Powershell Scripting Guide

PV6.0.0 SV100

Table of Contents

Overview 1

Using PowerShell Script Editor 2

Configuring PowerShell to Use the OneStream Client API 2

Learning PowerShell 4

Using the OneStream Client API in a PowerShell Script 4

Overview

PowerShell is an object-oriented programming language and interactive command line shell for Microsoft Windows. It was designed to automate system tasks, such as batch processing, and create systems management tools for commonly implemented processes. PowerShell includes more than 130 standard command line tools for functions that formerly required users to create scripts in VB, VBScript or C#.

PowerShell offers a variety of ways to automate tasks which include:

Cmdlets

Very small .NET classes that appear as system commands

Scripts

Combinations of cmdlets and associated logic

Executables

Standalone tools

Instantiation of standard .NET classes

PowerShell integrates with the .NET environment and can also be embedded in other applications. Over one hundred cmdlets are included and can be used separately or combined with others to automate more complex tasks. Users can also create and share cmdlets.

PowerShell is built into Windows Server 2008 and Windows 7 and included as an optionally installed feature. In addition, the Windows Task Scheduler can be used to automate PowerShell script execution.

Using PowerShell Script Editor

To run PowerShell on Windows 7 or a later version, begin typing PowerShell at the Windows Start Button, or go to All Programs |Accessories |Windows PowerShell.

There are two programs used to interact with PowerShell:

Windows PowerShell ISE

This is the integrated scripting environment, or Script Editor. The editor allows users to type PowerShell commands as well as edit and run PowerShell script files which are text files with a ps1 extension.

Windows PowerShell

This program is a command line execution tool that looks like a DOS prompt. It allows a user to run a command or a script file, but it does not perform editing/creating scripts as well.

Configuring PowerShell to Use the OneStream Client API

Before PowerShell can be used to interact with the OneStream XF client API, three configuration steps must be completed on each machine used for PowerShell script execution. First, execute a PowerShell command enabling the execution of unsigned scripts. Second, create or alter the PowerShell execution and IDE configuration files, so the script engine understands how to use the .Net Framework v4.0. Finally, OneStream Studio must be installed on each machine executing PowerShell scripts.

Allowing Execution of Unsigned Scripts

The first time this runs, the following line needs to run at a PowerShell command prompt. This will allow PowerShell to run unsigned scripts created on the local computer.

```
set-executionpolicy remotesigned
```

Configuration for .Net Framework v.4.0

In order to use the OneStreamClientApi with PowerShell, PowerShell needs to be configured to use the .NET Framework v4.0. In order to do this, modify or create two configuration files if they do not already exist.

Configuration File Folder

```
C:\Windows\System32\WindowsPowerShell\v1.0
```

File 1 (Configuration for Execution)

```
powershell.exe.config
```

File 2 (Configuration for IDE)

```
powershell_ise.exe.config
```

Required File Contents (Must be added to each configuration file)

```
<?xml version="1.0"?>
<configuration>
<startup useLegacyV2RuntimeActivationPolicy="true">
<supportedRuntime version="v4.0.30319"/>
<supportedRuntime version="v2.0.50727"/>
</startup>
</configuration>
```

Refer to the following web resources for more information on this process.

http://stackoverflow.com/questions/2094694/how-can-i-run-powershell-with-the-net-4-runtime http://tfl09.blogspot.com/2010/08/using-newer-versions-of-net-with.html.

Install OneStream Studio

The OneStream Studio product installation includes the Client API installation used by PowerShell scripts to interact with the OneStream server.

Learning PowerShell

Microsoft provides extensive resources to help IT professionals get the most out of PowerShell.

Refer to the following web resource in order to learn more about scripting with PowerShell.
http://technet.microsoft.com/en-us/scriptcenter/powershell.aspx

Using the OneStream Client API in a PowerShell Script

OneStream provides a client API (OneStreamClientApi) specifically designed to enable PowerShell scripts to call a OneStream XF function. This API exposes functions for authentication and Data Management. Over time, OneStream expands the number of functions exposed to this API.

> **Note:** The client API component is installed as part of the OneStream Studio installation.

This API offers a simple set of functions providing the script writer with the ability to connect to the OneStream server, authenticate, execute OneStream Data Management Sequences, and perform some basic data retrieval.

Client API Object Hierarchy

OneStreamClientAPI

LogonInfo

- Type: LogonInfo

SI

- Type: SessionInfo

Authentication

- Logon
    - Parameters:
        - string webServerUrl
        - string userName
        - string password
        - XFClientAuthenticationType clientAuthenticationType
    - Return Value:
        - LogonInfo
- Logoff

    - Parameters:
        - None
    - Return Value:
        - None

- OpenApplication
    - Parameters:
        - string application
    - Return Value:
    - LogonInfo

- LogonAndOpenApplication
    - Parameters:
        - string webServerUrl
        - string username
        - string password
        - string application
        - XFClientAuthenticationType clientAuthenticationType

        - Return Value:
            - LogonInfo
- EncryptPassword
    - Parameters:
        - string clearTextPassword
        - XFClientAuthenticationType clientAuthenticationType
    - Return Value:
        - string

DataManagement

- ExecuteSequence
    - Parameters:
        - string sequenceName
        - string customSubstVarsAsCommaSeparatedPairs
    - Return Value:
        - DataMgmtResult
- ExecuteStep

    - Parameters:
        - string dataMgmtGroupName
        - string stepName
        - string customSubstVarsAsCommaSeparatedPairs
    - Return Value:
        - DataMgmtResult

DataProvider

- GetAdoDataSetForCubeViewCommand
    - Parameters:
        - string cubeViewName
        - bool dataTablePerCubeViewRow
        - CubeViewDataTableOptions dataTableOptions
        - string resultDataTableName
        - Dictionary<string, string> customSubstVars
        - bool throwExceptionOnError
    - Return Value:
        - DataSet

GetAdoDataSetForSqlCommand

- Parameters:
    - DbLocation
    - string xfExternalDBConnectionName
    - string sqlQuery
    - string resultDataTableName
    - Dictionary<string, string> customSubstVars
    - bool throwExceptionOnError
- Return Value:
    - DataSet

GetAdoDataSetForMethodCommand

- Parameters:
    - XFCommandMethodTypeId xfCommandMethodType
    - string methodQuery
    - string resultDataTableName
    - Dictionary<string, string> customSubstVars
    - bool throwExceptionOnError

- Return Value:
    - DataSet