSP).

For each route item in the list, the following can be specified:

- **Address**: VPNv2/*ProfileName*/RouteList/*routeRowId*/Address
- **Prefix size**: VPNv2/*ProfileName*/RouteList/*routeRowId*/Prefix
- **Exclusion route**: VPNv2/*ProfileName*/RouteList/*routeRowId*/ExclusionRoute

  Windows VPN platform now supports the ability to specify exclusion routes that specifically should not go over the physical interface.

Routes can also be added at connect time through the server for UWP VPN apps.

Force tunnel configuration

In a force tunnel configuration, all traffic will go over VPN. This is the default configuration and takes effect if no routes are specified.

The only implication of this setting is the manipulation of routing entries. In the case of a force tunnel, VPN V4 and V6 default routes (for example. 0.0.0.0/0) are added to the routing table with a lower metric than ones for other interfaces. This sends traffic through the VPN as long as there isn't a specific route on the physical interface itself.

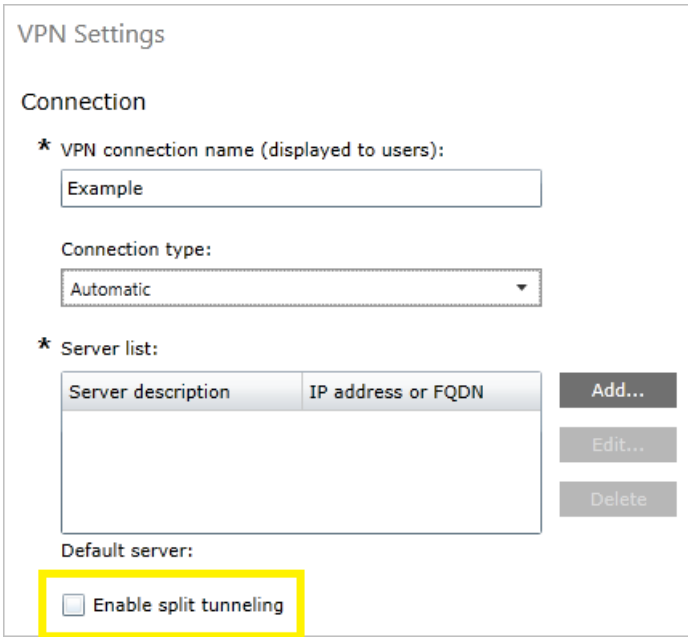
For built-in VPN, this decision is controlled using the MDM setting **VPNv2/ProfileName/NativeProfile/RoutingPolicyType**.

For a UWP VPN plug-in, this property is directly controlled by the app. If the VPN plug-in indicates the default route for IPv4 and IPv6 as the only two Inclusion routes, the VPN platform marks the connection as Force Tunneled.

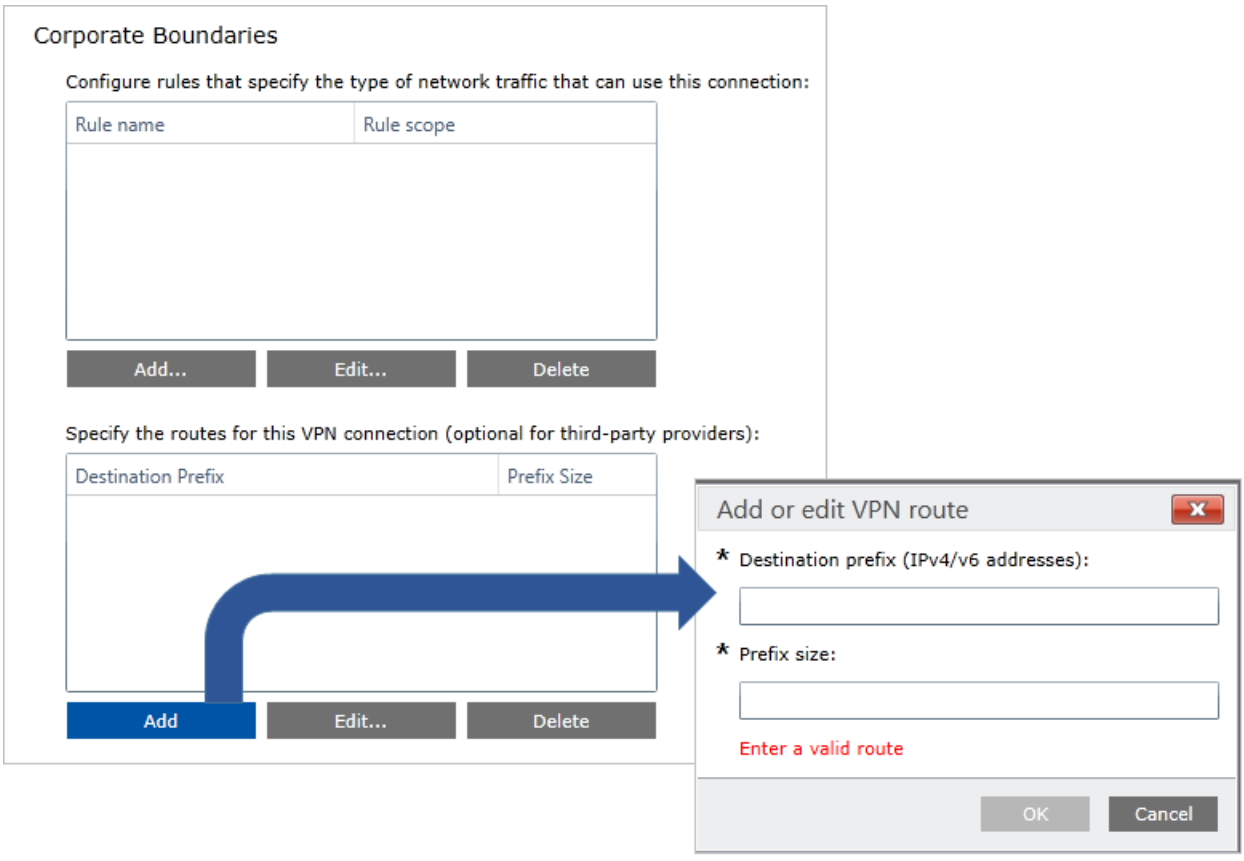
Configure routing

See VPN profile options and VPNv2 CSP for XML configuration.

When you configure a VPN profile in Microsoft Intune, you select a checkbox to enable split tunnel configuration.



Next, in **Corporate Boundaries**, you add the routes that should use the VPN connection.



Related topics

- VPN technical guide
- VPN connection types
- VPN authentication options
- VPN and conditional access
- VPN name resolution
- VPN auto-triggered profile options

- VPN security features
- VPN profile options

VPN authentication options

3/26/2021 • 2 minutes to read • Edit Online

Applies to

- Windows 10
- Windows 10 Mobile

In addition to older and less-secure password-based authentication methods (which should be avoided), the built-in VPN solution uses Extensible Authentication Protocol (EAP) to provide secure authentication using both user name and password, and certificate-based methods. You can only configure EAP-based authentication if you select a built-in VPN type (IKEv2, L2TP, PPTP or Automatic).

Windows supports a number of EAP authentication methods.

METHOD	DETAILS
EAP-Microsoft Challenge Handshake Authentication Protocol version 2 (EAP-MSCHAPv2)	• User name and password authentication • Winlogon credentials - can specify authentication with computer sign-in credentials

EAP-Transport Layer Security (EAP-TLS)	• Supports the following types of certificate authentication
	  ◦ Certificate with keys in the software Key Storage Provider (KSP)
	  ◦ Certificate with keys in Trusted Platform Module (TPM) KSP
	  ◦ Smart card certficates
	  ◦ Windows Hello for Business certificate
	• Certificate filtering
	  ◦ Certificate filtering can be enabled to search for a particular certificate to use to authenticate with
	  ◦ Filtering can be Issuer-based or Enhanced Key Usage (EKU)-based
	• Server validation - with TLS, server validation can be toggled on or off
	  ◦ Server name - specify the server to validate
	  ◦ Server certificate - trusted root certificate to validate the server
	  ◦ Notification - specify if the user should get a notification asking whether to trust the server or not

METHOD	DETAILS

Protected Extensible Authentication Protocol (PEAP)	• Server validation - with PEAP, server validation can be toggled on or off
	  ◦ Server name - specify the server to validate
	  ◦ Server certificate - trusted root certificate to validate the server
	  ◦ Notification - specify if the user should get a notification asking whether to trust the server or not
	• Inner method - the outer method creates a secure tunnel inside while the inner method is used to complete the authentication
	  ◦ EAP-MSCHAPv2
	  ◦ EAP-TLS
	• Fast Reconnect: reduces the delay between an authentication request by a client and the response by the Network Policy Server (NPS) or other Remote Authentication Dial-in User Service (RADIUS) server. This reduces resource requirements for both client and server, and minimizes the number of times that users are prompted for credentials.
	• Cryptobinding: By deriving and exchanging values from the PEAP phase 1 key material (**Tunnel Key**) and from the PEAP phase 2 inner EAP method key material (**Inner Session Key**), it is possible to prove that the two authentications terminate at the same two entities (PEAP peer and PEAP server). This process, termed "cryptobinding", is used to protect the PEAP negotiation against "Man in the Middle" attacks.

Tunneled Transport Layer Security (TTLS)	• Inner method
	  ◦ Non-EAP
	  ◦ Password Authentication Protocol (PAP)
	  ◦ CHAP
	  ◦ MSCHAP
	  ◦ MSCHAPv2
	  ◦ EAP
	  ◦ MSCHAPv2
	  ◦ TLS
	• Server validation: in TTLS, the server must be validated. The following can be configured:
	  ◦ Server name
	  ◦ Trusted root certificate for server certificate
	  ◦ Whether there should be a server validation notification

For a UWP VPN plug-in, the app vendor controls the authentication method to be used. The following credential types can be used:

- Smart card
- Certificate
- Windows Hello for Business
- User name and password

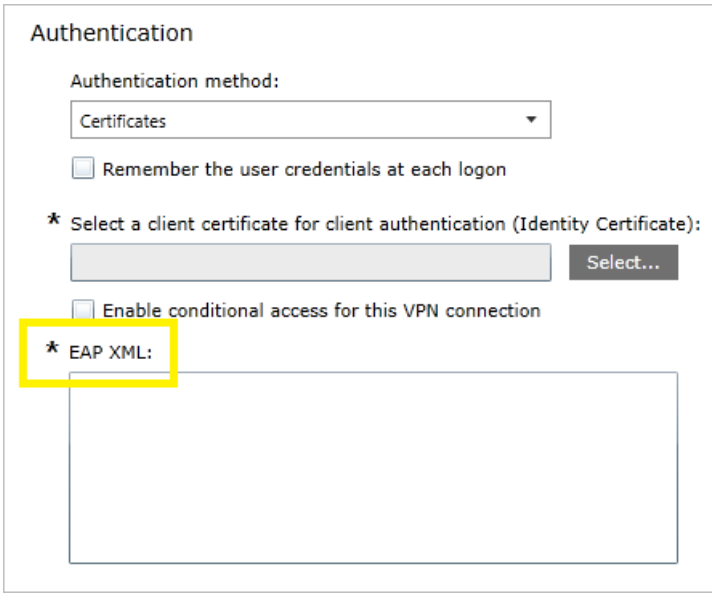
- One-time password
- Custom credential type

Configure authentication

See EAP configuration for EAP XML configuration.

> **NOTE**
> To configure Windows Hello for Business authentication, follow the steps in EAP configuration to create a smart card certificate. Learn more about Windows Hello for Business.

The following image shows the field for EAP XML in a Microsoft Intune VPN profile. The EAP XML field only appears when you select a built-in connection type (automatic, IKEv2, L2TP, PPTP).



Related topics

- VPN technical guide
- VPN connection types
- VPN routing decisions
- VPN and conditional access
- VPN name resolution
- VPN auto-triggered profile options
- VPN security features
- VPN profile options

VPN and conditional access

3/26/2021 • 5 minutes to read • Edit Online

Applies to: Windows 10 and Windows 10 Mobile

The VPN client is now able to integrate with the cloud-based Conditional Access Platform to provide a device compliance option for remote clients. Conditional Access is a policy-based evaluation engine that lets you create access rules for any Azure Active Directory (Azure AD) connected application.

> **NOTE**
> Conditional Access is an Azure AD Premium feature.

Conditional Access Platform components used for Device Compliance include the following cloud-based services:

- Conditional Access Framework
- Azure AD Connect Health
- Windows Health Attestation Service (optional)
- Azure AD Certificate Authority - It is a requirement that the client certificate used for the cloud-based device compliance solution be issued by an Azure Active Directory-based Certificate Authority (CA). An Azure AD CA is essentially a mini-CA cloud tenant in Azure. The Azure AD CA cannot be configured as part of an on-premises Enterprise CA. See also Always On VPN deployment for Windows Server and Windows 10.
- Azure AD-issued short-lived certificates - When a VPN connection attempt is made, the Azure AD Token Broker on the local device communicates with Azure Active Directory, which then checks for health based on compliance rules. If compliant, Azure AD sends back a short-lived certificate that is used to authenticate the VPN. Note that certificate authentication methods such as EAP-TLS can be used. When that certificate expires, the client will again check with Azure AD for health validation before a new certificate is issued.
- Microsoft Intune device compliance policies - Cloud-based device compliance leverages Microsoft Intune Compliance Policies, which are capable of querying the device state and define compliance rules for the following, among other things.
    - Antivirus status
    - Auto-update status and update compliance
    - Password policy compliance
    - Encryption compliance
    - Device health attestation state (validated against attestation service after query)

The following client-side components are also required:

- HealthAttestation Configuration Service Provider (CSP)
- VPNv2 CSP DeviceCompliance node settings
- Trusted Platform Module (TPM)

VPN device compliance

At this time, the Azure AD certificates issued to users do not contain a CRL Distribution Point (CDP) and are not suitable for Key Distribution Centers (KDCs) to issue Kerberos tokens. For users to gain access to on-premises resources such as files on a network share, client authentication certificates must be deployed to the Windows profiles of the users, and their VPNv2 profiles must contain the <SSO> section.

Server-side infrastructure requirements to support VPN device compliance include:

- The VPN server should be configured for certificate authentication.
- The VPN server should trust the tenant-specific Azure AD CA.
- For client access using Kerberos/NTLM, a domain-trusted certificate is deployed to the client device and is configured to be used for single sign-on (SSO).

After the server side is set up, VPN admins can add the policy settings for conditional access to the VPN profile using the VPNv2 DeviceCompliance node.

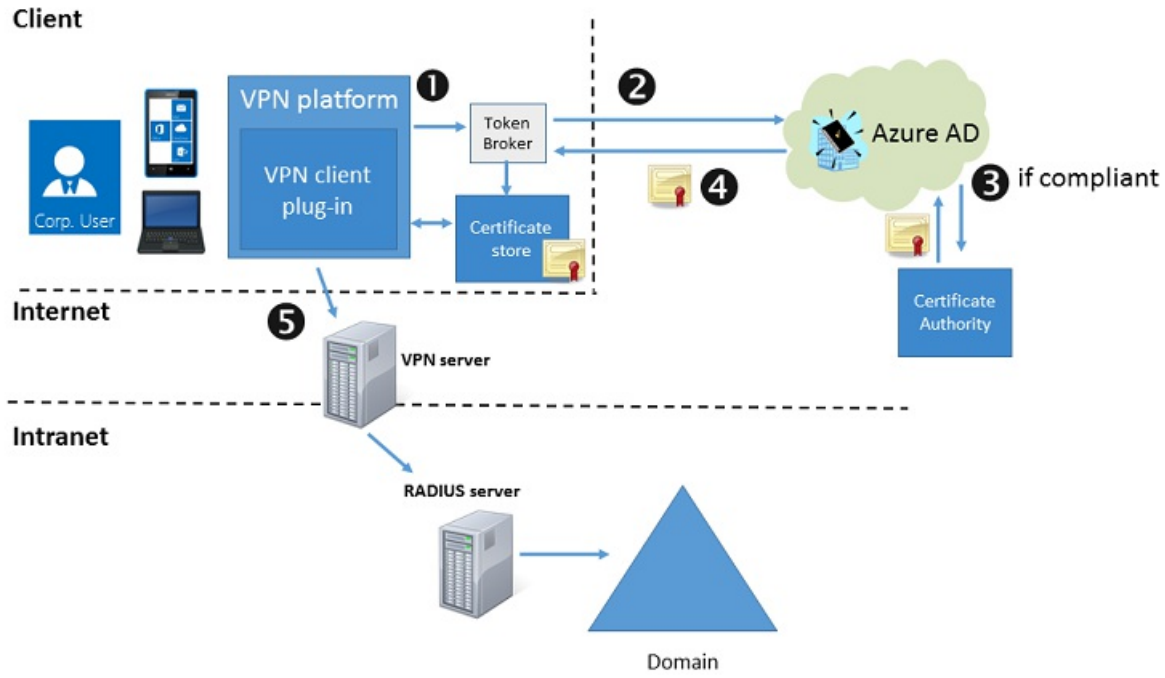
Two client-side configuration service providers are leveraged for VPN device compliance.

- VPNv2 CSP DeviceCompliance settings:
  - **Enabled**: enables the Device Compliance flow from the client. If marked as **true**, the VPN client attempts to communicate with Azure AD to get a certificate to use for authentication. The VPN should be set up to use certificate authentication and the VPN server must trust the server returned by Azure AD.
  - **Sso**: entries under SSO should be used to direct the VPN client to use a certificate other than the VPN authentication certificate when accessing resources that require Kerberos authentication.
  - **Sso/Enabled**: if this field is set to **true**, the VPN client looks for a separate certificate for Kerberos authentication.
  - **Sso/IssuerHash**: hashes for the VPN client to look for the correct certificate for Kerberos authentication.
  - **Sso/Eku**: comma-separated list of Enhanced Key Usage (EKU) extensions for the VPN client to look for the correct certificate for Kerberos authentication.
- HealthAttestation CSP (not a requirement) - functions performed by the HealthAttestation CSP include:
  - Collects TPM data used to verify health states
  - Forwards the data to the Health Attestation Service (HAS)
  - Provisions the Health Attestation Certificate received from the HAS
  - Upon request, forward the Health Attestation Certificate (received from HAS) and related runtime information to the MDM server for verification

> **NOTE**
>
> Currently, it is required that certificates used for obtaining Kerberos tickets must be issued from an on-premises CA, and that SSO must be enabled in the user's VPN profile. This will enable the user to access on-premises resources.
>
> In the case of AzureAD-only joined devices (not hybrid joined devices), if the user certificate issued by the on-premises CA has the user UPN from AzureAD in Subject and SAN (Subject Alternative Name), the VPN profile must be modified to ensure that the client does not cache the credentials used for VPN authentication. To do this, after deploying the VPN profile to the client, modify the *Rasphone.pbk* on the client by changing the entry **UseRasCredentials** from 1 (default) to 0 (zero).

Client connection flow

The VPN client side connection flow works as follows:



When a VPNv2 Profile is configured with <DeviceCompliance> <Enabled>true</Enabled> the VPN client uses this connection flow:

1. The VPN client calls into Windows 10's Azure AD Token Broker, identifying itself as a VPN client.
2. The Azure AD Token Broker authenticates to Azure AD and provides it with information about the device trying to connect. The Azure AD Server checks if the device is in compliance with the policies.
3. If compliant, Azure AD requests a short-lived certificate.
4. Azure AD pushes down a short-lived certificate to the Certificate Store via the Token Broker. The Token Broker then returns control back over to the VPN client for further connection processing.
5. The VPN client uses the Azure AD-issued certificate to authenticate with the VPN server.

Configure conditional access

See VPN profile options and VPNv2 CSP for XML configuration.

Learn more about Conditional Access and Azure AD Health

- Azure Active Directory conditional access
- Getting started with Azure Active Directory Conditional Access
- Control the health of Windows 10-based devices
- Tip of the Day: The Conditional Access Framework and Device Compliance for VPN (Part 1)
- Tip of the Day: The Conditional Access Framework and Device Compliance for VPN (Part 2)
- Tip of the Day: The Conditional Access Framework and Device Compliance for VPN (Part 3)
- Tip of the Day: The Conditional Access Framework and Device Compliance for VPN (Part 4)

Related topics

- VPN technical guide

- VPN connection types
- VPN routing decisions
- VPN authentication options
- VPN name resolution
- VPN auto-triggered profile options
- VPN security features
- VPN profile options

VPN name resolution

3/26/2021 • 2 minutes to read • Edit Online

Applies to

- Windows 10
- Windows 10 Mobile

When the VPN client connects to the VPN server, the VPN client receives the client IP address. The client may also receive the IP address of the Domain Name System (DNS) server and the IP address of the Windows Internet Name Service (WINS) server.

The name resolution setting in the VPN profile configures how name resolution should work on the system when VPN is connected. The networking stack first looks at the Name Resolution Policy table (NRPT) for any matches and tries a resolution in the case of a match. If no match is found, the DNS suffix on the most preferred interface based on the interface metric is appended to the name (in the case of a short name) and a DNS query is sent out on the preferred interface. If the query times out, the DNS suffix search list is used in order and DNS queries are sent on all interfaces.

Name Resolution Policy table (NRPT)

The NRPT is a table of namespaces that determines the DNS client's behavior when issuing name resolution queries and processing responses. It is the first place that the stack will look after the DNSCache.

There are 3 types of name matches that can set up for NRPT:

- Fully qualified domain name (FQDN) that can be used for direct matching to a name
- Suffix match results in either a comparison of suffixes (for FQDN resolution) or the appending of the suffix (in case of a short name)
- Any resolution should attempt to first resolve with the proxy server/DNS server with this entry

NRPT is set using the **VPNv2/*ProfileName*/DomainNameInformationList** node of the VPNv2 CSP. This node also configures Web proxy server or domain name servers.

Learn more about NRPT

DNS suffix

This setting is used to configure the primary DNS suffix for the VPN interface and the suffix search list after the VPN connection is established.

Primary DNS suffix is set using the **VPNv2/*ProfileName*/DnsSuffix** node.

Learn more about primaryDNS suffix

Persistent

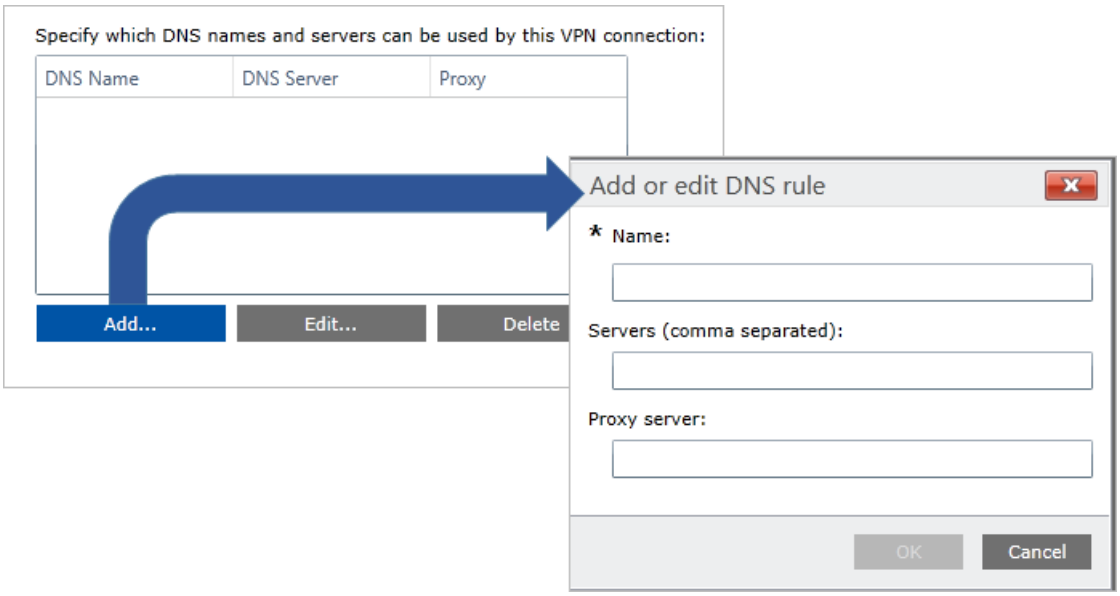
You can also configure *persistent* name resolution rules. Name resolution for specified items will only be performed over the VPN.

Persistent name resolution is set using the **VPNv2/*ProfileName*/DomainNameInformationList//*dniRowId*/Persistent** node.

Configure name resolution

See VPN profile options and VPNv2 CSP for XML configuration.

The following image shows name resolution options in a VPN Profile configuration policy using Microsoft Intune.



The fields in **Add or edit DNS rule** in the Intune profile correspond to the XML settings shown in the following table.

FIELD	XML
Name	VPNv2/*ProfileName*/DomainNameInformationList/*dnsRowId*/DomainName
Servers (comma separated)	VPNv2/*ProfileName*/DomainNameInformationList/*dnsRowId*/DnsServers
Proxy server	VPNv2/*ProfileName*/DomainNameInformationList/*dnsRowId*/WebServers

Related topics

- VPN technical guide
- VPN connection types
- VPN routing decisions
- VPN authentication options
- VPN and conditional access
- VPN auto-triggered profile options
- VPN security features
- VPN profile options

VPN auto-triggered profile options

4/29/2021 • 3 minutes to read • Edit Online

Applies to

- Windows 10
- Windows 10 Mobile

In Windows 10, a number of features were added to auto-trigger VPN so users won't have to manually connect when VPN is needed to access necessary resources. There are three different types of auto-trigger rules:

- App trigger
- Name-based trigger
- Always On

> **NOTE**
> Auto-triggered VPN connections will not work if Folder Redirection for AppData is enabled. Either Folder Redirection for AppData must be disabled or the auto-triggered VPN profile must be deployed in system context, which changes the path to where the rasphone.pbk file is stored.

App trigger

VPN profiles in Windows 10 can be configured to connect automatically on the launch of a specified set of applications. You can configure desktop or Universal Windows Platform (UWP) apps to trigger a VPN connection. You can also configure per-app VPN and specify traffic rules for each app. See Traffic filters for more details.

The app identifier for a desktop app is a file path. The app identifier for a UWP app is a package family name.

Find a package family name (PFN) for per-app VPN configuration

Name-based trigger

You can configure a domain name-based rule so that a specific domain name triggers the VPN connection.

Name-based auto-trigger can be configured using the VPNv2/*ProfileName*/DomainNameInformationList/dniRowId/AutoTrigger setting in the VPNv2 Configuration Service Provider (CSP).

There are four types of name-based triggers:

- Short name: for example, if **HRweb** is configured as a trigger and the stack sees a DNS resolution request for **HRweb**, the VPN will be triggered.
- Fully-qualified domain name (FQDN): for example, if **HRweb.corp.contoso.com** is configured as a trigger and the stack sees a DNS resolution request for **HRweb.corp.contoso.com**, the VPN will be triggered.
- Suffix: for example, if **.corp.contoso.com** is configured as a trigger and the stack sees a DNS resolution request with a matching suffix (such as **HRweb.corp.contoso.com**), the VPN will be triggered. For any short name resolution, VPN will be triggered and the DNS server will be queried for the *ShortName***corp.contoso.com**.
- All: if used, all DNS resolution should trigger VPN.

Always On

Always On is a feature in Windows 10 which enables the active VPN profile to connect automatically on the following triggers:

- User sign-in
- Network change
- Device screen on

When the trigger occurs, VPN tries to connect. If an error occurs or any user input is needed, the user is shown a toast notification for additional interaction.

When a device has multiple profiles with Always On triggers, the user can specify the active profile in **Settings** > **Network & Internet** > **VPN** > *VPN profile* by selecting the **Let apps automatically use this VPN connection** checkbox. By default, the first MDM-configured profile is marked as **Active**. Devices with multiple users have the same restriction: only one profile and therefore only one user will be able to use the Always On triggers.

Preserving user Always On preference

Windows has a feature to preserve a user's AlwaysOn preference. In the event that a user manually unchecks the "Connect automatically" checkbox, Windows will remember this user preference for this profile name by adding the profile name to the value **AutoTriggerDisabledProfilesList**.

Should a management tool remove or add the same profile name back and set **AlwaysOn** to **true**, Windows will not check the box if the profile name exists in the following registry value in order to preserve user preference.

Key: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\RasMan\Config
Value: AutoTriggerDisabledProfilesList
Type: REG_MULTI_SZ

Trusted network detection

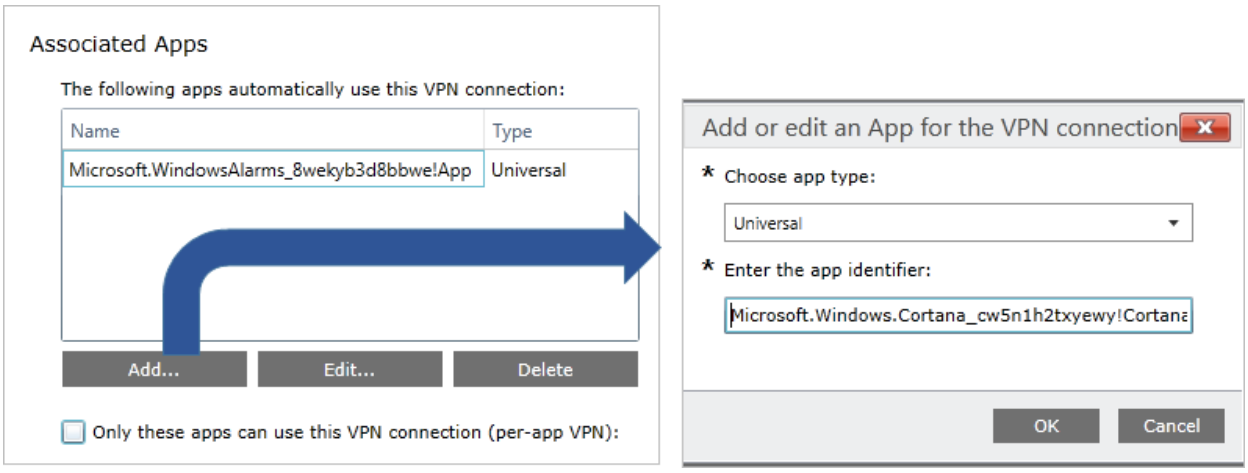
This feature configures the VPN such that it would not get triggered if a user is on a trusted corporate network. The value of this setting is a list of DNS suffices. The VPN stack will look at the DNS suffix on the physical interface and if it matches any in the configured list and the network is private or provisioned by MDM, then VPN will not get triggered.

Trusted network detection can be configured using the VPNv2/*ProfileName*/TrustedNetworkDetection setting in the VPNv2 CSP.

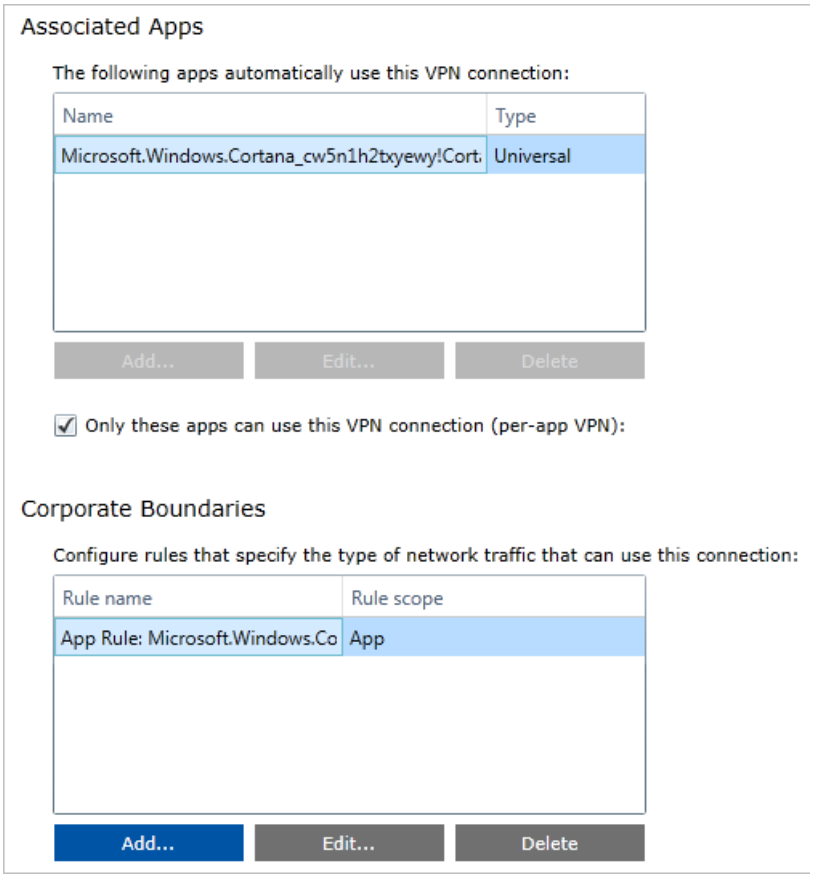
Configure app-triggered VPN

See VPN profile options and VPNv2 CSP for XML configuration.

The following image shows associating an app to a VPN connection in a VPN Profile configuration policy using Microsoft Intune.



After you add an associated app, if you select the **Only these apps can use this VPN connection (per-app VPN)** checkbox, the app becomes available in **Corporate Boundaries**, where you can configure rules for the app. See Traffic filters for more details.



Related topics

- VPN technical guide
- VPN connection types
- VPN routing decisions
- VPN authentication options
- VPN and conditional access
- VPN name resolution
- VPN security features
- VPN profile options

VPN security features

3/26/2021 • 3 minutes to read • Edit Online

Applies to

- Windows 10
- Windows 10 Mobile

Windows Information Protection (WIP) integration with VPN

Windows Information Protection provides capabilities allowing the separation and protection of enterprise data against disclosure across both company and personally owned devices, without requiring additional changes to the environments or the apps themselves. Additionally, when used with Rights Management Services (RMS), WIP can help to protect enterprise data locally.

The **EdpModeId** node in the VPNv2 Configuration Service Provider (CSP) allows a Windows 10 VPN client to integrate with WIP, extending its functionality to remote devices. Use case scenarios for WIP include:

- Core functionality: File encryption and file access blocking
- UX policy enforcement: Restricting copy/paste, drag/drop, and sharing operations
- WIP network policy enforcement: Protecting intranet resources over the corporate network and VPN
- Network policy enforcement: Protecting SMB and Internet cloud resources over the corporate network and VPN

The value of the **EdpModeId** is an Enterprise ID. The networking stack will look for this ID in the app token to determine whether VPN should be triggered for that particular app.

Additionally, when connecting with WIP, the admin does not have to specify AppTriggerList and TrafficFilterList rules separately in this profile (unless more advanced configuration is needed) because the WIP policies and App lists automatically take effect.

Learn more about Windows Information Protection

Traffic Filters

Traffic Filters give enterprises the ability to decide what traffic is allowed into the corporate network based on policy. Network admins can use Traffic Filters to effectively add interface specific firewall rules on the VPN Interface. There are two types of Traffic Filter rules:

- App-based rules. With app-based rules, a list of applications can be marked to allow only traffic originating from these apps to go over the VPN interface.
- Traffic-based rules. Traffic-based rules are 5-tuple policies (ports, addresses, protocol) that can be specified to allow only traffic matching these rules to go over the VPN interface.

There can be many sets of rules which are linked by OR. Within each set, there can be app-based rules and traffic-based rules; all the properties within the set will be linked by AND. In addition, these rules can be applied at a per-app level or a per-device level.

For example, an admin could define rules that specify:

- The Contoso HR App must be allowed to go through the VPN and only access port 4545.
- The Contoso finance apps are allowed to go over the VPN and only access the Remote IP ranges of

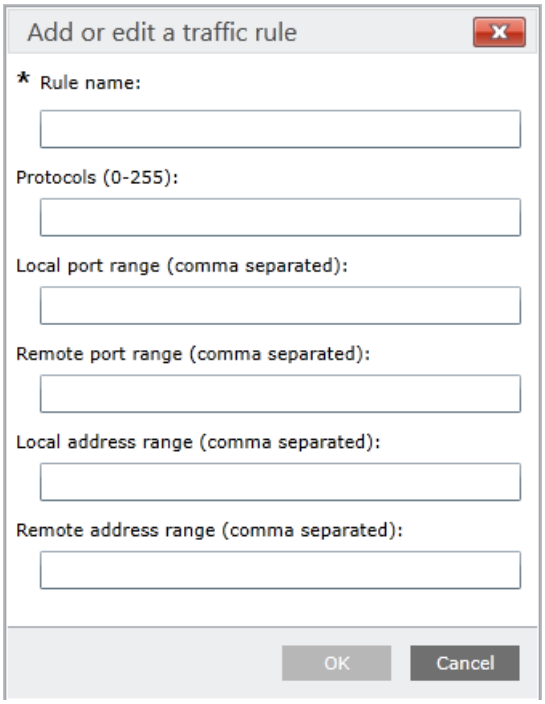
10.10.0.40 - 10.10.0.201 on port 5889.

- All other apps on the device should be able to access only ports 80 or 443.

Configure traffic filters

See VPN profile options and VPNv2 CSP for XML configuration.

The following image shows the interface to configure traffic rules in a VPN Profile configuration policy, using Microsoft Intune.



LockDown VPN

A VPN profile configured with LockDown secures the device to only allow network traffic over the VPN interface. It has the following features:

- The system attempts to keep the VPN connected at all times.
- The user cannot disconnect the VPN connection.
- The user cannot delete or modify the VPN profile.
- The VPN LockDown profile uses forced tunnel connection.
- If the VPN connection is not available, outbound network traffic is blocked.
- Only one VPN LockDown profile is allowed on a device.

> **NOTE**
> For built-in VPN, LockDown VPN is only available for the Internet Key Exchange version 2 (IKEv2) connection type.

Deploy this feature with caution, as the resultant connection will not be able to send or receive any network traffic without the VPN being connected.

Related topics

- VPN technical guide
- VPN connection types
- VPN routing decisions

- VPN authentication options
- VPN and conditional access
- VPN name resolution
- VPN auto-triggered profile options
- VPN profile options

VPN profile options

3/26/2021 • 4 minutes to read • Edit Online

Applies to

- Windows 10
- Windows 10 Mobile

Most of the VPN settings in Windows 10 can be configured in VPN profiles using Microsoft Intune or Microsoft Endpoint Configuration Manager. All VPN settings in Windows 10 can be configured using the **ProfileXML** node in the VPNv2 configuration service provider (CSP).

> **NOTE**
> If you're not familiar with CSPs, read Introduction to configuration service providers (CSPs) first.

The following table lists the VPN settings and whether the setting can be configured in Intune and Configuration Manager, or can only be configured using **ProfileXML**.

PROFILE SETTING	CAN BE CONFIGURED IN INTUNE AND CONFIGURATION MANAGER
Connection type	yes
Routing: split-tunnel routes	yes, except exclusion routes
Routing: forced-tunnel	yes
Authentication (EAP)	yes, if connection type is built-in
Conditional access	yes
Name resolution: NRPT	yes
Name resolution: DNS suffix	no
Name resolution: persistent	no
Auto-trigger: app trigger	yes
Auto-trigger: name trigger	yes
Auto-trigger: Always On	yes
Auto-trigger: trusted network detection	no
LockDown	no
Windows Information Protection (WIP)	yes

PROFILE SETTING	CAN BE CONFIGURED IN INTUNE AND CONFIGURATION MANAGER
Traffic filters	yes
Proxy settings	yes, by PAC/WPAD file or server and port

> **NOTE**
> VPN proxy settings are only used on Force Tunnel Connections. On Split Tunnel Connections, the general proxy settings are used.

The ProfileXML node was added to the VPNv2 CSP to allow users to deploy VPN profile as a single blob. This is particularly useful for deploying profiles with features that are not yet supported by MDMs. You can get additional examples in the ProfileXML XSD topic.

Sample Native VPN profile

The following is a sample Native VPN profile. This blob would fall under the ProfileXML node.

```
<VPNProfile>
  <ProfileName>TestVpnProfile</ProfileName>
  <NativeProfile>
    <Servers>testServer.VPN.com</Servers>
    <NativeProtocolType>IKEv2</NativeProtocolType>

    <!--Sample EAP profile (PEAP)-->
    <Authentication>
      <UserMethod>Eap</UserMethod>
      <Eap>
       <Configuration>
          <EapHostConfig xmlns="http://www.microsoft.com/provisioning/EapHostConfig">
            <EapMethod>
              <Type xmlns="http://www.microsoft.com/provisioning/EapCommon">25</Type>
              <VendorId xmlns="http://www.microsoft.com/provisioning/EapCommon">0</VendorId>
              <VendorType xmlns="http://www.microsoft.com/provisioning/EapCommon">0</VendorType>
              <AuthorId xmlns="http://www.microsoft.com/provisioning/EapCommon">0</AuthorId>
            </EapMethod>
            <Config xmlns="http://www.microsoft.com/provisioning/EapHostConfig">
              <Eap xmlns="http://www.microsoft.com/provisioning/BaseEapConnectionPropertiesV1">
                <Type>25</Type>
                <EapType xmlns="http://www.microsoft.com/provisioning/MsPeapConnectionPropertiesV1">
                  <ServerValidation>
                    <DisableUserPromptForServerValidation>true</DisableUserPromptForServerValidation>
                    <ServerNames></ServerNames>
                    <TrustedRootCA>d2 d3 8e ba 60 ca a1 c1 20 55 a2 e1 c8 3b 15 ad 45 01 10 c2
</TrustedRootCA>
                    <TrustedRootCA>d1 76 97 cc 20 6e d2 6e 1a 51 f5 bb 96 e9 35 6d 6d 61 0b 74
</TrustedRootCA>
                  </ServerValidation>
                  <FastReconnect>true</FastReconnect>
                  <InnerEapOptional>false</InnerEapOptional>
                  <Eap xmlns="http://www.microsoft.com/provisioning/BaseEapConnectionPropertiesV1">
                    <Type>13</Type>
                    <EapType xmlns="http://www.microsoft.com/provisioning/EapTlsConnectionPropertiesV1">
                      <CredentialsSource>
                        <CertificateStore>
                          <SimpleCertSelection>true</SimpleCertSelection>
                        </CertificateStore>
                      </CredentialsSource>
                      <ServerValidation>
                        <DisableUserPromptForServerValidation>true</DisableUserPromptForServerValidation>
```

```
                        <ServerNames></ServerNames>
                        <TrustedRootCA>d2 d3 8e ba 60 ca a1 c1 20 55 a2 e1 c8 3b 15 ad 45 01 10 c2
</TrustedRootCA>
                        <TrustedRootCA>d1 76 97 cc 20 6e d2 6e 1a 51 f5 bb 96 e9 35 6d 6d 61 0b 74
</TrustedRootCA>
                      </ServerValidation>
                      <DifferentUsername>false</DifferentUsername>
                      <PerformServerValidation
xmlns="http://www.microsoft.com/provisioning/EapTlsConnectionPropertiesV2">true</PerformServerValidation>
                      <AcceptServerName
xmlns="http://www.microsoft.com/provisioning/EapTlsConnectionPropertiesV2">false</AcceptServerName>
                      <TLSExtensions
xmlns="http://www.microsoft.com/provisioning/EapTlsConnectionPropertiesV2">
                        <FilteringInfo
xmlns="http://www.microsoft.com/provisioning/EapTlsConnectionPropertiesV3">
                          <EKUMapping>
                            <EKUMap>
                              <EKUName>AAD Conditional Access</EKUName>
                              <EKUOID>1.3.6.1.4.1.311.87</EKUOID>
                            </EKUMap>
                          </EKUMapping>
                          <ClientAuthEKUList Enabled="true">
                            <EKUMapInList>
                              <EKUName>AAD Conditional Access</EKUName>
                            </EKUMapInList>
                          </ClientAuthEKUList>
                        </FilteringInfo>
                      </TLSExtensions>
                    </EapType>
                  </Eap>
                  <EnableQuarantineChecks>false</EnableQuarantineChecks>
                  <RequireCryptoBinding>true</RequireCryptoBinding>
                  <PeapExtensions>
                    <PerformServerValidation
xmlns="http://www.microsoft.com/provisioning/MsPeapConnectionPropertiesV2">true</PerformServerValidation>
                    <AcceptServerName
xmlns="http://www.microsoft.com/provisioning/MsPeapConnectionPropertiesV2">false</AcceptServerName>
                  </PeapExtensions>
                </EapType>
              </Eap>
            </Config>
          </EapHostConfig>
        </Configuration>
      </Eap>
    </Authentication>

    <!--Sample routing policy: in this case, this is a split tunnel configuration with two routes
configured-->
    <RoutingPolicyType>SplitTunnel</RoutingPolicyType>
    <DisableClassBasedDefaultRoute>true</DisableClassBasedDefaultRoute>
  </NativeProfile>
    <Route>
    <Address>192.168.0.0</Address>
    <PrefixSize>24</PrefixSize>
  </Route>
  <Route>
    <Address>10.10.0.0</Address>
    <PrefixSize>16</PrefixSize>
  </Route>

  <!--VPN will be triggered for the two apps specified here-->
  <AppTrigger>
    <App>
      <Id>Microsoft.MicrosoftEdge_8wekyb3d8bbwe</Id>
    </App>
  </AppTrigger>
  <AppTrigger>
    <App>
      <Id>C:\windows\system32\ping.exe</Id>
```

```
    </App>
  </AppTrigger>

  <!--Example of per-app VPN. This configures traffic filtering rules for two apps. Internet Explorer is
configured for force tunnel, meaning that all traffic allowed through this app must go over VPN. Microsoft
Edge is configured as split tunnel, so whether data goes over VPN or the physical interface is dictated by
the routing configuration.-->
  <TrafficFilter>
    <App>
      <Id>%ProgramFiles%\Internet Explorer\iexplore.exe</Id>
    </App>
    <Protocol>6</Protocol>
    <LocalPortRanges>10,20-50,100-200</LocalPortRanges>
    <RemotePortRanges>20-50,100-200,300</RemotePortRanges>
    <RemoteAddressRanges>30.30.0.0/16,10.10.10.10-20.20.20.20</RemoteAddressRanges>
    <RoutingPolicyType>ForceTunnel</RoutingPolicyType>
  </TrafficFilter>
  <TrafficFilter>
    <App>
      <Id>Microsoft.MicrosoftEdge_8wekyb3d8bbwe</Id>
    </App>
    <LocalAddressRanges>3.3.3.3/32,1.1.1.1-2.2.2.2</LocalAddressRanges>
  </TrafficFilter>

  <!--Name resolution configuration. The AutoTrigger node configures name-based triggering. In this profile,
the domain "hrsite.corporate.contoso.com" triggers VPN.-->
  <DomainNameInformation>
    <DomainName>hrsite.corporate.contoso.com</DomainName>
    <DnsServers>1.2.3.4,5.6.7.8</DnsServers>
    <WebProxyServers>5.5.5.5</WebProxyServers>
    <AutoTrigger>true</AutoTrigger>
  </DomainNameInformation>
  <DomainNameInformation>
    <DomainName>.corp.contoso.com</DomainName>
    <DnsServers>10.10.10.10,20.20.20.20</DnsServers>
    <WebProxyServers>100.100.100.100</WebProxyServers>
  </DomainNameInformation>

  <!--EDPMode is turned on for the enterprise ID "corp.contoso.com". When a user accesses an app with that
ID, VPN will be triggered.-->
  <EdpModeId>corp.contoso.com</EdpModeId>
  <RememberCredentials>true</RememberCredentials>

  <!--Always On is turned off, and triggering VPN for the apps and domain name specified earlier in the
profile will not occur if the user is connected to the trusted network "contoso.com".-->
  <AlwaysOn>false</AlwaysOn>
  <DnsSuffix>corp.contoso.com</DnsSuffix>
  <TrustedNetworkDetection>contoso.com</TrustedNetworkDetection>
  <Proxy>
    <Manual>
      <Server>HelloServer</Server>
    </Manual>
    <AutoConfigUrl>Helloworld.Com</AutoConfigUrl>
  </Proxy>

  <!--Device compliance is enabled and an alternate certificate is specified for domain resource
authentication.-->
  <DeviceCompliance>
        <Enabled>true</Enabled>
        <Sso>
            <Enabled>true</Enabled>
            <Eku>This is my Eku</Eku>
            <IssuerHash>This is my issuer hash</IssuerHash>
        </Sso>
    </DeviceCompliance>
</VPNProfile>
```

Sample plug-in VPN profile

The following is a sample plug-in VPN profile. This blob would fall under the ProfileXML node.

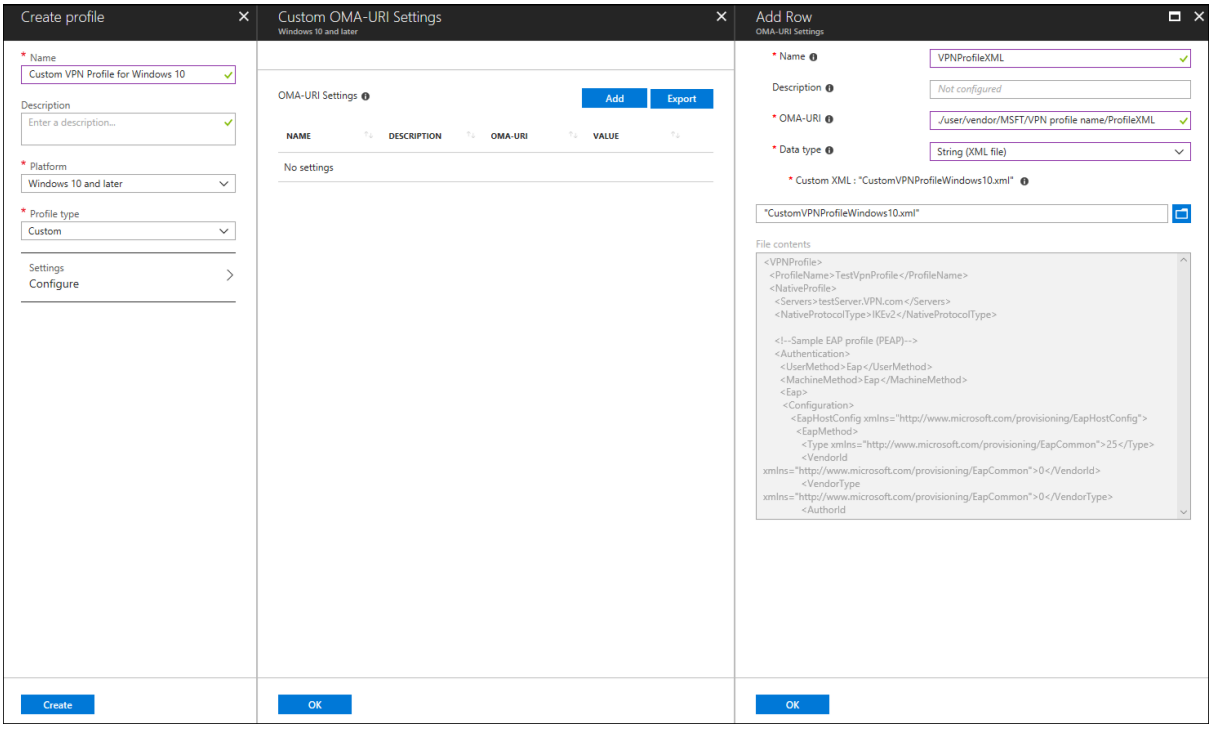
```
<VPNProfile>
 <ProfileName>TestVpnProfile</ProfileName>
 <PluginProfile>
  <ServerUrlList>testserver1.contoso.com;testserver2.contoso..com</ServerUrlList>
  <PluginPackageFamilyName>JuniperNetworks.JunosPulseVpn_cw5n1h2txyewy</PluginPackageFamilyName>
  <CustomConfiguration>&lt;pulse-
schema&gt;&lt;isSingleSignOnCredential&gt;true&lt;/isSingleSignOnCredential&gt;&lt;/pulse-schema&gt;
</CustomConfiguration>
 </PluginProfile>
 <Route>
  <Address>192.168.0.0</Address>
  <PrefixSize>24</PrefixSize>
 </Route>
 <Route>
  <Address>10.10.0.0</Address>
  <PrefixSize>16</PrefixSize>
 </Route>
 <AppTrigger>
  <App>
   <Id>Microsoft.MicrosoftEdge_8wekyb3d8bbwe</Id>
  </App>
 </AppTrigger>
 <AppTrigger>
  <App>
   <Id>%ProgramFiles%\Internet Explorer\iexplore.exe</Id>
  </App>
 </AppTrigger>
 <TrafficFilter>
  <App>
   <Id>%ProgramFiles%\Internet Explorer\iexplore.exe</Id>
  </App>
  <Protocol>6</Protocol>
  <LocalPortRanges>10,20-50,100-200</LocalPortRanges>
  <RemotePortRanges>20-50,100-200,300</RemotePortRanges>
  <RemoteAddressRanges>30.30.0.0/16,10.10.10.10-20.20.20.20</RemoteAddressRanges>
  <!--<RoutingPolicyType>ForceTunnel</RoutingPolicyType>-->
 </TrafficFilter>
 <TrafficFilter>
  <App>
   <Id>Microsoft.MicrosoftEdge_8wekyb3d8bbwe</Id>
  </App>
  <LocalAddressRanges>3.3.3.3/32,1.1.1.1-2.2.2.2</LocalAddressRanges>
 </TrafficFilter>
 <TrafficFilter>
  <App>
   <Id>Microsoft.MicrosoftEdge_8wekyb3d8bbwe</Id>
  </App>
  <Claims>O:SYG:SYD:(A;;CC;;;AU)</Claims>
  <!--<RoutingPolicyType>SplitTunnel</RoutingPolicyType>-->
 </TrafficFilter>
 <DomainNameInformation>
  <DomainName>corp.contoso.com</DomainName>
  <DnsServers>1.2.3.4,5.6.7.8</DnsServers>
  <WebProxyServers>5.5.5.5</WebProxyServers>
  <AutoTrigger>false</AutoTrigger>
 </DomainNameInformation>
 <DomainNameInformation>
  <DomainName>corp.contoso.com</DomainName>
  <DnsServers>10.10.10.10,20.20.20.20</DnsServers>
  <WebProxyServers>100.100.100.100</WebProxyServers>
 </DomainNameInformation>
 <!--<EdpModeId>corp.contoso.com</EdpModeId>-->
 <RememberCredentials>true</RememberCredentials>
```

```
 <AlwaysOn>false</AlwaysOn>
 <DnsSuffix>corp.contoso.com</DnsSuffix>
 <TrustedNetworkDetection>contoso.com,test.corp.contoso.com</TrustedNetworkDetection>
 <Proxy>
  <Manual>
   <Server>HelloServer</Server>
  </Manual>
  <AutoConfigUrl>Helloworld.Com</AutoConfigUrl>
 </Proxy>
</VPNProfile>
```

Apply ProfileXML using Intune

After you configure the settings that you want using ProfileXML, you can apply it using Intune and a **Custom Configuration (Windows 10 Desktop and Mobile and later)** policy.

1. Sign into the Azure portal.
2. Go to **Intune** > **Device Configuration** > **Profiles**.
3. Click **Create Profile**.
4. Enter a name and (optionally) a description.
5. Choose **Windows 10 and later** as the platform.
6. Choose **Custom** as the profile type and click **Add**.
7. Enter a name and (optionally) a description.
8. Enter the OMA-URI **./user/vendor/MSFT/VPNv2/*VPN profile name*/ProfileXML**.
9. Set Data type to **String (XML file)**.
10. Upload the profile XML file.
11. Click **OK**.



12. Click **OK**, then **Create**.
13. Assign the profile.

Learn more

- Create VPN profiles to connect to VPN servers in Intune
- VPNv2 configuration service provider (CSP) reference

- How to Create VPN Profiles in Configuration Manager

Related topics

- VPN technical guide
- VPN connection types
- VPN routing decisions
- VPN authentication options
- VPN and conditional access
- VPN name resolution
- VPN auto-triggered profile options
- VPN security features

How to configure Diffie Hellman protocol over IKEv2 VPN connections

3/26/2021 • 2 minutes to read • Edit Online

Applies To: Windows Server (Semi-Annual Channel), Windows Server 2016, Windows 10

In IKEv2 VPN connections, the default configuration for Diffie Hellman group is Group 2, which is not secure for IKE exchanges. To secure the connections, update the configuration of VPN servers and clients by running VPN cmdlets.

VPN server

For VPN servers that run Windows Server 2012 R2 or later, you need to run Set-VpnServerConfiguration to configure the tunnel type. This makes all IKE exchanges on IKEv2 tunnel use the secure configuration.

```
Set-VpnServerConfiguration -TunnelType IKEv2 -CustomPolicy
```

On an earlier versions of Windows Server, run Set-VpnServerIPsecConfiguration. Since `Set-VpnServerIPsecConfiguration` doesn't have `-TunnelType`, the configuration applies to all tunnel types on the server.

```
Set-VpnServerIPsecConfiguration -CustomPolicy
```

VPN client

For VPN client, you need to configure each VPN connection. For example, run Set-VpnConnectionIPsecConfiguration (version 4.0) and specify the name of the connection:

```
Set-VpnConnectionIPsecConfiguration -ConnectionName <String>
```

How to use single sign on (SSO) over VPN and Wi-Fi connections

3/26/2021 • 4 minutes to read • Edit Online

This topic explains requirements to enable Single Sign-On (SSO) to on-premises domain resources over WiFi or VPN connections. The scenario is:

- You connect to a network using Wi-Fi or VPN.
- You want to use the credentials that you use for the WiFi or VPN authentication to also authenticate requests to access a domain resource you are connecting to, without being prompted for your domain credentials separately.

For example, you want to connect to a corporate network and access an internal website that requires Windows integrated authentication.

At a high level, the way this works is that the credentials that are used for the connection authentication are put in Credential Manager as the default credentials for the logon session. Credential Manager is a place where credentials in the OS are can be stored for specific domain resources based on the targetname of the resource. For VPN, the VPN stack saves its credential as the session default. For WiFi, EAP does it.

The credentials are put in Credential Manager as a "*Session" credential. A "*Session" credential implies that it is valid for the current user session. The credentials are also cleaned up when the WiFi or VPN connection is disconnected.

When the user tries to access a domain resource, using Edge for example, Edge has the right Enterprise Authentication capability so WinInet can release the credentials that it gets from the Credential Manager to the SSP that is requesting it. For more information about the Enterprise Authentication capability, see App capability declarations.

The local security authority will look at the device application, such as a Universal Windows Platform (UWP) application, to see if it has the right capability. If the app is not UWP, it does not matter. But if it is a UWP app, it will look at the device capability for Enterprise Authentication. If it does have that capability and if the resource that you are trying to access is in the Intranet zone in the Internet Options (ZoneMap), then the credential will be released. This behavior helps prevent credentials from being misused by untrusted third parties.

Intranet zone

For the Intranet zone, by default it only allows single-label names, such as Http://finance. If the resource that needs to be accessed has multiple domain labels, then the workaround is to use the Registry CSP.

Setting the ZoneMap

The ZoneMap is controlled using a registry that can be set through MDM. By default, single-label names such as http://finance are already in the intranet zone. For multi-label names, such as http://finance.net, the ZoneMap needs to be updated.

MDM Policy

OMA URI example:

./Vendor/MSFT/Registry/HKU/S-1-5-21-2702878673-795188819-444038987-2781/Software/Microsoft/Windows/CurrentVersion/Internet%20Settings/ZoneMap/Domains/`<domain name>`/*

as an Integer Value of 1 for each of the domains that you want to SSO into from your device. This adds the specified domains to the Intranet Zone of the Edge browser.

Credential requirements

For VPN, the following types of credentials will be added to credential manager after authentication:

- Username and password
- Certificate-based authentication:
  - TPM KSP Certificate
  - Software KSP Certificates
  - Smart Card Certificate
  - Windows Hello for Business Certificate

The username should also include a domain that can be reached over the connection (VPN or WiFi).

User certificate templates

If the credentials are certificate-based, then the elements in the following table need to be configured for the certificate templates to ensure they can also be used for Kerberos client authentication.

TEMPLATE ELEMENT	CONFIGURATION
SubjectName	The user's distinguished name (DN) where the domain components of the distinguished name reflects the internal DNS namespace when the SubjectAlternativeName does not have the fully qualified UPN required to find the domain controller. This requirement is particularly relevant in multi-forest environments as it ensures a domain controller can be located.
SubjectAlternativeName	The user's fully qualified UPN where a domain name component of the user's UPN matches the organizations internal domain's DNS namespace. This requirement is particularly relevant in multi-forest environments as it ensures a domain controller can be located when the SubjectName does not have the DN required to find the domain controller.
Key Storage Provider (KSP)	If the device is joined to Azure AD, a discrete SSO certificate is used.
EnhancedKeyUsage	One or more of the following EKUs is required: - Client Authentication (for the VPN) - EAP Filtering OID (for Windows Hello for Business) - SmartCardLogon (for Azure AD joined devices) If the domain controllers require smart card EKU either: - SmartCardLogon - id-pkinit-KPClientAuth (1.3.6.1.5.2.3.4) Otherwise: - TLS/SSL Client Authentication (1.3.6.1.5.5.7.3.2)

NDES server configuration

The NDES server is required to be configured so that incoming SCEP requests can be mapped to the correct template to be used. For more information, see Configure certificate infrastructure for SCEP.

Active Directory requirements

You need IP connectivity to a DNS server and domain controller over the network interface so that authentication can succeed as well.

The domain controllers will need to have appropriate KDC certificates for the client to trust them as domain controllers, and since phones are not domain-joined, the root CA of the KDC's certificate must be in the Third-Party Root CA or Smart Card Trusted Roots store.

The domain controllers must be using certificates based on the updated KDC certificate template Kerberos Authentication. This is because Windows 10 Mobile requires strict KDC validation to be enabled. This requires that all authenticating domain controllers run Windows Server 2016, or you'll need to enable strict KDC validation on domain controllers that run previous versions of Windows Server. For more information, see Enabling Strict KDC Validation in Windows Kerberos.

Optimizing Office 365 traffic for remote workers with the native Windows 10 VPN client

3/26/2021 • 14 minutes to read • Edit Online

This article describes how to configure the recommendations in the article Optimize Office 365 connectivity for remote users using VPN split tunneling for the *native Windows 10 VPN client*. This guidance enables VPN administrators to optimize Office 365 usage while still ensuring that all other traffic goes over the VPN connection and through existing security gateways and tooling.

This can be achieved for the native/built-in Windows 10 VPN client using a *Force Tunneling with Exclusions* approach. This allows you to define IP-based exclusions *even when using force tunneling* in order to "split" certain traffic to use the physical interface while still forcing all other traffic via the VPN interface. Traffic addressed to specifically defined destinations (like those listed in the Office 365 optimize categories) will therefore follow a much more direct and efficient path, without the need to traverse or "hairpin" via the VPN tunnel and back out of the corporate network. For cloud-services like Office 365, this makes a huge difference in performance and usability for remote users.

> **NOTE**
> The term *force tunneling with exclusions* is sometimes confusingly called "split tunnels" by other vendors and in some online documentation. For Windows 10 VPN, the term *split tunneling* is defined differently as described in the article VPN routing decisions.

Solution Overview

The solution is based upon the use of a VPN Configuration Service Provider Reference profile (VPNv2 CSP) and the embedded ProfileXML. These are used to configure the VPN profile on the device. Various provisioning approaches can be used to create and deploy the VPN profile as discussed in the article Step 6. Configure Windows 10 client Always On VPN connections.

Typically, these VPN profiles are distributed using a Mobile Device Management solution like Intune, as described in VPN profile options and Configure the VPN client by using Intune.

To enable the use of force tunneling in Windows 10 VPN, the `<RoutingPolicyType>` setting is typically configured with a value of *ForceTunnel* in your existing Profile XML (or script) by way of the following entry, under the `<NativeProfile></NativeProfile>` section:

```
<RoutingPolicyType>ForceTunnel</RoutingPolicyType>
```

In order to define specific force tunnel exclusions, you then need to add the following lines to your existing Profile XML (or script) for each required exclusion, and place them outside of the `<NativeProfile></NativeProfile>` section as follows:

```
<Route>
 <Address>[IP addresses or subnet]</Address>
 <PrefixSize>[IP Prefix]</PrefixSize>
 <ExclusionRoute>true</ExclusionRoute>
</Route>
```

Entries defined by the `[IP Addresses or Subnet]` and `[IP Prefix]` references will consequently be added to the routing table as *more specific route entries* that will use the Internet-connected interface as the default gateway, as opposed to using the VPN interface. You will need to define a unique and separate `<Route></Route>` section for each required exclusion.

An example of a correctly formatted Profile XML configuration for force tunnel with exclusions is shown below:

```
<VPNProfile>
 <NativeProfile>
  <RoutingPolicyType>ForceTunnel</RoutingPolicyType>
 </NativeProfile>
 <Route>
  <Address>203.0.113.0</Address>
  <PrefixSize>24</PrefixSize>
  <ExclusionRoute>true</ExclusionRoute>
 </Route>
 <Route>
  <Address>198.51.100.0</Address>
  <PrefixSize>22</PrefixSize>
  <ExclusionRoute>true</ExclusionRoute>
 </Route>
</VPNProfile>
```

NOTE

The IP addresses and prefix size values in this example are used purely as examples only and should not be used.

Solution Deployment

For Office 365, it is therefore necessary to add exclusions for all IP addresses documented within the optimize categories described in Office 365 URLs and IP address ranges to ensure that they are excluded from VPN force tunneling.

This can be achieved manually by adding the IP addresses defined within the *optimize* category entries to an existing Profile XML (or script) file, or alternatively the following script can be used which dynamically adds the required entries to an existing PowerShell script, or XML file, based upon directly querying the REST-based web service to ensure the correct IP address ranges are always used.

An example of a PowerShell script that can be used to update a force tunnel VPN connection with Office 365 exclusions is provided below.

```
# Copyright (c) Microsoft Corporation.  All rights reserved.
#
# THIS SAMPLE CODE AND INFORMATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND,
# WHETHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED
# WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A PARTICULAR PURPOSE.
# IF THIS CODE AND INFORMATION IS MODIFIED, THE ENTIRE RISK OF USE OR RESULTS IN
# CONNECTION WITH THE USE OF THIS CODE AND INFORMATION REMAINS WITH THE USER.

<#
.SYNOPSIS
    Applies or updates recommended Office 365 optimize IP address exclusions to an existing force tunnel
Windows 10 VPN profile
.DESCRIPTION
    Connects to the Office 365 worldwide commercial service instance endpoints to obtain the latest
published IP address ranges
    Compares the optimized IP addresses with those contained in the supplied VPN Profile (PowerShell or XML
file)
    Adds or updates IP addresses as necessary and saves the resultant file with "-NEW" appended to the file
name
```

```
.PARAMETERS
    Filename and path for a supplied Windows 10 VPN profile file in either PowerShell or XML format
.NOTES
    Requires at least Windows 10 Version 1803 with KB4493437, 1809 with KB4490481, or later
.VERSION
    1.0
#>

param (
    [string]$VPNprofilefile
)

$usage=@"

This script uses the following parameters:

VPNprofilefile - The full path and name of the VPN profile PowerShell script or XML file

EXAMPLES

To check a VPN profile PowerShell script file:

Update-VPN-Profile-Office365-Exclusion-Routes.ps1 -VPNprofilefile [FULLPATH AND NAME OF POWERSHELL SCRIPT
FILE]

To check a VPN profile XML file:

Update-VPN-Profile-Office365-Exclusion-Routes.ps1 -VPNprofilefile [FULLPATH AND NAME OF XML FILE]

"@

# Check if filename has been provided #
if ($VPNprofilefile -eq "")
{
   Write-Host "`nWARNING: You must specify either a PowerShell script or XML filename!" -ForegroundColor Red

    $usage
    exit
}

$FileExtension = [System.IO.Path]::GetExtension($VPNprofilefile)

# Check if XML file exists and is a valid XML file #
if ( $VPNprofilefile -ne "" -and $FileExtension -eq ".xml")
{
    if ( Test-Path $VPNprofilefile )
    {
        $xml = New-Object System.Xml.XmlDocument
        try
        {
            $xml.Load((Get-ChildItem -Path $VPNprofilefile).FullName)

        }
        catch [System.Xml.XmlException]
        {
            Write-Verbose "$VPNprofilefile : $($_.toString())"
            Write-Host "`nWARNING: The VPN profile XML file is not a valid xml file or incorrectly
formatted!" -ForegroundColor Red
            $usage
            exit
        }
    }else
    {
        Write-Host "`nWARNING: VPN profile XML file does not exist or cannot be found!" -ForegroundColor Red
        $usage
        exit
    }
}
```

```
# Check if VPN profile PowerShell script file exists and contains a VPNPROFILE XML section #
if ( $VPNprofilefile -ne "" -and $FileExtension -eq ".ps1")
{
    if ( (Test-Path $VPNprofilefile) )
    {
        if (-Not $(Select-String -Path $VPNprofilefile -Pattern "<VPNPROFILE>") )
        {
            Write-Host "`nWARNING: PowerShell script file does not contain a valid VPN profile XML section
or is incorrectly formatted!" -ForegroundColor Red
            $usage
            exit
        }
    }else
    {
        Write-Host "`nWARNING: PowerShell script file does not exist or cannot be found!"-ForegroundColor
Red
        $usage
        exit
    }
}

# Define Office 365 endpoints and service URLs #
$ws = "https://endpoints.office.com"
$baseServiceUrl = "https://endpoints.office.com"

# Path where client ID and latest version number will be stored #
$datapath = $Env:TEMP + "\endpoints_clientid_latestversion.txt"

# Fetch client ID and version if data file exists; otherwise create new file #
if (Test-Path $datapath)
{
    $content = Get-Content $datapath
    $clientRequestId = $content[0]
    $lastVersion = $content[1]

}else
{
    $clientRequestId = [GUID]::NewGuid().Guid
    $lastVersion = "0000000000"
    @($clientRequestId, $lastVersion) | Out-File $datapath
}

# Call version method to check the latest version, and pull new data if version number is different #
$version = Invoke-RestMethod -Uri ($ws + "/version?clientRequestId=" + $clientRequestId)

if ($version[0].latest -gt $lastVersion)
{

    Write-Host
    Write-Host "A new version of Office 365 worldwide commercial service instance endpoints has been
detected!" -ForegroundColor Cyan

    # Write the new version number to the data file #
    @($clientRequestId, $version[0].latest) | Out-File $datapath
}

# Invoke endpoints method to get the new data #
$uri = "$baseServiceUrl" + "/endpoints/worldwide?clientRequestId=$clientRequestId"

# Invoke endpoints method to get the data for the VPN profile comparison #
$endpointSets = Invoke-RestMethod -Uri ($uri)
$Optimize = $endpointSets | Where-Object { $_.category -eq "Optimize" }
$optimizeIpsv4 = $Optimize.ips | Where-Object { ($_).contains(".") } | Sort-Object -Unique

# Temporarily include additional IP address until Teams client update is released
$optimizeIpsv4 += "13.107.60.1/32"

# Process PowerShell script file start #
if ($VPNprofilefile -ne "" -and $FileExtension -eq ".ps1")
```

```
if ($VPNprofilefile -ne "" -and $FileExtension -eq ".ps1")
{
    Write-host "`nStarting PowerShell script exclusion route check...`n" -ForegroundColor Cyan

    # Clear Variables to allow re-run testing #

    $ARRVPN=$null              # Array to hold VPN addresses from VPN profile PowerShell file #
    $In_Opt_Only=$null         # Variable to hold IP addresses that only appear in the optimize list #
    $In_VPN_Only=$null         # Variable to hold IP addresses that only appear in the VPN profile
PowerShell file #

    # Extract the Profile XML from the ps1 file #

    $regex = '(?sm).*^*.<VPNProfile>\r?\n(.*?)\r?\n</VPNProfile>.*'

    # Create xml format variable to compare with the optimize list #

    $xmlbody=(Get-Content -Raw $VPNprofilefile) -replace $regex, '$1'
    [xml]$VPNprofilexml="<VPNProfile>"+$xmlbody+"</VPNProfile>"

        # Loop through each address found in VPNPROFILE XML section #
        foreach ($Route in $VPNprofilexml.VPNProfile.Route)
        {
        $VPNIP=$Route.Address+"/"+$Route.PrefixSize
        [array]$ARRVPN=$ARRVPN+$VPNIP
        }

    # In optimize address list only #
    $In_Opt_Only= $optimizeIpsv4 | Where {$ARRVPN -NotContains $_}

    # In VPN list only #
    $In_VPN_only =$ARRVPN | Where {$optimizeIpsv4 -NotContains $_}
    [array]$Inpfile = get-content $VPNprofilefile

    if ($In_Opt_Only.Count -gt 0 )
    {
        Write-Host "Exclusion route IP addresses are unknown, missing, or need to be updated in the VPN
profile`n" -ForegroundColor Red

         [int32]$insline=0

            for ($i=0; $i -lt $Inpfile.count; $i++)
            {
                if ($Inpfile[$i] -match "</NativeProfile>")
                {
                $insline += $i # Record the position of the line after the NativeProfile section ends #
                }
            }
            $OFS = "`r`n"
                foreach ($NewIP in $In_Opt_Only)
                {
                    # Add the missing IP address(es) #
                    $IPInfo=$NewIP.Split("/")
                    $InpFile[$insline] += $OFS+"    <Route>"
                    $InpFile[$insline] += $OFS+"      <Address>"+$IPInfo[0].Trim()+"</Address>"
                    $InpFile[$insline] += $OFS+"      <PrefixSize>"+$IPInfo[1].Trim()+"</PrefixSize>"
                    $InpFile[$insline] += $OFS+"      <ExclusionRoute>true</ExclusionRoute>"
                    $InpFile[$insline] += $OFS+"    </Route>"
                }
             # Update fileName and write new PowerShell file #
             $NewFileName=(Get-Item $VPNprofilefile).Basename + "-NEW.ps1"
             $OutFile=$(Split-Path $VPNprofilefile -Parent)+"\"+$NewFileName
             $InpFile | Set-Content $OutFile
             Write-Host "Exclusion routes have been added to VPN profile and output to a separate PowerShell
script file; the original file has not been modified`n" -ForegroundColor Green
    }else
    {
        Write-Host "Exclusion route IP addresses are correct and up to date in the VPN profile`n" -
ForegroundColor Green
        $OutFile=$VPNprofilefile
```

```
        $Outfile=$VPNprofilefile
    }

if ( $In_VPN_Only.Count -gt 0 )
{
    Write-Host "Unknown exclusion route IP addresses have been found in the VPN profile`n" -ForegroundColor
Yellow

        foreach ($OldIP in $In_VPN_Only)
        {
            [array]$Inpfile = get-content $Outfile
            $IPInfo=$OldIP.Split("/")
            Write-Host "Unknown exclusion route IP address"$IPInfo[0]"has been found in the VPN profile - Do
you wish to remove it? (Y/N)`n" -ForegroundColor Yellow
            $matchstr="<Address>"+$IPInfo[0].Trim()+"</Address>"
            $DelAns=Read-host
                if ($DelAns.ToUpper() -eq "Y")
                {
                    [int32]$insline=0
                        for ($i=0; $i -lt $Inpfile.count; $i++)
                        {
                            if ($Inpfile[$i] -match $matchstr)
                            {
                                $insline += $i # Record the position of the line for the string match #
                            }
                        }
                        # Remove entries from XML #
                        $InpFile[$insline-1]="REMOVETHISLINE"
                        $InpFile[$insline]="REMOVETHISLINE"
                        $InpFile[$insline+1]="REMOVETHISLINE"
                        $InpFile[$insline+2]="REMOVETHISLINE"
                        $InpFile[$insline+3]="REMOVETHISLINE"
                        $InpFile=$InpFile | Where-Object {$_ -ne "REMOVETHISLINE"}

                        # Update filename and write new PowerShell file #
                        $NewFileName=(Get-Item $VPNprofilefile).Basename + "-NEW.xml"
                        $OutFile=$(Split-Path $VPNprofilefile -Parent)+"\"+$NewFileName
                        $Inpfile | Set-content $OutFile
                        Write-Host "`nAddress"$IPInfo[0]"exclusion route has been removed from the VPN
profile and output to a separate PowerShell script file; the original file has not been modified`n" -
ForegroundColor Green

                }else
                {
                    Write-Host "`nExclusion route IP address has *NOT* been removed from the VPN profile`n"
-ForegroundColor Green
                }
        }
 }
}

# Process XML file start #
if ($VPNprofilefile -ne "" -and $FileExtension -eq ".xml")
{
    Write-host "`nStarting XML file exclusion route check...`n" -ForegroundColor Cyan

    # Clear variables to allow re-run testing #
    $ARRVPN=$null              # Array to hold VPN addresses from the XML file #
    $In_Opt_Only=$null         # Variable to hold IP Addresses that only appear in optimize list #
    $In_VPN_Only=$null         # Variable to hold IP Addresses that only appear in the VPN profile XML file
#

    # Extract the Profile XML from the XML file #
    $regex = '(?sm).*^*.<VPNProfile>\r?\n(.*?)\r?\n</VPNProfile>.*'

    # Create xml format variable to compare with optimize list #
    $xmlbody=(Get-Content -Raw $VPNprofilefile) -replace $regex, '$1'
    [xml]$VPNRulesxml="$xmlbody"

    # Load the exclusion addresses found in VPNPROFILE file #
```

```
        # Loop through each address found in VPNPROFILE file #
        foreach ($Route in $VPNRulesxml.VPNProfile.Route)
        {
            $VPNIP=$Route.Address+"/"+$Route.PrefixSize
            [array]$ARRVPN=$ARRVPN+$VPNIP
        }

    # In optimize address list only #
    $In_Opt_Only= $optimizeIpsv4 | Where {$ARRVPN -NotContains $_}

    # In VPN list only #
    $In_VPN_only =$ARRVPN | Where {$optimizeIpsv4 -NotContains $_}
    [System.Collections.ArrayList]$Inpfile = get-content $VPNprofilefile

        if ($In_Opt_Only.Count -gt 0 )
        {
            Write-Host "Exclusion route IP addresses are unknown, missing, or need to be updated in the VPN
profile`n" -ForegroundColor Red

                foreach ($NewIP in $In_Opt_Only)
                {
                    # Add the missing IP address(es) #
                    $IPInfo=$NewIP.Split("/")
                    $routes += "<Route>`n"+"`t<Address>"+$IPInfo[0].Trim()+"
</Address>`n"+"`t<PrefixSize>"+$IPInfo[1].Trim()+"
</PrefixSize>`n"+"`t<ExclusionRoute>true</ExclusionRoute>`n"+"</Route>`n"
                }
            $inspoint = $Inpfile.IndexOf("</VPNProfile>")
            $Inpfile.Insert($inspoint,$routes)

            # Update filename and write new XML file #
            $NewFileName=(Get-Item $VPNprofilefile).Basename + "-NEW.xml"
            $OutFile=$(Split-Path $VPNprofilefile -Parent)+"\"+$NewFileName
            $InpFile | Set-Content $OutFile
            Write-Host "Exclusion routes have been added to VPN profile and output to a separate XML file;
the original file has not been modified`n`n" -ForegroundColor Green

        }else
        {
            Write-Host "Exclusion route IP addresses are correct and up to date in the VPN profile`n" -
ForegroundColor Green
            $OutFile=$VPNprofilefile
        }

        if ( $In_VPN_Only.Count -gt 0 )
        {
            Write-Host "Unknown exclusion route IP addresses found in the VPN profile`n" -ForegroundColor
Yellow

            foreach ($OldIP in $In_VPN_Only)
            {
                [array]$Inpfile = get-content $OutFile
                $IPInfo=$OldIP.Split("/")
                Write-Host "Unknown exclusion route IP address"$IPInfo[0]"has been found in the VPN profile
- Do you wish to remove it? (Y/N)`n" -ForegroundColor Yellow
                $matchstr="<Route>"+"<Address>"+$IPInfo[0].Trim()+"</Address>"+"
<PrefixSize>"+$IPInfo[1].Trim()+"</PrefixSize>"+"<ExclusionRoute>true</ExclusionRoute>"+"</Route>"
                $DelAns=Read-host
                    if ($DelAns.ToUpper() -eq "Y")
                    {
                        # Remove unknown IP address(es) #
                        $inspoint = $Inpfile[0].IndexOf($matchstr)
                        $Inpfile[0] = $Inpfile[0].Replace($matchstr,"")

                        # Update filename and write new XML file #
                        $NewFileName=(Get-Item $VPNprofilefile).Basename + "-NEW.xml"
                        $OutFile=$(Split-Path $VPNprofilefile -Parent)+"\"+$NewFileName
                        $Inpfile | Set-content $OutFile
                        Write-Host "`nAddress"$IPInfo[0]"exclusion route has been removed from the VPN
```

```
profile and output to a separate XML file; the original file has not been modified`n" -ForegroundColor Green

                    }else
                    {
                        Write-Host "`nExclusion route IP address has *NOT* been removed from the VPN
profile`n" -ForegroundColor Green
                    }
                }
            }
}
```

Version Support

This solution is supported with the following versions of Windows:

- Windows 10 1903/1909 and newer: Included, no action needed
- Windows 10 1809: At least KB4490481
- Windows 10 1803: At least KB4493437
- Windows 10 1709 and lower: Exclusion routes are not supported
- Windows 10 Enterprise 2019 LTSC: At least KB4490481
- Windows 10 Enterprise 2016 LTSC: Exclusion routes are not supported
- Windows 10 Enterprise 2015 LTSC: Exclusion routes are not supported

Microsoft strongly recommends that the latest available Windows 10 cumulative update always be applied.

Other Considerations

You should also be able to adapt this approach to include necessary exclusions for other cloud-services that can be defined by known/static IP addresses; exclusions required for Cisco WebEx or Zoom are good examples.

Examples

An example of a PowerShell script that can be used to create a force tunnel VPN connection with Office 365 exclusions is provided below, or refer to the guidance in Create the ProfileXML configuration files to create the initial PowerShell script:

```
# Copyright (c) Microsoft Corporation.  All rights reserved.
#
# THIS SAMPLE CODE AND INFORMATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND,
# WHETHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED
# WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A PARTICULAR PURPOSE.
# IF THIS CODE AND INFORMATION IS MODIFIED, THE ENTIRE RISK OF USE OR RESULTS IN
# CONNECTION WITH THE USE OF THIS CODE AND INFORMATION REMAINS WITH THE USER.

<#
.SYNOPSIS
    Configures an AlwaysOn IKEv2 VPN Connection using a basic script
.DESCRIPTION
    Configures an AlwaysOn IKEv2 VPN Connection with proxy PAC information and force tunneling
.PARAMETERS
    Parameters are defined in a ProfileXML object within the script itself
.NOTES
    Requires at least Windows 10 Version 1803 with KB4493437, 1809 with KB4490481, or later
.VERSION
    1.0
#>
```

```
<#-- Define Key VPN Profile Parameters --#>
$ProfileName = 'Contoso VPN with Office 365 Exclusions'
$ProfileNameEscaped = $ProfileName -replace ' ', '%20'

<#-- Define VPN ProfileXML --#>
$ProfileXML = '<VPNProfile>
      <RememberCredentials>true</RememberCredentials>
    <DnsSuffix>corp.contoso.com</DnsSuffix>
    <AlwaysOn>true</AlwaysOn>
    <TrustedNetworkDetection>corp.contoso.com</TrustedNetworkDetection>
<NativeProfile>
        <Servers>edge1.contoso.com</Servers>
        <RoutingPolicyType>ForceTunnel</RoutingPolicyType>
        <NativeProtocolType>IKEv2</NativeProtocolType>
        <Authentication>
            <MachineMethod>Certificate</MachineMethod>
        </Authentication>
    </NativeProfile>
    <Route>
      <Address>13.107.6.152</Address>
      <PrefixSize>31</PrefixSize>
      <ExclusionRoute>true</ExclusionRoute>
    </Route>
    <Route>
      <Address>13.107.18.10</Address>
      <PrefixSize>31</PrefixSize>
      <ExclusionRoute>true</ExclusionRoute>
    </Route>
    <Route>
      <Address>13.107.128.0</Address>
      <PrefixSize>22</PrefixSize>
      <ExclusionRoute>true</ExclusionRoute>
    </Route>
    <Route>
      <Address>23.103.160.0</Address>
      <PrefixSize>20</PrefixSize>
      <ExclusionRoute>true</ExclusionRoute>
    </Route>
    <Route>
      <Address>40.96.0.0</Address>
      <PrefixSize>13</PrefixSize>
      <ExclusionRoute>true</ExclusionRoute>
    </Route>
    <Route>
      <Address>40.104.0.0</Address>
      <PrefixSize>15</PrefixSize>
      <ExclusionRoute>true</ExclusionRoute>
    </Route>
    <Route>
      <Address>52.96.0.0</Address>
      <PrefixSize>14</PrefixSize>
      <ExclusionRoute>true</ExclusionRoute>
    </Route>
    <Route>
      <Address>131.253.33.215</Address>
      <PrefixSize>32</PrefixSize>
      <ExclusionRoute>true</ExclusionRoute>
    </Route>
    <Route>
      <Address>132.245.0.0</Address>
      <PrefixSize>16</PrefixSize>
      <ExclusionRoute>true</ExclusionRoute>
    </Route>
    <Route>
      <Address>150.171.32.0</Address>
      <PrefixSize>22</PrefixSize>
      <ExclusionRoute>true</ExclusionRoute>
    </Route>
    <Route>
```

```
    <Route>
      <Address>191.234.140.0</Address>
      <PrefixSize>22</PrefixSize>
      <ExclusionRoute>true</ExclusionRoute>
    </Route>
    <Route>
      <Address>204.79.197.215</Address>
      <PrefixSize>32</PrefixSize>
    <ExclusionRoute>true</ExclusionRoute>
    </Route>
    <Route>
      <Address>13.107.136.0</Address>
      <PrefixSize>22</PrefixSize>
      <ExclusionRoute>true</ExclusionRoute>
    </Route>
    <Route>
      <Address>40.108.128.0</Address>
      <PrefixSize>17</PrefixSize>
      <ExclusionRoute>true</ExclusionRoute>
    </Route>
    <Route>
      <Address>52.104.0.0</Address>
      <PrefixSize>14</PrefixSize>
      <ExclusionRoute>true</ExclusionRoute>
    </Route>
    <Route>
      <Address>104.146.128.0</Address>
      <PrefixSize>17</PrefixSize>
      <ExclusionRoute>true</ExclusionRoute>
    </Route>
    <Route>
      <Address>150.171.40.0</Address>
      <PrefixSize>22</PrefixSize>
      <ExclusionRoute>true</ExclusionRoute>
    </Route>
    <Route>
      <Address>13.107.60.1</Address>
      <PrefixSize>32</PrefixSize>
      <ExclusionRoute>true</ExclusionRoute>
    </Route>
    <Route>
      <Address>13.107.64.0</Address>
     <PrefixSize>18</PrefixSize>
    <ExclusionRoute>true</ExclusionRoute>
    </Route>
    <Route>
      <Address>52.112.0.0</Address>
      <PrefixSize>14</PrefixSize>
      <ExclusionRoute>true</ExclusionRoute>
    </Route>
    <Route>
      <Address>52.120.0.0</Address>
      <PrefixSize>14</PrefixSize>
    <ExclusionRoute>true</ExclusionRoute>
    </Route>
    <Proxy>
            <AutoConfigUrl>http://webproxy.corp.contoso.com/proxy.pac</AutoConfigUrl>
      </Proxy>
</VPNProfile>'

<#-- Convert ProfileXML to Escaped Format --#>
$ProfileXML = $ProfileXML -replace '<', '&lt;'
$ProfileXML = $ProfileXML -replace '>', '&gt;'
$ProfileXML = $ProfileXML -replace '"', '&quot;'

<#-- Define WMI-to-CSP Bridge Properties --#>
$nodeCSPURI = './Vendor/MSFT/VPNv2'
$namespaceName = "root\cimv2\mdm\dmmap"
$className = "MDM_VPNv2_01"
```

```
<#-- Define WMI Session --#>
$session = New-CimSession

<#-- Detect and Delete Previous VPN Profile --#>
try
{
    $deleteInstances = $session.EnumerateInstances($namespaceName, $className, $options)
    foreach ($deleteInstance in $deleteInstances)
    {
        $InstanceId = $deleteInstance.InstanceID
        if ("$InstanceId" -eq "$ProfileNameEscaped")
        {
            $session.DeleteInstance($namespaceName, $deleteInstance, $options)
            $Message = "Removed $ProfileName profile $InstanceId"
            Write-Host "$Message"
        } else {
            $Message = "Ignoring existing VPN profile $InstanceId"
            Write-Host "$Message"
        }
    }
}
catch [Exception]
{
    $Message = "Unable to remove existing outdated instance(s) of $ProfileName profile: $_"
    Write-Host "$Message"
    exit
}

<#-- Create VPN Profile --#>
try
{
    $newInstance = New-Object Microsoft.Management.Infrastructure.CimInstance $className, $namespaceName
    $property = [Microsoft.Management.Infrastructure.CimProperty]::Create("ParentID", "$nodeCSPURI",
'String', 'Key')
    $newInstance.CimInstanceProperties.Add($property)
    $property = [Microsoft.Management.Infrastructure.CimProperty]::Create("InstanceID",
"$ProfileNameEscaped", 'String', 'Key')
    $newInstance.CimInstanceProperties.Add($property)
    $property = [Microsoft.Management.Infrastructure.CimProperty]::Create("ProfileXML", "$ProfileXML",
'String', 'Property')
    $newInstance.CimInstanceProperties.Add($property)

    $session.CreateInstance($namespaceName, $newInstance, $options)
    $Message = "Created $ProfileName profile."
    Write-Host "$Message"
    Write-Host "$ProfileName profile summary:"
    $session.EnumerateInstances($namespaceName, $className, $options)
}
catch [Exception]
{
    $Message = "Unable to create $ProfileName profile: $_"
    Write-Host "$Message"
    exit
}

$Message = "Script Complete"
Write-Host "$Message"
```

An example of an Intune-ready XML file that can be used to create a force tunnel VPN connection with Office 365 exclusions is provided below, or refer to the guidance in Create the ProfileXML configuration files to create the initial XML file.

NOTE

This XML is formatted for use with Intune and cannot contain any carriage returns or whitespace.

```
<VPNProfile><RememberCredentials>true</RememberCredentials><DnsSuffix>corp.contoso.com</DnsSuffix>
<AlwaysOn>true</AlwaysOn><TrustedNetworkDetection>corp.contoso.com</TrustedNetworkDetection><NativeProfile>
<Servers>edge1.contoso.com</Servers><RoutingPolicyType>ForceTunnel</RoutingPolicyType>
<NativeProtocolType>IKEv2</NativeProtocolType><Authentication><MachineMethod>Certificate</MachineMethod>
</Authentication></NativeProfile><Route><Address>13.107.6.152</Address><PrefixSize>31</PrefixSize>
<ExclusionRoute>true</ExclusionRoute></Route><Route><Address>13.107.18.10</Address>
<PrefixSize>31</PrefixSize><ExclusionRoute>true</ExclusionRoute></Route><Route>
<Address>13.107.128.0</Address><PrefixSize>22</PrefixSize><ExclusionRoute>true</ExclusionRoute></Route>
<Route><Address>23.103.160.0</Address><PrefixSize>20</PrefixSize><ExclusionRoute>true</ExclusionRoute>
</Route><Route><Address>40.96.0.0</Address><PrefixSize>13</PrefixSize><ExclusionRoute>true</ExclusionRoute>
</Route><Route><Address>40.104.0.0</Address><PrefixSize>15</PrefixSize><ExclusionRoute>true</ExclusionRoute>
</Route><Route><Address>52.96.0.0</Address><PrefixSize>14</PrefixSize><ExclusionRoute>true</ExclusionRoute>
</Route><Route><Address>131.253.33.215</Address><PrefixSize>32</PrefixSize>
<ExclusionRoute>true</ExclusionRoute></Route><Route><Address>132.245.0.0</Address>
<PrefixSize>16</PrefixSize><ExclusionRoute>true</ExclusionRoute></Route><Route>
<Address>150.171.32.0</Address><PrefixSize>22</PrefixSize><ExclusionRoute>true</ExclusionRoute></Route>
<Route><Address>191.234.140.0</Address><PrefixSize>22</PrefixSize><ExclusionRoute>true</ExclusionRoute>
</Route><Route><Address>204.79.197.215</Address><PrefixSize>32</PrefixSize>
<ExclusionRoute>true</ExclusionRoute></Route><Route><Address>13.107.136.0</Address>
<PrefixSize>22</PrefixSize><ExclusionRoute>true</ExclusionRoute></Route><Route>
<Address>40.108.128.0</Address><PrefixSize>17</PrefixSize><ExclusionRoute>true</ExclusionRoute></Route>
<Route><Address>52.104.0.0</Address><PrefixSize>14</PrefixSize><ExclusionRoute>true</ExclusionRoute></Route>
<Route><Address>104.146.128.0</Address><PrefixSize>17</PrefixSize><ExclusionRoute>true</ExclusionRoute>
</Route><Route><Address>150.171.40.0</Address><PrefixSize>22</PrefixSize>
<ExclusionRoute>true</ExclusionRoute></Route><Route><Address>13.107.60.1</Address>
<PrefixSize>32</PrefixSize><ExclusionRoute>true</ExclusionRoute></Route><Route>
<Address>13.107.64.0</Address><PrefixSize>18</PrefixSize><ExclusionRoute>true</ExclusionRoute></Route>
<Route><Address>52.112.0.0</Address><PrefixSize>14</PrefixSize><ExclusionRoute>true</ExclusionRoute></Route>
<Route><Address>52.120.0.0</Address><PrefixSize>14</PrefixSize><ExclusionRoute>true</ExclusionRoute></Route>
<Proxy><AutoConfigUrl>http://webproxy.corp.contoso.com/proxy.pac</AutoConfigUrl></Proxy></VPNProfile>
```