

Server Suite

Auditing and Analysis Scripting Guide

Version: 2024.x

Publication Date: 7/14/2025

Server Suite Auditing and Analysis Scripting Guide

Version: 2024.x, Publication Date: 7/14/2025

© Delinea, 2025

Warranty Disclaimer

DELINEA AND ITS AFFILIATES, AND/OR ITS AND THEIR RESPECTIVE SUPPLIERS, MAKE NO REPRESENTATIONS ABOUT THE SUITABILITY OF THE INFORMATION CONTAINED IN THE DOCUMENTS AND RELATED GRAPHICS, THE SOFTWARE AND SERVICES, AND OTHER MATERIAL PUBLISHED ON OR ACCESSIBLE THROUGH THIS SITE FOR ANY PURPOSE. ALL SUCH MATERIAL IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND. DELINEA AND ITS AFFILIATES, AND/OR ITS AND THEIR RESPECTIVE SUPPLIERS, HEREBY DISCLAIM ALL WARRANTIES AND CONDITIONS WITH REGARD TO SUCH MATERIAL, INCLUDING ALL IMPLIED WARRANTIES AND CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT.

THE MATERIAL PUBLISHED ON THIS SITE COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN. DELINEA AND ITS AFFILIATES, AND/OR ITS AND THEIR RESPECTIVE SUPPLIERS, MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE MATERIAL DESCRIBED HEREIN AT ANY TIME.

Disclaimer of Liability

IN NO EVENT SHALL DELINEA AND ITS AFFILIATES, AND/OR ITS AND THEIR RESPECTIVE SUPPLIERS, BE LIABLE FOR ANY SPECIAL, INDIRECT, OR CONSEQUENTIAL DAMAGES (INCLUDING LOSS OF USE, DATA, PROFITS OR OTHER ECONOMIC ADVANTAGE) OR ANY DAMAGES WHATSOEVER, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE, OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF SOFTWARE, DOCUMENTS, PROVISION OF OR FAILURE TO PROVIDE SERVICES, OR MATERIAL AVAILABLE FROM THIS SITE.

Table of Contents

Auditing and Analysis Scripting Guide	i
About this Guide	1
Intended Audience	1
Using this Guide	1
Compatibility and Limitations of This Guide	1
Developing Scripts for Administrative Tasks	1
Getting Started with cmdlets For Powershell	2
Managing Unix Information from a Windows Computer	2
Writing Programs in Other Languages	2
Accessing Audit Information Using Native Interfaces	2
Installing the Audit Module for PowerShell	3
About the Standalone Package	4
Running the Setup Program	4
Importing the cmdlets into the Windows PowerShell Console	4
Managing Audit-Related Objects with Windows PowerShell Scripts	5
Using cmdlets to Manage Auditing	5
Preparing the Environment to Run cmdlets	6
Setting the Preferred Domain Controller	6
Setting the Logging Level	7
Running cmdlets under Another Account	7
Organizing cmdlet Operations in a Sequence	7
Checking for Valid Licenses	8
Specifying Parameters using Different Formats	8
Working with Sample Scripts	9
Writing Your Own Scripts	10
Exporting Specific Session Fields For A Report	11
Checking the status of agents and collectors	11
Recommendations for Writing Custom Scripts	11
Executing Custom Scripts	12
Getting Information about the cmdlet Available	13
Auditing-Related Objects and Properties	13
CdaAccessAccount	13
CdaAdPrincipal	14
CdaAgent	14
CdaAuditEvent	15
CdaAuditRole	16
CdaAuditRoleAssignment	16
CdaAuditRoleRight	17
CdaAuditScope	17

Table of Contents

CdaAuditSession	17
CdaAuditSessionTag	19
CdaAuditSessionDataIntegrityStatus	19
CdaAuditStore	19
CdaAuditStoreRight	20
CdaCollector	20
CdaDatabase	21
CdaDetailedExecution	22
CdaInstallation	23
CdaInstallationRight	23
CdaManagementDatabase	24
CdaManagementDatabaseRight	24
CdaMonitoredExecution	25
CdaMonitoredFile	26
CdaQuery	26
CdaQueryRight	27
CdaSearchCriteria	27
CdaUnixCommand	28
CdaUnixCommandTranscript	29
CdaUserEvent	29
CdaWindowsEvent	29

About this Guide

This guide describes the Audit Module for PowerShell command set. These PowerShell cmdlets run on Windows computers and can be used to automate auditing-related management tasks, such as the creation of new audit store databases. You can also use the cmdlets to get or set properties for an installation and perform other administrative tasks. For example, you can write scripts to find and remove sessions matching specific criteria, export audit trail events, or manage audit roles and auditor assignments.

Intended Audience

This guide provides information for auditing infrastructure administrators who want to use PowerShell scripts to manage auditing-related features and components of Server Suite software. This document supplements the help provided within the PowerShell environment using the get-help function. Whereas the get-help function describes each cmdlet in detail, this document provides an introduction to the Auditing Module for Windows PowerShell objects and how you can use PowerShell cmdlets and scripts to perform auditing-related tasks.

This guide assumes general knowledge of PowerShell scripts and syntax, and of the Windows PowerShell modules used to write scripts for Active Directory. You should also be familiar with basic Active Directory operations, such as connecting to a domain controller and managing objects and attributes.

In addition to scripting skills, you should be familiar with Server Suite architecture, terms, and concepts, and know how to perform administrative tasks for the Audit & Monitoring Service and for the platforms you support.

Using this Guide

This guide discusses audit-related administrative tasks using PowerShell-based command-line programs. This information is intended to help you develop scripts for managing the auditing infrastructure, including collectors, audited computers, the audit management database, and the active and attached audit store databases and performing other administrative tasks on Windows computers. With scripts, you can automate the administrative or analytical tasks you might otherwise perform using Audit Manager or Audit Analyzer.

Compatibility and Limitations of This Guide

The information in this guide is intended for use with Server Suite 2015, or later. Although intended to be accurate and up-to-date, interfaces are subject to change without notice and can become incompatible or obsolete when a newer version of the software is released.

In general, application programming interfaces are also intended to be backward-compatible, but are not guaranteed to work with older versions of the software. Because the authentication and privilege elevation cmdlets are subject to change, enhancement, or replacement, the information in this guide can also become incomplete, obsolete, or unsupported in future versions. If you are unsure whether this guide is appropriate for the version of the software you have installed, you can consult the Delinea website or Delinea Support to find out if another version of this guide is available.

Developing Scripts for Administrative Tasks

The Audit Module for PowerShell consists of the following:

Developing Scripts for Administrative Tasks

- Application programming interfaces in the form of PowerShell command-line programs, or cmdlets, that are packaged in dynamic link libraries (.DLLs).
- A PowerShell help file that includes complete cmdlet reference information and this scripting guide.
- Sample scripts to illustrate administrative tasks.

On Windows computers, you can use the Audit Module for PowerShell to develop your own custom scripts that access, create, or modify auditing components or auditing-related information, such as session activity and audit trail events.

Getting Started with cmdlets For Powershell

The Audit Module for PowerShell consists of “cmdlets” that you can use to manage Server Suite-specific information. A “cmdlet” is a lightweight command-line program that runs in the Windows PowerShell environment. In most cases, cmdlets perform a basic operation and return a Microsoft .NET Framework object to the next command in the pipeline.

The cmdlets in the Audit Module for PowerShell module enable you to access, create, modify, and remove information about Server Suite auditing components and auditing-related information. Using the cmdlets you can manage the entire auditing infrastructure, including installation properties, collectors, audited computers, the audit management database, and the active and attached audit store databases. You can also use cmdlets to manage permissions, audit roles, and role assignments and to work with captured session activity and audit trail events. By combining cmdlets into scripts, you can also automate common administrative tasks, such as the creation of new audit store databases.

Managing Unix Information from a Windows Computer

You can use the cmdlets to work with information for any Server Suite-managed computer where you have enabled the auditing service. However, you can only run the cmdlets on Windows-based computers that have the Windows PowerShell command-line shell available. If you want to develop scripts that run directly on UNIX computers, you can use the ADEdit program (adedit). However, the ADEdit application only provides functionality similar to the cmdlets for access control and privilege management. You cannot use ADEdit to develop scripts for auditing-specific tasks. For detailed information about using ADEdit, see the *ADEdit Command Reference and Scripting Guide*.

Writing Programs in Other Languages

If you want to develop programs or scripts that run on Windows but outside of the Windows PowerShell environment, you can use the Component Object Model (COM) interface that is available as part of the auditing software development kit (SDK). For information about auditing-specific objects you can use the COM-based applications, see the *Database Management Guide*.

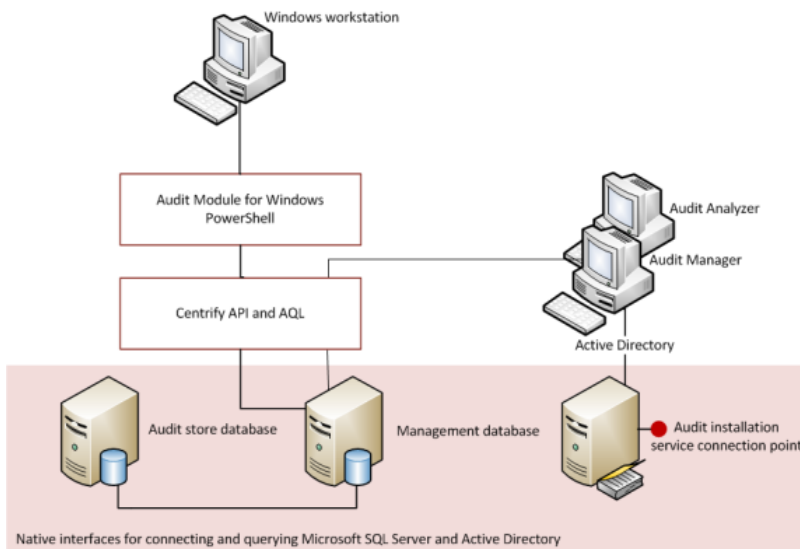
Accessing Audit Information Using Native Interfaces

The Audit Module for PowerShell cmdlets connect to Active Directory or to Microsoft SQL Server databases to access audit information. You can, therefore, write PowerShell scripts to automate procedures that you would otherwise perform interactively using Audit Manager or Audit Analyzer.

Installing the Audit Module for PowerShell

The cmdlets rely on the underlying interfaces provided by Microsoft Active Directory Service Interfaces (ADSI), Microsoft SQL Server AQL query language, and Server Suite Windows API objects. The ADSI and AQL layers provide low-level functions that permit applications to read and write data. The cmdlets provide a task and object-based level of abstraction for retrieving and manipulating Server Suite audit information so that you do not need to know the details of how the data is stored or how to use any of the underlying ADSI or AQL functions directly.

The following figure illustrates how the Audit Module for PowerShell provides a layer of abstraction between the data stored in Active Directory, the management database, the audit store databases, and your scripting environment.



The Audit Module for PowerShell provides a logical view of the auditing infrastructure and captured information, eliminating the need to know the details of how data is stored in the management database or the audit store databases when performing common administrative tasks. The cmdlets also provide a simple method for accessing audit-related objects without needing to write complex AQL queries.

Using the cmdlets, you can write scripts that automatically create and make active new audit store databases or delete sessions that are no longer of interest. In most cases, the cmdlets enable you to perform exactly the same tasks from the command line that you would otherwise perform interactively using Audit Manager or Audit Analyzer.

Installing the Audit Module for PowerShell

You can install the Audit Module for PowerShell from the Server Suite setup program or as a separate package. It includes the auditing-related cmdlets for Windows PowerShell, sample scripts, and documentation for performing common administrative tasks using PowerShell scripts. This chapter describes how to install the software on a Windows computer.

The following topics are covered:

- About the standalone package
- Running the setup program

- Importing the cmdlets into the Windows PowerShell console

About the Standalone Package

The cmdlets that run in Windows PowerShell are defined in dynamic link libraries that can be installed on any computer where you install other Windows-based components, such as Audit Manager or Audit Analyzer. You can also install these libraries separately, along with sample scripts and documentation, onto computers where no Server Suite software is installed.

If you did not install the Audit Module for PowerShell as a component of a Server Suite installation, you can install it separately. The Audit Module for PowerShell files are available in the same disk image from which you installed Server Suite.

Running the Setup Program

You run the setup program to install the Audit Module for PowerShell files.

To run the standalone setup program

1. Select the downloaded file, right-click, then select **Extract All** to extract the compressed files to a folder.
2. In the Windows disk image for Server Suite, navigate to the /DirectAudit/Powershell folder and double-click the standalone executable to start the setup program.

For example, double click the Delinea DirectAudit PowerShell64.exe file.

3. At the Welcome page, click **Next**.
4. Select **I accept the terms in the License Agreement**, then click **Next**.
5. Accept the default location or click **Change** to choose a different location, then click **OK**.

If you accept the default location, the cmdlets are available in a separate Audit Module for PowerShell console.

If you want the cmdlets to be available in the default Windows PowerShell console with other PowerShell modules, select the following location:

`C:\windows\System32\windowsPowerShell\v1.0\Modules\Centrify.DirectAudit.PowerShell`

6. Verify the path to the Centrify.DirectAudit.PowerShell folder, then click **Next**.
7. Click **Install**.
8. Click **Finish** to complete the installation.

Importing the cmdlets into the Windows PowerShell Console

If you install the Audit Module for PowerShell in the default location, it is a self-contained Windows PowerShell console. If you install the files in the location for system modules so that cmdlets from other modules are available in the same console, you should import the Audit Module for PowerShell into your default Windows PowerShell console.



Note: According to Microsoft, the Import System Modules task is available only Windows 7 and Windows Server 2008 R2 when Windows PowerShell 3.0 is not installed on the computer. Beginning in Windows PowerShell 3.0, modules are imported automatically the first time that you use a cmdlet in the module.

To import the auditing module

Managing Audit-Related Objects with Windows PowerShell Scripts

1. On the Start menu, select Windows PowerShell to display a menu extension with a list of Tasks.
2. On the Tasks menu, select **Import System Modules** to import the auditing module and open the Windows PowerShell console.
3. Verify the installation and import completed successfully by typing the following command:

```
get-command *-Cda*
```

You should see a listing of the audit cmdlets, similar to the following:

CommandType	Name	Version	Source
-----	-----	-----	-----
Cmdlet	Attach-CdaDatabase	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Detach-CdaDatabase	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Export-CdaAuditSessionRecording	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Get-CdaActiveDatabase	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Get-CdaAgent	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Get-CdaAuditEvent	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Get-CdaAuditRole	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Get-CdaAuditRoleAssignment	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Get-CdaAuditRoleRight	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Get-CdaAuditSession	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Get-CdaAuditSessionDataIntegrityStatus	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Get-CdaAuditSessionReviewer	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Get-CdaAuditStore	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Get-CdaAuditStoreRight	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Get-CdaCollector	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Get-CdaDatabase	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Get-CdaDetailedExecution	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Get-CdaInstallation	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Get-CdaInstallationRight	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Get-CdaManagementDatabase	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Get-CdaManagementDatabaseRight	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Get-CdaMonitoredExecution	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Get-CdaMonitoredFile	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Get-CdaQuery	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Get-CdaQueryRight	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Get-CdaUnixCommand	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Get-CdaUnixCommandTranscript	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Get-CdaUserEvent	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Get-CdaWindowsEvent	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	New-CdaAuditRole	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	New-CdaAuditRoleAssignment	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	New-CdaAuditStore	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	New-CdaDatabase	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	New-CdaSearchCriteria	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Publish-CdaInstallation	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Remove-CdaAgent	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Remove-CdaInstallation	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Remove-CdaAuditRoleAssignment	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Remove-CdaAuditRoleRight	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Remove-CdaCollector	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Remove-CdaDatabase	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Set-CdaActiveDatabase	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Set-CdaAuditRole	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Set-CdaAuditSession	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Set-CdaAuditSessionReviewer	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Set-CdaAuditStore	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Set-CdaConfiguration	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Set-CdaDatabase	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Set-CdaInstallation	3.5.2.570	Centrify.DirectAudit.PowerShell
Cmdlet	Set-CdaManagementDatabase	3.5.2.570	Centrify.DirectAudit.PowerShell

Managing Audit-Related Objects with Windows PowerShell Scripts

This chapter provides an overview of how you can use the cmdlets to access and manage audit information stored in Microsoft SQL Server databases and Active Directory using Windows PowerShell scripts. For more examples of how to perform common administrative and auditing analysis tasks using the cmdlets in PowerShell scripts, see the samples included with the software.

Using cmdlets to Manage Auditing

The Audit Module for PowerShell provides cmdlets that perform operations on objects that correspond to the core elements of Server Suite data. The core elements of Server Suite data for auditing are the following:

- Audited computers with the Server Suite auditing services
- Collectors that transfer audited activity from audited computers to the active audit store database
- Active and attached audit store databases
- Management database
- Audit installation
- User sessions
- Audit trail events
- Audit roles
- Audit role assignments

You can use the cmdlets to create, access, modify, and remove information associated with these core elements of Server Suite data for auditing. Most of the cmdlets perform one of the following basic operations:

- New-CdaXxx cmdlets create new Server Suite objects, such as a new audit role or a new audit store database.
- Get-CdaXxx cmdlets get the properties of a specified object.
- Set-CdaXxx cmdlets set or change the properties of a specified object.
- Remove-CdaXxx cmdlets delete a specified object.

In addition to these basic operations, there are cmdlets for attaching or detaching an audit store database, exporting session activity to a file, and for publishing installation information to Active Directory.

For reference information describing the use and parameters for each cmdlet, you can use the `get-help` function within the PowerShell console. For example, if you want to see a description and syntax summary for the `New-CdaAuditStore` cmdlet, type the following command in the PowerShell console:

```
get-help New-CdaAuditStore
```

If you want to see more detailed information about a cmdlet's parameters and code examples, you can use the `-detailed` or `-full` option. For example, type the following command in the PowerShell console:

```
get-help New-CdaAuditStore -detailed
```

Preparing the Environment to Run cmdlets

Because the Audit Module for PowerShell cmdlets run in the context of a domain account, you don't need to make an explicit connection to an Active Directory domain.

Setting the Preferred Domain Controller

If there is more than one domain controller in the current domain, you can use the `SetCdaConfiguration` cmdlet to specify the preferred domain controller server to which you want to connect. The following example illustrates how to connect to the preferred domain controller for the `finance.acme` domain:

```
PS C:\> Set-CdaConfiguration -DomainController "win-2012r2dc.finance.acme"
```

You can also use the `SetCdaConfiguration` cmdlet to connect to an auditing installation in another trusted forest. In this case, specify the domain controller that is in the other trusted forest.

Setting the Logging Level

You can use the `Set-CdaConfiguration` cmdlet to specify a logging level for running cmdlets. The following example illustrates how to use the `Set-CdaConfiguration` cmdlet to enable verbose logging:

```
PS C:\> Set-CdaConfiguration -LogLevel "verbose"
```

The default path to the log file is `C:\Program Files\Common Files\Centrify Shared\Logs\DirectAudit_*date-time*.log`.

Running cmdlets under Another Account

Some Audit Module for PowerShell cmdlets require permission to connect and update Microsoft SQL Server. If your login credentials do not have the required permissions, you can run cmdlets under another account. To run cmdlets as another user, you can use the standard PowerShell `Start-Process -Credential` to specify a different user name, then type the user's password when prompted, or you can right-click the Audit Module for PowerShell menu item, then select **Run As Administrator** to run the cmdlets as the local administrator.

Organizing cmdlet Operations in a Sequence

There is no fixed sequence in which cmdlets must be called. There is, however, a logical sequence to follow to make information available from one to another. For example, to get all of the audit roles in an installation, you might first want to identify the installation object you want to work with before you call the `Get-CdaAuditRole` cmdlet. To accomplish this, you could organize the calls in the following sequence:

```
$site = Get-CdaInstallation -Name "production"
Get-CdaAuditRole $site
```

Similarly, before converting an active database into an attached database, you might organize the calls to create a new audit store database, then set the new database to be the active audit store database:

```
$install = Get-CdaInstallation -Name "site1"

$auditStore = Get-CdaAuditStore -Installation $install -Name "auditstore1"

// Use New-CdaDatabase to create and attach the new database
$newDB = New-CdaDatabase -AuditStore $auditStore -Name "audit-us-Oct2014" -Server "sql_
server1.domain.com\da" -Database "audit-us-Oct2014"

// Set the newly created database as the active database for this audit store
Set-CdaActiveDatabase -AuditStore $auditStore -Database $newDB

// Create another new database
$newDB2 = New-CdaDatabase -AuditStore $auditStore -Name "audit-us-Nov2014" -Server "sql_
server1.domain.com\da" -Database "audit-us-Nov2014"

// Set the second database as active database
Set-CdaActiveDatabase -AuditStore $auditStore -Database $newDB2

// Detach the first database if it is no longer needed
Detach-CdaDatabase -Database $newDB
```

In most cases, you can determine from the parameters of a cmdlet whether you need to call another cmdlet first. For example, most `Set-Cda*xxx*` or `Remove-Cda*xxx*` cmdlets, you must call the corresponding `GetCda*xxx*` cmdlet

to obtain the object first. For example, to delete the "forensics" audit role from the "production" audit installation, you could call the cmdlets as follows:

```
Get-CdaAuditRole -Installation "production" -Name "forensics" | RemoveCdaAuditRole
```

In this example, the `Get-CdaAuditRole` cmdlet retrieves "forensics" from the specified installation and passes it to the `Remove-CdaAuditRole` cmdlet.

Checking for Valid Licenses

All of the cmdlets check for a valid license before performing the requested action. The license check succeeds only if one of the following conditions is true:

- There is at least one evaluation license that has not expired.
- There is at least one workstation license.
- There is at least one server license.

If the license check fails, the cmdlet displays an error and stops running. If the license check succeeds, the result is cached. The next time a cmdlet tries to access the same forest, it uses the cached result rather than performing the license check again. Note that the cache is only effective in one PowerShell console. If another PowerShell console runs a cmdlet accessing the same forest, the cmdlet in that console performs a separate license check.

Specifying Parameters using Different Formats

For certain types of parameters, you can specify a value using any one of several different supported formats. For example, you can specify a user principal for a `CdaAdPrincipal` object type by providing the information that identifies the user in any of the following formats:

- distinguished name (DN) for the user.
- security identifier for the user (SID).
- `sAMAccountName` attribute for the user in either the `sAMAccountName@domain` format or `domain\sAMAccountName` format.
- in a stored user object.

The following formats are all valid for specifying an Active Directory user principal:

```
New-CdaRoleAssignment -AuditRole $role -Assignee "cn=ben,cn=Users,dc=acme,dc=com"
```

```
New-CdaRoleAssignment -AuditRole $role -Assignee "S-1-5-21-12345678-98765432-500"
```

```
New-CdaRoleAssignment -AuditRole $role -Assignee "ben@acme.com"
```

```
New-CdaRoleAssignment -AuditRole $role -Assignee "acme\ben"
```

```
New-CdaRoleAssignment -AuditRole $role -Assignee $userObject
```

The following table lists the supported formats for each type of parameter.

Type	Supported parameter formats
CdaInstallation	You can specify an installation name as string, for example, "DefaultInstallation," or using a CdaInstallation object.
CdaAdPrincipal	<p>You can specify Active Directory users, groups, or computers using any of the following formats:</p> <ul style="list-style-type: none"> - Distinguished name string - SID string - <i>sAMAccountName@domain</i> - <i>domain\sAMAccountName</i> <p>You can specify Active Directory users, groups, or computers using a CdaAdPrincipal object.</p>
CdaAccessAccount	<p>You can specify a Windows account name or a SQL Server login account name and password, for example.</p> <p>For a Windows user account, all of the same formats listed for a CdaAdPrincipal object are supported.</p> <p>For SQL Server login accounts, the format is "sql:sql_name:sql_password". The password can be empty.</p>
CdaAuditScope	You can specify the audit scope using the Active Directory site name as a string, for example, "default-first-site" or by specifying a network subnet definition as a string, for example, "192.168.100.0/24".

If a parameter is not listed in the table, you must specify the object instance returned by a previously cmdlet. For example, you can use the Get-CdaAuditStore cmdlet to return an object instance of the audit store then use that object instance for parameters in other cmdlets that require it.

```
# Get the audit store object instance and store it in $cdaAuditStoreObject
```

```
$cdaAuditStoreObject = Get-CdaAuditStore -Installation "DefaultInstallation" -Name "Default-First-Site"
```

```
# Use the audit store object instance to specify a parameter value
```

```
Attach-CdaDatabase -AuditStore $cdaAuditStoreObject -Name "audit-store-db" -Server "win2012\instance1" -Database "audit-store-database"
```

Working with Sample Scripts

The Audit Module for PowerShell includes some sample scripts that you can use to do database rotation; rotating a database involves creating a new audit store database and making the new database the active database for the

installation. You can copy and modify the sample scripts to use the code directly in your environment or study the syntax used in the script to serve as an example for writing your own custom scripts.

- `db_rotation.ps1`

This PowerShell script demonstrates how to use the following cmdlets to do database rotation:

- `New-CdaDatabase`
- `Set-CdaDatabase`
- `Detach-CdaDatabase`

- `db_rotation_sql_script.ps1`

This PowerShell script demonstrates how to do database rotation using the SQL scripts and PowerShell cmdlets.

- `SetupDatabase.sql`, `SetupServer.sql`

You can modify the file path settings in these SQL scripts to create a new audit store database using the `New-CdaDatabase` cmdlet with the `DatabaseScriptFile` and `ServerScriptFile` parameters. This case is useful if you want to create a new audit store database with a customized database file folder path, database log file folder path, or database full-text catalog root path.

If you want to use these SQL scripts and you also want to customize the `StoredProcedureAccount`, `IsAwsRds`, and `IntegrityCheckEnabled` settings, you also need to customize the settings in the SQL scripts instead of using the `StoredProcedureAccount`, `IsAwsRds`, and `IntegrityCheckEnabled` parameters. These three parameters can't be used along with the `DatabasesScriptFile` and `ServerScriptFile` parameters.

To run the sample script:

1. Open the Audit Module for PowerShell.
2. Verify you have permission to execute scripts.

`Get-ExecutionPolicy`

In most cases, the permission to execute scripts is restricted. You can use the `SetExecutionPolicy` to allow execution. For example:

`Set-ExecutionPolicy Unrestricted`

For more information about execution policies and the options available, use the `gethelp` function.

3. Verify you are in the directory where the `db_rotation.ps1` script is located.
4. Execute the sample script.

Writing Your Own Scripts

You can combine Audit Module for PowerShell cmdlets with native Windows PowerShell cmdlets to perform many common administrative tasks. The following examples illustrate how you can combine the Audit Module for PowerShell cmdlets and native cmdlets to accomplish a simple but specific goal.

- [Exporting Specific Session Fields for a Report](#)
- [Checking the Status of Agents and Collectors](#)

Exporting Specific Session Fields For A Report

By default, the cmdlet for getting session information might return more information than you want for a simple report. If you want to narrow down the fields returned, you can use a native cmdlet, such as `Select-Object`, to specify the fields of interest, then another native cmdlet, such as `Out-File`, to export the results to a text file. For example, to limit the results to a few key fields, you might specify a command similar to this:

```
Get-CdaAuditSession -Installation "installation-name"
| Select-Object -Property User, Machine, StartTime, EndTime,
State, | Out-File c:\samplesession.txt
```

This example pipes the results from the Audit Module for PowerShell `GetCdaAuditSession` cmdlet to the `Select-Object` cmdlet, then uses another pipe to send the resulting output to a file.

Checking the status of agents and collectors

You can use Server Suite cmdlets to get status information for agents and collectors and combine those cmdlets with native or custom cmdlets to schedule checking for connectivity to run on a regular basis and to trigger an email notification if the status returned for the agent or collector in any interval is `Disconnected`. For example, to check for disconnected agents, you might specify a command similar to this:

```
Get-CdaAgent -i "*installation-name*" | where { $_.Status -eq "Disconnected" }
```

To check for agents that haven't connected to the collector since a specific time, you might specify a command similar to this:

```
Get-CdaAgent -i "*installation-name*" | where { $_.LastUpdateTime -lt ([DateTime]"12:00:00
AM, 12/29/2014") }
```

Similarly, you can use the `Get-CdaCollector` cmdlet to check for connectivity between a collector and the Microsoft SQL Server database you are using as the active audit store database.

```
Get-CdaCollector -i "*installation_name*" | where { $_.LastUpdateTime -lt "10:00:00 AM,
12/17/2014" }
```

You can include these cmdlets in scripts that run automatically using a task scheduler to check for connectivity issues at the interval you specify, such as once a day or once a week, and to send the results to specified channels, such as an email message or SNMP trap, using a cmdlet such as `Send-MailMessage`.

Recommendations for Writing Custom Scripts

Most cmdlets and scripts return information efficiently without any special handling or any noticeable effect on performance. If you plan to write custom scripts that could potentially return large data sets, however, you should consider ways to improve performance. For example, if you are writing a script that exports a large number of sessions or reports on activity for a large audit installation, you might want to use the following recommendations as guidelines.

- When testing the performance of the script, use the standard `Measure-Command` cmdlet to accurately measure cmdlet and script performance.

The `Measure-Command` cmdlet ignores the time it takes to print all of the results returned to the PowerShell console. In many cases, the execution of a query or script is efficient, but rendering the results in the PowerShell console might make the cmdlet or script performance seem unacceptable.

- Avoid using the PowerShell pipeline if your cmdlet or script returns large data collections.

For example, you might use `foreach` in a script instead of using the pipeline to improve performance.

Use syntax similar to this:

```
foreach ($cmd in Get-CdaUnixCommand -Session $s) { *action_on_each_cmd* }
```

Instead of:

```
Get-CdaUnixCommand -Session $s | *action_on_each_cmd*
```

- Cache the data, if possible, by writing the results to a file.

For example, use syntax similar to this:

```
$cmds = Get-CdaUnixCommand -Session $s
```

```
Out-File -InputObject $cmds -FilePath file
```

Instead of:

```
Get-CdaUnixCommand -Session $s | Out-File -FilePath file
```

- Use `Export-Csv` instead of `Out-File` if possible. The `Export-Csv` cmdlet writes results to a file faster than the `Out-File` cmdlet.
- If you are writing a script that generates a very large data set—for example, reporting information for a global audit installation—you might want to use the native .NET `FileStream` function. The `FileStream` function is the fastest way to write content to a file.

For example, you might use a code snippet like this:

```
$fs = New-Object IO.FileStream <file>, 'Append','write','Read'
```

```
$fw = New-Object System.IO.StreamWriter $fs
```

```
$cmds = Get-CdaUnixCommand -Session $s
```

```
foreach ($cmd in $cmds) {$fw.WriteLine("{0} {1} {2}",
```

```
$cmd.Sequence, $cmd.Time, $cmd.Command)}
```

```
$fw.Close()
```

```
$fs.Dispose() ``
```

Executing Custom Scripts

In most cases, the permission to execute scripts is restricted. You can use the native PowerShell cmdlet `Get-ExecutionPolicy` to check whether you have permission to execute scripts using your current account credentials and the native `Set-ExecutionPolicy` cmdlet to specify an execution policy.

To check and update the execution policy for scripts

1. Open the Audit Module for PowerShell.
2. Verify you have permission to execute scripts.
`Get-ExecutionPolicy`
3. Run the `SetExecutionPolicy` cmdlet to change the execution policy. For example:
`Set-ExecutionPolicy Unrestricted`

Auditing-Related Objects and Properties

For more information about execution policies and the options available, use the `get-help` function.

4. Verify you are in the directory where your scripts are located.
5. Execute the sample script.

Getting Information about the cmdlet Available

You can use the `get-help` command with different options to get summary or detailed information about the cmdlets available in the Audit Module for PowerShell or about the specific cmdlets you want to use. For example, you can use `get-help` with the full command-line option to see complete reference information for a specified cmdlet or `get-help -example` to display only the examples for a specified cmdlet.

To see the current list of cmdlets available open the Audit Module for PowerShell, then run the following command:

```
get-help *-cda*
```

This command displays a summary of the Audit Module for PowerShell cmdlets similar to the following partial list of commands:

Name	Category	Module	Synopsis
-----	-----	-----	-----
Attach-CmdDatabase	Cmdlet	Centrify.Directory.P...	Attach an existing audit store database to an audit store.
Detach-CmdDatabase	Cmdlet	Centrify.Directory.P...	Detaches an attached audit store database from an audit store.
Export-CmdAuditSessionRecording	Cmdlet	Centrify.Directory.P...	Exports the video capture recording of an audited session.
Get-CmdAuditDatabase	Cmdlet	Centrify.Directory.P...	Gets the current active database for the specified audit store.
Get-CmdAgent	Cmdlet	Centrify.Directory.P...	Gets one or more audited computers.
Get-CmdAuditEvent	Cmdlet	Centrify.Directory.P...	Gets audit trail events.
Get-CmdAuditRole	Cmdlet	Centrify.Directory.P...	Gets an existing audit role.
Get-CmdAuditRoleAssignment	Cmdlet	Centrify.Directory.P...	Gets an existing audit role assignment.
Get-CmdAuditRoleRight	Cmdlet	Centrify.Directory.P...	Gets the audit role rights granted to trustees in a DirectoryAudit installation.
Get-CmdAuditSession	Cmdlet	Centrify.Directory.P...	Gets audit sessions matching the criteria you specify.
Get-CmdAuditSessionDataIntegr...	Cmdlet	Centrify.Directory.P...	Gets the data integrity status for audited sessions in order to detect possible data tampering.
Get-CmdAuditSessionReviewer	Cmdlet	Centrify.Directory.P...	Gets the Active Directory users and groups who have been designated as session reviewers.
Get-CmdAuditStore	Cmdlet	Centrify.Directory.P...	Gets the audit stores for a specified audit installation.
Get-CmdAuditStoreRight	Cmdlet	Centrify.Directory.P...	Gets the audit store rights granted to trustees in a DirectoryAudit installation.
Get-CmdCollection	Cmdlet	Centrify.Directory.P...	Gets the collections for a specified audit installation.
Get-CmdDatabase	Cmdlet	Centrify.Directory.P...	Gets a specified audit store database.
Get-CmdDetailedSession	Cmdlet	Centrify.Directory.P...	Gets detailed command execution details, if advanced monitoring is enabled.
Get-CmdInstallation	Cmdlet	Centrify.Directory.P...	Gets the audit installation for the current Active Directory forest.
Get-CmdInstallationRight	Cmdlet	Centrify.Directory.P...	Gets the installation rights granted to trustees in a DirectoryAudit installation.
Get-CmdManagementDatabase	Cmdlet	Centrify.Directory.P...	Gets the management database for an installation.
Get-CmdManagementDatabaseRight	Cmdlet	Centrify.Directory.P...	Gets the management database rights granted to trustees in a DirectoryAudit installation.
Get-CmdMonitorExecution	Cmdlet	Centrify.Directory.P...	Gets monitored command execution details, if advanced monitoring is enabled.
Get-CmdMonitorFile	Cmdlet	Centrify.Directory.P...	Gets monitored file activity, if advanced monitoring is enabled.
Get-CmdQuery	Cmdlet	Centrify.Directory.P...	Gets the queries in a DirectoryAudit installation.
Get-CmdQueryRight	Cmdlet	Centrify.Directory.P...	Gets the query rights granted to trustees in a DirectoryAudit installation.
Get-CmdTaskCommand	Cmdlet	Centrify.Directory.P...	Gets the task commands associated during an audited session.
Get-CmdTaskCommandDescription	Cmdlet	Centrify.Directory.P...	Gets the task command descriptions that were displayed during an audited session.
Get-CmdEventExec	Cmdlet	Centrify.Directory.P...	Gets user activity events.
Get-CmdEventExec	Cmdlet	Centrify.Directory.P...	Gets the Windows events captured for an audited session.
New-CmdAuditRole	Cmdlet	Centrify.Directory.P...	Creates a new audit role.
New-CmdAuditRoleAssignment	Cmdlet	Centrify.Directory.P...	Creates a new audit role assignment.
New-CmdAuditStore	Cmdlet	Centrify.Directory.P...	Creates a new audit store.
New-CmdDatabase	Cmdlet	Centrify.Directory.P...	Creates a new audit store database.
New-CmdSessionCriteria	Cmdlet	Centrify.Directory.P...	Creates a new session criteria object.
Publish-CmdInstallation	Cmdlet	Centrify.Directory.P...	Publishes or synchronizes installation information in Active Directory to other auditing components on...
Remove-CmdAgent	Cmdlet	Centrify.Directory.P...	Deletes an audited computer database record from the audit installation.
Remove-CmdAuditRole	Cmdlet	Centrify.Directory.P...	Deletes an existing audit role.
Remove-CmdAuditRoleAssignment	Cmdlet	Centrify.Directory.P...	Deletes an existing audit role assignment.
Remove-CmdAuditSession	Cmdlet	Centrify.Directory.P...	Deletes an existing session.
Remove-CmdCollection	Cmdlet	Centrify.Directory.P...	Deletes an existing collection.
Remove-CmdDatabase	Cmdlet	Centrify.Directory.P...	Deletes the specified database file.
Remove-CmdAuditDatabase	Cmdlet	Centrify.Directory.P...	Updates the database to use as the active audit store database.
Set-CmdAuditRole	Cmdlet	Centrify.Directory.P...	Updates properties for an existing audit role.
Set-CmdAuditSession	Cmdlet	Centrify.Directory.P...	Updates properties for an existing audit session.
Set-CmdAuditSessionReviewer	Cmdlet	Centrify.Directory.P...	Updates the list of users and groups who have been designated session reviewers.
Set-CmdAuditStore	Cmdlet	Centrify.Directory.P...	Updates properties for an existing audit store.
Set-CmdConfiguration	Cmdlet	Centrify.Directory.P...	Defines the preferred domain controller and logging settings to use for the DirectoryAudit PowerShell se...
Set-CmdDatabase	Cmdlet	Centrify.Directory.P...	Updates properties for an existing audit store database.
Set-CmdInstallation	Cmdlet	Centrify.Directory.P...	Updates properties for an existing audit installation.
Set-CmdManagementDatabase	Cmdlet	Centrify.Directory.P...	Updates properties for an existing audit management database.

Auditing-Related Objects and Properties

Most Audit Module for PowerShell cmdlets return object instances either directly or as properties of other objects.

This section provides an alphabetical listing of the objects and the properties of each object defined in the Audit Module for PowerShell. Note that not all properties are available as parameters in the PowerShell cmdlets.

CdaAccessAccount

Represents a Windows user or SQL Server login account with access to auditing components. The following properties are defined for this object.

Property	Type	Description
AccountName	String	Name of the Windows user or SQL Server login account.
Type	Enum	Account type. The valid values are: 1 if the account is a Windows account that uses Windows authentication. 2 if the account is a Microsoft SQL Server login account that uses SQL Server authentication.

CdaAdPrincipal

Represents an Active Directory principal. The principal can be an Active Directory user, group, or computer account. You can use the Class property to identify the type of principal. Only the account name for the principal is stored in the database. The following properties are defined for this object.

Property	Type	Description
Class	String	Principal type of the Active Directory object.
DistinguishedName	String	Distinguished name of the Active Directory object.
Domain	String	Domain name for the Active Directory principal.
GUID	Guid	Globally unique identifier (GUID) for the Active Directory object.
Name	String	Name of the Active Directory object.
SamAccountName	String	The sAMAccountName attribute for the Active Directory principal.
SID	Security identifier	The security identifier (SID) for the Active Directory principal.

CdaAgent

Represents an audited computer where the auditing service is enabled. The following properties are defined for this object.

Property	Type	Description
AuditedSystemType	Enum	Specifies whether the audited systems are system-based ("SystemBased") or vault-based ("VaultBased"). This parameter is optional. If you do not specify this parameter, the results include a list of both types of audited systems. System-based describes a Windows or UNIX computer that is running an agent. You can access these systems either directly or from the Privileged Access Service Admin Portal. Vault-based describes a Windows or UNIX computer or a network device that is not running an agent (agentless). You can access these systems from the Privileged Access Service Admin Portal. Note: Some properties display different values for vault-based systems: * Version: this property is empty because vault-based systems are agentless * Status: this property displays as none * StartupTime: this property displays as a default date-time value of "1/1/0001 12:00:00 AM" * UpTime: this property displays as 00:00:00
LastUpdateTime	DateTime	Time at which the auditing service agent was last updated.
MachineAddress	String	IP address of the computer hosting the auditing service.
MachineName	String	Name of the computer hosting the auditing service.
MachineSid	String	Security identifier string for the computer hosting the auditing service.
StartupTime	DateTime	Time at which the auditing service agent first started.
Status	Enum	Status of the auditing service. The valid values are: Connected Disconnected
Type	Enum	Type of operating system running on the computer hosting the auditing service. The valid values are: 0 – Unknown 1 – UNIX 2 – Windows
UpTime	TimeSpan	Total time the auditing service agent was connected time from the startup time to the last update time.
Version	String	Auditing service agent version number.

CdaAuditEvent

Represents an audit trail event. The following properties are defined for this object.

Property	Type	Description
Description	String	Description of the audit trail event.

Property	Type	Description
EventId	Integer	Event identifier for the audit trail event. Event instances that share the same event type will also have the same EventId.
EventName	String	Name of the audit trail event.
Machine	String	Computer name associated with the audit trail event.
Parameters	String Array	List of parameters for this audit trail event.
Result	String	Result returned by the audit trail event.
SessionId	String	Identifying string for the session associated with the audit trail event, if there is one.
SessionUri	String	The uniform resource identifier (URI) for the session associated with the audit trail event, if there is one.
Time	DateTime	Date and time the audit trail event occurred.
Uniqueld	Long	Unique identifier for the event instance.
User	String	User name associated with the audit trail event.

CdaAuditRole

Represents an audit role. The following properties are defined for this object.

Property	Type	Description
Definition	String	String that defines the criteria used in the audit role to specify the sessions to include.
Description	String	Description of the audit role.
Name	String	Name of the audit role.
Privilege	Enum array	User privileges for the audited sessions that match the criteria specified for this audit role.

CdaAuditRoleAssignment

Represents an audit role assignment. The following properties are defined for this object.

Property	Type	Description
Assignee	CdaAdPrincipal	User, group, or computer account assigned to the audit role.
AuditRole	CdaAuditRole	Name of the audit role being assigned.

CdaAuditRoleRight

This object represents the rights granted to a trustee on the audit role. The following properties are defined for this object.

Property	Type	Description
AuditRole	CdaAuditRole	The audit role
Trustee	String	Trustee name in format <DOMAIN>\<User account name> Note: The consistent name format is shown in the Audit Manager console. For an orphan trustee, it shows SID in SDDL format.
TrusteeType	String	Indicate the type of the trustee,for example Active Directory User or Group
Rights	string[]	The collection of rights granted to the trustee on the audit role. Possible rights: Full Control Change Permissions Change Role Membership Change Role Definition

CdaAuditScope

Represents the audit scope for an audit store. The following properties are defined for this object.

Property	Type	Description
Definition	String	String that defines the audit scope. If the audit scope is an Active Directory site, this property is the site name. If the scope is a subnet, this property is the IP address and subnet mask.
Type	Enum	The type of audit scope. The valid values are: 1 if the audit scope is an Active Directory site. 2 if the audit scope is a network subnet segment.

CdaAuditSession

Represents an audited user session. The following properties are defined for this object.

Property	Type	Description
AuditStore	String	Name of the audit store.

Property	Type	Description
ClientAddress	String	Client IP address.
ClientName	String	Client name.
Comment	String array	Comments that have been added by reviewers to the session.
EndTime	DateTime	Session end time.
IsADUser	Boolean	Indicates whether the user is an Active Directory user.
Machine	String	Host name of the computer where the session ran.
MachineAddress	String	Computer IP address of the computer where the session ran.
MachinePrincipal	String	Computer principal name of the computer where the session ran.
ReviewedBy	ADUser	Name of the user who last updated the review status for the session.
ReviewStatus	Enum	Session review status. The valid values are: 0 for None 1 for ToBeReviewed 2 for Reviewed 3 for PendingForAction 4 for KeepForever 5 for ToBeDeleted
ReviewTime	DateTime	Date and time of the last review status update for the session.
SessionID	String	Globally unique identifier (GUID) for the object.
Size	Integer	Total size of the session in KB.
StartTime	DateTime	Session start time.
State	Enum	Status of the session. The valid values are: -1 for Unknown 0 for InProgress 1 for Terminated 2 for Disconnected 3 for Completed
Tags	String	The tags associated with the audit session
Type	Enum	Session type. The valid values are: 1 if the session is a Windows session 2 if the session is a UNIX session
Uri	String	The uniform resource identifier (URI) for replaying the session in the session player.
User	String	User name associated with the session.
UserDisplayName	String	User display name associated with the session.
Zone	String	Server Suite zone name.

CdaAuditSessionTag

Represents the keyword tag that is associated with an audited session. The following properties are defined for this object.

Property	Type	Description
AuditStoreDatabaseId	int	The ID of the audit store database
Id	long	The ID of the tag
Mode	string	The mode of the tag, which indicates if the session was tagged by an automatic process or manually tagged. The possible values are: Manual Automatic
ReplayTimestamp	DateTime	The session replay time of the tag in the audit session
Session	CdaAuditSession	The audit session(s) that the tag is associated with
Tag	string	The tag
Tagger	string	The user name of the auditor who tagged the audit session
TagTimestamp	DateTime	The timestamp when the audit session was tagged

CdaAuditSessionDataIntegrityStatus

If you've enabled the audit store database for data integrity checking, this object refers to the session's data integrity status. Data integrity checking provides the ability to detect if auditing data has been tampered. The following properties are defined for this object.

Property	Type	Description
Session	CdaAuditSession	The audited user session
Status	Integer	Unknown = -1 Passed = 0 Not Enabled = 1 Session Not Found = 2 Missing Final Thumbprint = 3 Invalid Final Thumbprint = 4 Missing Thumbprint = 5 Invalid Thumbprint = 6 Failed = 7
StatusMessage	String	The friendly display message of the status
Source	String	The source name of the audit session data (which contains the database table name and record Id)

CdaAuditStore

Represents an audit store. The following properties are defined for this object.

Property	Type	Description
Affinity	AffinityType enum	The type agents that the audit store serves, either Windows, UNIX, or both. The possible values are: WindowsAndUnix - 0 Windows - 1 Unix - 2
Name	String	Name of the audit store.
Scopes	CdaAuditScope[]	Audit store scopes.
TrustedAgentEnabled	Boolean	Whether the trusted agent filter is enabled or not.
TrustedAgents	CdaComputer[]	Trusted agent computers.
TrustedCollectorEnabled	Boolean	Whether the trusted collector filter is enabled or not.
TrustedCollectors	CdaComputer[]	Trusted collector service computers.

CdaAuditStoreRight

This object represents the rights granted to a trustee on the audit store. The following properties are defined for this object.

Property	Type	Description
AuditStore	CdaAuditStore	The audit store
Trustee	String	Trustee name in format <DOMAIN>\<User account name> Note: The consistent name format is shown in the Audit Manager console. For an orphan trustee, it shows SID in SDDL format.
TrusteeType	String	Indicate the type of the trustee,for example Active Directory User or Group
Rights	string[]	The collection of rights granted to the trustee on the auditstore. Possible rights: Full Control Change Permissions Modify Name Manage Scopes Manage SQL Logins Manage Collectors Manage Audited Systems Manage Databases Manage Database Trace

CdaCollector

Represents a collector service computer. The following properties are defined for this object.

Property	Type	Description
AuditStoreDatabase	String	The audit store database the collector connects to.

Property	Type	Description
LastUpdateTime	DateTime	The date and time at which the collector received the last update.
MachineAddress	String	IP address of the computer hosting the collector service.
MachineName	String	Name of the computer hosting the collector service.
PortNumber	Integer	Collector connection port number.
Sid	String	Security identifier string for the computer hosting the collector service.
StartupTime	DateTime	The date and time at which the collector first connected to the audit store database.
Status	Enum	Status of the collector service. The valid values are: Connected Disconnected
UpTime	TimeSpan	Total time the collector was connected time from startup to the last update time.
Version	String	Collector service version number.

CdaDatabase

Represents an audit store database. The following properties are defined for this object.

Property	Type	Description
ActiveEndTime	DateTime	The date and time at which the database stopped being the active database.
ActiveStartTime	DateTime	The date and time this database became the active database.
AllowedCollectors	CdaAccessAccount[]	Allowed collector accounts.
AllowedManagementServers	CdaAccessAccount[]	Allowed management database accounts.
AuditStore	CdaAuditStore	The audit store object instance for the database.
CollectorCount	Integer	Number of collectors connected to the database.
Database	String	Microsoft SQL Server database name for the audit store database.
DiskUsage	Integer	Database file size, in 8KB pages.

Property	Type	Description
IsActive	Boolean	Specifies whether this is the active database for the audit store.
Name	String	Display name of the audit store database.
RecordCount	Integer	Number of session records in the database.
Server	String	Microsoft SQL Server host name and instance.
Status	Enum	Database status. The valid values are: Connected Disconnected
Version	String	Database version number.

CdaDetailedExecution

Represents detailed command execution details, if advanced monitoring is enabled. The following properties are defined for this object.

Property	Type	Description
User	String	The user name associated with the event
Machine	String	The computer name associated with the event
Time	DateTime	The date and time when the command was executed
EnteredCommand	String	The name of the entered command
ExecutedCommand	String	The name of the executed command
CommandArguments	String	The command arguments
RunAsUser	String	The run as user name
AccessStatus	String	The access status: Succeeded or Failed
AccessStatusDetails	String	The detailed message about the status
CurrentDirectory	String	The current directory of the command execution
ProcessId	String	The process ID of the command execution
ParentProcessId	String	The process ID of the parent process of the command execution

CdaInstallation

Represents an audit installation. The installation defines the scope of the auditing infrastructure and audit data available for review and play back. The following properties are defined for this object.

Property	Type	Description
DisableSelfDelete	Boolean	Indicates whether users can delete their own sessions. This installation-wide option takes precedence over the permissions granted to a user account. If you set this option to be True, users cannot delete their own sessions regardless of the rights granted to their audit roles.
DisableSelfReview	Boolean	Indicates whether users can update the review status or the comments on their own sessions. This installation-wide option takes precedence over the permissions granted to a user account. If you set this option to be True, users cannot update the review status or add comments for their own sessions regardless of the rights granted to their audit roles.
EnableVideoCapture	Boolean	Indicates whether the video capture auditing of user activity is enabled or not.
ManagementDatabase	CdaManagementDatabase	The default connected management database for the installation.
Name	String	Name of the installation.
NotificationImage	String	Name of the notification banner image file in base64 string format.
NotificationMessage	String	Name of the file containing the notification message text.
PublishLocations	String Array	One or more Active Directory locations where the installation service connection point is published.

CdaInstallationRight

This object represents the rights granted to a trustee on the DA Installation. The following properties are defined for this object.

Property	Type	Description
Trustee	String	Trustee name in format <DOMAIN>\<User account name> Note: The consistent name format is shown in the Audit Manager console. For an orphan trustee, it shows SID in SDDL format.
TrusteeType	String	Indicate the type of the trustee,for example Active Directory User or Group
Rights	string[]	The collection of rights granted to the trustee on the DA Installation. Possible rights: Full Control Change Permissions Modify Name Manage Management Database List Manage Audit Store List Manage Collectors Manage Audited Systems Manage Audit Role Manage Queries Manage Publications Manage Licenses Manage Notification Manage Audit Option View

CdaManagementDatabase

Represents an audit management database. The following properties are defined for this object.

Property	Type	Description
AllowedIncomingUsers	CdaUser[]	Allowed incoming users of the management database.
Database	String	Microsoft SQL Server database name for the management database.
Name	String	Display name of the management database.
OutgoingAccount	CdaAccessAccount	Outgoing account of the management database.
Scope	CdaAuditScope[]	Audit store scopes defined for the management database.
Server	String	Microsoft SQL Server host name and instance name.
Status	Enum	Status of the management database. The valid values are: Connected Disconnected

CdaManagementDatabaseRight

This object represents the rights granted to a trustee on the management database. The following properties are defined for this object.

Property	Type	Description
ManagementDatabase	CdaManagementDatabase	The management database

Property	Type	Description
Trustee	String	Trustee name in format <DOMAIN>\<User account name> Note: The consistent name format is shown in the Audit Manager console. For an orphan trustee, it shows SID in SDDL format.
TrusteeType	String	Indicate the type of the trustee,for example Active Directory User or Group
Rights	string[]	The collection of rights granted to the trustee on the management database. Possible rights: Full Control Change Permissions Modify Name Manage Scopes Remove Database Manage SQL Logins Manage Database Trace

CdaMonitoredExecution

Represents monitored command execution details, if advanced monitoring is enabled. The following properties are defined for this object.

Property	Type	Description
User	String	The user name associated with the event
Machine	String	The computer name associated with the event
Time	DateTime	The date and time when the command was executed
Command	String	The name of the executed command
CommandArguments	String	The command arguments
RunAsUser	String	The run as user name
AccessStatus	String	The access status: Succeeded or Failed
AccessStatusDetails	String	The detailed message about the status
CurrentDirectory	String	The current directory of the command execution
ProcessId	String	The process ID of the command execution
ParentProcessId	String	The process ID of the parent process of the command execution

CdaMonitoredFile

Represents monitored file details, if advanced monitoring is enabled. The following properties are defined for this object.

Property	Type	Description
User	String	The user name associated with the event
Machine	String	The computer name associated with the event
Time	DateTime	The date and time when the command was executed
FileName	String	The filename of the file being accessed
Command	String	The name of the executed command
RunAsUser	String	The run as user name
SystemCallName	String	The name of the system call
AccessType	String	The type of the file access: Write or ChangeAttribute
AccessStatus	String	The access status: Succeeded or Failed
AccessStatusDetails	String	The detailed message about the status
CurrentDirectory	String	The current directory of the command execution
ProcessId	String	The process ID of the command execution
ParentProcessId	String	The process ID of the parent process of the command execution

CdaQuery

This object represents a query. The following properties are defined for this object.

Property	Type	Description
Name	String	The query name
Description	String	The query description
IsPredefined	boolean	Whether this query is predefined or not

CdaQueryRight

This object represents the rights granted to a trustee on the query. The following properties are defined for this object.

Property	Type	Description
Query	CdaQuery	The query
Trustee	String	Trustee name in format <DOMAIN>\<User account name> Note: The consistent name format is shown in the Audit Manager console. For an orphan trustee, it shows SID in SDDL format.
TrusteeType	String	Indicate the type of the trustee,for example Active Directory User or Group
Rights	string[]	The collection of rights granted to the trustee on the query. Here are the possible rights: Full Control Change Permissions Read Delete Modify

CdaSearchCriteria

Represents a search criteria object that defines the filters to use to find sessions that can be passed to other cmdlets. For example, you can create a search criteria object to define the sessions that are applicable for a given audit role. The following properties are defined for this object.

Property	Type	Description
Application	String array	Filter sessions by using the Windows application name used.
AuditStore	String	Filter sessions by using the name of the audit store.
ClientName	String	Filter sessions by using the client name of the session.
Comment	String array	Filter sessions by using the comments that have been added by reviewers to the session.
Group	String array	Filter sessions by using the session owner's Active Directory security group.
Installation	String	Filter sessions by using the name of the audit installation.
Machine	String	Filter sessions by using the host name of the computer where the session ran.
ReviewStatus	Enum	Filter sessions by using the session review status. The valid values are: 0 for None 1 for ToBeReviewed 2 for Reviewed 3 for PendingForAction 4 for KeepForever 5 for ToBeDeleted

Property	Type	Description
State	Enum	Filter sessions by using the status of the session. The valid values are: 0 for InProgress 1 for Terminated 2 for Disconnected 3 for Completed
TimeAfter	DateTime	Filter sessions that ran after a specific date and time.
TimeBefore	DateTime	Filter sessions that ran before a specific date and time.
TimeBetween	DateTime	Filter sessions that ran between a start time and an end time.
Type	Enum	Filter sessions by using the session type. The valid values are: 1 if the session is a Windows session 2 if the session is a UNIX session
UnixCommand	String array	Filter sessions by using the UNIX command line input and output.
UnixCommandName	String array	Filter sessions by the UNIX command name only.
UnixCommandTimeAfter	DateTime	Filter sessions that ran after a specific date and time based on the UNIX command input time.
UnixCommandTimeBefore	DateTime	Filter sessions that ran before a specific date and time based on the UNIX command input time.
UnixCommandTimeBetween	DateTime	Filter sessions that ran between a start and end time based on the UNIX command input time.
UnixOutput	Text	Filter sessions by using the UNIX terminal output text captured in the session.
User	String	Filter sessions by using the user name associated with the session.

CdaUnixCommand

Represents an indexed UNIX command captured in an audited session. The following properties are defined for this object.

Property	Type	Description
Command	String	Text of the UNIX command line that was executed.
Sequence	Integer	Sequence number that identifies where in the indexed list of events this event occurs.

Property	Type	Description
Session	CdaAuditSession	The session object.
Time	DateTime	Date and time when the command was executed.

CdaUnixCommandTranscript

Represents the UNIX command input and output captured in an audited session. The following properties are defined for this object.

Property	Type	Description
EndTime	DateTime	The time at which the capture of this command ended.
LineNumber	Integer	The line number at which the text displayed in the terminal.
Role	String	The DirectAuthorize role assigned to this command.
Session	CdaAuditSession	The session object.
StartTime	DateTime	The time at which the capture of this command started.
Text	String	The text displayed in the terminal.
Ticket	String	The trouble ticket assigned to this command.
Type	Enum	Indicates whether the captured text was input or output.

CdaUserEvent

Represents a user event. The following properties are defined for this object.

Property	Type	Description
User	String	User name associated with the user event.
Machine	String	Computer name associated with the audit trail event.
Time	DateTime	The date and time when the command was executed.
Activity	String	A brief description of the user event.

CdaWindowsEvent

Represents an indexed Windows event captured in an audited session. The following properties are defined for this object.

Property	Type	Description
Application	String	Application name associated with the event.
Desktop	String	Desktop name associated with the event if the event occurred when using a desktop access right.
IsAudited	Boolean	Indicates whether this event occurred when using an audited role with a desktop right.
Sequence	Integer	Sequence number that identifies where in the indexed list of events this event occurs.
Time	DateTime	Date and time when the event occurred.
Title	String	Windows title bar text for the application when the event occurred.
Type	Enum	Type of event. The most common event types indicate when a new window or a new application starts or when the title of an existing windows changes.