# Centrify Server Suite 2017

## *Auditing and Analysis Scripting Guide*

February 2017

Centrify Corporation

## Legal notice

This document and the software described in this document are furnished under and are subject to the terms of a license agreement or a non-disclosure agreement. Except as expressly set forth in such license agreement or non-disclosure agreement, Centrify Corporation provides this document and the software described in this document “as is” without warranty of any kind, either express or implied, including, but not limited to, the implied warranties of merchantability or fitness for a particular purpose. Some states do not allow disclaimers of express or implied warranties in certain transactions; therefore, this statement may not apply to you.

This document and the software described in this document may not be lent, sold, or given away without the prior written permission of Centrify Corporation, except as otherwise permitted by law. Except as expressly set forth in such license agreement or non-disclosure agreement, no part of this document or the software described in this document may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, or otherwise, without the prior written consent of Centrify Corporation. Some companies, names, and data in this document are used for illustration purposes and may not represent real companies, individuals, or data.

This document could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein. These changes may be incorporated in new editions of this document. Centrify Corporation may make improvements in or changes to the software described in this document at any time.



U.S. Government Restricted Rights: If the software and documentation are being acquired by or on behalf of the U.S. Government or by a U.S. Government prime contractor or subcontractor (at any tier), in accordance with 48 C.F.R. 227.7202-4 (for Department of Defense (DOD) acquisitions) and 48 C.F.R. 2.101 and 12.212 (for non-DOD acquisitions), the government’s rights in the software and documentation, including its rights to use, modify, reproduce, release, perform, display or disclose the software or documentation, will be subject in all respects to the commercial license rights and restrictions provided in the license agreement.

Centrify, DirectControl, DirectAuthorize, DirectAudit, DirectSecure, DirectControl Express, Centrify User Suite, and Centrify Server Suite are registered trademarks and Centrify for Mobile, Centrify for SaaS, Centrify for Mac, DirectManage, Centrify Express, DirectManage Express, Centrify Identity Platform, Centrify Identity Service, and Centrify Privilege Service are trademarks of Centrify Corporation in the United States and other countries. Microsoft, Active Directory, Windows, and Windows Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and other countries.

Centrify software is protected by U.S. Patents 7,591,005; 8,024,360; 8,321,523; 9,015,103 B2; 9,112,846; 9,197,670; and 9,378,391.

The names of any other companies and products mentioned in this document may be the trademarks or registered trademarks of their respective owners. Unless otherwise noted, all of the names used as examples of companies, organizations, domain names, people and events herein are fictitious. No association with any real company, organization, domain name, person, or event is intended or should be inferred.

# Contents

# About this guide

The *Auditing and Analysis Scripting Guide* describes the Centrify Audit Module for Windows 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

The *Auditing and Analysis Scripting Guide* provides information for auditing infrastructure administrators who want to use PowerShell scripts to manage auditing-related features and components of Centrify 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 Centrify architecture, terms, and concepts, and know how to perform administrative tasks for Centrify DirectManage Audit 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.

The guide provides the following information:

- Chapter 3, "Developing scripts for administrative tasks," provides an introduction to using Windows PowerShell to perform auditing-related administrative activity.
- Chapter 4, "Installing the audit module for PowerShell," describes how to download and install the module as a separate package.
- Chapter 5, "Managing audit-related objects with Windows PowerShell scripts," describes how to use the cmdlets to connect to Active Directory and perform access control and privilege management tasks.
- Chapter 6, "Auditing-related objects and properties," lists the objects defined by the Centrify DirectManage PowerShell module, and the properties of each object.

# Compatibility and limitations of this guide

The information in this guide is intended for use with Centrify 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 Centrify DirectManage Access 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 Centrify website or Centrify Support to find out if another version of this guide is available.

# Conventions used in this guide

The following conventions are used in this guide:

- `Fixed-width` font is used for sample code, program names, program output, file names, and commands that you type at the command line. When *`italicized`*, the fixed-width font is used to indicate variables.
- **Bold** text is used to indicate commands, buttons, or user interface text.
- *Italics* are used for book titles and to emphasize specific words or terms

# Finding information about Centrify products

Centrify Server Suite includes extensive documentation targeted for specific audiences, functional roles, or topics of interest. However, most of the information in the documentation set is intended for administrators, application developers, or security architects after you have purchased the software or licensed specific features. If you want to learn more about Centrify and Centrify products and features, start by visiting the Centrify website. From the Centrify website, you can download data sheets and evaluation software, view video demonstrations and technical presentations about Centrify products, and get the latest news about upcoming events and webinars.

# Contacting Centrify

You can contact Centrify by visiting our website, www.centrify.com. On the website, you can find information about Centrify office locations worldwide, email and phone numbers for contacting Centrify sales, and links for following Centrify on social media. If you have questions or comments, we look forward to hearing from you.

# Getting additional support

If you have a Centrify account, click Support on the Centrify website to log on and access the Centrify Customer Support Portal. From the support portal, you can search knowledge base articles, open and view support cases, connect with other Centrify users on customer forums, and access additional resources—such as online training, how-to videos, and diagnostic tools.

# Developing scripts for administrative tasks

The Centrify Audit Module for Windows PowerShell consists of the following:

- 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 Centrify Audit Module for Windows PowerShell to develop your own custom scripts that access, create, or modify Centrify 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 Centrify-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 Centrify module enable you to access, create, modify, and remove information about Centrify 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 Centrify-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 *Centrify Database Management Guide*.

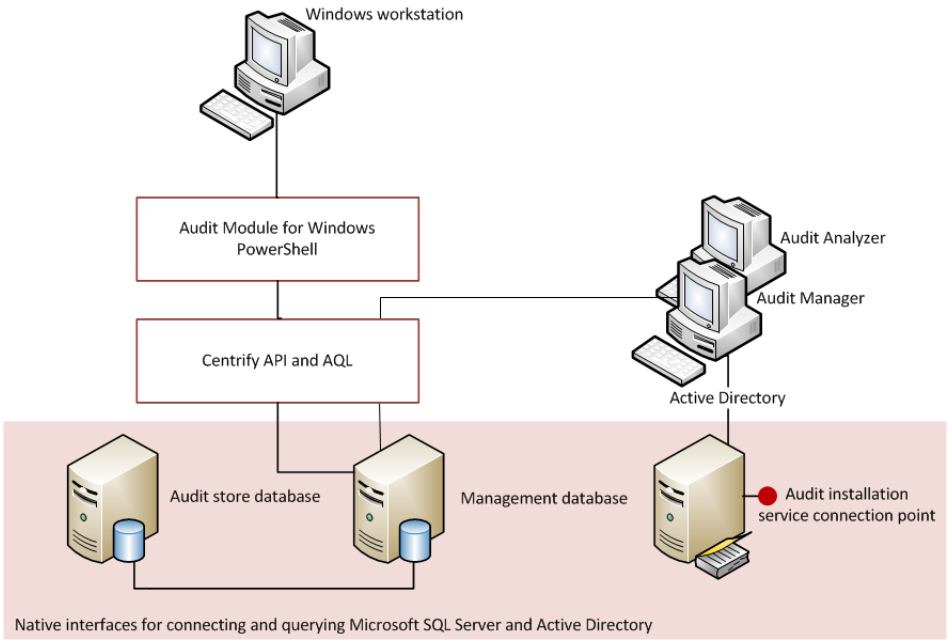
# Accessing audit information using native interfaces

The Audit Module for Windows 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.

The cmdlets rely on the underlying interfaces provided by Microsoft Active Directory Service Interfaces (ADSI), Microsoft SQL Server AQL query language, and Centrify 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 Centrify 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 Windows 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 Windows 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 Windows PowerShell from the Centrify 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

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 Centrify software is installed.

If you did not install the Audit Module for PowerShell as a component of a Centrify Server Suite Enterprise Edition installation, you can install it separately. The DirectAudit PowerShell files are available in the same disk image from which you installed Centrify 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 Centrify Server Suite, navigate to the /DirectAudit/Powershell folder and double-click the standalone executable to start the setup program.

   For example, double click the `Centrify_DirectAudit_PowerShellwin64.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 Windows 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 Windows 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 Centrify Audit module into your default Windows PowerShell console.

### To import the Centrify DirectManage Audit module:

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 Centrify Audit 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 Centrify audit cmdlets, similar to the following:

```
CommandType     Name                               Definition
-----------     ----                               ----------
Cmdlet          Attach-CdaDatabase                 Attach-CdaDatabase -AuditSto...
Cmdlet          Detach-CdaDatabase                 Detach-CdaDatabase -Database...
Cmdlet          Export-CdaAuditSessionRecording    Export-CdaAuditSessionRecord...
Cmdlet          Get-CdaActiveDatabase              Get-CdaActiveDatabase -Audit...
Cmdlet          Get-CdaAgent                       Get-CdaAgent -Installation <...
Cmdlet          Get-CdaAuditEvent                  Get-CdaAuditEvent -Installat...
Cmdlet          Get-CdaAuditRole                   Get-CdaAuditRole -Installati...
Cmdlet          Get-CdaAuditRoleAssignment         Get-CdaAuditRoleAssignment -...
Cmdlet          Get-CdaAuditSession                Get-CdaAuditSession -Install...
Cmdlet          Get-CdaAuditStore                  Get-CdaAuditStore -Installat...
Cmdlet          Get-CdaCollector                   Get-CdaCollector -Installati...
Cmdlet          Get-CdaDatabase                    Get-CdaDatabase -AuditStore ...
Cmdlet          Get-CdaInstallation                Get-CdaInstallation [-Name <...
Cmdlet          Get-CdaManagementDatabase          Get-CdaManagementDatabase -I...
Cmdlet          Get-CdaUnixCommand                 Get-CdaUnixCommand -Session ...
Cmdlet          Get-CdaUnixCommandTranscipt        Get-CdaUnixCommandTranscript...
Cmdlet          Get-CdaWindowsEvent                Get-CdaWindowsEvent -Session...
Cmdlet          New-CdaAuditRole                   New-CdaAuditRole -Installati...
Cmdlet          New-CdaAuditRoleAssignment         New-CdaAuditRoleAssignment -...
Cmdlet          New-CdaAuditStore                  New-CdaAuditStore -Installat...
Cmdlet          New-CdaDatabase                    New-CdaDatabase -AuditStore ...
Cmdlet          New-CdaSearchCriteria              New-CdaSearchCriteria [-Type...
Cmdlet          Publish-CdaInstallation            Publish-CdaInstallation -Ins...
Cmdlet          Remove-CdaAgent                    Remove-CdaAgent -Agent <CdaA...
Cmdlet          Remove-CdaAuditRole                Remove-CdaAuditRole -AuditRo...
Cmdlet          Remove-CdaAuditRoleAssignment      Remove-CdaAuditRoleAssignmen...
Cmdlet          Remove-CdaAuditSession             Remove-CdaAuditSession -Sess...
Cmdlet          Remove-CdaCollector                Remove-CdaCollector -Collect...
Cmdlet          Set-CdaActiveDatabase              Set-CdaActiveDatabase -Audit...
Cmdlet          Set-CdaAuditRole                   Set-CdaAuditRole -AuditRole ...
Cmdlet          Set-CdaAuditSession                Set-CdaAuditSession -Session...
Cmdlet          Set-CdaAuditStore                  Set-CdaAuditStore -AuditStor...
Cmdlet          Set-CdaConfiguration               Set-CdaConfiguration [-Domai...
Cmdlet          Set-CdaDatabase                    Set-CdaDatabase -Database <C...
Cmdlet          Set-CdaInstallation                Set-CdaInstallation -Install...
Cmdlet          Set-CdaManagementDatabase          Set-CdaManagementDatabase -M...
```

# 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.

The following topics are covered:

- Using cmdlets to manage auditing
- Preparing the environment to run cmdlets
- Organizing cmdlet operations in a sequence
- Checking for valid licenses
- Specifying parameters using different formats
- Working with sample scripts
- Writing your own scripts
- Getting information about the cmdlet available

## Using cmdlets to manage auditing

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

- Audited computers with the Centrify 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 Centrify data for auditing. Most of the cmdlets perform one of the following basic operations:

- `New-Cda`*`xxx`* cmdlets create new Centrify objects, such as a new audit role or a new audit store database.
- `Get-Cda`*`xxx`* cmdlets get the properties of a specified object.
- `Set-Cda`*`xxx`* cmdlets set or change the properties of a specified object.
- `Remove-Cda`*`xxx`* 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 Windows 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 `Set-CdaConfiguration` 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 `Set-CdaConfiguration` 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 Windows 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 `Get-Cda`*`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" | Remove-CdaAuditRole
```

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 | You can specify Active Directory users, groups, or computers using any of the following formats:<br>• Distinguished name string<br>• SID string<br>• *sAMMAccountName@domain*<br>• *domain**sAMAccountName*<br>You can specify Active Directory users, groups, or computers using a CdaAdPrincipal object. |
| CdaAccessAccount | You can specify a Windows account name or a SQL Server login account name and password, e.g.<br>For a Windows user account, all of the same formats listed for a CdaAdPrincipal object are supported.<br>For SQL Server login accounts, the format is “sql:*sql_name*:*sql_password*”. The password can be empty. |
| 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
```

```
$cdaAduitStoreObject = Get-CdaAuditStore -Installation
"DefaultInstallation" -Name "Default-First-Site"
# Use the audit store object instance to specify a parameter
value
Attach-CdaDatabase -AuditStore $cdaAduitStoreObject -Name
"audit-store-db" -Server "win2012\instance1" -Database
"audit-store-database"
```

# Working with sample scripts

The Audit Module for PowerShell includes a sample script—`db_rotation.ps1`—to demonstrate how you can use cmdlets to create a new audit store database and make the new database the active database for an installation, a process sometimes referred to as database rotation. You can copy and modify the sample script 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.

To run the sample script:

1. Open the Centrify 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 `Set-ExecutionPolicy` to allow execution. For example:

   ```
   Set-ExecutionPolicy Unrestricted
   ```

   For more information about execution policies and the options available, use the `get-help` 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 Centrify 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 Centrify 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 Centrify 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 `Get-CdaAuditSession` 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 Centrify 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 snipper 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 Centrify Audit Module for PowerShell.
2. Verify you have permission to execute scripts.

   ```
   Get-ExecutionPolicy
   ```

3 Run the `Set-ExecutionPolicy` cmdlet to change the execution policy. For example:

```
Set-ExecutionPolicy Unrestricted
```

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 Centrify 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  Synopsis
----                    --------  --------
Attach-CdaDatabase                Cmdlet    Attach an
existing audit store d...
Detach-CdaDatabase                Cmdlet    Detaches an
attached audit store...
Export-CdaAuditSessionRecording   Cmdlet    Exports the
video capture record...
Get-CdaActiveDatabase          Cmdlet   Gets the current
active database...
Get-CdaAgent                      Cmdlet    Gets one or
more audited computers.
```

```
Get-CdaAuditEvent                Cmdlet    Gets audit
trail events.
Get-CdaAuditRole                 Cmdlet    Gets an existing
audit role.
Get-CdaAuditRoleAssignment       Cmdlet    Gets an
existing audit role assi...
Get-CdaAuditSession              Cmdlet    Gets audit
sessions matching the...
Get-CdaAuditSessionReviewer      Cmdlet    Gets the Active
Directory users ...
Get-CdaAuditStore                Cmdlet    Gets the audit
stores for a spec...
Get-CdaCollector                 Cmdlet    Gets the
collectors for a specif...
Get-CdaDatabase                  Cmdlet    Gets a specified
audit store dat...
Get-CdaInstallation              Cmdlet    Gets the audit
installation for ...
Get-CdaManagementDatabase        Cmdlet    Gets the
management databases fo...
Get-CdaUnixCommand               Cmdlet    Gets the UNIX
command executed d...
Get-CdaUnixCommandTranscript     Cmdlet    Gets the UNIX
terminal text stri...
Get-CdaWindowsEvent              Cmdlet    Gets the Windows
events captured...
New-CdaAuditRole                 Cmdlet    Creates a new
audit role.
New-CdaAuditRoleAssignment       Cmdlet    Creates a new
audit role assignm...
New-CdaAuditStore                Cmdlet    Creates a new
audit store.
New-CdaDatabase                  Cmdlet    Creates a new
audit store database.
New-CdaSearchCriteria            Cmdlet    Creates a new
search criteria ob...
Publish-CdaInstallation          Cmdlet    Publishes or
synchronizes instal...
Remove-CdaAgent                  Cmdlet    Deletes an
audited computer data...
```

```
Remove-CdaAuditRole                 Cmdlet    Deletes an
exiting audit role.
Remove-CdaAuditRoleAssignment       Cmdlet    Deletes an
existing audit role a...
Remove-CdaAuditSession              Cmdlet    Deletes an
existing session.
Remove-CdaCollector                 Cmdlet    Deletes an
existing collector.
Remove-CdaDatabase                  Cmdlet    Deletes the
specified database f...
Set-CdaActiveDatabase               Cmdlet    Updates the
database to use as t...
Set-CdaAuditRole                    Cmdlet    Updates
properties for an existi...
Set-CdaAuditSession                 Cmdlet    Updates
properties for an existi...
Set-CdaAuditSessionReviewer         Cmdlet    Updates the
list of users and gr...
Set-CdaAuditStore                   Cmdlet    Updates
properties for an existi...
Set-CdaConfiguration                Cmdlet    Defines the
preferred domain con...
Set-CdaDatabase                     Cmdlet    Updates
properties for an existi...
Set-CdaInstallation                 Cmdlet    Updates
properties for an existi...
Set-CdaManagementDatabase           Cmdlet    Updates
properties for an existi...
```

# Auditing-related objects and properties

Most Audit Module for PowerShell cmdlets return object instances either directly or as properties of other objects. This chapter provides an alphabetical listing of the objects and the properties of each object defined in the Audit Module for Windows PowerShell. Note that not all properties are available as parameters in the PowerShell cmdlets.

"CdaUnixCommand" on page 46

"CdaUnixCommandTranscript" on page 47

"CdaUserEvent" on page 48

"CdaWindowsEvent" on page 48

# CdaAccessAccount

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

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

object.

# 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 |
|---|---|---|
| `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. |

| Property | Type | Description |
|---|---|---|
| `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:<br>• Connexted<br>• Disconnected |
| `Type` | `Enum` | Type of operating system running on the computer hosting the auditing service. The valid values are:<br>0 — Unknown<br>1 — UNIX<br>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. |
| `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. |
| `UniqueId` | `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. |

# 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:<br>• 1 if the audit scope is an Active Directory site.<br>• 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. |
| `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. |

| Property | Type | Description |
|---|---|---|
| `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:<br>• 0 for None<br>• 1 for ToBeReviewed<br>• 2 for Reviewed<br>• 3 for PendingForAction<br>• 4 for KeepForever<br>• 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. |

| Property | Type | Description |
|---|---|---|
| State | Enum | Status of the session. The valid values are:<br>• -1 for Unknown<br>• 0 for InProgress<br>• 1 for Terminated<br>• 2 for Disconnected<br>• 3 for Completed |
| Type | Enum | Session type. The valid values are:<br>• 1 if the session is a Windows session<br>• 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 | Centrify zone name. |

# CdaAuditStore

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

| Property | Type | Description |
|---|---|---|
| `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. |

# CdaAuditStoreDatabase

Represents an audit store database. The following properties are

| 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. |
| 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. |

| Property | Type | Description |
|---|---|---|
| `Status` | `Enum` | Database status. The valid values are:<br>• Connected<br>• Disconnected |
| `Version` | `String` | Database version number. |

defined for this object.

# 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. |
| `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. |

| Property | Type | Description |
|---|---|---|
| `Status` | `Enum` | Status of the collector service. The valid values are:<br>• Connected<br>• Disconnected |
| `UpTime` | `TimeSpan` | Total time the collector was connected time from startup to the last update time. |
| `Version` | `String` | Collector service version number. |

# 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. |

| Property | Type | Description |
|---|---|---|
| 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. |

# 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:<br>• Connected<br>• Disconnected |

# 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. |

| Property | Type | Description |
|---|---|---|
| ReviewStatus | Enum | Filter sessions by using the session review status. The valid values are:<br>• 0 for None<br>• 1 for ToBeReviewed<br>• 2 for Reviewed<br>• 3 for PendingForAction<br>• 4 for KeepForever<br>• 5 for ToBeDeleted |
| State | Enum | Filter sessions by using the status of the session. The valid values are:<br>• 0 for InProgress<br>• 1 for Terminated<br>• 2 for Disconnected<br>• 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:<br>• 1 if the session is a Windows session<br>• 2 if the session is a UNIX session |

| Property | Type | Description |
|---|---|---|
| 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 UNIC 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. |

| Property | Type | Description |
|---|---|---|
| `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. |