BeyondTrust

Privileged Identity 5.5.4
PowerShell API Guide

 TC:3/12/2019

Table of Contents

Introduction to the BeyondTrust Privileged Identity API

Privileged Identity is a solution designed to:

- Discover systems, devices, and accounts in your network
- Manage the passwords or SSH keys for those discovered accounts

With BeyondTrust PI's API support, it is possible to perform day-to-day operations without ever using the web application or management console. Common uses for API access include programmatic retrieval of passwords, integration into third-party applications, workflow establishment, system and identity orchestration, etc.

Programmatic access to Privileged Identity can occur through multiple web service endpoints.

The web service supports a REST/JSON format and is required for the Privileged Identity web application to function. Use of the API does not bypass the standard delegation system. Any identity making a programmatic call must still be delegated the proper permissions like website users in order to perform any actions.

For discovery and management, the target systems need to be online and have network connectivity with Privileged Identity.

Regular Authentication

When the service is installed, certain parameters are configured and affect configurations in IIS and the web service. For example, if you installed the web service during installation and configured it to use **Anonymous Authentication** and SSL, any attempts to access the web service using an alternate authentication method results in an error.

If the web service is configured to use **Anonymous Authentication**, you must pass username, password, and authenticator information at log in. If the web service is configured to use **Integrated Windows Authentication** (IWA), you can login without providing further information, or you may pass username, password, and authenticator information.

In any scenario, an authentication token is required to log in and to perform additional commands.

Multiple Authentication Scenarios

If you have a scenario where users connect to trusted Windows machines and wish for them to be able to login without supplying a username and password, you must install the web service with **Integrated Windows Authentication** support. However, if you have clients or processes that must programmatically access Privileged Identity without integrated authentication, follow the steps below:

1. Go to the host system supporting the web service, **%inetpub%\wwwroot\erpmwebservice**.
2. Copy the **ErpmWebService** folder.
3. Place the copy in **%inetpub%\wwwroot**.
4. Delete the current **web.config** file in this new directory.
5. Copy the required **web.config example** file.
6. Rename it to **web.config**.
7. In IIS, right-click **Convert to application**, and convert the **ERPMWebServiceAnon** directory to an application.

Delegations & Access

Regardless of which method is used to programmatically access Privileged Identity, the calling user must be authenticated and must have proper delegations to perform the requested action. Before any actions can occur, a user must be granted the global **Logon** permission. The permission can be directly assigned or inherited.

A successful login provides the calling user an authentication token. This token is passed to all subsequent calls as **AuthenticationToken**.

Authentication tokensl have the same idle life-time expiration, which defaults to 20 minutes in both IIS and Privileged Identity settings. If a token sits idle for 20 minutes or the user logs out, the token expires, and the user must log in to obtain a new authentication token.

Permissions Required for Management set Manipulation in Powershell Commandlets

With **Global management set permissions on the management set**, the management set is assigned to the delegation identity in the global delegation dialog.

Alternatively, use **Change Group Membership Permission on the specific management set**. This can be assigned on a per-management-set basis by configuring per-management-set permissions in the console or through the API.

Web Service vs Web Application Dependency

While the web service communicates with the database directly and is responsible for its own client communications, the web service is dependent on the web application's configuration options. A web service installed on a system also hosting a web application inherits that particular web application's settings. A web service installed on a system not hosting a web application must have a web server's registry configuration exported and manually imported to the web service host. Changes can be made to the configuration by directly editing the registry or the registry import file.

Database connectivity is key. If the database is unavailable, the web application is unable to provide any services to calling users.

URI Information

The REST API is accessed at **serverName/ErpmWebService/AuthService.svc/REST**. REST help pages are available at **serverName/ErpmWebService/AuthService.svc/REST/help**.

PowerShell Cmdlets

This guide documents PowerShell cmdlets you can use to extend the management of Privileged Identity to a shell / scripting environment.

The PowerShell cmdlets can run from any system that supports PowerShell 3.0+. The PowerShell commands requires the Privileged Identity web service to be installed, functional, and accessible to you. Before you install the PowerShell cmdlets, consider the following::

- How will authentication occur? Windows integrated? Anonymous?

  > ***Note:*** *We strongly recommend against using certificate-based authentication because PowerShell is known to refuse client certificates, resulting in a "Could not establish a secure channel" error message. For password-less PowerShell authentication, we recommend using Integrated Windows Authentication.*

- Is SSL enabled?
- What port is the web service listening on?
- What is the full URL to the web service?

There are three sets of PowerShell cmdlets distributed with Privileged Identity:

- **LSClientAgentCommandlets:** Provides web application and management console-equivalent functionality.
- functionality for web application, web service, and zone processor deployment and management.

If using the PowerShell profile files, LSClientAgentCommandlets is automatically imported when you start PowerShell. The other two modules can be imported using the import-module command. If needed, modify the profile to include these extra cmdlets.

> ***Note:*** *Program configuration such as data store or solution email configuration cannot be performed programmatically and must be done by the management console.*

Install the PowerShell Cmdlets

PowerShell Cmdlets

Cmdlets can be distributed to any Windows computer as long as network connectivity to the target web service has been established. Before using PowerShell cmdlets, make sure the following is in place:

1. Ensure Prerequisites Are Met
2. Check and Set the Execution Policy
3. Create Folders and Distribute the Cmdlets
4. Configure the Client

Ensure Prerequisites Are Met

Windows PowerShell 3.0+ is required. Previous versions of Windows need to download and install **Windows Management Framework** (WMF). WMF version 4+ is recommended. WMF 4.0 requires Microsoft .NET Framework 4.5+..

1. Open **PowerShell** or **PowerShell ISE**.
2. Run the following command:

   ```
   Get-Host
   ```

Check and Set the Execution Policy

> ***Note:*** *To set the execution policy, administrator privileged are required.*

Set the execution policy to **AllSigned**, **RemoteSigned**, or **Unrestricted** to use the PowerShell cmdlets. Also, if you leverage these cmdlets from both PowerShell x64 and x86, you must take the following steps:

1. Open **PowerShell** or **PowerShell ISE**.
2. Run the following command:

   ```
   Get-ExecutionPolicy
   ```

   If the execution policy is set to **Restricted**, the execution policy must be changed. Otherwise, your system is ready to use the cmdlets.

3. If the execution policy must be changed, open an administrative **PowerShell** or **PowerShell ISE**.
4. Run the following command:

   ```
   Set-ExecutionPolicy -ExecutionPolicy RemoteSigned
   ```

5. Click **Yes** on the security warning..
6. Run the following command to verify the execution policy is properly set:

   ```
   Get-ExecutionPolicy
   ```

7. Close **PowerShell**.

Create Folders and Distribute the Cmdlets

This process describes one possible way to deploy and configure a PowerShell environment. There are three sets of PowerShell cmdlets distributed with Privileged Identity:

- **LSClientAgentCommandlets:** Provides web application and management console-equivalent functionality.
- **LSClientUpdateConfiguration:** Provides some management functionality for the web application, web service, and zone processor.
- **LSClientUpdatePassword:** Provides functionality for working with the **Offline Account Update** feature.

The required PowerShell files are located in the Privileged Identity installation path at `\SupplementalInstallers\LSCPowerShellCmdlets`.

1. In the target user's profile, create the following folder structure:

   ```
   %userprofile%\Documents\WindowsPowerShell\Modules.
   ```

2. Copy the desired cmdlets to the **Modules** subdirectory in the user's profile. **LSClientAgentCommandlets** is most common.
3. Copy the two profile.ps1 files to the **WindowsPowerShell** subdirectory in the user's profile. **Microsoft.PowerShell_profile.ps1** is for the standard PowerShell environment while **Microsoft.PowerShellISE_profile.ps1** is for PowerShell ISE. These affect both x64 and x86 environments.
4. Launch **PowerShell**.
5. Run the following command to validate the desired modules loaded:

   ```
   Get-Module -Name LSClient*
   ```

By default, the profile automatically loads **LSClientAgentCommandlets** only. If it should automatically load any of the cmdlets, edit the appropriate profile file, and modify the existing import-module command or add a new import-module command. You can also run the import-module command at any time to load the modules by hand.

Future PowerShell upgrades require re-copying the three folders and their modules to the target systems and overwriting the previous versions.

Configure the Client

The client must know information about the web service endpoint it is communicating with, specifically the endpoint URL and how to authenticate.

There are three way to set the client's configuration:

- Use PowerShell
- Push the configuration from the management console
- Edit the registry

To use PowerShell, configure the client settings and define where the web service is hosted. Use the **Set-LSClientWebServiceSettings** cmdlet to run the configuration. The syntax is as follows:

```
Set-LSClientWebServiceSettings [-EnableWebService] <bool> [-WebServiceAddress <string>] [-
IntegratedAuth <bool>] [-ClientCert <string>] [-SSLEnabled <bool>] [-PassException] [-Trace] [-
RunAs <pscredential>] [<CommonParameters>]
```

The variables can be entered in any order. The variables are as follows:

- **EnableWebService:** If configuring the host to use the web service, this value should be set to `$true`.
- **ClientCert:** If using user certificates to perform login, specify the friendly name of the user certificate as shown in the user's certificate store. If using **Integrated Windows Authentication**, omit this variable or if passing a username and password. To use certificate-based authentication, the web service must be configured with SSL and accept client certificates.

  > ***Note:*** *We strongly recommend against using certificate-based authentication because PowerShell is known to refuse client certificates, resulting in a* ***Could not establish a secure channel*** *error message. This is a known issue with Windows PowerShell. For password-less PowerShell authentication, we recommend using Integrated Windows Authentication.*

- **IntegratedAuth:** If using **Integrated Windows Authentication**, this value should be set to `$true`. If you are passing a username and password or using client certificates, set this value to `$false`. :
  - Set the web service and website to enable **Integrated Windows Authentication**
  - Set **Anonymous Authentication** to disabled
  - Set the web application global option to permit **Integrated Windows Authentication**
- **SSLEnabled:** If the website uses SSL, set this value to `$true`. Be aware that enabling SSL also changes the default listening port from 80 to 443.
- **WebServiceAddress:** Enter the full URL, including the protocol and port to the web service page and authservice.svc, such as

  `https://webserver.domain.int:65535/erpmwebservice/AuthService.Svc.`

Any item entered can be changed at any time by re-running the above command or by manipulating the registry at `HKLM\Software\WoW6432Node\Lieberman\ClientAccountManagement\GlobalSettings`. The registry values are appropriately named.

To view the client's current settings, run the cmdlet **Get-LSClientSettings** with no parameters.

Because this writes to the system's registry key, the **Set-LSClientWebServiceSettings** cmdlet must be ran as an administrator.

```
Set-LSClientWebServiceSettings -EnableWebService $True -IntegratedAuth $True -SSLEnabled $True -
WebServiceAddress https://lsdslscprd.lsds.int/erpmwebservice/authservice.svc
```

Client Configuration Cmdlet Alternative

An alternative cmdlet is **Set-LSClientSettings**. This cmdlet configures the client for the web service. The syntax is as follows:

```
Set-LSClientSettings [-WebserverName] <string> [-Page] <string> [-SSLEnabled] <string> [-
VerboseLogging] <string> [-ClientCert] <string> [-IntegratedAuth] <string> [-CustomPort] <string>
[-EnableWebService] <bool> [-WebServiceAddress] <string> [[-UserCertStore] <bool>] [-
PassException] [-Trace] [-RunAs <pscredential>] [<CommonParameters>]
```

This cmdlet adds the following parameters to those noted above:

- **CustomPort:** This value should be configured any time the website is not listening on the default port of 80. If the port ever changes from port 80, this value should be configured.
- **Page:** This value is not required for configuring the web service communication.
- **VerboseLogging:** This value is optional and supplies all logging messages to the local client. This significantly slows down operations and should normally be set to `$false`.
- **WebserverName:** This is the name (or FQDN) of the host providing the web service. Consider if the host requires a full FQDN or can be accessed by a host name. This is especially important when using SSL because the certificate supplies the entire

host name. If a proper SSL handshake cannot be established, the cmdlets will not connect. Also, if the web service is being hosted by an NLB cluster, supply the virtual name or IP of the cluster rather than the name or IP of a specific host.

```
Set-LSClientSettings - ClientCert '' -CustomPort 443 -EnableWebService $True -IntegratedAuth 1 -
Page /pwcweb/ClientAgentRequests.asp -SSLEnabled 1 -VerboseLogging 0 -WebserverName
lsdslscprd.lsds.int -WebServiceAddress https://lsdslscprd.lsds.int/erpmwebservice/authservice.svc
-UserCertStore $False
```

Any item can be changed at any time by re-running the above command or by manipulating the registry at `HKLM\Software\WoW6432Node\Lieberman\ClientAccountManagement\GlobalSettings`. The registry values are appropriately named.

To view the clients current settings, run the cmdlet **Get-LSClientSettings** with no parameters.

Because this writes to the system's registry key, the **Set-LSClientWebServiceSettings** cmdlet must be ran as an administrator.

PowerShell Cmdlet Reference

Use the standard PowerShell command `Get-Help cmdlet_Name` syntax to receive further information about any available cmdlet.

Most cmdlets also support the `PassException` and `Trace` clauses to pass back additional logging information that otherwise are suppressed.

Objects and Enumerations

Objects and enumerations used by the PowerShell cmdlets are in the `LSClientAgentCommandlets.RouletteWebService` namespace.

Creating an object in PowerShell that is defined in the web service appears like:

```
$ObjectName = New-Object -TypeName LSClientAgentCommandlets.RouletteWebService.ObjectName
```

To create a new `TargetIdentificationInfo` object,, use the command:

```
$oSampleTarget = New-Object -TypeName
LSClientAgentCommandlets.RouletteWebService.TargetIdentificationInfo
```

To assign an enumerated value that is defined in the web service to a variable of a structure, use:

```
$object.value = [LSClientAgentCommandlets.RouletteWebService.EnumerationName]::EnumerationValue
```

To create a `TargetIdentificationInfo` object named `$oSampleTarget` and to set a member variable named `Type` (of type `ETargetType`) to the `OS_Windows` enumeration value, use this command:

```
$oSampleTarget.Type = [LSClientAgentCommandlets.RouletteWebservice.ETargetType]::OS_Windows
```

The following pages outline the current PowerShell cmdlets.

PowerShell: Login

Get-LSLoginToken

Get-LSLoginToken is used to obtain an **AuthenticationToken** and is the first step to performing any subsequent operations. A successful authentication provides an **AuthenticationToken**, which is passed to other subsequent commands.

Related Calls

- **SOAP:** DoLogin
- **REST:** Login (POST)

Syntax

```
Get-LSLoginToken [[-Authenticator] <string>] [[-Username] <string>] [[-Password] <string>] [[-
Credential] <pscredential>] [[-MFAToken] <string>] [-PassException] [-Trace] [-RunAs
<pscredential>] [<CommonParameters>]
```

Parameters

- **Authenticator:** The name of authentication server entry as seen in the web application or authentication server entry label. This value is not supplied for Integrated Windows Authentication, explicit accounts, certificates. It is still required when using the **Credential** option.
- **Username:**The name of the user attempting to authenticate. This value is not supplied for Integrated Windows Authentication, certificates, or when using the Credential option.
- **Password:** The password for the user attempting to authenticate. This value is not supplied for Integrated Windows Authentication, certificates, or when using the Credential option.
- **Credential:** If you must supply a username and password, rather than passing them directly to the cmdlet, you can create a **PSCredential** object. See Microsoft documentation for more information.
    - If running interactively, use `Get-Credential` and PowerShell prompts you for the username and password:

      `$myId = Get-Credential`

      You can now pass `$myId` to the **Credential** option.
    - If running in non-interactive mode:

      `$secpasswd = ConvertTo-SecureString “PlainTextPassword” -AsPlainText -Force`

      `$myUnPw = New-Object System.Management.Automation.PSCredential (“username”, $secpasswd)`
    - You can now pass `$myUnPw` to the **Credential** option.
- **MFAToken:** This value is required if the user is configured to **Require OATH/Yubico MFA**. This option is only supported for users configured for OATH Token MFA.

Example Requests

- Integrated Windows Authentication or Certificate.

  ```
  Get-LSLoginToken
  ```

- Program explicit authentication:

  ```
  Get-LSLoginToken -Username loginName -Password loginPassword
  ```

- Standard authentication using LDAP directory, Windows domain, RADIUS, or LDAP authentication:

  ```
  Get-LSLoginToken -Authenticator DIRECTORY/DOMAIN/RADIUS_name -Username loginName -Password
  loginPassword
  ```

Output Success

Theoutput ia an authentication token and is passed to other commands as **AuthenticationToken**.

Example:

```
TTQ7T84EQ2X57QP9B725HA8OE7OQB9W6
```

Output Fail

Login can fail for a number of reasons, such as a entering the wrong username/password combination or the lack of proper delegation.

- **Bad Password or Username**

  Login failed, or the username was not found.

- **Time restrictions for operation or Logon permission not granted**

  Identity cannot perform operation at this time, or the identity has time restrictions.

Note: *There is not a PowerShell logout cmdlet.*

PowerShell: Get-LSLoginSAMLToken

Get-LSLoginSAMLToken is the first step in performing any subsequent operations using SAML-based authentication. A successful authentication provides an **AuthenticationToken**, which passes to other subsequent commands. To use this cmdlet, you must first log into your SAML provider and retrieve a base64 SAML response. This is passed directly to the cmdlet.

Related Calls

- **SOAP:** DoLoginSAML
- **REST:** LoginSAML (POST)

Syntax

```
Get-LSLoginSAMLToken [-SAMLResponseBase64] <string> [-PassException] [-Trace] [-RunAs
<pscredential>] [<CommonParameters>]
```

Options

- **SAMLResponseBase64:** The base64 response received from the SAML provider.

Example Request

- Login

```
Get-LSLoginSAMLToken - SAMLResponseBase64 $samlResponse
```

Output Success

The output is an authentication token, which is passed to other commands as **AuthenticationToken**.

Example

```
TTQ7T84EQ2X57QP9B725HA8OE7OQB9W6
```

Output Fail

Login can fail for a number of reasons such as entering the wrong username/password combination, the lack of proper delegation, or an incorrect SAML response.

- **Bad Password or Username**

  Login failed, or the username was not found.

- **Time restrictions for operation or Logon permission not granted**

  Identity cannot perform the operation at this time, or the identity has time restrictions

- **Bad SAML response**

  There was an unexpected token "token_data".

- **Login Attempt falls outside of NotOnOrAfter condition**

  A session failed to generate, or the login attempt failed with the result: 2147746477.

PowerShell: Auditing

Use PowerShell cmdlets to view web application audit logs.

Get-LSListWebAuditLogs

Get-LSListWebAuditLogs returns the web application audit logs for a given date range, if specified.

Permissions Required

- View Web Audit Logs

Related Commands

- **SOAP:** LoggingOps_GetWebAuditLog
- **REST:** Logs/WebLogs (GET)

Syntax

```
Get-LSListWebAuditLogs [-AuthenticationToken] <string> [-StartTime <datetime>] [-EndTime
<datetime>] [-PassException] [-Trace] [-RunAs <pscredential>]
```

Parameters

- **AuthenticationToken:** The authentication token of the calling user.
- **StartTime:** *(Optional)* The start date and time of the date range.
- **EndTime:** *(Optional)*The end date and time of the date range.

Parameter Help

The start and end time parameters may be entered in one of two ways:

- Parsing normal date/time values: `[System.DateTime]::Parse("12/29/2016 12:42:00 AM")`
- Entering the time as follows (string): `"YYYY-DD-HHThh:mm:ss". For example: "2016-12-29T00:42:00"`

Example Request

The request accepts a start and end date for the audit logs. If a date range is not specified, the entire audit log history is returned.

No date range specified

```
Get-LSListWebAuditLogs - AuthenticationToken $tok
```

Date range specified

```
#Create start date
$startdate = [System.DateTime]::Parse("05/10/2017 00:00:00 AM")
#Create end date
$enddate = [System.DateTime]::Parse("05/17/2017 11:59:59 PM")
#Get audit logs
Get-LSListWebAuditLogs -AuthenticationToken $tok
```

Output Success

If the command is successful, the audit logs for the given date range are returned.

Example Success Output

```
ExtensionData         : System.Runtime.Serialization.ExtensionDataObject
ClientAgentOperation  : False
HTTPSEnabled          : False
IPAddress             : fe80::c1e0:93f:1f9f:c1d5%12
JobID                 : 0
OperationAccount      :
OperationResult       : OPERATION_SUCCESS
OperationSystem       :
ServerName            :
Timestamp             : 5/12/2017 3:07:08 PM
Username              : user@example.com
WebOperation          : LOGOFF
WebServiceOperation   : False
ExtensionData         : System.Runtime.Serialization.ExtensionDataObject
ClientAgentOperation  : False
HTTPSEnabled          : False
IPAddress             : fe80::c1e0:93f:1f9f:c1d5%12
JobID                 : 0
OperationAccount      :
OperationResult       : OPERATION_SUCCESS
OperationSystem       :
ServerName            :
Timestamp             : 5/12/2017 3:07:21 PM
Username              : user@example.com
WebOperation          : LOGON
WebServiceOperation   : False
```

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Improperly formatted date/time value**

  The value **bad_string_Value_goes_here** cannot be parsed as the type **DateTime**.

PowerShell: Jobs & Job Settings

Use PowerShell to add, edit, list, and delete jobs. Jobs are created for interactive and scheduled management activities.

During job creation, the cmdlets creating jobs cannot set a schedule, except for the "-RunNow" flag. To control the job's schedule, use "Set-LSJobSchedule" on page 1.

Get-LSListJobs

Get-LSListJobs returns job information for jobs matching a particular filter such as the type of job or last result of a job.

Permissions Required

- All Access

Related Commands

- **REST:** Job (GET)

Syntax

```
Get-LSListJobs [-AuthenticationToken] <string> [-MaxCountReturned <int>] [-JobOperationType
<EJobOperation> {Unknown | SendMessage | Reboot | AbortReboot | PasswordChange | Refresh_All |
Refresh_SystemInfo | WebOperation | UpdateManagementSet | ActivityReport_Manager |
ActivityReport_Admin | Refresh_SystemInfoAndCredentialReferences | GenReport_StoredPasswordsTest
| GenReport_StoredPasswords | Refresh_TrustInfo | Refresh_TrustInfoForDomain | Refresh_
CredentialReferences | Refresh_InstanceAccounts_SQLServer | Refresh_InstanceAccounts_MySQL |
Refresh_InstanceData_SQLServer | Refresh_InstanceData_MySQL | Refresh_InstanceAccounts_Oracle |
Refresh_InstanceData_Oracle | GenReport_ComplianceReportingDataSnapshot | Refresh_InstanceData_
CustomAccountStore | Refresh_InstanceAccounts_CustomAccountStore | AccountElevation | Refresh_
InstanceData_Oracle_InternetDirectory | Refresh_InstanceAccounts_Oracle_InternetDirectory |
Refresh_InstanceData_Novell_eDirectory | Refresh_InstanceAccounts_Novell_eDirectory | Refresh_
InstanceAccounts_Sybase | Refresh_InstanceData_Sybase | Refresh_InstanceData_IBM_Tivoli |
Refresh_InstanceAccounts_IBM_Tivoli | Refresh_InstanceData_BMCThroughIPMI | Refresh_
InstanceAccounts_BMCThroughIPMI | Refresh_InstanceData_ViewDS | Refresh_InstanceAccounts_ViewDS |
UpdateSSHKeyData | Refresh_SelectedData | Refresh_InstanceAccounts_PostgreSQL | Refresh_
InstanceData_PostgreSQL | GenReport_SecurityPolicyCheck | Ops_AppDataStoreMaintenance | Refresh_
InstanceSystems_CustomAccountStore | Refresh_InstanceAccounts_Teradata | Refresh_InstanceData_
Teradata | Refresh_InstanceData_XeroxPhaser | GenReport_AuditSettings | GenReport_EventLogEvents
| GenReport_EventLogInfos | GenReport_FilePermissions | GenReport_Files | GenReport_GlobalGroups
| GenReport_GroupMembership | GenReport_GlobalGroupMembership | GenReport_UserGroupMembership |
GenReport_IEUpdates | GenReport_InstalledSoftware | GenReport_LocalGroups | GenReport_LocalUsers
| GenReport_LoggedOnUsers | GenReport_NetShares | GenReport_Policies | GenReport_RegistryKeys |
GenReport_Rights | GenReport_TrustAccounts | GenReport_UnixAccounts | GenReport_VNCInstances |
GenReport_WindowsUpdates | GenReport_WMI | GenReport_SystemInfo | GenReport_NetUses | GenReport_
NetSessions}] [-LastResult <ELastResult> {Unknown | HasNotRun | Incomplete_InProcess | Complete_
WithFailures_CanRetry | Complete_WithFailures_NoRetry | Complete_NoFailures_Rescheduled |
Complete_NoFailures | Complete_WithFailures_NoRetry_Rescheduled | Disabled | MissedRun_
Rescheduled | MissedRun | Incomplete_PartialRun}] [-PassException] [-Trace] [-RunAs
<pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** The AuthenticationToken of the calling user.
- **JobOperationTypeI:** *(Optional)* This is an enumerated value and can be passed as an integer or string value. The filtering options for job type include:
    - 0 or Unknown
    - 1 or SendMessage
    - 2 or Reboot
    - 3 or AbortReboot
    - 4 or PasswordChange
    - 5 or Refresh_All
    - 6 or Refresh_SystemInfo
    - 7 or WebOperation
    - 8 or UpdateManagementSet
    - 9 or ActivityReport_Manager
    - 10 or ActivityReport_Admin
    - 11 or Refresh_SystemInfoAndCredentialReferences
    - 12 or GenReport_StoredPasswordsTest
    - 13 or GenReport_StoredPasswords
    - 14 or Refresh_TrustInfo
    - 15 or Refresh_TrustInfoForDomain
    - 16 or Refresh_CredentialReferences
    - 17 or Refresh_InstanceAccounts_SQLServer
    - 18 or Refresh_InstanceAccounts_MySQL
    - 19 or Refresh_InstanceData_SQLServer
    - 20 or Refresh_InstanceData_MySQL
    - 21 or Refresh_InstanceAccounts_Oracle
    - 22 or Refresh_InstanceData_Oracle
    - 23 or GenReport_ComplianceReportingDataSnapshot
    - 24 or Refresh_InstanceData_CustomAccountStore
    - 25 or Refresh_InstanceAccounts_CustomAccountStore
    - 26 or AccountElevation
    - 27 or Refresh_InstanceData_Oracle_InternetDirectory
    - 28 or Refresh_InstanceAccounts_Oracle_InternetDirectory
    - 29 or Refresh_InstanceData_Novell_eDirectory
    - 30 or Refresh_InstanceAccounts_Novell_eDirectory
    - 31 or Refresh_InstanceAccounts_Sybase
    - 32 or Refresh_InstanceData_Sybase
    - 33 or Refresh_InstanceData_IBM_Tivoli
    - 34 or Refresh_InstanceAccounts_IBM_Tivoli

- 35 or Refresh_InstanceData_BMCThroughIPMI
- 36 or Refresh_InstanceAccounts_BMCThroughIPMI
- 37 or Refresh_InstanceData_ViewDS
- 38 or Refresh_InstanceAccounts_ViewDS
- 39 or UpdateSSHKeyData
- 40 or Refresh_SelectedData
- 41 or Refresh_InstanceAccounts_PostgreSQL
- 42 or Refresh_InstanceData_PostgreSQL
- 43 or GenReport_SecurityPolicyCheck
- 44 or Ops_AppDataStoreMaintenance
- 45 or Refresh_InstanceSystems_CustomAccountStore
- 46 or Refresh_InstanceAccounts_Teradata
- 47 or Refresh_InstanceData_Teradata
- 48 or Refresh_InstanceData_XeroxPhaser
- 49 or GenReport_AuditSettings
- 50 or GenReport_EventLogEvents
- 51 or GenReport_EventLogInfos
- 52 or GenReport_FilePermissions
- 53 or GenReport_Files
- 54 or GenReport_GlobalGroups
- 55 or GenReport_GroupMembership
- 56 or GenReport_GlobalGroupMembership
- 57 or GenReport_UserGroupMembership
- 58 or GenReport_IEUpdates
- 59 or GenReport_InstalledSoftware
- 60 or GenReport_LocalGroups
- 61 or GenReport_LocalUsers
- 62 or GenReport_LoggedOnUsers
- 63 or GenReport_NetShares
- 64 or GenReport_Policies
- 65 or GenReport_RegistryKeys
- 66 or GenReport_Rights
- 67 or GenReport_TrustAccounts
- 68 or GenReport_UnixAccounts
- 69 or GenReport_VNCInstances
- 70 or GenReport_WindowsUpdates
- 71 or GenReport_WMI
- 72 or GenReport_SystemInfo
- 73 or GenReport_NetUses
- 74 or GenReport_NetSessions

- **LastResult:** *(Optional)* This is is an enumerated value and can be passed as an integer or string value. Fiilters jobs based on **Last Result**. Options include:
  - 0 or Unknown
  - 1 or HasNotRun
  - 2 or Incomplete_InProcess
  - 3 or Complete_WithFailures_CanRetry
  - 4 or Complete_WithFailures_NoRetry
  - 5 or Complete_NoFailures_Rescheduled
  - 6 or Complete_NoFailures
  - 7 or Complete_WithFailures_NoRetry_Rescheduled
  - 8 or Disabled
  - 9 or MissedRun_Rescheduled
  - 10 or MissedRun
  - 11 or Incomplete_PartialRun
- **MaxCountReturned:** *(Optional)* Defines the maximum number of jobs to return.

Example Request

No Options - Return Everything

```
Get-LSListJobs - AuthenticationToken $tok
```

Filter for Job Type: AccountElevation

```
Get-LSListJobs -AuthenticationToken $tok - JobOperationType AccountElevation
```

Filter for password change jobs with a result of Complete_WithFailures_NoRetry

```
Get-LSListJobs -AuthenticationToken $tok -JobOperationType PasswordChange - LastResult Complete_
WithFailures_NoRetry
```

Output Success

Output is all jobs and descriptions matching any defined filters.

Example Success Output

```
ExtensionData          : System.Runtime.Serialization.ExtensionDataObject
AssociatedGroup        : [Web Job]
Comment                : Account Elevation Job created by Web Application (user lsds\lscadmin) - i
                         like being elevated
CreatedBy              : lsds\erpmweb
CreationTimeUTC        : 4/24/2017 8:21:35 PM
JobID                  : 1195
JobOperation           : AccountElevation
JobType                : OriginatedFromWebApp
```

```
LastResult             : HasNotRun
LastRunTimeUTC         : 4/24/2017 8:21:35 PM
NextRunTimeUTC         : 4/24/2017 8:21:35 PM
PullSystemsFromGroup   : False
RefreshGroupBeforeRun  : False
RunJobOnNewSystems     : False
```

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Invalid identifier name for JobOperationType or LastResult**

  Parameter **JobOperationType** could not be binded, or paramter **LastResult** could not be binded.

PowerShell: Get-LSJobSchedule

Get-LSJobSchedule returns the schedule parameters for a target job ID.

Permissions Required

- Delegated permissions on the job

Related Commands

- **SOAP:** JobOps_GetJobSchedule
- **REST:** Job/Schedule (GET)

Syntax

```
Get-LSJobSchedule [-AuthenticationToken] <string> [-JobID] <string> [-PassException] [-Trace] [-
RunAs <pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** The authentication of the calling user.
- **JobID:** The ID of the job you are retrieving a schedule for.

Example Request

```
Get-LSJobSchedule -AuthenticationToken $tok - JobID 1
```

Output Success

The scheduling options for the job are provided.

Example Success Output

```
ExtensionData                    : System.Runtime.Serialization.ExtensionDataObject
DayOfMonth                       : 0
DayOfWeek                        : 0
DayOfYear                        : 0
DaysBits                         : 0
EveryNDays                       : 30
Hours                            : 0
Minutes                          : 0
MonthOfYear                      : 0
NextRetryTimeUTC                 : 3/23/2017 5:00:01 AM
NumberOfRetries                  : 0
Reboot                           : False
RetryEnabled                     : True
RunWindowMinutes                 : 0
ScheduleType                     : SCHEDULE_TYPE_N_DAYS
```

```
SchedulingPeriod                        : 0
UpdateNextRunTimeForPartialCompletion : False
```

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Non-existent JobID**

  The job could not be found.

PowerShell: Get-LSJobStatus

Get-LSJobStatus obtains met data for a particular job, such as current status..

Permissions Required

- Delegated permissions on the target job.

Related Commands

- **SOAP:** JobOps_GetJobStatus
- **REST:** Job (GET)

Syntax

```
Get-LSJobStatus [-AuthenticationToken] <string> [-JobID] <string> [-PassException] [-Trace] [-
RunAs <pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** The authentication token of the calling user.
- **JobID:** The ID for the job.

Example Request

```
Get-LSJobStatus -AuthenticationToken $tok - JobID 1
```

Output Success

A successful run returns metadata about the job.

Example Success Output

```
ExtensionData         : System.Runtime.Serialization.ExtensionDataObject
AssociatedGroup       : Default
Comment               : Management Set Update Job
CreatedBy             : lsds\lscadmin
CreationTimeUTC       : 12/20/2016 8:43:36 PM
JobID                 : 1
JobOperation          : UpdateManagementSet
JobType               : Interactive_Console
LastResult            : Incomplete_InProcess
LastRunTimeUTC        : 3/23/2017 5:00:01 AM
NextRunTimeUTC        : 3/23/2017 5:00:00 AM
PullSystemsFromGroup  : False
RefreshGroupBeforeRun : False
RunJobOnNewSystems    : False
```

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Non-existent JobID**

  The job could not be found.

PowerShell: Get-LSJobAccountElevationSettings

Get-LSJobAccountElevationSettings obtains the account elevation settings for a specific job..

Permissions Required

- Delegated permissions on the target job.

Related Commands

- **SOAP:** JobOps_GetJobElevationSettings
- **REST:** Job/WindowsElevation (POST)

Syntax

```
Get-LSJobAccountElevationSettings [-AuthenticationToken] <string> [-JobID] <string> [-
PassException] [-Trace] [-RunAs <pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** The authentication token of the calling user.
- **JobID:** The ID for the job.

Example Request

```
Get-LSJobAccountElevationSettings -AuthenticationToken $tok2 - JobID 68
```

Output Success

A successful run returns metadata about the job.

Example Success Output

```
ExtensionData              : System.Runtime.Serialization.ExtensionDataObject
AccountElevatedState       : REMOVED
AccountNameToElevate       : lsds\lscadmin
DomainElevationGroup       : Administrators
ElevateToDomainGlobalGroup : True
ElevationGroup             : Administrators
ExpirationEmailAddress     :
ExpirationEmailMinutes     : 20
ExpirationEmailSent        : False
MinutesBeforeRemoval       : 360
MinutesBeforeRemovalGlobal : 360
SendExpirationEmail        : False
SendFailureEmail           : False
SendSuccessEmail           : False
```

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Non-existent JobID**

  The job could not be found

- **Job is not an account elevation job**

  Information for the job could not be found.

PowerShell: Get-LSJobPasswordChangeSettings

Get-LSJobPasswordChangeSettings obtains the current status and other metadata of a particular job.

Permissions Required

- Delegated permissions on the target job.

Related Commands

- **SOAP:** JobOps_GetJobPasswordChangeSettings
- **REST:** Job/PasswordChange (GET)

Syntax

```
Get-LSJobPasswordChangeSettings [-AuthenticationToken] <string> [-JobID] <string> [-
PassException] [-Trace] [-RunAs <pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** The authentication token of the calling user.
- **JobID:** The ID for the job.

Example Request

```
Get-LSJobPasswordChangeSettings -AuthenticationToken $tok2 - JobID 18
```

Output Success

A successful run returns certain metadata about the job.

Example Success Output

```
ExtensionData                    : System.Runtime.Serialization.ExtensionDataObject
AccountComment                   :
AccountType                      : ACCOUNT_TYPE_CUSTOM_ACCOUNT
AddMissing                       : False
AddType                          : ACCOUNT_TYPE_USER
CancelIfCheckedOut               : False
ChangeLoginAccount               : False
ChangeRootAccount                : False
ChangeTwice                      : False
ClearAutoLogon                   : False
ConfigFile                       :
ConnectionType                   : SSH
CurrentPassword                  :
DisableAccountLockout            : False
DomainName                       : Azure Active Directory
EmailOnChange                    :
```

```
ExplicitPassword                  :
FirstCharacterSetBits             : 15
FullAccountName                   : James.Kirk@cloudstuff.onmicrosoft.com
HostCodePage                      : 1
KeepAccountLockedOutUntilComplete : False
KeyLabel                          :
LastCharacterSetBits              : 15
LoginName                         :
LoginPassword                     :
MiddleCharactersSetBits           : 15
MinLettersLcase                   : 0
MinLettersUcase                   : 0
MinNumbers                        : 0
MinSymbols                        : 0
NewAccountName                    :
PasswordChangeType                : PWD_CHANGE_TYPE_EXPLICIT
PasswordCharacterSetBits          : 15
PasswordCompatibilityLevel        : PWD_COMPAT_LAN_MANAGER
PasswordConstraints               :
LSClientAgentCommandlets.RouletteWebService.PasswordChangeConstraints
PasswordLength                    : 14
PasswordPropagationSettings       :
LSClientAgentCommandlets.RouletteWebService.PasswordPropagationSettings
PasswordPropagationTargets        :
LSClientAgentCommandlets.RouletteWebService.PasswordPropagationTargets
PasswordSecurityOptions           : 0
PasswordSegments                  : 1
PreventUsernameInPassword         : False
ReEnableAccountAfterSetTimeHours  : False
ReEnableAccountIfOperationFails   : False
RenameAccount                     : False
SendEmailOnChange                 : False
SerializedUtilityIDs              :
StoredAccountName                 :
StoredNamespace                   :
StoredSystemName                  :
SymbolsSetOverride                :
TerminalType                      : 0
Unique                            : True
UnlockAccount                     : False
UpdateAutoLogon                   : False
UpdatedAccountIsRootAccount       : False
UseSavedPasswords                 : False
UseStoredLoginPassword            : False
```

Additional Information

Additional information is stored in the following objects:

- **PasswordConstraints:** Password constraints such as filtered characters and relative positions.
- **PasswordPropagationSettings:** The propagation scope settings of the job.
- **PasswordPropagationTargets:** The sub-systems the job will attempt to propagate.

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Non-existent JobID**

  The job could not be found.

- **Job is not a password change job**

  Information for the job could not be found.

PowerShell: Get-LSJobPreAndPostRunSettings

Get-LSJobPreAndPostRunSettings obtains the pre and post-run job settings for a specific JobID.

Permissions Required

- Delegated permissions on the target job.

Related Commands

- **SOAP:** JobOps_GetPreAndPostRunSettings
- **REST:** Job/PreAndPostRun (GET)

Syntax

```
Get-LSJobPreAndPostRunSettings [-AuthenticationToken] <string> [-JobID] <string> [-PassException]
[-Trace] [-RunAs <pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** The authentication token of the calling user.
- **JobID:** The ID for the job.

Example Request

```
Get-LSJobPreAndPostRunSettings -AuthenticationToken $tok2 - JobID 68
```

Output Success

A successful run returns metadata about the job's pre and post-run operations..

Example Success Output

```
ExtensionData      : System.Runtime.Serialization.ExtensionDataObject
PostRunApplication : c:\utils\sdnutil.exe
PostRunArgs        : -Op Close -Targ vn-custx
PostRunExe         : True
PreRunAbortFail    : True
PreRunApplication  : c:\utils\sdnutil.exe
PreRunArgs         : -Op Open -Targ vn-custx
PreRunExe          : True
PreRunWait         : True
```

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Non-existent JobID**

  The job could not be found.

PowerShell: Get-LSJobSSHKeyChangeSettings

Get-LSJobSSHKeyChangeSettings obtains the current status and other metadata of a particular SSH key change job.

Permissions Required

- Delegated permissions on the target job.

Related Commands

- **SOAP:**JobsOps_GetJobKeyChangeSettings
- **REST:** Job/SSHKeyChange (GET)

Syntax

```
Get-LSJobSSHKeySettings [-AuthenticationToken] <string> [-JobID] <string> [-PassException] [-
Trace] [-RunAs <pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** The authentication token of the calling user.
- **JobID:** The ID for the job.

Example Request

- **Get-LSJobPasswordChangeSettings -AuthenticationToken $tok2** - JobID 18

Output Success

A successful run returns metadata about the job.

Example Success Output

```
ExtensionData                : System.Runtime.Serialization.ExtensionDataObject
DeleteKeyFileOnRemoteSystems : True
GenerateNewKeyEachRun        : True
KeyLabel                     : 2B:39:27:15:8C:17:61:32:59:9F:FC:04:A5:EB:B8:11
KeyLengthBits                : 4096
KeyType                      : 0
OldKeyLabel                  :
OldKeySig                    :
OldPublicKey                 :
RemoveOldKey                 : True
UpdateKeyReferences          : True
```

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Non-existent JobID**

  The job could not be found.

- **Job is not an SSH key change job**

  Information for the job could not be found.

PowerShell: Get-LSListJobMessagesForJob

Get-LSListJobMessagesForJob returns the logged operation messages for a job. These are not the verbose messages seen in the text log for specific job. These are general logs as seen in the **Logging** tab of the job in the management console.

Permissions Required

- Delegated permissions on the target job.

Related Commands

- **SOAP:** JobOps_GetJobLogForJob
- **REST:** Job/Logs (GET)

Syntax

```
Get-LSListJobMessagesForJob [-AuthenticationToken] <string> [-JobID] <string> [-StartWindow
<datetime>] [-EndWindow <datetime>] [-PassException] [-Trace] [-RunAs <pscredential>]
[<CommonParameters>]
```

Parameters

- **AuthenticationToken:** The authentication token of the calling user.
- **JobID:** The JobID of the job you are retrieving the messages for.
- **StartWindow:** *(Optional)* The start date and time of the date range.
- **EndWindow:** *(Optional)* The end date and time of the date range.

Parameter Help

The start and end time parameters may be entered in one of two ways:

- Parse normal date/time values: `[System.DateTime]::Parse("12/29/2016 12:42:00 AM")`
- Enter the time as follows (string): `"YYYY-DD-HHThh:mm:ss". For example: "2016-12-29T00:42:00"`

Example Request

The request accepts a start and end date for the audit logs. If neither date range is specified, the entire audit log history is returned.

No date range specified

```
Get-LSListJobMessagesForJob -AuthenticationToken $tok2 - JobID 68
```

Date range specified

```
#Create start date
$startdate = [System.DateTime]::Parse("12/29/2016 00:42:00 AM")
#Create end date
```

```
$enddate = [System.DateTime]::Parse("12/29/2016 00:46:00 AM")
#Get audit logs
Get-LSListJobMessagesForJob -AuthenticationToken $tok2 -JobID 68 -StartWindow $startdate -EndWindow
$enddate
```

Output Success

If the command is successful, the audit logs for the given date range are returned.

Example Success Output

```
ExtensionData   : System.Runtime.Serialization.ExtensionDataObject
InstanceName    : DBAG01
Message         : Account lsds\lscadmin has been removed from group Administrators on system DBAG01
OperationEntity : DBAG01
OperationLevel  : 2
TimeStamp       : 12/29/2016 12:46:07 AM
ExtensionData   : System.Runtime.Serialization.ExtensionDataObject
InstanceName    : DBAG02
Message         : Account lsds\lscadmin has been removed from group Administrators on system DBAG02
OperationEntity : DBAG02
OperationLevel  : 2
TimeStamp       : 12/29/2016 12:46:07 AM
ExtensionData   : System.Runtime.Serialization.ExtensionDataObject
InstanceName    : DBAG01
Message         : Account lsds\lscadmin has been temporarily elevated to group Administrators on
                  system DBAG01, elevation will expire in 360 minutes
OperationEntity : DBAG01
OperationLevel  : 2
TimeStamp       : 12/29/2016 12:42:02 AM
ExtensionData   : System.Runtime.Serialization.ExtensionDataObject
InstanceName    : DBAG02
Message         : Account lsds\lscadmin has been temporarily elevated to group Administrators on
                  system DBAG02, elevation will expire in 360 minutes
OperationEntity : DBAG02
OperationLevel  : 2
TimeStamp       : 12/29/2016 12:42:02 AM
```

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Invalid JobID**

  The job could not be found.

- **Improperly formatted date/time value**

  The value 'bad_string_value_goes_here' cannot be parsed as the type 'DateTime'.

PowerShell: Get-LSListSystemStatusForJob

Get-LSListSystemStatusForJob lists the status of the work performed against each system in the target job ID.

Permissions Required

- Delegated permissions on the target job.

Related Commands

- **SOAP:** JobOps_GetSystemStatusForJob
- **REST:** Job/System (GET)

Syntax

```
Get-LSListSystemStatusForJob [-AuthenticationToken] <string> [-JobID] <string> [-PassException]
[-Trace] [-RunAs <pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** The authentication token of the calling user.
- **JobID:** The ID of the job you retrieving log messages for.

Example Request

```
Get-LSListSystemStatusForJob -AuthenticationToken $tok2 - JobID 68
```

Output Success

The output is the status of the work performed against each system in the target job ID; one entry per system.

Example Success Output

```
ExtensionData       : System.Runtime.Serialization.ExtensionDataObject
AccountStoreName    : DBAG01
LastRunTime         : 12/28/2016 6:46:07 PM
SystemJobLastResult : WORK_COMPLETE
SystemName          : DBAG01
ExtensionData       : System.Runtime.Serialization.ExtensionDataObject
AccountStoreName    : DBAG02
LastRunTime         : 12/28/2016 6:46:07 PM
SystemJobLastResult : WORK_COMPLETE
SystemName          : DBAG02
```

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Non-existent JobID**

  The job could not be found.

PowerShell: New-LSJobAccountElevation

New-LSJobAccountElevation creates a new account elevation job.

Permissions Required

- Elevate Any Account

Related Commands

- **SOAP:** JobOps_CreateAccountElevationJon
- **REST:** Job/WindowsElevation (POST)

Syntax

```
New-LSJobAccountElevation [-AuthenticationToken] <string> [-SystemName] <string> [-AccountName]
<string> [-GroupName] <string> [-ExpirationEmail] <string> [-ElevationDuration] <int> [-RunNow]
[-ElevateToDomainGroup] [-SendExpirationEmail <bool>] [-SendSuccessEmail <bool>] [-
SendFailureEmail <bool>] <string> [-PassException] [-Trace] [-RunAs <pscredential>]
[<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.
- **AccountName:** Target account to elevate. (Format should be DomainName\UserName)
- **ElevationDuration:** Amount of time in minutes to elevate the target account.
- **ExpirationEmail:** Email address to send the elevation expiration notice to.
- **GroupName:** Target group to add the target account to.
- **SystemName:** Target system hosting the group the target account will be added to.
- **ElevateToDomainGroup:** *(Optional)* Include if the target account is being added to a global security group rather than a domain local group and if the target system is a domain controller.
- **SendExpirationEmail:** *(Optional)* Set to **$True** to send an email prior to account de-elevation.
- **SendFailureEmail:** *(Optional)* Set to **$True** to send an email to the expiration email address if the account elevation fails.
- **SendSuccessEmail:** *(Optional)* Set to **$True** to send an email to the expiration email address if the account elevation succeeds.

Example Requests

Minimal Request to a [Domain] Local Group

```
New-LSJobAccountElevation - AccountName lsds\fred -AuthenticationToken $tok -ElevationDuration
240 -ExpirationEmail fred@lsds.int -GroupName "Backup Operators" -SystemName dbsmash2008
```

Minimal request to a Global Security Group

```
New-LSJobAccountElevation - AccountName lsds\fred -AuthenticationToken $tok -ElevationDuration
240 -ExpirationEmail fred@lsds.int -GroupName "SQL Server Admins" -SystemName lsds.int -
```

```
ElevateToDomainGroup
```

Minimal Request to a [Domain] Local Group sending Email Notifications

```
New-LSJobAccountElevation - AccountName lsds\fred -AuthenticationToken $tok -ElevationDuration
240 -ExpirationEmail fred@lsds.int -GroupName "Backup Operators" -SystemName dbsmash2008 -
SendExpirationEmail $true -SendFailureEmail $true -SendSuccessEmail $true
```

Output Success

The output message includes the JobID.

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Elevation duration too large**

  Value was either too large or too small for an Int32.

- **Elevation duration was longer than maximum allowed elevation duration**

  Elevation job could not be create, or the levation duration is longer than the maximum allowed.

PowerShell: New-LSJobAddSystem

New-LSJobAddSystem adds a system to an existing job.

Permissions Required

- Delegated permissions on the job.

Related Commands

- **SOAP:** JobOps_AddSystemToJob
- **REST:** Job/System (PUT)

Syntax

```
New-LSJobAddSystem [-AuthenticationToken] <string> [-JobID] <string> [-SystemName] <string> [-
PassException] [-Trace] [-RunAs <pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.
- **JobID**: The JobID of the job you are adding to the system to.
- **SystemName:** The name of the system you are adding to the job.

Example Request

```
New-LSJobAddSystem -AuthenticationToken $tok -JobID 1259 - SystemName lscw-2012
```

Output Success

The output message indicates system was added to the job.

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Invalid JobID**

  The job could not be found.

PowerShell: New-LSJobClone

New-LSJobClone create a duplicate job based off of a source job.

Permissions Required

- All Access

Related Commands

- SOAP: JobOps_CloneJob
- REST: Job/Clone (POST)

Syntax

```
New-LSJobClone [-AuthenticationToken] <string> [-JobID] <string> [-PassException] [-Trace] [-
RunAs <pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.
- **JobID:** The ID of the job you are adding the system to.

Example Request

```
New-LSJobClone -AuthenticationToken $tok - JobID 1259
```

Output Success

The output message provides the JobID..

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Invalid JobID**

  The job could not be found.

PowerShell: New-LSJobRefreshAndDiscoveryIPMI

New-LSJobRefreshAndDiscoveryIPMI scans a target IPMI device and retrieves a list of accounts from the device.

Permissions Required

- All Access

Related Commands

- **SOAP:** JobOps_CreateRefreshIPMISystemJob
- **REST:** Job/RefreshIPMI (POST)

Syntax

```
New-LSJobRefreshAndDiscoveryIPMI [-AuthenticationToken] <string> [-SystemName] <string> [-RunNow]
[-PassException] [-Trace] [-RunAs <pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.
- **SystemName:** The target IPMI device to scan.
- **RunNow:** *(Optional)* When used, the job will be set to run in the next minute.

Example Request

```
New-LSJobRefreshAndDiscoveryIPMI -AuthenticationToken $tok - SystemName 192.168.1.151
```

Output Success

The output message states that a job was created.

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

PowerShell: New-LSJobSSHKeyChange

New-LSJobSSHKeyChange creates a new SSH key update job.

Permissions Required

- All Access

Related Commands

- **SOAP:** JobOps_CreateKeyChangeJob
- **REST:** Job/SSHKeyChange (POST)

Syntax

```
New-LSJobSSHKeyChange [-AuthenticationToken] <string> [-KeyLabel] <string> [-KeyLength] <int> [[-
UpdateReferences] <bool>] [[-RemoveOldKey] <bool>] [[-GenerateNewKeyEachRun] <bool>] [[-
RemoveOldKeyFiles] <bool>] [-RunNow] [-PassException] [-Trace] [-RunAs <pscredential>]
[<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.
- **KeyLabel:** The label of the SSH key to update.
- **KeyLength:** The length of the newly generated key. Values must be set to the valid length for the key type. If an invalid key length is set, the default value of 2048 bits is used.
- **UpdateReferences:** Set to **$True** to update SSH key files on target systems by adding new key reference.
- **RemoveOldKey:** Set to **$True** to remove old SSH key references from target systems.
- **GenerateNewKeyEachRun:** When set to **$True**, a new key is generated, stored, and updated in the solution database for every job run and does not perform any subsequent updates to target systems.
- **RemoveOldKeyFiles:** Set to **$True** to delete previous SSH keys left behind on target systems.

Example Request

```
New-LSJobSSHKeyChange -AuthenticationToken $tok -KeyLabel dbsmashlnx -KeyLength 4096 -
UpdateReferences $true -RemoveOldKey $true -GenerateNewKeyEachRun $true - RemoveOldKeyFiles $true
```

Output Success

The output message provides the JobID.

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

PowerShell: New-LSJobWindowsChangeAdministratorPassword

New-LSJobWindowsChangeAdministratorPassword creates a new password change job targeting the built-in Window administrator account (RID500). New jobs receive all default password generation settings.

Permissions Required

- All Access

Syntax

```
New-LSJobWindowsChangeAdministratorPassword [-AuthenticationToken] <string> [-SystemName]
<string> [-PasswordLength] <int> [-RunNow] [-PassException] [-Trace] [-RunAs <pscredential>]
[<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.
- **SystemName:** The name of the target system.
- **PasswordLength:** The length of the new random password. The maximum length is 127 characters.

Example Request

```
New-LSJobWindowsChangeAdministratorPassword -AuthenticationToken $tok -SystemName lsdslscprd -
PasswordLength 20
```

Output Success

The output message provides the Job ID.

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

PowerShell: New-LSJobWindowsChangePassword

New-LSJobWindowsChangePassword creates a new password change job targeting a Windows account. New jobs receive all default password generation settings.

If you are looking to change passwords for account types other than Windows, such as Linux or Oracle, start with this cmdlet, and then use "Set-LSJobPasswordChangeSettings" on page 1 to format the job for non-Windows platforms.

Permissions Required

- All Access

Syntax

```
New-LSJobWindowsChangePassword [-AuthenticationToken] <string> [-SystemName] <string> [-
AccountName] <string> [-CreateAccountIfNotFound] <bool> [-PasswordLength] <int> [[-MinutesDelay]
<int>] [-RunNow] [-PassException] [-Trace] [-RunAs <pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.
- **SystemName:** The name of the target system.
- **AccountName:** The name of the target account account.
- **CreateAccountIfNotFound:** Set to $true to create the account of the account name is not found.
- **PasswordLength:** The length of the new random password. The maximum length is 127 characters.

Example Request

```
New-LSJobWindowsChangePassword -AuthenticationToken $tok -SystemName lsdslscprd -AccountName
firecall -CreateAccountIfNotFound $False - PasswordLength 20
```

Output Success

The output message provides the JobID.

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

PowerShell: New-LSJobWindowsRefreshAndDiscovery

New-LSJobWindowsRefreshAndDiscovery creates a job to refresh target Windows systems and discovers local accounts and account usage.

Permissions Required

- All Access

Related Commands

- **SOAP:** JobOps_CreateRefreshWindowsSystemAndUsageJob
- **REST:** Job/RefreshAndDiscoverWindows (POST)

Syntax

```
New-LSJobWindowsRefreshAndDiscovery [-AuthenticationToken] <string> [-SystemName] <string> [-
RunNow] [-PassException] [-Trace] [-RunAs <pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.
- **SystemName:** The name of the target system..

Example Request

```
New-LSJobWindowsRefreshAndDiscovery -AuthenticationToken $tok -SystemName lsdslscprd - RunNow
```

Output Success

The output message provides the Job ID.

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

PowerShell: Remove-LSJob

Remove-LSJob deletes a job.

Permissions Required

- All Access

Related Commands

- **SOAP:** JobOps_DeleteJob
- **REST:** Job (DELETE)

Syntax

```
Remove-LSJob [-AuthenticationToken] <string> [-JobID] <string> [-PassException] [-Trace] [-RunAs
<pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.
- **JobID:** The ID of the job needing to be deleted.

Example Request

```
Remove-LSJob -AuthenticationToken $tok - JobID 1339
```

Output Success

The output message states the job was successfully removed.

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Invalid JobID**

  The job could not be found.

PowerShell: Remove-LSJobSystem

Remove-LSJobSystem removes a system from a job.

Permissions Required

- Delegated control of the job

Related Commands

- **SOAP:** JobOps_RemoveSystemFromJob
- **REST:** Job/Syste, (DELETE)

Syntax

```
Remove-LSJobSystem [-AuthenticationToken] <string> [-JobID] <string> [-SystemName] <string> [-
PassException] [-Trace] [-RunAs <pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.
- **JobID:** The ID of the target job.
- **SystemName:** The system to remove from the job.

Example Request

```
Remove-LSJobSystem -AuthenticationToken $tok -JobID 1339 - SystemName lsdslscprd
```

Output Success

The output message the system was successfully removed.

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Invalid JobID**

  The job could not be found.

PowerShell: Set-LSJobAccountElevationSettings

Set-LSJobAccountElevationSettings updates the elevation settings of an elevation job.

Permissions Required

- Delegated control of the job.

Related Commands

- **SOAP:** JobOps_SetJobElevationSettings
- **REST:** Job/WindowsElevation

Syntax

```
Set-LSJobAccountElevationSettings [-AuthenticationToken] <string> [-JobID] <string> [-
ElevationSettings] <ElevationSettings> [-PassException] [-Trace] [-RunAs <pscredential>]
[<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.
- **JobID:** TheID of the target job to be updated.
- **ElevationSettings:** The updated account elevation settings.
    - **AccountElevatedState:** The current state of the account to elevate or de-elevate. Valid states are:
        - NOT_ELEVATED
        - ELEVATED
        - REMOVED
    - **AccountNameToElevate:** The target account needing to be elevated.
    - **DomainElevationGroup:** The target group to elevate to when targeting a domain controller.
    - **ElevateToDomainGlobalGroup:** Set to `$true` if the target group is a global security group, or set to `$false` if targeting a [domain] local group.
    - **ElevationGroup:** The target [domain] local group if `ElevateToDomainGlobalGroup` is `$false`.
    - **ExpirationEmailAddress:** The email address to notify of impending elevation expiration. Also configure `ExpirationEmailMinutes`.
    - **ExpirationEmailMinutes:** The number of minutes prior to elevation expiration to send the expiration email.
    - **ExpirationEmailSent:** Set to `$false` to indicate the expiration email has not been sent.
    - **MinutesBeforeRemoval:** The number of minutes to elevate the account into a [domain] local group.
    - **MinutesBeforeRemovalGlobal:** The number of minutes to elevate the account into a global security group.
    - **SendExpirationEmail:** Set to `$true` to send an elevation expiration email.
    - **SendFailureEmail:** Set to `$true` to send an email to `ExpirationEmailAddress` that the elevation/de-elevation failed.
    - **SendSuccessEmail:** Set to `$true` to send an email to `ExpirationEmailAddress` that the elevation/de-elevation succeeded.

Example Request

```
#create the object and define the account elevation settings
$eSettings = New-Object -TypeName LSClientAgentCommandlets.RouletteWebService.ElevationSettings
$eSettings.AccountElevatedState = "NOT_ELEVATED"
$eSettings.AccountNameToElevate = "lsc\bob"
$eSettings.DomainElevationGroup = ""
$eSettings.ElevateToDomainGlobalGroup = $false
$eSettings.ElevationGroup = "Administrators"
$eSettings.ExpirationEmailAddress = "bob@lsds.int"
$eSettings.ExpirationEmailMinutes = 20
$eSettings.ExpirationEmailSent = $false
$eSettings.MinutesBeforeRemoval = 360
$eSettings.MinutesBeforeRemovalGlobal = 360
$eSettings.SendExpirationEmail = $true
$eSettings.SendFailureEmail = $true
$eSettings.SendSuccessEmail = $true
Set-LSJobAccountElevationSettings -JobID 1284 -ElevationSettings $eSettings -AuthenticationToken
$tok
```

Output Success

The output message states the job was successfully updated.

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Invalid JobID or Job is not an account elevation job**

  The job could not be found.

PowerShell: Set-LSJobAccountElevationExtension

Set-LSJobAccountElevationExtension changes the de-elevation time, thereby extending or minimizing the elevation time on an account elevation job.

Permissions Required

- Elevate any account

Related Commands

- **SOAP:** JobOps_SetJobElevationExtension
- **REST:** Job/WindowsElevation/Extend (POST)

Syntax

```
Set-LSJobAccountElevationExtension [-AuthenticationToken] <string> [-JobID] <string> [-
ElevationExtension] <ElevationExtension> [-PassException] [-Trace] [-RunAs <pscredential>]
[<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.
- **JobID:** The ID of the target elevation job.
- **ElevationExtension:** The new escalation duration or de-elevation time.

Example Request

Set a Specific Date and Time for De-Elevation

```
#Create ElevationExtension object
$EEO = New-Object -TypeName LSClientAgentCommandlets.RouletteWebService.ElevationExtension
$EEO.ExpirationUTC = Get-Date -Date 2017-06-02T19:35:00
$EEO.ExtensionMinutes = 0
Set-LSJobAccountElevationExtension -AuthenticationToken $tok -JobID 1243 -ElevationExtension $EEO
```

Add More Time (Minutes) from Now

```
#Create ElevationExtension object
$EEOX = New-Object -TypeName LSClientAgentCommandlets.RouletteWebService.ElevationExtension
$EEOX.ExtensionMinutes = 120
Set-LSJobAccountElevationExtension -AuthenticationToken $tok -JobID 1243 -ElevationExtension $EEOX
```

Output Success

The output message states the job was updated.

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Invalid JobID**

  The job could not be found.

- **Elevation duration too large**

  Value was either too large or too small for an Int32.

- **Elevation duration was longer than maximum allowed elevation duration**

  Elevation job could not be created. The elevation duration is longer than the maximum allowed.

PowerShell: Set-LSJobComment

Set-LSJobComment replaces an existing job's comment or sets a new comment for a job. This comment is visible in the web application and management console.

Permissions Required

- Delegated control of the job.

Related Commands

- **SOAP:** JobOps_SetJobComment
- **REST:** Job/Comment (PUT)

Syntax

```
Set-LSJobComment [-AuthenticationToken] <string> [-JobID] <string> [-Comment] <string> [-
PassException] [-Trace] [-RunAs <pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.
- **JobID:** The ID of the target job.
- **Comment:** The job comment you wish to add..

Example Request

```
Set-LSJobComment -JobID 1284 - Comment "Recurring local admin password spin" -AuthenticationToken
$tok
```

Output Success

The output message states the job was successfully updated.

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Invalid JobID**

  The job could not be found.

PowerShell: Set-LSJobPasswordChangeSettings

Set-LSJobPasswordChangeSettings allows full [re-]configuration of an existing password change job. For example, if a password change job was initially created as a Windows password change job or did not originally have propagation settings, this cmdlet is used to reconfigure the job as a Linux or Oracle password change job and to add propagation settings.

Permissions Required

- Delegated control of the job.

Related Commands

- **SOAP:** JobOps_SetJobPasswordChangeSettings
- **REST:** Job/PasswordChange (PUT)

Syntax

```
Set-LSJobPasswordChangeSettings [-AuthenticationToken] <string> [-JobID] <string> [-
PasswordChangeSettings] <PasswordChangeSettings> [-PassException] [-Trace] [-RunAs
<pscredential>] [<CommonParameters>]
```

Parameters

Set-LSPasswordChangeSettings has multiple options. Some of these options are enumerated values and list types. To aid in the description of the available parameters, the parameters are divided into their respective sections.

InputArgs

- **AuthenticationToken:** Authentication token of the calling user.
- **JobID:** The ID of the password change job to be updated.

InputArgs\PasswordChangeSettings

- **AccountComment:** *(Optional)* The comment for the target managed account. This is visible in the web application.
- **AccountType:** *(Optional)* For Windows password change jobs, this value identifies if you are targeting a Windows systems' built-in administrator, built-in guest, a regular user account, or if the job is targeting another platform such as SQL Server or IPMI. Valid values are:
  - ACCOUNT_TYPE_USER
  - ACCOUNT_TYPE_ADMINISTRATOR - set `FullAccountName` to `*Administrator`.
  - ACCOUNT_TYPE_GUEST - set `FullAccountName` to `*Guest`.
  - ACCOUNT_TYPE_SQLSERVER_SA_ACCOUNT
  - ACCOUNT_TYPE_LINUX_ACCOUNT
  - ACCOUNT_TYPE_CISCO_ROUTER
  - ACCOUNT_TYPE_AS400_ACCOUNT
  - ACCOUNT_TYPE_UNIX_ACCOUNT
  - ACCOUNT_TYPE_MYSQL_ACCOUNT

    - ACCOUNT_TYPE_ORACLE_ACCOUNT
    - ACCOUNT_TYPE_CUSTOM_ACCOUNT
    - ACCOUNT_TYPE_LDAP
    - ACCOUNT_TYPE_SYBASE
    - ACCOUNT_TYPE_OS390_ACCOUNT
    - ACCOUNT_TYPE_DRAC
    - ACCOUNT_TYPE_IPMI
    - ACCOUNT_TYPE_3270_ACCOUNT
    - ACCOUNT_TYPE_DSRM
- **AddMissing:** *(Optional)* For a Windows password change job where the `AccountType` is set to `ACCOUNT_TYPE_USER`, if the user does not exist, it can be added if the value is set to true.
- **AddType:** *(Optional)* For Windows password change jobs, set any of the following values if you are creating the target account. If it is missing, (`AddMissing` must be set to `true`). This setting defines what group the missing user will be placed into on the target machine. Valid values are:
    - ACCOUNT_TYPE_GUEST
    - ACCOUNT_TYPE_USER
    - ACCOUNT_TYPE_ADMINISTRATOR
- **CancelIfCheckedOut:** *(Optional)* If set to **True**, the job will not run if the password is currently checked out to a user.
- **ChangeLoginAccount:** *(Optional)* For SSH-based jobs, set the option to change the login account when set to `true`.
- **ChangeRootAccount:** *(Optional)* For SSH-based jobs, set the option login account is root when set to `true`.
- **ChangeTwice:** *(Optional)* For Windows password change jobs, will spin the password for the target account twice when set to `true`.
- **ClearAutoLoginAccount:** *(Optional)* For Windows password change jobs, will remove the any configured automatic login account when set to `true`.
- **ConfigFile:** *(Optional)* This is used for database instance names and for SSH/Telnet-based jobs, this defines the name (and possibly the path) for configuration response file to use for the password change process. For database jobs, this specifies the database instance/service name.
- **ConnectionType:** *(Optional)* For SSH/Telnet jobs, set the value to either `SSH` or `TELNET`.
- **CurrentPassword:** *(Optional)* For SSH/Telnet jobs, this is the password for the target account, if needed.
- **DisableAccountLockout:** Not used during job creation.
- **DomainName:** Not used during job creation.
- **EmailOnChange:** *(Optional)* Will email the clear text password to the target email address when `SendEmailOnChange` is set to `true`.
- **ExplicitPassword:** *(Optional)* Define the password to set on the target account if setting a static password.
- **FirstCharacterSetBits:** Defines the valid characters for the first character position. Values are cumulative, e.g. a value of 15 enable all possible character types. Possible values are:
    - **1** = include upper case letters
    - **2** = include lower case letters
    - **4** = include numbers
    - **8** = include symbols
- **FullAccountName:** Supply the name of the target account. If running against the Windows built-in administrator or built-in guest, set the name to `*Administrator` or `*Guest respectively`.

- **HostCodePage:** Not used during job creation.
- **KeepAccountLockedOutUntilComplete:** *(Optional)* For Windows and Oracle database jobs, when UnlockAccount is set to `true`, this will clear the account lockout flag of the target account AFTER the password change and propagation completes when set to `true`. If not defined or set to false, the account will be unlocked as soon as the password change job begins.
- **KeyLabel:** *(Optional)* For SSH jobs, this identifies the SSH key to use for authentication.
- **LastCharacterSetBits:** Defines the valid characters for the last character position. Values are cumulative, e.g. a value of 15 enable all possible character types. Possible values are:
    - **1** = include upper case letters
    - **2** = include lower case letters
    - **4** = include numbers
    - **8** = include symbols
- **LoginName:** *(Optional)* For target systems that require a named login account, specify the name of the login account, such as SSH, Telnet, IPMI, jobs, etc.
- **LoginPassword:** *(Optional)* Define the static login password for `LoginName` when the option `UseSavedPasswords` is set to `false`.
- **MiddleCharactersSetBits:** Defines the valid characters for the middle character position. Values are cumulative, e.g. a value of 15 enable all possible character types. Possible values are:
    - **1** = include upper case letters
    - **2** = include lower case letters
    - **4** = include numbers
    - **8** = include symbols
- **MinLettersLcase:** Define the minimum number of lower case letters.
- **MinLettersUcase:** Define the minimum number of upper case letters.
- **MinNumbers:** Define the minimum number of numbers.
- **MinSymbols:** Define the minimum number of symbols.
- **NewAccountName:** *(Optional)* For Windows password change jobs targeting the built-in administrator or guest, this defines the new name for the account.
- **PasswordChangeType:** Valid values are:
    - **PWD_CHANGE_TYPE_GEN_RANDOM:** Set a random password.
    - **PWD_CHANGE_TYPE_EXPLICIT:** Set a static password.
- **PasswordCharacterSetBits:** Defines the valid characters for all character positions. Values are cumulative, e.g. a value of 15 enable all possible character types. Possible values are:
    - **1** = include upper case letters
    - **2** = include lower case letters
    - **4** = include numbers
    - **8** = include symbols
- **PasswordCompatibilityLevel:**Valid values are:
    - **PWD_COMPAT_LAN_MANAGER:** Sets LanMan compatible password constraints.
    - **PWD_COMPAT_NT4:** Sets NT4 compatible password constraints.
    - **PWD_COMPAT_W2K:** Sets Windows 2000 and later compatible password constraints.

- **PasswordLength:** *(Optional)* The desired length for a random password. Use when setting a random password. The minimum length is 3 characters and the maximum length is limited based on `PasswordCompatibilityLevel` configuration. Maximum values are:
  - **14:** When set to PWD_COMPAT_LAN_MANAGER or PWD_COMPAT_NT4.
  - **127:** When set to PWD_COMPAT_W2K.
- **PasswordSecurityOptions:** *(Optional)* Possible values are:
  - **1** = symbol in middle
  - **2** = no repeated characters
  - **3** = both symbol in middle and no repeated characters
- **PasswordSegments:** Defines how many segments the password will be broken into for later retrieval. Set to 1 store the password as 1 segment, meaning only one identity will be required to retrieve the whole password.
- **PreventUsernameInPassword:** For random passwords, set the value to true to ensure the username does not appear anywhere in a random password (statistically improbable!).
- **ReEnableAccountAfterSetTimeHours:** Not used.
- **ReEnableAccountIfOperationFails:** Not used.
- **RenameAccount:** *(Optional)* For Windows password change jobs, set to true to rename the target account and define `NewAccountName`.
- **SendEmailOnChange:** *(Optional)* When set to true, this will send the password in clear text via email to the email address defined in `EmailOnChange`.
- **SerializedUtilityIDs:** For SSH/Telnet jobs, these are the IDs of the utility accounts that may be used as tertiary login credentials during the password change job. Multiple IDs are separated by a semi-colon, for example `"1062;1064;15"`. The IDs are translated into utility account IDs in the answer file based on the order they are entered here. In the example above, `1062` would be `utilityAccount_1`.
- **StoredAccountName:** *(Optional)* For non-Windows password change jobs that will use a managed (and central account, e.g. from a directory), to login and change the target account. You must also specify `StoredNameSpace` and `StoredSystemName`. `UseStoredLoginPassword` must also be set to `true`.
- **StoredNamespace:** *(Optional)* For non-Windows password change jobs that will use a managed (and central account, e.g. from a directory), to login and change the target account. You must also specify `StoredAccountName` and `StoredSystemName`. `UseStoredLoginPassword` must also be set to `true`.
- **StoredSystemName:** *(Optional)* For non-Windows password change jobs that will use a managed (and central account, e.g. from a directory), to login and change the target account. You must also specify `StoredNameSpace` and `StoredAccountName`. `UseStoredLoginPassword` must also be set to `true`.
- **SymbolsSetOverride:** *(Optional)* When setting a random password, if desired, define the allowed special symbols for the random password. If not defined, all symbols will be allowed.
- **TerminalType:** Not used during job creation.
- **Unique:** Set to `true` to define the target account will get a unique random password, should multiple systems be defined in `SystemsList`. If set to `false` or not included and a random password is being set and multiple systems are included in `SystemsList`, the target account's password will be set the same across all target systems. Further, the account will be opted out of password re-randomization following password retrieval via the web application.
- **UnlockAccount:** *(Optional)* For Windows and Oracle database jobs, this will clear the account lockout flag of the target account when set to true.
- **UpdateAutoLogon:** *(Optional)* For Windows password change jobs, this will set the current account to be the automatic login account for the target systems.
- **UpdatedAccountIsRootAccount:** *(Optional)* For SSH/Telnet jobs, set to `true` when the target account is a root account.

- **UseSavedPasswords:** *(Optional)* For jobs that define a login account on the job, set to `true` when it is desired to use the stored password for the account. Set to `false`, when the password defined in the job, `LoginPassword`, should be used instead of any stored password.
- **UseStoredLoginPassword:** *(Optional)* For jobs that must use a login account on the job, set to true when it is desired to use a managed credential for the login account. You must also define `StoredAccountName`, `StoredNameSpace`, and `StoredSystemName`.

InputArgs\PasswordChangeSettings\PasswordConstraints

Many items are derived from the **PasswordChangeSettings** previously defined. Listed below are the new items for which there is no duplicate **PasswordChangeSettings** element.

- **DefaultPasswordFilterCompliance:** Not used.
- **FailGenerationOnMissingPassfiltDLL:** *(Optional)* Set to `false` to avoid failing the job if a custom password filter is not defined or unavailable.
- **PathToPassfiltDLL:** *(Optional)* Set to empty value to avoid system trying to use a custom passfilt.dll password filter. Otherwise, define the absolute path the the custom password filter.
- **SymbolsExcludeProblematicWithAPIs:** *(Optional)* Set to `true` to avoid using symbols known to be problematic with scripts and APIs. These symbols include: `/\:;"'`
- **SymbolsExcluded** *(Optional)* Define symbols to exclude from password change jobs.

InputArgs\PasswordChangeSettings\PasswordPropagationSettings

PasswordPropagationSettings defines the scope of propagation. In other words, what systems will be targeted for password propagation once the password change is made successfully.

- **ConstrainToManagedSystems:** *(Optional)* Set to `true` to limit the scope of propagation to only systems that are managed by Privileged Identity.
- **ConstrainToMembersOfGroup:** *(Optional)* Set to `true` to limit propagation scope to the systems in a specific management set. You must also define the `GroupName`.
- **ConstrainToSystemsWithNonzeroInUse:** Not used.
- **ExcludeDomainControllers:** *(Optional)* Set to `true` to avoid attempting propagation of the new password to domain controllers which may otherwise be included in the propagation scope. `ExcludeSystemWithAccount` must also be set to `true`.
- **ExcludeSystemWithAccount:** *(Optional)* Set to true to avoid scanning of and attempted propagation to the system where the password was changed.
- **GroupName:** *(Optional)* If `ConstraintoMembersOfGroup` is set to `true`, define the management set to limit propagation scope to.
- **PropagateToSystemWithAccountOnly:** *(Optional)* Set to `true` to scan only the system where the account password was updated. E.g. a local system account where only the local system uses the account.
- **PropagateToTrustingDomains:** *(Optional)* Set to `true` to cause Privileged Identity to enumerate all trusting domains and attempt scanning and propagating to those trusting systems.

InputArgs\PasswordChangeSettings\PropagationTargets\ListTargets

This defines what sub-systems to propagate to such as Windows Services, Scheduled Tasks, etc.. Create zero or more repetitions of `<PasswordPropagationTarget>` for each target sub-system. Each propagation target will be wrapped in a `<PasswordPropagationTarget>` tag.

- **ConfigurationData:** *(Optional)* This data varies by target. See "PowerShell: PropagationTargets ConfigurationsData" on page 68 for more information on each propagation target type.
- **DescriptiveName:** *(Optional)* A friendly name for the propagation type.
- **Enabled:** *(Optional)* Set to `true` to enable the propagation type for the job.
- **PasswordChangeJobID:** Not used.
- **RestrictBySystemSet:** *(Optional)* Set to true to limit the propagation type's scope to a specific list of systems. This is useful to ensure a certain type of propagation found on only a subset of systems included in the job's propagation scope are checked for a specific type of propagation. For example, if a job's propagation scope encompasses 1,000 systems, but only 10 of those systems run SharePoint, setting this option and defining `SystemSet` would configure the SharePoint propagation type to scan only those 10 systems if they were in their own management set.
- **SystemSet:** *(Optional)* Define the name of a management set when `RestrictBySystemSet` is set to `true`.
- **TargetSystemType_Linux:** *(Optional)* Set to `true` to enable this propagation type for Linux systems (systems under the Linux/Unix node) included in the job's propagation scope.
- **TargetSystemType_Windows:** *(Optional)* Set to `true` to enable this propagation type for Windows systems included in the job's propagation scope.
- **TypeName:** *(Optional)* If configuring propagations, this value must be defined. Valid values are:
    - **builtin:WindowsServices:** Windows services.
    - **builtin:WindowsScheduler:** Windows scheduled tasks.
    - **builtin:WindowsSchedulerAtAccount:** Windows AT identity.
    - **builtin:COMPlus:** Windows COM.
    - **builtin:DCOM:** Windows DCOM.
    - **builtin:IIS6Metabase:** Windows IIS6 (anonymous, app pool, network credentials).
    - **builtin:IIS7ConfigFiles:** Windows IIS7 and later (anonymous, app pool, network credentials).
    - **builtin:SCOM:** Microsoft SCOM RunAs accounts.
    - **builtin:SqlServer:** SQL Server Credentials (not to be confused with SQL Server Logins).
    - **builtin:NetConfig:** IIS asp.net connection strings.
    - **builtin:ReplaceInFiles:** String replacement within files.
    - **builtin:RunProcess:** Run an arbitrary process.
    - **builtin:Sharepoint:** Microsoft SharePoint server.
    - **builtin:IBM WebSphere Application Server:** IBM WebSphere Server.
    - **builtin:Oracle WebLogic Server:** Oracle Web Logic Server.
    - **builtin:SAP Server:** SAP.
    - **builtin:UpdateLogonCache:** Windows logon cache.
    - **builtin:UpdateAutoLogon:** Windows automatic logon account.
    - **builtin:SQLReportingServices:** Microsoft SQL Reporting Services Action Account.

Example Request

Change Password Settings

```
#Get authentication token
$tok = get-lslogintoken
```

```
#Get ALL current settings for job 1402, we are only adding propagation settings
$jid = 1402
$js = Get-LSJobPasswordChangeSettings -JobID $jid -AuthenticationToken $tok
#Change the account type to Oracle, reset the target name, set the database service name
$js.CancelIfCheckedOut = $true
$js.FirstCharacterSetBits = 15
$js.LastCharacterSetBits = 15
$js.MiddleCharactersSetBits = 15
$js.MinLettersLcase = 1
$js.MinLettersUcase = 1
$js.MinNumbers = 1
$js.MinSymbols = 1
$js.PasswordChangeType = "PWD_CHANGE_TYPE_GEN_RANDOM"
$js.PasswordCharacterSetBits = 15
$js.PasswordCompatibilityLevel = "PWD_COMPAT_W2K"
$js.PasswordLength = 32
$js.PasswordSecurityOptions = 3
$js.PasswordSegments = 2
$js.PreventUsernameInPassword = $true
#Set password constraints not set with primary object
$js.PasswordConstraints = New-Object -TypeName
LSClientAgentCommandlets.RouletteWebService.PasswordChangeConstraints
$Js.PasswordConstraints.FailGenerationOnMissingPassfiltDLL = $false
$js.PasswordConstraints.SymbolsExcludeProblematicWithAPIs = $true
#Update the job with the new settings
Set-LSJobPasswordChangeSettings -AuthenticationToken $tok -JobID $jid -PasswordChangeSettings $js
```

Change Job Type to an Oracle Password Change Job

```
#Get ALL current settings for job 1402, we are only adding propagation settings
$jid = 1402
$js = Get-LSJobPasswordChangeSettings -JobID $jid -AuthenticationToken $tok
#Change the account type to Oracle, reset the target name, set the database service name
$js.AccountType = "ACCOUNT_TYPE_ORACLE_ACCOUNT"
$js.FullAccountName = "sys"
$js.ConfigFile = "orcl.lsc.ent"
#Configure the login account for the target system - assuming previously managed [common] account
$js.LoginName = "erpmrpmdb"
$js.UseSavedPasswords = $true
#Update the job with the new settings
Set-LSJobPasswordChangeSettings -AuthenticationToken $tok -JobID $jid -PasswordChangeSettings $js
```

Add Propagation Settings to an Existing Job

```
#Get ALL current settings for job 1304, we are only adding propagation settings
$jid = 1304
$js = Get-LSJobPasswordChangeSettings -JobID $jid -AuthenticationToken $tok
#Create new job settings object and configure it to be equivalent to original job settings
$nPCS = New-Object -TypeName LSClientAgentCommandlets.RouletteWebService.PasswordChangeSettings
$nPCS = $js
#Set new objects password constraints equal to old job's password constraints
$nPCS.PasswordConstraints = New-Object -TypeName
```

```
LSClientAgentCommandlets.RouletteWebService.PasswordChangeConstraints
$nPCS.PasswordConstraints = $js.PasswordChangeConstraints
#Define propagation scope to limit propagation to a specific management set called 'Web Servers'
$nPCS.PasswordPropagationSettings = New-Object -TypeName
LSClientAgentCommandlets.RouletteWebService.PasswordPropagationSettings
$nPCS.PasswordPropagationSettings.ConstrainToManagedSystems = $false
$nPCS.PasswordPropagationSettings.ConstrainToMembersOfGroup = $true
$nPCS.PasswordPropagationSettings.ConstrainToSystemsWithNonzeroInUse = $false
$nPCS.PasswordPropagationSettings.ExcludeDomainControllers = $false
$nPCS.PasswordPropagationSettings.ExcludeSystemWithAccount = $false
$nPCS.PasswordPropagationSettings.GroupName = "Web Servers"
$nPCS.PasswordPropagationSettings.PropagateToSystemWithAccountOnly = $false
$nPCS.PasswordPropagationSettings.PropagateToTrustingDomains = $false
#Create a new propagation targets (settings) list
$nPCS.PasswordPropagationTargets = New-Object -TypeName
LSClientAgentCommandlets.RouletteWebService.PasswordPropagationTargets
$nPCS.PasswordPropagationTargets.ListTargets = New-Object 'System.Collections.Generic.List
[LSClientAgentCommandlets.RouletteWebService.PasswordPropagationTarget]'
#Add one or more propagation sub-system
$nPCPropTar1 = New-Object -TypeName
LSClientAgentCommandlets.RouletteWebService.PasswordPropagationTarget
$nPCPropTar1.ConfigurationData = ""
$nPCPropTar1.DescriptiveName = "IIS Servers"
$nPCPropTar1.Enabled = $true
$nPCPropTar1.TargetSystemType_Linux = $false
$nPCPropTar1.TargetSystemType_Windows = $true
$nPCPropTar1.TypeName = "builtin:NetConfig"
$nPCPropTar2 = New-Object -TypeName
LSClientAgentCommandlets.RouletteWebService.PasswordPropagationTarget
$nPCPropTar2.ConfigurationData = ""
$nPCPropTar2.DescriptiveName = "IIS Servers"
$nPCPropTar2.Enabled = $true
$nPCPropTar2.TargetSystemType_Linux = $false
$nPCPropTar2.TargetSystemType_Windows = $true
$nPCPropTar2.TypeName = "builtin:COMPlus"
#Add propagation sub-systems to the propagation targets list
$nPCS.PasswordPropagationTargets.ListTargets.Add($nPCPropTar1)
$nPCS.PasswordPropagationTargets.ListTargets.Add($nPCPropTar2)
#Update the original job with the new settings
Set-LSJobPasswordChangeSettings -JobID $jid -AuthenticationToken $tok -PasswordChangeSettings $nPCS
```

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Invalid JobID**

  The job could not be found.

PowerShell: PropagationTargets ConfigurationsData

This section defines any ConfigurationData requirements for each propagation target under **InputArgs\PasswordChangeSettings\PropagationTargets\ListTargets**.

The data varies by target and not all propagation targets have configuration data.

builtin:WindowsServices

Used for Windows services. If the Windows services should be restarted following update, set the ConfigurationData to:

```
<Settings CompactMode="1"/>
```

If the Windows services should NOT be restarted following update, set the ConfigurationData to:

```
<Settings CompactMode="1" m_bRestartServicesAfterUpdate="0"/>
```

builtin:WindowsScheduler

Used for Windows scheduled tasks. There is no ConfigurationData for this item.

builtin:WindowsSchedulerAtAccount

Used for Windows AT identity. There is no ConfigurationData for this item.

builtin:COMPlus

Used for Windows COM/MTS applications. There is no ConfigurationData for this item.

builtin:DCOM

Used for Windows DCOM applications. There is no ConfigurationData for this item.

builtin:IIS6Metabase

Used for Windows IIS6 (anonymous, app pool, network credentials). There is no ConfigurationData for this item.

builtin:IIS7ConfigFiles - Windows IIS7 and later (anonymous, app pool, network credentials)

Used for Windows IIS7 and later (anonymous, app pool, network credentials). There is no ConfigurationData for this item.

builtin:SCOM

Used for Microsoft SCOM RunAs accounts. There is no ConfigurationData for this item.

builtin:SqlServer

Used for SQL Server Credentials (not to be confused with SQL Server Logins). You must define the target named instance for the SQL Server credentials propagation in `m_sInstanceName`.

```
<Settings CompactMode="1" m_sInstanceName="docs_sys"/>
```

builtin:NetConfig

Used for IIS asp.net connection strings. There is no ConfigurationData for this item.

builtin:ReplaceInFiles

Used for string replacement within files. The following ConfigurationData must be defined and set::

- **listFileTargetsForReplace:** Add one entry for each file to search in the specific propagation and also define...
  - **sLocalFilePath:** The double quoted path to the file to check for replacement. Path is local relative to target system.
  - **bCreateReferenceBackup:** Set to 1 to create a back of the original file and then define `sReferenceBackupFileFormatString`.
  - **sReferenceBackupFileFormatString:** The double quoted name of the backup file. Replaceable arguments are %filename% and %timestamp%. Default value is "Backup of %filename% (original)".
  - **bBackupExistingFile:** Default value is "0". Set to "1" to create multiple backups of the original file, up to `dwMaxNumberOfBackups`. You must also define `bBackupFileFormatString` for the secondary backup names. Replaceable arguments are %filename% and %timestamp%. Default value is "Backup of %filename% (original)". The default value is "Backup of %filename% at %timestamp%".
  - **dwMaxNumberOfBackups:** If multiple backups of the original file will be kept, define how many will be kept. Default value is "5".
  - **bReplaceTextFileExistingTypeOnly:** Set to "1" to use the native text file type to determine which text type to search for. Set to "0" to specify the text type. Then define `bReplaceASCII` and `bReplaceUnicode`.
  - **bReplaceASCII:** When `bReplaceTextFileExistingTypeOnly` is set to "0", set `bReplaceASCII` to "1" to search for ASCII text.
  - **bReplaceUNICODE:** When `bReplaceTextFileExistingTypeOnly` is set to "0", set `bReplaceUNICODE` to "1" to search for UNICODE text.
  - **bUseRegexSearch:** Set to "1" to use a regex search to find the old password and define the `sRegexSeach` parameter. Set to "0" to let Privileged Identity attempt to locate the previous password in the target files (password must have been previously managed/imported).
  - **sRegexBuilderString:** Set to an empty value, ""
  - **sRegexSearch:** The regex search pattern to use for the string replacement.
  - **dwSubExpressionNumber_Username:** Not used, set to "0".
  - **dwSubExpressionNumber_Password:** Set to "1".
  - **dwSubExpressionNumber_Description:** Not used, set to "0".

Following is an example ConfigurationData for a file called "/usr/bin/redoar/clpwd.py" that will perform a regex search for "password = (.*)":

```
<Settings CompactMode="1"><listFileTargetsForReplace sLocalFilePath="/usr/bin/redoar/clpwd.py"
bCreateReferenceBackup="1" sReferenceBackupFileFormatString="Backup of %filename% (original)"
bBackupExistingFile="0" sBackupFileFormatString="Backup of %filename% at %timestamp%"
dwMaxNumberOfBackups="5" bReplaceTextFileExistingTypeOnly="1" bReplaceASCII="1"
bReplaceUNICODE="1" bUseRegexSearch="1" sRegexBuilderString="" sRegexSearch="password = (.*)"
dwSubExpressionNumber_Username="0" dwSubExpressionNumber_Password="1" dwSubExpressionNumber_
Description="0"/></Settings>
```

builtin:RunProcess

Used to run an arbitrary process. The following ConfigurationData must be defined and set:

- **listFileTargetsForReplace:** Add one entry for each file to search in the specific propagation and also define:
  - **m_bOperationSupportsPropagation:** Set to "1".
  - **m_sPropagationCommandLineApp:** Define the path to the file to run on the target or local system. All forward and backslashes must be escaped by a backslash, e.g. "`c:\temp\file.exe`" should be written as "`c:\\temp\\file.exe`".
  - **m_sPropagationCommandLineParams:** The replaceable arguments for the propagation. Valid values are:
    - **%AccountDomain%:** Domain of the account being changed.
    - **%OldUsername%:** Current username of the account being changed.
    - **%NewUsername%:** New username of the account being changed if changing the account name.
    - **%OldPassword%:** Current password for the account being changed.
    - **%NewPassword%:** New password for the account being changed.
    - **%System%:** Target system which the change is being propagated to as entered in Privileged Identity.
    - **%SystemNetName%:** Network name for the system which change is being propagated to.
  - **m_sPropagationCommandLineFormat:** Defines the order for processing the file name and its command line parameters. Recommended value is "%Application% %Parameters%".
  - **m_eRunLocation:** Defines the location to run the program from. Valid values are:
    - "1" - run on the system performing the password change.
    - "2" - run on the target system. If this value is set, you must also define `m_eRemoteRunAsCredentialsType`.
  - **m_eRemoteRunAsCredentialsType:**Defines which credentials to use to run the program when `meRunLocation` is set to "2". Valid Values areL
    - "2" - run under explicitly defined credentials. You must also define `m_sRemoteRunAsExplicitUsername` and `m_sRemoteRunAsExplicitPassword`.
    - "3" - run under the account used to connect to the system.
    - "4" - run as the account being updated.
  - **m_sRemoteRunAsExplicitUsername:** When `m_eRemoteRunAsCredentialType` is set to "2", define the username to run the process as. If supplying a pre-windows 2000 username, e.g. "demo\bob", supply the name escaped like "demo\\bob".
  - **m_sRemoteRunAsExplicitPassword:** When `m_eRemoteRunAsCredentialType` is set to "2", define the password for the username the process will run as. Backslashes should be escaped with another backslash, e.g. "\\" and other special characters should be turned into their XML/HTML equivalents, for example, a double quote would be passed as "`&quote;`". This is typically automatically performed by the management console.
  - **m_bCopyDirectReferencedFiles:** Set to "1" to copy the target files from the source Privileged Identity machine to the target system. The file must exist in the exact same location on the source machine that it will be copied to on the target machines. If this value is set to "1", then you must also define `m_sFileCopyDestinationDirectory`.
  - **m_bCopyOtherFiles:** Set to "1" if you will wish to copy secondary files to the target system. Then define the `Settings/m_listFilesToCopy` section.
  - **m_ListFilesToCopy:** If `m_bCopyOtherFiles` is set to "1", define one or more entries for each secondary file to copy to the target system. Then define `sSourceFileName` and `sDestinationFileName`.
    - **sSourceFileName:** The escaped path to the source file on the Privileged Identity host system.
    - **sDestinationFileName:** The escaped path to the destination location on the target server (including target file name).
    - **dwFlags:** Set to "0"

builtin:Sharepoint

Used for Microsoft SharePoint server. There is no ConfigurationData for this item.

builtin:IBM WebSphere Application Server

Used to update IBM WebSphere Server where the target account matches an account name in WebSphere. If managing local WebSphere accounts, it is recommended to manage IBM WebSphere directly.

If using this propagation type, the following ConfigurationData must be defined and set:

- **m_strDefaultPort:** *(Optional)* Set the non-SSL port to connect to. Set `m_bUseSSL` to "0".
- **m_strSSLPort:** *(Optional)* Set the SSL port to connect to. Set `m_bUseSSL` to "1".
- **m_bUseSSL:** Set to "1" to use SSL and define `m_strSSLPort`. Set to "0" to not use SSL and define `m_strDefaultPort`.
- **m_strLoginUser:** The login name of the user.
- **m_strLoginPassword:** The XML/HTML escaped password for the login user.

```
<Settings CompactMode="1" m_strDefaultPort="9080" m_strSSLPort="9443" m_bUseSSL="1" m_
strLoginUser="wsadmin" m_strLoginPassword="P@ssw0rd"/>
```

builtin:Oracle WebLogic Server

Used to update Oracle WebLogic where the target account matches an account name in WebLogic. If managing local WebLogic accounts, it is recommended to manage Oracle WebLogic directly.

If using this propagation type, the following ConfigurationData must be defined and set:

- **m_strDefaultPort:** *(Optional)* Set the non-SSL port to connect to. Set `m_bUseSSL` to "0".
- **m_strSSLPort:** *(Optional)* Set the SSL port to connect to. Set `m_bUseSSL` to "1".
- **m_bUseSSL:** Set to "1" to use SSL and define `m_strSSLPort`. Set to "0" to not use SSL and define `m_strDefaultPort`.
- **m_strLoginUser:** The login name of the user.
- **m_strLoginPassword:** The XML/HTML escaped password for the login user.

```
<Settings CompactMode="1" m_strDefaultPort="8080" m_strSSLPort="8443" m_bUseSSL="1" m_
strLoginUser="wladmin" m_strLoginPassword="P@ssw0rd"/>
```

builtin:SAP Server

Used to update SAP local accounts where the target account matches an account name in SAP. If managing local SAP accounts, it is recommended to manage the SAP instances directly.

If using this propagation type, the following ConfigurationData must be defined and set:

- **m_iSystemNumber:** Define the system number you are connecting directly. If connecting using a gateway server, this value will be ignored.
- **m_strClient:** Define your client number if you are connecting directly. If connecting using a gateway server, this value will be ignored.
- **m_strUser:** Define the name of the management user to connect to SAP as.
- **m_strPassword:** Supply the escaped value for the password to connect as.
- **m_blsGatewayServer:** Set to "1" to indicate the target SAP server is a gateway server and define `m_strPath`, `m_nPort` or `m_bSecurePortEnabled` and `m_nSecurePort`.

- **m_nPort:** The unsecured port to connect to if `m_bIsGatewayServer` is set to "1" and `m_bSecurePortEnabled` is set to "0". If `m_bSecurePortEnabled` is set to "1", or `m_bSecurePortEnabled` is set to "0", this value will be ignored.
- **m_bSecurePortEnabled:** Defines SSL will be used to connect through a Netweaver Gateway server if set to "1" and `m_bIsGatewayServer` is also set to "1".
- **m_nSecurePort:** The secured port to connect to if `m_bIsGatewayServer` is set to "1" and `m_bSecurePortEnabled` is set to "1". If `m_bSecurePortEnabled` is set to "0" or `m_bSecurePortEnabled` is set to "0", this value will be ignored.
- **m_strPath:** The path on the Netweaver server's URL to locate the Privileged Identity integration. This value is required if `m_bIsGatewayServer` is set to "1".

```
<Settings CompactMode="1" m_iSystemNumber="1" m_strClient="2" m_strUser="sap*" m_
strPassword="P@ssw0rd" m_bIsGatewayServer="1" m_nPort="55636" m_bSecurePortEnabled="1" m_
nSecurePort="65535" m_strPath="/sap/opu/odata/LIEBSOFT/ERPM_USER_MGMT"/>
```

builtin:UpdateLogonCache

Used for Windows logon cache. There is no ConfigurationData for this item.

builtin:UpdateAutoLogon

Windows automatic logon account. There is no ConfigurationData for this item.

builtin:JavaClientLocalStore

DEPRECATED - Update Privileged Identity local Java client. There is no ConfigurationData for this item.

builtin:SQLReportingServices

Used for Microsoft SQL Reporting Services Action Account. There is no ConfigurationData for this item.

PowerShell: Set-LSJobPasswordSpin

Set-LSJobPasswordSpin generates a new password randomization job. This method always uses fixed random password generation settings, and the password change job is set to run 30 minutes from the start time.

The fixed settings are based on management console elements:

- Password Compatibility: Windows 2000 and later compatible
- Password May Contain: all character types
- Create a unique password for each account: enabled
- Password Character Constraints, character minimums: 0, the password may or may not contain any of the characters
- Position Constraints: all character types in all positions
- Constraints on Symbols: Use any symbols
- Other Arbitrary Constraints: none

Use "Set-LSJobSchedule" on page 1 to change the job scheduling options once the job is created.

If these default settings cannot be used, it is recommended to edit this job's settings using "PowerShell: Set-LSJobPasswordChangeSettings" on page 1, or create anew job using"PowerShell: New-LSJobWindowsChangePassword" on page 1.

Permissions Required

- All Access

Related Commands

- **SOAP:** JobOps_SpinPassword
- **REST:** Job/SpinPassword

Syntax

```
Set-LSJobPasswordSpin [-AuthenticationToken] <string> [-SystemName] <string> [-Namespace]
<string> [-AccountName] <string> [-AccountType] <EAccountType> {ACCOUNT_TYPE_USER | ACCOUNT_TYPE_
ADMINISTRATOR | ACCOUNT_TYPE_GUEST | ACCOUNT_TYPE_SQLSERVER_SA_ACCOUNT | ACCOUNT_TYPE_LINUX_
ACCOUNT | ACCOUNT_TYPE_CISCO_ROUTER | ACCOUNT_TYPE_AS400_ACCOUNT | ACCOUNT_TYPE_UNIX_ACCOUNT |
ACCOUNT_TYPE_MYSQL_ACCOUNT | ACCOUNT_TYPE_ORACLE_ACCOUNT | ACCOUNT_TYPE_CUSTOM_ACCOUNT | ACCOUNT_
TYPE_LDAP | ACCOUNT_TYPE_SYBASE | ACCOUNT_TYPE_OS390_ACCOUNT | ACCOUNT_TYPE_DRAC | ACCOUNT_TYPE_
IPMI | ACCOUNT_TYPE_3270_ACCOUNT | ACCOUNT_TYPE_DSRM} [-AssetTag] <string> [-PassException] [-
Trace] [-RunAs <pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.
- **AccountName:** The name of the target account.
- **AccountType:** For Windows password change jobs, this value identifies if you are targeting a Windows systems' built-in administrator, built-in guest, a regular user, or if the job will target another platform such as SQL Server or IPMI. Valid values are:

- ACCOUNT_TYPE_USER
- ACCOUNT_TYPE_ADMINISTRATOR - set `AccountName` to `*Administrator`.
- ACCOUNT_TYPE_GUEST - set `AccountName` to `*Guest`.
- ACCOUNT_TYPE_SQLSERVER_SA_ACCOUNT
- ACCOUNT_TYPE_LINUX_ACCOUNT
- ACCOUNT_TYPE_CISCO_ROUTER
- ACCOUNT_TYPE_AS400_ACCOUNT
- ACCOUNT_TYPE_UNIX_ACCOUNT
- ACCOUNT_TYPE_MYSQL_ACCOUNT
- ACCOUNT_TYPE_ORACLE_ACCOUNT
- ACCOUNT_TYPE_CUSTOM_ACCOUNT
- ACCOUNT_TYPE_LDAP
- ACCOUNT_TYPE_SYBASE
- ACCOUNT_TYPE_OS390_ACCOUNT
- ACCOUNT_TYPE_DRAC
- ACCOUNT_TYPE_IPMI
- ACCOUNT_TYPE_3270_ACCOUNT
- ACCOUNT_TYPE_DSRM

- **AssetTag:** Sets the asset tag to be assigned to the target system.
- **Namespace:** Lists the namespace of the system.
- **SystemName:** The name of the target system.

Example Request

```
Set-LSJobPasswordSpin - AccountName *Administrator -AccountType ACCOUNT_TYPE_ADMINISTRATOR -
AssetTag Cookie -AuthenticationToken $tok -Namespace DBAG01 -SystemName DBAG01
```

Output Success

The output message provides the new Job ID.

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

PowerShell: Set-LSJobPreAndPostRunSettings

Set-LSJobPreAndPostRunSettings sets and replaces pre and post-run job settings on the target job.

Permissions Required

- Delegated control of the job.

Related Commands

- **SOAP:** JobOps_SetPreAndPostRunSettings
- **REST:** Job/PreAndPostRun (PUT)

Syntax

```
Set-LSJobPreAndPostRunSettings [-AuthenticationToken] <string> [-JobID] <string> [-
JobPreAndPostSettings] <JobPreAndPostSettings> [-PassException] [-Trace] [-RunAs <pscredential>]
[<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.
- **JobID:** The ID of the target job.
- **PreAndPostRunSettings:** The pre and/or post-run settings to apply to the target job.
    - **PostRunApplication:** The path of the executable to run on the Privileged Identity host after the job completes.
    - **PostRunArguments:** Command line arguments for the post-run executable.
    - **PostRunExe:** Set to `$true` to run a PostRun application.
    - **PreRunAbortFail:** Set to `$true` to abort the the job if the pre-run operation fails or returns a non-zero code.
    - **PreRunApplication:** The path of the executable to run on the Privileged Identity host, before the job starts.
    - **PreRunArgs:** Command line arguments for the pre-run executable.
    - **PreRunExe:** Set to `$true` to run a PreRun application.
    - **PreRunWait:** Set to `$true` to wait for the PreRun application to exit and supply a non-zero return code before continuing to process the job. Set to `$false` to run the PreRun operation and immediately continue processing the password change.

Example Request

```
#Create Pre and post run object
$preandpost = New-Object -TypeName LSClientAgentCommandlets.RouletteWebService.JobPreAndPostSettings
$preandpost.PostRunApplication = "c:\utils\sdnutil.exe"
$preandpost.PostRunArgs = "-Op Close -Targ vn-custx"
$preandpost.PostRunExe = $true
$preandpost.PreRunAbortFail = $true
$preandpost.PreRunApplication = "c:\utils\sdnutil.exe"
$preandpost.PreRunArgs = "Op Open -Targ vn-custx"
```

```
$preandpost.PreRunExe = $true
$preandpost.PreRunWait = $true
#Update the job
Set-LSJobPreAndPostRunSettings -AuthenticationToken $tok -JobID $jid -JobPreAndPostSettings
$preandpost
```

Output Success

The output message states the job's pre and post run settings were updated.

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Invalid JobID**

  The job could not be found.

PowerShell: Set-LSJobRun

Set-LSJobRun updates the specified job's next run time to run now. Run now is "now" plus 1 minute to account for transaction submission delays.

Permissions Required

- Delegated control of the job.

Related Commands

- **SOAP:** JobsOps_RunJob
- **REST:** Job/RunNow (POST)

Syntax

```
Set-LSJobRun [-AuthenticationToken] <string> [-JobID] <string> [-PassException] [-Trace] [-RunAs
<pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.
- **JobID:** The ID of the target job.

Example Request

```
Set-LSJobRun -AuthenticationToken $tok - JobID $jid
```

Output Success

The output message states the job schedule was updated.

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Invalid JobID**

  The job could not be found.

PowerShell: Set-LSJobSchedule

Set-LSJobSchedule sets a job schedule for a target job.

Permissions Required

- Delegated control of the job.

Related Commands

- **SOAP:** JobOps_SetJobSchedule
- **REST:** Job/Schedule (PUT)

Syntax

```
Set-LSJobSchedule [-AuthenticationToken] <string> [-JobID] <string> [-ScheduleInfo]
<ScheduleInfo> [-PassException] [-Trace] [-RunAs <pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.
- **JobID:** The ID of the target job.
- **ScheduleInfo:** The job's schedule object.
    - **DayOfMonth:** *(Optional)* Set the day of the month to run the job. Valid values are 1-31. For months with less than 31 days, the job will run on the last day of the month.
    - **DayOfWeek:** *(Optional)* Set the day of the week to run the job. Values are:
        - **0** = Sunday
        - **1** = Monday
        - **2** = Tuesday
        - **3** = Wednesday
        - **4** = Thursday
        - **5** = Friday
        - **6** = Saturday
    - **DayOfYear:** Not used.
    - **DaysBits:** Not used.
    - **EveryNDays:** *(Optional)* Set the amount of days for the job to reoccur.
    - **Hours:** *(Optional)* Set the hour at which the job will run.
    - **Minutes:** *(Optional)* Set the minutes into the hour (Hours) when the job will run.
    - **MonthOfYear:** *(Optional)* Set the month (number) the job to run. Values are:
        - **1** = January
        - **2** = February
        - **3** = March
        - **4** = April

        - **5** = May
        - **6** = June
        - **7** = July
        - **8** = August
        - **9** = September
        - **10** = October
        - **11** = November
        - **12** = December
    - **NextRetryUTC:** *(Optional)* For new jobs, this value should not be used. Expected format is ISO 8601, YYYY-MM-DDThh:mm:ss.
    - **NumberOfRetries:** *(Optional)* Set the number of retries for the job. If it is not defined, the system default is used.
    - **Reboot:** *(Optional)* Set to **True** to reboot Windows systems following a password change. Systems defined in `SystemList` are affected.
    - **RetryEnabled:** *(Optional)* Set to `$true` to enable retries in the event of a failure.
    - **RunWindowMinutes:** *(Optional)* Set to 1 or more minutes for the job to run by the specified time plus the run window duration, or the job will be skipped. Set to 0 or do not define to run the job despite missing the originally scheduled time.
    - **ScheduleType:** *(Optional)* Define the type of schedule (one time, recurring, etc.) for the job. If not defined, the default value is SCHEDULE_TYPE_INTERACTIVE, which means it must run manually or run immediately based on other scheduling options. Valid values are:
        - **SCHEDULE_TYPE_UNKNOWN**
        - **SCHEDULE_TYPE_IMMEDIATELY**
        - **SCHEDULE_TYPE_HOURLY:** Job will run once every hour. Set `Minutes`.
        - **SCHEDULE_TYPE_DAILY:**Job will run once everyday on the specified time. Set `Hours` and `Minutes`.
        - **SCHEDULE_TYPE_WEEKLY:** Job will run once every week on the specified day and time. Set `Hours`, `Minutes` and `DayOfWeek`.
        - **SCHEDULE_TYPE_MONTHLY:** Job will run once every month on the specified day and time. `DayOfMonth`, `Hours`, and `Minutes`.
        - **SCHEDULE_TYPE_YEARLY:** Job will run once every year on the specified month, day and time. Define `DayOfMonth`, `MonthOfYear`, `Hours`, and `Minutes`.
        - **SCHEDULE_TYPE_DAYS_OF_WEEK:** Job will run every set day of week. Set `DayOfWeek`.
        - **SCHEDULE_TYPE_ONCE:** Job will run once at some point in the future. Define `DayOfMonth`, `MonthOfYear`, `Hours`, and `Minutes`.
        - **SCHEDULE_TYPE_N_DAYS:** Job will run every N days. Set integer value for `EveryNDays`.
        - **SCHEDULE_TYPE_INTERACTIVE:** *(Default)* Job will run based on `NextRunTimeUTC`.
        - **SCHEDULE_TYPE_N_HOURS:** Job will run every N hours. Set `Hours` for the number of hours, and set `Minutes` for the number of minutes to offset.
    - **SchedulingPeriod:** Not used.
    - **UpdateNextRunTimeForPartialCompletion:** *(Optional)* Set to `$true` to define whether jobs with multiple systems should update the next run time based on the management console display. Default value is `$false`.

Example Request

```
#Set schedule info
$sked = New-Object -TypeName LSClientAgentCommandlets.RouletteWebService.ScheduleInfo
$sked.DayOfMonth = 27
$sked.Hours = 0
$sked.Minutes = 30
$sked.NextRetryTimeUTC = "2017-08-03T12:01:02"
$sked.NumberOfRetries = 3
$sked.RetryEnabled = $true
$sked.RunWindowMinutes = 20
$sked.ScheduleType = "SCHEDULE_TYPE_MONTHLY"
$sked.UpdateNextRunTimeForPartialCompletion = $true
#Set the new schedule
Set-LSJobSchedule -AuthenticationToken $tok -JobID $jid -ScheduleInfo $sked
```

Output Success

The output message states the job schedule was updated successfully.

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Invalid JobID**

  The job could not be found.

PowerShell: Set-LSJobSSHKeyChangeSettings

Set-LSJobSSHKeyChangeSettings defines SSH key change job settings for an existing job.

Permissions Required

- Delegated control of the job

Related Commands

- **SOAP:** JobOps_SetJobKeyChangeSettings
- **REST:** Job/SSHKeyChange (PUT)

Syntax

```
Set-LSJobSSHKeyChangeSettings [-AuthenticationToken] <string> [-JobID] <string> [-
JobSSHKeyChangeSettings] <JobSSHKeyChangeSettings> [-PassException] [-Trace] [-RunAs
<pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.
- **JobID:** Job ID of the target job.
- **JobSSHKeySettings:** The new SSH key settings for the job.
    - **DeleteKeyFileOnRemoteSystems:** When set to `$true`, a new key is generated and stored in the solution database on the first run only and removes any physical files found where it has the authority to remove key files.
    - **GenerateNewKeyForEachRun:** When set to `$true`, a new key is generated, stored, and updated in the solution database on every job run and does not perform any subsequent updates to target systems.
    - **KeyLabel:** The key label to be updated.
    - **KeyLengthBits:** Length of the new key. The bit length defaults to the current length of the key. Available options are 2048, 3072, and 4096 bits.
    - **KeyType:** This functionality is limited to OpenSSH v2 RSA type keys and cannot be configured. Set this value to 0.
    - **OldKeyLabel:** Not used.
    - **OldKeySig:** Not used.
    - **OldPublicKey:** Not used.
    - **RemoveOldKey:** When set to `$true`, a new key will be generated and stored in the database on the first run only. Then, the previous key references will be removed from the authorized key files on the target systems, which breaks any access reliant on the old key.
    - **UpdateKeyReferences:** When set to `$true`, a new key will be generated and stored in the solution database on the first job run only and updates the authorized key files on the target systems. Subsequent job runs will try to update the target system's authorized keys to reference the new SSH key on the first job run. If a system was offline or otherwise inaccessible when the job ran previously, it will be updated on a subsequent run. This job will distribute key files to the systems. It only updates the authorized key files on the target systems.

Example Request

```
#Set schedule info
$keyset = New-Object -TypeName LSClientAgentCommandlets.RouletteWebService.JobSSHKeyChangeSettings
$keyset.DeleteKeyFileOnRemoteSystems = $false
$keyset.GenerateNewKeyEachRun = $true
$keyset.KeyLabel = "dbsmashlnx"
$keyset.KeyLengthBits = 3072
$keyset.KeyType = 0
$keyset.RemoveOldKey = $false
$keyset.UpdateKeyReferences = $true
#Set the new schedule
Set-LSJobSSHKeyChangeSettings -AuthenticationToken $tok -JobID $jid -JobSSHKeyChangeSettings $keyset
```

Output Success

The output message states the job settings were updated.

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Invalid JobID**

  The job could not be found.

PowerShell: Set-LSSharedCredentialList

Set-LSSharedCredentialList changes the settings for a shared credential list.

Permissions Required

- Manage External Lists
- All Access

Related Commands

- **SOAP:**
- **REST:** SharedCredentialList (PUT)

Syntax

```
Set-LSSharedCredentialList [-AuthenticationToken] <string> [-Comment] <string> [-
SharedCredentialListName] <string> [-PassException] [-Trace] [-RunAs <pscredential>]
[<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.
  - **AllowAdd:** Set to `$true` to allow new passwords to be added to the shared credential list.
  - **AllowDelegate:** Set to `$true` to allow other users to manage the delegation of the shared credential list.
  - **AllowEdit:** Set to `$true` to allow existing passwords in the shared credential list to be edited.
  - **Comment:** Include a comment.
  - **Count:** Number of passwords included in the shared credential list.
  - **Name:** The name of the shared credential list.
  - **RowID:**
  - **NewListName:** (string) Provide a new name for the shared credential list, if needed.

Example Request

```
Set-LSSharedCredentialLists - AuthenticationToken HGECUWK11TYGBTCOOXAEZFIN38PL9OPK -Comment
TestComment -SharedCredentialListName "AB List" -NewListName "AC List"
```

Output Success

The output states the shared credential list has been updated.

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Invalid List**

  The shared credential list could not be found.

- **Edit Failed**

  Failed to edit the shared credential list.

PowerShell: Delegations

Use PowerShell cmdlets to set, edit, or remove delegations. Delegations are used to grant access to passwords, account elevation, file store, etc.

Get-LSListDelegationAccountMasks

Get-LSListDelegationAccountMasks returns the entire list of defined account masks.

Permissions Required

- View Delegations

Related Commands

- **SOAP:** DelegationOps_GetAccountMaskPermissionsList
- **REST:** Delegation/AccountMask (GET)

Syntax

```
Get-LSListDelegationAccountMasks [-AuthenticationToken] <string> [-PassException] [-Trace] [-
RunAs <pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.

Example Request

```
Get-LSListDelegationAccountMasks - AuthenticationToken $tok
```

Output Success

If no account masks defined are defined, the output list is empty. If lists are defined, the output contains the account mask and associated identity.

Example Success Output

```
ExtensionData                       AccountMask                       IdentityName

-------------                       -----------                       ------------

System.Runtime.Serialization.Exte... admin*                           pat
```

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

PowerShell: Get-LSListDelegationIdentities

Get-LSListDelegationIdentities returns the entire list of delegated identities and their global permissions. Use"Get-LSListDelegationPermissions" on page 1 to return a list of all permissions for all permission types and for all identities.

Permissions Required

- View Delegations

Related Commands

- **SOAP:** DelegationOps_GetIdentities
- **REST:** Delegation/Identity (GET)

Syntax

```
Get-LSListDelegationIdentities [-AuthenticationToken] <string> [-PassException] [-Trace] [-RunAs
<pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.

Example Request

```
Get-LSListDelegationIdentities - AuthenticationToken $tok
```

Output Success

If there are not any delegation identities defined, the output list is empty. If identities are defined, the output contains the list of the identities and their global permissions.

Example Success Output

```
ExtensionData                          : System.Runtime.Serialization.ExtensionDataObject
AccountName                            : Recovery User
AlertOnRecovery                        : False
AlertOnRequest                         : False
DisplayName                            :
EmailAddress                           :
IsDomainAccount                        : MANAGER_TYPE_ROLE
Password                               :
PermissionAccessRemoteSessions         : False
PermissionAddPasswordsForManagedSystems : False
PermissionAllAccess                    : False
PermissionCreateRefreshSystemJob       : False
PermissionEditDelegation               : False
PermissionEditPasswordLists            : False
PermissionEditStoredPasswords          : False
```

```
PermissionElevateAccountPermissions     : False
PermissionElevateAnyAccountPermissions  : False
PermissionGrantPasswordRequests         : False
PermissionIgnorePasswordCheckout        : False
PermissionLogon                         : True
PermissionPersonalStore                 : False
PermissionRequestPasswords              : False
PermissionRequestRemoteAccess           : False
PermissionRequireOATH                   : False
PermissionRequireRSASecurID             : False
PermissionSelfRecovery                  : False
PermissionViewAccounts                  : True
PermissionViewDashboards                : False
PermissionViewDelegation                : False
PermissionViewFileStore                 : False
PermissionViewJobs                      : False
PermissionViewPasswordActivity          : False
PermissionViewPasswordHistory           : False
PermissionViewPasswords                 : True
PermissionViewScheduler                 : False
PermissionViewSystems                   : False
PermissionViewWebLogs                   : False
```

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

PowerShell: Get-LSListDelegationManagementSetsForIdentity

Get-LSListDelegationManagementSetsForIdentity lists the management sets for which a specific identity has global permissions.

Permissions Required

- View Delegation

Related Commands

- **SOAP:** DelegationOps_GetManagedGroupsforIdentity
- **REST:** Delegation/Identity/ManagementSet (GET)

Syntax

```
Get-LSListDelegationManagementSetsForIdentity [-AuthenticationToken] <string> [-IdentityName]
<string> [-PassException] [-Trace] [-RunAs <pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.
- **IdentityName:** The name of the target identity to retrieve permissions for.

Example Request

```
Get-LSListDelegationManagementSetsForIdentity -AuthenticationToken $tok - IdentityName $id
```

Output Success

A successful output provides all management sets associated with the identity. If no management sets are defined for the identity, the output list is empty.

Example Success Output

```
ExtensionData                                          ManagedGroupName

-------------                                          ----------------

System.Runtime.Serialization.ExtensionDataObject       CLR Tests

System.Runtime.Serialization.ExtensionDataObject       LinuxTest

System.Runtime.Serialization.ExtensionDataObject       All Windows Systems
```

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Invalid or non-existent identity**

  The database call did not return any data, or the file was not found.

PowerShell: Get-LSListDelegationPermissions

Get-LSListDelegationPermissions returns all identities and all permissions across all delegated objects.

Permissions Required

- View Delegation

Related Commands

- **SOAP:** DelegationOps_GetPermissions
- **REST:** Delegation/Permission (GET)

Syntax

```
Get-LSListDelegationPermissions [-AuthenticationToken] <string> [-PassException] [-Trace] [-RunAs
<pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.

Example Request

```
Get-LSListDelegationPermissions - AuthenticationToken $tok
```

Output Success

Output provides all global permissions for identities as well as object specific permissions.

Example Success Output

```
ExtensionData                          : System.Runtime.Serialization.ExtensionDataObject
AccountName                            :
AccountStoreType                       : 0
Application                            :
FileName                               :
IdentityID                             : 12
IdentityName                           : pat
JobID                                  : 0
ManagementSetName                      :
Namespace                              :
OfflineTenant                          :
PermissionAccessJobs                   : False
PermissionAccessPersonalPasswords      : False
PermissionAccessRemoteSessions         : False
PermissionAddPassword                  : False
PermissionAlertOnChange                : False
PermissionAlertOnIncident              : False
```

```
PermissionAllAccess                         : False
PermissionChangeDelegation                  : False
PermissionChangePasswords                   : False
PermissionChangePasswordsOnManagedSystems   : False
PermissionChangeSharedCredentialLists       : False
PermissionCreateModifyManagementSets        : False
PermissionCreateRefreshJobs                 : False
PermissionDeletePassword                    : False
PermissionEditWebPanels                     : False
PermissionElevateAnyAccount                 : False
PermissionGrantPasswordRequests             : False
PermissionIgnorePasswordCheckout            : False
PermissionLogon                             : False
PermissionModifyElevationJob                : False
PermissionModifyPasswordChangeJob           : False
PermissionModifyRefreshJob                  : False
PermissionRead                              : False
PermissionRequestPasswords                  : False
PermissionRequestRemoteAccess               : False
PermissionRequire2Factor                    : False
PermissionRequireOATH                       : False
PermissionSelfAccountElevation              : False
PermissionSelfRecovery                      : False
PermissionType                              : PERMISSION_TYPE_ON_SHARED_CREDENTIAL_LIST
PermissionViewAccounts                      : True
PermissionViewDashboards                    : False
PermissionViewDelegation                    : False
PermissionViewFileStore                     : False
PermissionViewPasswordActivity              : False
PermissionViewPasswordHistory               : False
PermissionViewPasswords                     : True
PermissionViewSchedulerService              : False
PermissionViewSystems                       : False
PermissionViewWebLogs                       : False
PermissionWrite                             : False
RestrictionDayOfMonthEnd                    : 0
RestrictionDayOfMonthStart                  : 0
RestrictionDayOfWeekEnd                     : 0
RestrictionDayOfWeekStart                   : 0
RestrictionEndTimeUTC                       : 1/1/1900 12:00:00 AM
RestrictionStartTimeUTC                     : 1/1/1900 12:00:00 AM
ScheduleRestrictionType                     : 0
SharedCredentialListName                    : List-4990
SystemName                                  :
```

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

PowerShell: Get-LSListDelegationPermissionsForSelfRecovery

Get-LSListDelegationPermissionsForSelfRecovery returns a list of all self-recovery permission entries.

Permissions Required

- View Delegation

Related Commands

- **SOAP:** DelegationOps_GetSelfRecoveryPermissionList
- **REST:** DelegationSelfRecoveryPermission (GET)

Syntax

```
Get-LSListDelegationPermissionsForSelfRecovery [-AuthenticationToken] <string> [-PassException]
[-Trace] [-RunAs <pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.

Example Request

```
Get-LSListDelegationPermissionsForSelfRecovery - AuthenticationToken $tok
```

Output Success

If there are not any self-recovery permissions set, the output list is empty. If permissions are defined, the output contains the self recovery rule and associated identity.

Example Success Output

```
ExtensionData : System.Runtime.Serialization.ExtensionDataObject
AccountName   : erpmlauncher
IdentityName  : pat
Namespace     : LSC
SystemName    : lsc.ent
```

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

PowerShell: Get-LSListDelegationPermissionsOnAccounts

Get-LSListDelegationPermissionsOnAccounts returns a list of identities and permissions assigned per-account delegations.

Permissions Required

- View Delegation

Related Commands

- **SOAP:** DelegationOps_GetPermissionsOnAccounts
- **REST:** Delegation/StoredCredential (GET)

Syntax

```
Get-LSListDelegationPermissionsOnAccounts [-AuthenticationToken] <string> [-PassException] [-
Trace] [-RunAs <pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.

Example Request

```
Get-LSListDelegationPermissionsOnAccounts - AuthenticationToken $tok
```

Output Success

If per-account delegation is defined, the output contains an entry for each per-account delegation.

Example Success Output

```
ExtensionData                  : System.Runtime.Serialization.ExtensionDataObject
AccountName                    : bob
AlertForChange                 : False
AlertForIncident               : False
IdentityName                   : pat
Namespace                      : DTVM
PermissionAllowRemoteSessions  : False
PermissionGrantPasswordRequests : False
PermissionRequestPasswords     : False
PermissionRequestRemoteAccess  : False
PermissionViewAccounts         : True
PermissionViewPasswords        : True
SystemName                     : dtvm
```

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

PowerShell: Get-LSListDelegationPermissionsOnFile

Get-LSListDelegationPermissionsOnFile returns a list of identities and permissions assigned to a specific file via file store delegations.

Permissions Required

- View Delegation

Related Commands

- **SOAP:** DelegationOps_StoredFile_GetPermissions
- **REST:** Delegation/File (GET)

Syntax

```
Get-LSListDelegationPermissionsOnFile [-AuthenticationToken] <string> [-FileID] <string> [-
PassException] [-Trace] [-RunAs <pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.
- **FileID:** The ID of the target file.

Example Request

```
Get-LSListDelegationPermissionsOnFile -AuthenticationToken $tok - FileID $fID
```

Output Success

If file store delegations are defined for a specific file, the output contains an entry for each file delegation.

Example Success Output

```
ExtensionData      : System.Runtime.Serialization.ExtensionDataObject
FileName           : LSC Certificate for Web-1.cer
IdentityName       : lsds\lscadmin
PermissionDelegate : True
PermissionDelete   : True
PermissionDownload : True
PermissionGrant    : True
PermissionRequest  : False
PermissionUpdate   : True
PermissionView     : True
ExtensionData      : System.Runtime.Serialization.ExtensionDataObject
FileName           : LSC Certificate for Web-1.cer
IdentityName       : pat
PermissionDelegate : False
PermissionDelete   : True
```

```
PermissionDownload : True
PermissionGrant    : False
PermissionRequest  : False
PermissionUpdate   : False
PermissionView     : True
```

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Invalid or non-existent FileID**

  The database call did not return any data, or the file was not found.

PowerShell: Get-LSListDelegationPermissionsOnJobs

Get-LSListDelegationPermissionsOnJobs returns a list of per-job permissions.

> ***Note:*** *Job permissions added prior to version 5.5.2.1 will be unavailable for reporting.*

Permissions Required

- View Delegation

Related Commands

- **SOAP:**DelegationOps_GetPermissionsOnJobs
- **REST:** Delegation/Job (GET)

Syntax

```
Get-LSListDelegationPermissionsOnJobs [-AuthenticationToken] <string> [-PassException] [-Trace]
[-RunAs <pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.

Example Request

```
Get-LSListDelegationPermissionsOnJobs - AuthenticationToken $tok
```

Output Success

If defined, a list of each delegation for each job is returned.

Example Success Output

```
ExtensionData                          : System.Runtime.Serialization.ExtensionDataObject
AccountName                            :
AccountStoreType                       : 0
Application                            :
FileName                               :
IdentityID                             : 12
IdentityName                           : pat
JobID                                  : 3
ManagementSetName                      :
Namespace                              :
OfflineTenant                          :
PermissionAccessJobs                   : False
PermissionAccessPersonalPasswords      : False
PermissionAccessRemoteSessions         : False
```

```
PermissionAddPassword                     : False
PermissionAlertOnChange                   : False
PermissionAlertOnIncident                 : False
PermissionAllAccess                       : False
PermissionChangeDelegation                : False
PermissionChangePasswords                 : False
PermissionChangePasswordsOnManagedSystems : False
PermissionChangeSharedCredentialLists     : False
PermissionCreateModifyManagementSets      : False
PermissionCreateRefreshJobs               : False
PermissionDeletePassword                  : False
PermissionEditWebPanels                   : False
PermissionElevateAnyAccount               : False
PermissionGrantPasswordRequests           : False
PermissionIgnorePasswordCheckout          : False
PermissionLogon                           : False
PermissionModifyElevationJob              : False
PermissionModifyPasswordChangeJob         : False
PermissionModifyRefreshJob                : False
PermissionRead                            : True
PermissionRequestPasswords                : False
PermissionRequestRemoteAccess             : False
PermissionRequire2Factor                  : False
PermissionRequireOATH                     : False
PermissionSelfAccountElevation            : False
PermissionSelfRecovery                    : False
PermissionType                            : PERMISSION_TYPE_ON_JOB
PermissionViewAccounts                    : False
PermissionViewDashboards                  : False
PermissionViewDelegation                  : False
PermissionViewFileStore                   : False
PermissionViewPasswordActivity            : False
PermissionViewPasswordHistory             : False
PermissionViewPasswords                   : False
PermissionViewSchedulerService            : False
PermissionViewSystems                     : False
PermissionViewWebLogs                     : False
PermissionWrite                           : True
RestrictionDayOfMonthEnd                  : 0
RestrictionDayOfMonthStart                : 0
RestrictionDayOfWeekEnd                   : 0
RestrictionDayOfWeekStart                 : 0
RestrictionEndTimeUTC                     : 1/1/1900 12:00:00 AM
RestrictionStartTimeUTC                   : 1/1/1900 12:00:00 AM
ScheduleRestrictionType                   : 0
SharedCredentialListName                  :
SystemName                                :
```

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

PowerShell: Get-LSListDelegationPermissionsOnManagementSets

Get-LSListDelegationPermissionsOnManagementSets returns a list of per-management set permissions.

Permissions Required

- View Delegation

Related Commands

- **SOAP:** DelegationOps_GetPermissionsOnManagementSets
- **REST:** DelegationManagementSEt (GET)

Syntax

```
Get-LSListDelegationPermissionsOnManagementSets [-AuthenticationToken] <string> [-PassException]
[-Trace] [-RunAs <pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.

Example Request

```
Get-LSListDelegationPermissionsOnManagementSets – AuthenticationToken $tok
```

Output Success

If defined, a list of each delegation for each management set is returned.

Example Success Output

```
ExtensionData                       : System.Runtime.Serialization.ExtensionDataObject
AlertForChange                      : False
AlertForIncident                    : True
IdentityName                        : pat
ManagementSetName                   : All Windows Systems
PermissionAllowRemoteSessions       : False
PermissionChangeGroupMembership     : False
PermissionElevateAccountPermissions : False
PermissionGrantPasswordRequests     : False
PermissionRequestPasswords          : False
PermissionRequestRemoteAccess       : False
PermissionViewAccounts              : False
PermissionViewPasswords             : False
PermissionViewSystems               : False
```

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

PowerShell: Get-LSListDelegationPermissionsOnSharedCredentialList

Get-LSListDelegationPermissionsOnSharedCredentialList returns a list of shared credential list permissions.

Permissions Required

Any of the following permissions:

- View Delegation
- Manage delegations on the target list
- Manage External Lists

Related Commands

- **SOAP:** DelegationOps_GetPermissionsForSharedCredentialList
- **REST:** Delegation/SharedCredentialList (GET)

Syntax

```
Get-LSListDelegationPermissionsOnManagementSets [-AuthenticationToken] <string> [-
SharedCredentialList] <string> [-PassException] [-Trace] [-RunAs <pscredential>]
[<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.
- **SharedCredentialList:** The shared credential list.

Example Request

```
Get-LSListDelegationPermissionsOnManagementSets -AuthenticationToken $tok - SharedCredentialList
$list
```

Output Success

If defined, a list of delegations for each shared credential list is returned.

Example Success Output

```
ExtensionData              : System.Runtime.Serialization.ExtensionDataObject
CredentialListName         : List-0001
IdentityName               : pat
PermissionAddPassword      : False
PermissionChangeDelegation : True
PermissionDeletePassword   : False
PermissionEditPassword     : False
PermissionGrantRequest     : False
PermissionRecoverPassword  : False
```

```
PermissionRequestPassword  : False
PermissionViewList         : True
```

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Invalid shared credential list or list not found**

  A shared credential list with the name specified could not be found.

PowerShell: Get-LSListDelegationPermissionsOnSystems

Get-LSListDelegationPermissionsOnSystems returns a list of per-system permissions.

Permissions Required

- View Delegation

Related Commands

- **SOAP:** DelegationOps_GetPermissionsonSystems
- **REST:** Delegation/System

Syntax

```
Get-LSListDelegationPermissionsOnSystems [-AuthenticationToken] <string> [-PassException] [-
Trace] [-RunAs <pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.

Example Request

```
Get-LSListDelegationPermissionsOnSystems - AuthenticationToken $tok
```

Output Success

If defined, a list of each delegation on each system is returned.

Example Success Output

```
ExtensionData                      : System.Runtime.Serialization.ExtensionDataObject
AlertForChange                     : False
AlertForIncident                   : False
IdentityName                       : pat
PermissionAllowRemoteSessions      : False
PermissionElevateAccountPermissions : False
PermissionGrantPasswordRequests    : False
PermissionRequestPasswords         : False
PermissionRequestRemoteAccess      : True
PermissionViewAccounts             : False
PermissionViewPasswords            : True
PermissionViewSystems              : False
SystemName                         : 10.1.0.95
```

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

PowerShell: Get-LSListDelegationRoleMapping

Get-LSListDelegationRoleMapping returns a list of delegated identity roles and their members.

Permissions Required

- View Delegation

Related Commands

- **SOAP:** DelegationOps_GetRoleMappingPermissionsList
- **REST:** Delegation/Role (GET)

Syntax

```
Get-LSListDelegationRoleMapping [-AuthenticationToken] <string> [-PassException] [-Trace] [-RunAs
<pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.

Example Request

```
Get-LSListDelegationRoleMapping – AuthenticationToken $tok
```

Output Success

If defined, a list of each identity role and its members is returned.

Example Success Output

```
ExtensionData               Authenticator      CredentialName           RoleName
-------------               -------------      --------------           --------
System.Runtime.Serializa... Azure AD.LSDS-AAD  K.Janeway@lickety.split  AAD-OAuth
System.Runtime.Serializa... Salesforce.LSSFDC  jerpm@lickety.split      SFDC-OAuth
System.Runtime.Serializa... lsds               dale                     Recovery User
System.Runtime.Serializa... lsds               fred                     Recovery User
System.Runtime.Serializa... SAML.OneLogin      phil@lickety.split       Administrator User
System.Runtime.Serializa... lsds.int           lscadmin                 Administrator User
```

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

PowerShell: New-LSDelegationIdentity

New-LSDelegationIdentity creates and adds a new delegation identity.

Permissions Required

- Manage Delegation

Related Commands

- **SOAP:** DelegationOps_CreateIdentity
- **REST:** Delegation/Identity (POST)

Syntax

```
New-LSDelegationIdentity [-AuthenticationToken] <string> [-IdentityName] <string> [-IdentityType]
<EManagerType> {MANAGER_TYPE_EXPLICIT_USER | MANAGER_TYPE_DOMAIN_USER | MANAGER_TYPE_DOMAIN_GROUP
| MANAGER_TYPE_SELF_RECOVERY | MANAGER_TYPE_ROLE | MANAGER_TYPE_RADIUS | MANAGER_TYPE_CERTIFICATE
| MANAGER_TYPE_LDAP_USER} [-Password] <string> [-PassException] [-Trace] [-RunAs <pscredential>]
[<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.
- **IdentityName:** The name of the new identity.
- **IdentityType:** Valid values are:
  - **MANAGER_TYPE_EXPLICIT_USER:** Explicit user account, `IdentityName` and `Password`, are required.
  - **MANAGER_TYPE_DOMAIN_USER:** Windows domain user.
  - **MANAGER_TYPE_DOMAIN_GROUP:** Domain-based global security group from a Windows domain.
  - **MANAGER_TYPE_SELF_RECOVERY:** Not used.
  - **MANAGER_TYPE_ROLE:** Privileged Identity role. User objects from LDAP sources must be added separately. Supply the name of the role as the `IdentityName`.
  - **MANAGER_TYPE_RADIUS:** RADIUS user. Supply the name of the `AuthenticationServerName\UserName` as the `IdentityName`.
  - **MANAGER_TYPE_CERTIFICATE:** Certificate-based identity. Supply the name of the user to be associated with the token as the `IdentityName`. Note the certificate must already have been enrolled in the management console.
  - **MANAGER_TYPE_LDAP_USER:** A specific user from an LDAP directory. Supply `AuthenticationServerName\UserName` name as the `IdentityName`.
- **Password:** The password for the identity. If your account is not an explicit user, specify an empty string with two single or double quotations.

Example Request

```
New-LSDelegationIdentity -AuthenticationToken $tok - IdentityName lsds\franklin -IdentityType
MANAGER_TYPE_DOMAIN_USER -Password ''
```

Output Success

The output states the identity was added.

Example Success Output

```
ExtensionData                          OperationMessage                    OperationSucceeded
-------------                          ----------------                    ------------------
System.Runtime.Serialization.Exte... Identity with name lsds\franklin...               True
```

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Identity already exists**

  An identity with name that name already exists, or the identity creation process failed.

PowerShell: New-LSDelegationManagementSetForIdentity

New-LSDelegationManagementSetForIdentity associates an existing management set with an existing delegation identity. Management sets are added at the global delegation level.

Permissions Required

- Manage Delegations

Related Commands

- **SOAP:** DelegationOps_GetManagedGroupsForIdentity
- **REST:** Delegation/Identity/ManagementSet (POST)

Syntax

```
New-LSDelegationManagementSetForIdentity [-AuthenticationToken] <string> [-IdentityName] <string>
[-ManagementSet] <string> [-PassException] [-Trace] [-RunAs <pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.
- **IdentityName:** The identity that will have a management set associated with it.
- **ManagementSet:** The name of the management set to be associated with the `IdentityName`.

Example Request

```
New-LSDelegationManagementSetForIdentity -AuthenticationToken $tok -IdentityName $id -
ManagementSet $msn
```

Output Success

The output states the identity is now managing the management set.

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Invalid IdentityName**

  The database call did not return any data, or the file was not found.

PowerShell: New-LSDelegationPermissionForSelfRecovery

New-LSDelegationPermissionForSelfRecovery creates new self-recovery delegations.

Permissions Required

- Manage Delegations

Related Commands

- **SOAP:** DelegationOps_SelfRecoveryPermission_Add
- **REST:** Delegation/SelfRecoveryPermission (POST)

Syntax

```
New-LSDelegationPermissionForSelfRecovery [-AuthenticationToken] <string> [-IdentityName]
<string> [-SystemName] <string> [-Namespace] <string> [-AccountName] <string> [-PassException] [-
Trace] [-RunAs <pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.
- **IdentityName:** The identity creating the new self-recovery rule.
- **SystemName:** The target system.
- **NameSpace:** The target namespace.
- **AccountName** The target account name associated with the `SystemName` and namespace.

Example Request

```
New-LSDelegationPermissionForSelfRecovery -AuthenticationToken $tok -IdentityName $id -SystemName
$sn -Namespace $ns - AccountName $an
```

Output Success

The output states the self-recovery permission was created.

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Delegation already Exists**

  Self-recovery permission for the identity already exists.

PowerShell: Remove-LSDelegationIdentity

Remove-LSDelegationIdentity removes a specific delegation identity as well as any permissions associated with the identity.

Permissions Required

- Manage Delegations

Related Commands

- **SOAP:** DelegationOps_DeleteIdentity
- **REST:** Delegation/Identity

Syntax

```
Remove-LSDelegationIdentity [-AuthenticationToken] <string> [-IdentityName] <string> [-
PassException] [-Trace] [-RunAs <pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.
- **IdentityName:** The name of the target identity being deleted.

Example Request

```
Remove-LSDelegationIdentity -AuthenticationToken $tok3 - IdentityName $idn3
```

Output Success

The output states the identity has been removed.

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Invalid identity specified**

  The delegation identity could not be found.

PowerShell: Remove-LSDelegationManagementSetFromIdentity

Remove-LSDelegationManagementSetFromIdentity modifies global delegation permissions to remove management set delegations from a target identity.

Permissions Required

- Manage Delegations

Related Commands

- **SOAP:** DelegationOps_RemoveManagedGroupFromIdentity
- **REST:** Delegation/ManagementSet (DELETE)

Syntax

```
Remove-LSDelegationManagementSetFromIdentity [-AuthenticationToken] <string> [-IdentityName]
<string> [-ManagementSet] <string> [-PassException] [-Trace] [-RunAs <pscredential>]
[<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.
- **IdentityName:** The name of the target identity.
- **ManagementSet:** The name of the management set to remove from the `IdentityName`.

Example Request

```
Remove-LSDelegationManagementSetFromIdentity -AuthenticationToken $tok -IdentityName $id -
ManagementSet $msn
```

Output Success

The output states the identity is no longer managing the management set.

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Invalid identity name**

  The database call did not return any data, or the file was not found.

- **No existing association of identity and management set** -or- **Invalid management set name**

  The identity is not managing the management set, or the deletion process failed.

PowerShell: Remove-LSDelegationPermissionAccountMask

Remove-LSDelegationPermissionAccountMask removes an account mask associated with a target identity.

Permissions Required

- Manage Delegations

Related Commands

- **SOAP:** DelegationOps_AccountMaskPermission_Remove
- **REST:** Delegation/AccountMask (DELETE)

Syntax

```
Remove-LSDelegationPermissionAccountMask [-AuthenticationToken] <string> [-IdentityName] <string>
[-AccountMask] <string> [-PassException] [-Trace] [-RunAs <pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.
- **IdentityName:** The target identity.
- **AccountMask:** The account mask to disassociate from the target identity.

Example Request

```
Remove-LSDelegationPermissionAccountMask -AuthenticationToken $tok -IdentityName $id -
AccountMask $am
```

Output Success

The output states the account mask associated with the identity was removed.

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Invalid identity name**

  The database call did not return any data, or the file was not found.

- **Invalid account mask**

  The account mask does not exist for the identity.

PowerShell: Remove-LSDelegationPermissionForSelfRecovery

Remove-LSDelegationPermissionForSelfRecovery removes a self-recovery permission associated with a target identity.

Permissions Required

- Manage Delegations

Related Commands

- **SOAP:** DelegationOps_SelfRecoveryPermission_Remove
- **REST:** Delegation/SelfRecoveryPermission (DELETE)

Syntax

```
Remove-LSDelegationPermissionForSelfRecovery [-AuthenticationToken] <string> [-IdentityName]
<string> [-SystemName] <string> [-Namespace] <string> [-AccountName] <string> [-PassException] [-
Trace] [-RunAs <pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.
- **IdentityName:** The target identity needing the permission removed.
- **SystemName:** The target system name defined in the self-recovery permission.
- **NameSpace:** The target namespace defined in the self-recovery permission.
- **AccountName:** The target account defined in the self-recovery permission.

Example Request

```
Remove-LSDelegationPermissionForSelfRecovery -AuthenticationToken $tok -IdentityName $id -
SystemName $sn -Namespace $ns - AccountName $an
```

Output Success

The output states the self-recovery permission has been removed for the identity.

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Invalid AccountName** -or- **Invalid SystemName** -or- **NameSpace** -or- **AccountName**

  No self-recovery permissions exist for the identity.

PowerShell: Remove-LSDelegationPermissionOnAccount

Remove-LSDelegationPermissionOnAccount removes a per-account permission.

Permissions Required

- Manage Delegations

Related Commands

- **SOAP:** DelegationOps_RemovePermissionOnAccount
- **REST:** Delegation/StoredCredential (DELETE)

Syntax

```
Remove-LSDelegationPermissionOnAccount [-AuthenticationToken] <string> [-IdentityName] <string>
[-SystemName] <string> [-Namespace] <string> [-AccountName] <string> [-PassException] [-Trace] [-
RunAs <pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.
- **IdentityName:** The target identity needing the permission removed.
- **SystemName:** The target system name defined in the per-account permission.
- **NameSpace:** The target namespace defined in the per-account permission.
- **AccountName:** The target account defined in the per-account permission.

Example Request

```
Remove-LSDelegationPermissionOnAccount -AuthenticationToken $tok -IdentityName $id -SystemName
$sn -Namespace $ns - AccountName $an
```

Output Success

The output states the per-account permissions was deleted.

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Invalid AccountName** -or- **InvalidSystemName** -or- **NameSpace** -or- **AccountName**

  The specified permission could not be found.

PowerShell: Remove-LSDelegationPermissionOnJob

Remove-LSDelegationPermissionOnJob removes permissions for a specific job from an identity.

Permissions Required

- Manage Delegations

Related Commands

- **SOAP:** DelegationOps_RemovePermissionOnJob
- **REST:** Delegation/Job (DELETE)

Syntax

```
Remove-LSDelegationPermissionOnJob [-AuthenticationToken] <string> [-IdentityName] <string> [-
JobID] <string> [-PassException] [-Trace] [-RunAs <pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.
- **IdentityName:** The target identity.
- **JobID:** The target job.

Example Request

```
Remove-LSDelegationPermissionOnJob -AuthenticationToken $tok -IdentityName $id - JobID $jid
```

Output Success

The output states permissions for the job were removed from the identity.

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Invalid AccountName**

  Log in failed, or the username provided was not found.

PowerShell: Remove-LSDelegationPermissionOnManagementSet

Remove-LSDelegationPermissionOnManagementSet removes the per-management set permissions assigned to a target identity for specific management set.

Permissions Required

- Manage Delegations

Related Commands

- **SOAP:** DelegationOps_RemovePermissionOnManagementSet
- **REST:** Delegation/ManagementSet (DELETE)

Syntax

```
Remove-LSDelegationPermissionOnManagementSet [-AuthenticationToken] <string> [-IdentityName]
<string> [-ManagementSet] <string> [-PassException] [-Trace] [-RunAs <pscredential>]
[<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.
- **IdentityName:** The target identity.
- **ManagementSet:** The target management set.

Example Request

```
Remove-LSDelegationPermissionOnManagementSet -AuthenticationToken $tok -IdentityName $id -
ManagementSet $msn
```

Output Success

The output states management set permissions were removed from the identity.

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Invalid identity** -or- **Invalid management set**

  The specific permission could not be found.

PowerShell: Remove-LSDelegationPermissionOnSharedCredentialList

Remove-LSDelegationPermissionOnSharedCredentialList removes an identity's permissions from a shared credential list.

Permissions Required

- Manage permissions for the target shared credential list

Related Commands

- **SOAP:** DelegationOps_RemovePermissionForSharedCredentialList
- **REST:** Delegation/SharedCredentialList (DELETE)

Syntax

```
Remove-LSDelegationPermissionOnSharedCredentialList [-AuthenticationToken] <string> [-
IdentityName] <string> [-SharedCredentialList] <string> [-PassException] [-Trace] [-RunAs
<pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.
- **IdentityName:** The target identity.
- **SharedCredentialList:** The target management set.

Example Request

```
Remove-LSDelegationPermissionOnSharedCredentialList -AuthenticationToken $tok -IdentityName $id -
SharedCredentialList $scl
```

Output Success

The output states the identity's permissions were removed from the shared credential list.

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Invalid identity**

  Log in failed, or the username provided was not found.

- **Invalid shared credential list**

  The shared credential list could not be found.

PowerShell: Remove-LSDelegationPermissionOnSystem

Remove-LSDelegationPermissionOnSystem removes a per-system permission.

Permissions Required

- Manage Delegations

Related Commands

- **SOAP:** DelegationOps_RemovePermissionOnSystem
- **REST:** Delegation/System (DELETE)

Syntax

```
Remove-LSDelegationPermissionOnSystem [-AuthenticationToken] <string> [-IdentityName] <string> [-
SystemName] <string> [-PassException] [-Trace] [-RunAs <pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.
- **IdentityName:** The target identity.
- **SystemName:** The target system.

Example Request

```
Remove-LSDelegationPermissionOnSystem -AuthenticationToken $tok -IdentityName $id - SystemName
$sn
```

Output Success

The output states the permission was removed from the identity.

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Invalid identity -or- Invalid system**

  The specific permission could not be found.

PowerShell: Remove-LSDelegationPermissionRoleMapping

Remove-LSDelegationPermissionRoleMapping removes usernames from identity role mappings.

Permissions Required

- Manage Delegations

Related Commands

- **SOAP:** DelegationOps_RoleMappingPermission_Remove
- **REST:** Delegation/Identity/Role (DELETE)

Syntax

```
Remove-LSDelegationPermissionRoleMapping [-AuthenticationToken] <string> [-IdentityName] <string>
[-AuthenticationServer] <string> [-CredentialName] <string> [-PassException] [-Trace] [-RunAs
<pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.
- **IdentityName:** The name of the identity role needing to be modified.
- **AuthenticationServer:** The authentication server entry associated with the CredentialName.
- **CredentialName:** The name of the account to remove from the identity-role.

Example Request

```
Remove-LSDelegationPermissionRoleMapping -AuthenticationToken $tok -IdentityName $in -
AuthenticationServer $as - CredentialName $cn
```

Output Success

The output states the permission for the role was removed from the authentication server.

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Invalid identity** -or- **Invalid authentication server** -or- **Invalid credential**

  Permission for the on authenticating server does not exist for the credential.

PowerShell: Set-LSDelegationIdentitySettings

Set-LSDelegationIdentitySettings configures the web application's global delegations for an existing target identity. This function replaces the existing settings and permissions for a delegation identity with those specified in the `IdentitySettings` parameter.

Permissions Required

- Manage Delegations

Related Commands

- **SOAP:** DelegationOps_SetIdentitySettings
- **REST:** Delegation/Identity (PUT)

Syntax

```
Set-LSDelegationIdentitySettings [-AuthenticationToken] <string> [-IdentitySettings]
<DelegationIdentity> [-PassException] [-Trace] [-RunAs <pscredential>] [<CommonParameters>]
```

Parameters

Any permissions not included should be set to 0 (false).

- **AuthenticationToken:** Authentication token of the calling user.
- **IdentitySettings:** Includes one enumerated type and multiple values.
  - **AccountName:** (string) The name of the identity.
  - **IsDomainAccount:** Enumerated value eManagerType. Available values are:
    - MANAGER_TYPE_EXPLICIT_USER
    - MANAGER_TYPE_DOMAIN_USER
    - MANAGER_TYPE_DOMAIN_GROUP
    - MANAGER_TYPE_SELF_RECOVERY
    - MANAGER_TYPE_ROLE
    - MANAGER_TYPE_RADIUS
    - MANAGER_TYPE_CERTFICATE
    - MANAGER_TYPE_LDAP_USER
  - **AlertOnRecovery:** *(boolean)* Used in conjunction with the `EmailAddress`. Set to 1 to enable email notifications when a password or access request is made for a system or account the identity can grant access requests for.
  - **AlertOnRequest:** *(boolean)* Used in conjunction with the `EmailAddress`. Set to 1 to enable email notifications when a password or access request is made for a system or account the identity can grant access requests for.
  - **DisplayName:** *(string)* Sets the display name of the account. If omitted uses the `AccountName`.
  - **EmailAddress:** *(string)* Email address for the identity.
  - **Password:** *(string)* Will set the password for an explicit account.
  - **PermissionAccessRemoteSessions:** *(boolean)* 0 or 1 to disable or enable access to remote sessions (RDP & SSH/Telnet).

- **PermissionAddPasswordsForManagedSystems:** *(boolean)* 0 or 1 to disable or enable adding/editing stored managed passwords.
- **PermissionAllAccess:** *(boolean)* 0 or 1 to disable or enable All Access.
- **PermissionCreateRefreshSystemJob:** *(boolean)* 0 or 1 to disable or enable creating new jobs.
- **PermissionEditDelegation:** *(boolean)* 0 or 1 to disable or enable managing delegations.
- **PermissionEditPasswordLists:** *(boolean)* 0 or 1 to disable or enable editing of passwords in password lists.
- **PermissionEditStoredPasswords:** *(boolean)* 0 or 1 to disable or enable editing stored passwords.
- **PermissionElevateAccountPermissions:** *(boolean)* 0 or 1 to disable or enable Elevate Account (self-elevation).
- **PermissionElevateAnyAccountPermissions:** *(boolean)* 0 or 1 to disable or enable Elevate Any Account.
- **PermissionGrantPasswordRequests:** *(boolean)* 0 or 1 to disable or enable Grant Password Requests.
- **PermissionIgnorePasswordCheckout:** *(boolean)* 0 or 1 to disable or enable Ignore Password Checkout (programmatic access only).
- **PermissionLogon:** *(boolean)* 0 or 1 to disable or enable web logon.
- **PermissionPersonalStore:** *(boolean)* 0 or 1 to disable or enable access to the personal password store.
- **PermissionRequestPasswords:** *(boolean)* 0 or 1 to disable or enable Request Passwords.
- **PermissionRequestRemoteAccess:** *(boolean)* 0 or 1 to disable or enable Request Passwords.
- **PermissionRequireOATH:** *(boolean)* 0 or 1 to disable or enable requirement for OATH two factor authentication.
- **PermissionRequireRSASecurID:** *(boolean)* 0 or 1 to disable or enable requirement of two factor authentication.
- **PermissionSelfRecovery:** *(boolean)* 0 or 1 to disable or enable access to Self-Recovery.
- **PermissionViewAccounts:** *(boolean)* 0 or 1 to disable or enable View Accounts.
- **PermissionViewDashboards:** *(boolean)* 0 or 1 to disable or enable access to the dashboards (when enabled).
- **PermissionViewDelegation:** *(boolean)* 0 or 1 to disable or enable Viewing Delegations.
- **PermissionViewFileStore:** *(boolean)* 0 or 1 to disable or enable access to File Repository.
- **PermissionViewJobs:** *(boolean)* 0 or 1 to disable or enable View Jobs.
- **PermissionViewPasswordActivity:** *(boolean)* 0 or 1 to disable or enable access to the Password Activity.
- **PermissionViewPasswordHistory:** *(boolean)* 0 or 1 to disable or enable access to managed Password History.
- **PermissionViewPasswords:** *(boolean)* 0 or 1 to disable or enable Recover Passwords.
- **PermissionViewScheduler:** *(boolean)* 0 or 1 to disable or enable Manage Scheduled Jobs.
- **PermissionViewSystems:** *(boolean)* 0 or 1 to disable or enable View Systems.
- **PermissionViewWebLogs:** *(boolean)* 0 or 1 to disable or enable View Web Logs.

Example Request

```
#Construct the enumerated value first
$MyUserY = New-Object -TypeName LSClientAgentCommandlets.RouletteWebService.DelegationIdentity
$MyUserY.IsDomainAccount = [LSClientAgentCommandlets.RouletteWebService.EManagerType]::MANAGER_TYPE_
DOMAIN_USER
#The next step is to fill in the remaining non-enumerated values to the object
$MyUserY.AccountName = "demo\user1"
$MyUserY.IsDomainAccount = [LSClientAgentCommandlets.RouletteWebService.EManagerType]::MANAGER_TYPE_
DOMAIN_USER
$MyUserY.EmailAddress = "corky@lsds.int"
$MyUserY.AlertOnRequest = 1
```

```
$MyUserY.PermissionLogon = 1
$MyUserY.DisplayName = "lsds\corky"
#Update the target identity is:
Set-LSDelegationIdentitySettings -AuthenticationToken $tok -IdentitySettings $MyUserY
```

Output Success

The output states the identity had permissions added and removed. Each permission added or removed is listed in a series, separated by commas.

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Invalid Identity**

  The identity could not be found, or the editing process failed.

PowerShell: Set-LSDelegationPermissionAccountMask

Set-LSDelegationPermissionAccountMask creates an account mask for a target identity.

Permissions Required

- Manage Delegations

Related Commands

- **SOAP:** DelegationOps_AccountMaskPermission_Add
- **REST:** Delegation/AccountMask (POST)

Syntax

```
Set-LSDelegationPermissionAccountMask [-AuthenticationToken] <string> [-IdentityName] <string> [-
AccountMask] <string> [-PassException] [-Trace] [-RunAs <pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.
- **IdentityName:** The target identity.
- **AccountMask:** The new account mask to associate with the identity.

Example Request

```
Set-LSDelegationPermissionAccountMask -AuthenticationToken $tok -IdentityName $id - AccountMask
$mask
```

Output Success

The output states an account mask was created for the identity.

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Invalid identity**

  The database call did not return any data, or the file was not found.

- **Account mask already exists**

  An account mask for the identity already exists.

PowerShell: Set-LSDelegationPermissionForIdentityOnFile

Set-LSDelegationPermissionForIdentityOnFile adds permissions to a file in the file store for an identity. If the identity already has permissions, those permissions are replaced.

Permissions Required

- Change permissions for the target file

Related Commands

- **SOAP:** DelegationOps_StoredFile_SetPermissions
- **REST:** Delegation/File (PUT)

Syntax

```
Set-LSDelegationPermissionForIdentityOnFile [-AuthenticationToken] <string> [-FilePermission]
<DelegationFilePermission> [-PassException] [-Trace] [-RunAs <pscredential>] [<CommonParameters>]
```

Parameters

Any permissions not included should be set to 0 (false).

- **AuthenticationToken:** Authentication token of the calling user.
- **IdentityName:** (string) Name of identity to modify /add permissions for.
- **DelegationFilePermission:** An object containing the file permissions.
    - **FileName:** (string) The name of the file.
    - **PermissionDelegate:** (boolean) 0 or 1 to disable or enable managing delegations for the file.
    - **PermissionDelete:** (boolean) 0 or 1 to disable or enable deleting the file.
    - **PermissionDownload:** (boolean) 0 or 1 to disable or enable downloading/ checking out the file.
    - **PermissionGrant:** (boolean) 0 or 1 to disable or enable granting requests for access to the file.
    - **PermissionRequest:** (boolean) 0 or 1 to disable or enable requesting the file.
    - **PermissionUpdate:** (boolean) 0 or 1 to disable or enable updating the file with a new version.
    - **PermissionView:** (boolean) 0 or 1 to disable or enable viewing the file.

Example Request

```
#Create the file permissions object and set permissions
$FilePerm = New-Object -TypeName
LSClientAgentCommandlets.RouletteWebService.DelegationFilePermission
$FilePerm.FileName = $fn
$FilePerm.IdentityName = $id
$FilePerm.PermissionView = 1
$FilePerm.PermissionDownload = 1
$FilePerm.PermissionDelete = 0
$FilePerm.PermissionUpdate = 0
```

```
$FilePerm.PermissionDelegate = 0
$FilePerm.PermissionRequest = 0
$FilePerm.PermissionGrant = 1
Set-LSDelegationPermissionForIdentityOnFile -AuthenticationToken $tok -FilePermission $FilePerm
```

Output Success

The output states the file permissoins were updated.

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Invalid identity**

  Log in failed, or the username provided was not found.

- **Invalid file or no permissions**

  No files were found with this name, or the appropriate permissions have not been granted to the view file.

PowerShell: Set-LSDelegationPermissionOnAccount

Set-LSDelegationPermissionOnAccount adds or updates per-account delegations. A stored password for the target system, account, and namespace must already exist to apply the per-account permissions.

Permissions Required

- Manage Delegations

Related Commands

- **SOAP:** DelegationOps_SetPermissionOnAccount
- **REST:** Delegation/StoredCredential (POST)

Syntax

```
Set-LSDelegationPermissionOnAccount [-AuthenticationToken] <string> [-PermissionOnAccount]
<DelegationPermissionOnAccount> [-PassException] [-Trace] [-RunAs <pscredential>]
[<CommonParameters>]
```

Parameters

Any permissions not included should be set to 0 (false).

- **AuthenticationToken:** Authentication token of the calling user.
- **DelegationPermissionOnAccount:**
  - **IdentityName:** *(string)* The name of the target identity.
  - **SystemName:** *(string)* The name of the target system.
  - **Namespace:** *(string)* The namespace of the target system. See Namespace Values for a list of pre-defined values.
  - **AccountName:** *(string)* The name of the target account on the target system.
  - **AlertForIncident:** *(boolean)* 0 or 1 to disable or enable alert emails during a password request for the target account when INCIDENT is selected.
  - **AlertForChange:** *(boolean)* 0 or 1 to disable or enable alert emails during a password request for the target account when CHANGE is selected.
  - **PermissionAllowRemoteSessions:** *(boolean)* 0 or 1 to disable or enable RDP/SSH/Telnet access with the target account (web site).
  - **PermissionGrantPasswordRequests:** *(boolean)* 0 or 1 to disable or enable granting password requests for the target account.
  - **PermissionRequestPasswords:** *(boolean)* 0 or 1 to disable or enable requesting access to the password.
  - **PermissionRequestRemoteAccess:** *(boolean)* 0 or 1 to disable or enable requesting remote access with the account.
  - **PermissionViewAccounts:** *(boolean)* 0 or 1 to disable or enable viewing of the account. This value should be set to 1 in order to view the account in the web site.
  - **PermissionViewPasswords:** *(boolean)* 0 or 1 to disable or enable recovering of the password.

Example Request

```
#Create the account permissions object and set permissions
$PAPerm = New-Object -TypeName
LSClientAgentCommandlets.RouletteWebService.DelegationPermissionOnAccount
$PAPerm.IdentityName = $id
$PAPerm.SystemName = $sn
$PAPerm.NameSpace = $ns
$PAPerm.AccountName = $an
$PAPerm.AlertForIncident = 0
$PAPerm.AlertForChange = 1
$PAPerm.PermissionViewAccounts = 1
$PAPerm.PermissionViewPasswords = 1
$PAPerm.PermissionRequestPasswords = 0
$PAPerm.PermissionRequestRemoteAccess = 0
$PAPerm.PermissionGrantPasswordRequests = 1
$PAPerm.PermissionAllowRemoteSessions = 1
Set-LSDelegationPermissionOnAccount -AuthenticationToken $tok -PermissionOnAccount $PAPerm
```

Output Success

The output states the delegation permission the identity were updated.

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Invalid identity**

  Log in failed, or the username provided was not found.

- **Invalid SystemName** -or- **Namespace** -or- **AccountName**

  The specified password was not found.

PowerShell: Set-LSDelegationPermissionOnJob

Set-LSDelegationPermissionOnJob grants full control of a job to an identity.

Permissions Required

- Manage Delegations

Related Commands

- **SOAP:** DelegationOps_SetPermissionOnJob
- **REST:** Delegation/Job (POST)

Syntax

```
Set-LSDelegationPermissionOnJob [-AuthenticationToken] <string> [-IdentityName] <string> [-JobID]
<string> [-PassException] [-Trace] [-RunAs <pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.
- **JobID:** The ID of the job.
- **IdentityName:** The target identity.

Example Request

```
Set-LSDelegationPermissionOnJob -AuthenticationToken $tok -IdentityName $id - JobID $jid
```

Output Success

The output states the delegation permission for the identity was updated.

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Invalid identity**

  Log in failed, or the username provided was not found.

PowerShell: Set-LSDelegationPermissionOnManagementSet

Set-LSDelegationPermissionOnManagementSet adds or updates per-management set delegations. The management set must already exist to apply the per-management set permissions.

Permissions Required

- Manage Delegations

Related Commands

- **SOAP:** DelegationOps_SetPermissionOnManagementSet
- **REST:** Delegation/Identity/ManagementSet (POST)

Syntax

```
Set-LSDelegationPermissionOnManagementSet [-AuthenticationToken] <string> [-
PermissionOnManagementSet] <DelegationPermissionOnManagementSet> [-PassException] [-Trace] [-
RunAs <pscredential>] [<CommonParameters>]
```

Parameters

Any permissions not included should be set to 0 (false).

- **AuthenticationToken:** Authentication token of the calling user.
- **DelegationPermissionOnManagementSet:**
  - **AlertForChange:** (boolean) 0 or 1 to disable or enable alert emails during a password request for the target account when CHANGE is selected.
  - **AlertForIncident:** (boolean) 0 or 1 to disable or enable alert emails during a password request for the target account when INCIDENT is selected.
  - **IdentityName:** Name of the target identity.
  - **ManagementSetName:** Name of the target management set.
  - **PermissionAllowRemoteSessions:** (boolean) 0 or 1 to disable or enable RDP/SSH/Telnet access to the target system (web site).
  - **PermissionChangeGroupMembership:** (boolean) 0 or 1 to disable or enable adding/removing systems to/from the management set.
  - **PermissionElevateAccountPermissions:** (boolean) 0 or 1 to enable the user for self elevation.
  - **PermissionGrantPasswordRequests:** 0 or 1 to disable or enable granting password requests for the target account.
  - **PermissionRequestPasswords:** (boolean) 0 or 1 to disable or enable requesting access to the password.
  - **PermissionRequestRemoteAccess:** (boolean) 0 or 1 to disable or enable requesting RDP/SSH/Telnet access to the target system (web site).
  - **PermissionViewAccounts:** (boolean) 0 or 1 to disable or enable viewing of the account. This value should be set to 1 in order to view the account in the web site.
  - **PermissionViewPasswords:** (boolean) 0 or 1 to disable or enable recovering of the password.

- **PermissionViewSystems:** (boolean) 0 or 1 to disable or enable viewing of the systems. This value should be set to 1 in order to view the systems in the web site. If this is set to 0, then the user will also be unable to view accounts in the web site regardless of the permission to do so.

Example Request

```
#Create the mgmt set permissions object and set permissions
$PMSPerm = New-Object -TypeName
LSClientAgentCommandlets.RouletteWebService.DelegationPermissionOnManagementSet
$PMSPerm.IdentityName = "lsds\cletus"
$PMSPerm.ManagementSetName = "Web Servers"
$PMSPerm.AlertForIncident = 0
$PMSPerm.AlertForChange = 1
$PMSPerm.PermissionViewAccounts = 1
$PMSPerm.PermissionViewSystems = 1
$PMSPerm.PermissionViewPasswords = 1
$PMSPerm.PermissionRequestPasswords = 0
$PMSPerm.PermissionGrantPasswordRequests = 1
$PMSPerm.PermissionAllowRemoteSessions = 1
$PMSPerm.PermissionRequestRemoteAccess = 0
$PMSPerm.PermissionElevateAccountPermissions = 1
$PMSPerm.PermissionChangeGroupMembership = 0
$x = Set-LSDelegationPermissionOnManagementSet -AuthenticationToken $tok -PermissionOnManagementSet
$PMSPerm
```

Output Success

The output states the delegation permissions for the identity were updated.

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Invalid identity**

  Log in failed, or the username provided was not found.

- **Invalid management set**

  The management set specified could not be found.

PowerShell: Set-LSDelegationPermissionOnSharedCredentialList

Set-LSDelegationPermissionOnSharedCredentialList adds or updates shared credential list delegations. The shared credential list set must already exist to apply the permissions.

Permissions Required

Either:

- Manage Permissions on the list
- Manage External Lists

Related Commands

- **SOAP:** DelegationOps_SetPermissionsForSharedCredentialList
- **REST:** Delegation/SharedCredentialList (POST)

Syntax

```
Set-LSDelegationPermissionOnSharedCredentialList [-AuthenticationToken] <string> [-
PermissionOnSharedCredentialList] <DelegationSharedCredentialListPermission> [-PassException] [-
Trace] [-RunAs <pscredential>] [<CommonParameters>]
```

Parameters

Any permissions not included should be set to 0 (false).

- **AuthenticationToken:** Authentication token of the calling user.
- **DelegationSharedCredentialListPermission:**
    - **CredentialListName:** The name of the shared credential list.
    - **IdentityName:** The name of the target identity.
    - **PermissionAddPassword:** *(boolean)* 0 or 1 to disable or enable adding passwords to the target SCL.
    - **PermissionChangeDelegation:** *(boolean)* 0 or 1 to disable or enable changing permissions on the target SCL.
    - **PermissionDeletePassword:** *(boolean)* 0 or 1 to disable or enable deleting passwords from the target SCL.
    - **PermissionEditPassword:** *(boolean)* 0 or 1 to disable or enable modifying passwords in the target SCL.
    - **PermissionGrantRequest:** *(boolean)* 0 or 1 to disable or enable granting requests to passwords in the target SCL.
    - **PermissionRecoverPassword:** *(boolean)* 0 or 1 to disable or enable viewing passwords from the target SCL.
    - **PermissionRequestPassword:** *(boolean)* 0 or 1 to disable or enable requesting passwords from the target SCL.
    - **PermissionViewList:** *(boolean)* 0 or 1 to disable or enable viewing the list of credentials stored in the SCL.

Example Request

```
#Create the SCL permissions object and set permissions
$PSCLPerm = New-Object -TypeName
LSClientAgentCommandlets.RouletteWebService.DelegationSharedCredentialListPermission
```

```
$PSCLPerm.CredentialListName = $scl
$PSCLPerm.IdentityName = $id
$PSCLPerm.PermissionChangeDelegation = 1
$PSCLPerm.PermissionAddPassword = 1
$PSCLPerm.PermissionEditPassword = 0
$PSCLPerm.PermissionDeletePassword = 1
$PSCLPerm.PermissionGrantRequest = 1
$PSCLPerm.PermissionRecoverPassword = 1
$PSCLPerm.PermissionRequestPassword = 0
$PSCLPerm.PermissionViewList = 1
Set-LSDelegationPermissionOnSharedCredentialList -AuthenticationToken $tok -
PermissionOnSharedCredentialList $PSCLPerm
```

Output Success

The output states the delegations for the shared credential list have been updated.

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Invalid list**

  The shared credential list could not be found.

PowerShell: Set-LSDelegationPermissionOnSystem

Set-LSDelegationPermissionOnSystem adds or updates per-system delegations. The system must already exist to apply the per-system permissions.

Permissions Required

- Manage delegations

Related Commands

- **SOAP:** DelegationOps_SetPermissionsOnSystem
- **REST:** Delegation/System (POST)

Syntax

```
Set-LSDelegationPermissionOnSystem [-AuthenticationToken] <string> [-PermissionOnSystem]
<DelegationPermissionOnSystem> [-PassException] [-Trace] [-RunAs <pscredential>]
[<CommonParameters>]
```

Parameters

Any permissions not included should be set to 0 (false).

- **AuthenticationToken:** Authentication token of the calling user.
- **DelegationPermissionOnSystem:**
    - **AlertForChange:** *(boolean)* 0 or 1 to disable or enable alert emails during a password request for the target account when CHANGE is selected.
    - **AlertForIncident:** *(boolean)* 0 or 1 to disable or enable alert emails during a password request for the target account when INCIDENT is selected.
    - **IdentityName:** Name of the target identity.
    - **PermissionAllowRemoteSessions:** *(boolean)* 0 or 1 to disable or enable RDP/SSH/Telnet access with the target account (web site).
    - **PermissionElevateAccountPermissions:** *(boolean)* 0 or 1 to disable or enable self-service account elevation.
    - **PermissionGrantPasswordRequests:** *(boolean)* 0 or 1 to disable or enable granting password requests for the target account.
    - **PermissionRequestPasswords:** *(boolean)* 0 or 1 to disable or enable requesting access to the password.
    - **PermissionRequestRemoteAccess:** *(boolean)* 0 or 1 to disable or enable requesting remote access to the system using RDP/SSH/Telnet access with the target account (web site).
    - **RemoteAccessPermissionViewAccounts:** *(boolean)* 0 or 1 to disable or enable viewing of the account. This value should be set to 1 in order to view the account in the web site.
    - **PermissionViewPasswords:** *(boolean)* 0 or 1 to disable or enable recovering of the password.
    - **PermissionViewSystems:** *(boolean)* 0 or 1 to disable or enable viewing the system. The value should be set to 1 in order to view the system and its accounts in the web interface.
    - **SystemName:** Name of the target system.

Example Request

```
#Create the system permissions object and set permissions
$PSPerm = New-Object -TypeName
LSClientAgentCommandlets.RouletteWebService.DelegationPermissionOnSystem
$PSPerm.AlertForChange = 1
$PSPerm.AlertForIncident = 0
$PSPerm.IdentityName = $id
$PSPerm.PermissionAllowRemoteSessions = 1
$PSPerm.PermissionElevateAccountPermissions = 1
$PSPerm.PermissionGrantPasswordRequests = 1
$PSPerm.PermissionRequestPasswords = 0
$PSPerm.PermissionRequestRemoteAccess = 0
$PSPerm.PermissionViewAccounts = 1
$PSPerm.PermissionViewPasswords = 1
$PSPerm.PermissionViewSystems = 1
$PSPerm.SystemName = $sn
Set-LSDelegationPermissionOnSystem -AuthenticationToken $tok - PermissionOnSystem $PSPerm
```

Output Success

The output states the delegation permission for the identity has been updated.

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Invalid identity**

  Log in failed, or the username provided was not found.

- **Invalid system**

  The system information could not be found.

PowerShell: Set-LSDelegationPermissionRoleMapping

Set-LSDelegationPermissionRoleMapping adds a user from an authentication server (LDAP) to a role.

Permissions Required

- Manage delegations

Related Commands

- **SOAP:** DelegationOps_RoleMappingPermission_Add
- **REST:** Delegation/Identity/Role (POST)

Syntax

```
Set-LSDelegationPermissionRoleMapping [-AuthenticationToken] <string> [-IdentityName] <string> [-
AuthenticationServer] <string> [-CredentialName] <string> [-PassException] [-Trace] [-RunAs
<pscredential>] [<CommonParameters>]
```

Parameters

- **AuthenticationToken:** Authentication token of the calling user.
- **AuthenticationServer:** The source authentication server entry.
- **IdentityName:** The target identity role.
- **CredentialName:** The user to add from the source authentication server to the identity role.

Example Request

```
Set-LSDelegationPermissionRoleMapping -AuthenticationToken $tok -IdentityName $in -
AuthenticationServer $as - CredentialName $id
```

Output Success

The output states the permission for the role was created.

Output Fail

- **Session previously expired**

  The session was invalid, or a duplicate web session was detected for this identity.

- **Invalid authentication token**

  An invalid authentication token was used, or the token was not found.

- **Invalid role**

  The role could not be found.

- **Invalid authentication server**

  The authentication server could not be found.