Active Directory Domain Database Synchronization

PowerShell Framework to Selectively Extract AD Objects to Database

Author: Hugh Scott

Version: 1.0

Date: July 5, 2019

# Copyright

This document is provided "as-is". Information and views expressed in this document, including URL and other Internet Web site references, may change without notice.

Some examples depicted herein are provided for illustration only and are fictitious. No real association or connection is intended or should be inferred.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes. You may modify this document for your internal, reference purposes.

© 2018 Microsoft Corporation. All rights reserved.

Microsoft, Active Directory, Bing, BizTalk, Forefront, Hyper-V, Internet Explorer, JScript, SharePoint, Silverlight, SQL Database, SQL Server, Visio, Visual Basic, Visual Studio, Win32, Windows, Windows Azure, Windows Intune, Windows PowerShell, Windows Server, and Windows Vista are trademarks of the Microsoft group of companies. All other trademarks are property of their respective owners.

# Version Control

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Changes** | **Comments** |
| 1.0 | 2019/07/05 | Hugh Scott | Original |  |

Table of Contents

[Copyright 2](#_Toc13230938)

[Version Control 3](#_Toc13230939)

[What’s New 5](#_Toc13230940)

[Background 6](#_Toc13230941)

[Setup Requirements and Permissions 6](#_Toc13230942)

[Server Installation Requirements 6](#_Toc13230943)

[Domain and Server Permissions 6](#_Toc13230944)

[Active Directory Domain Permissions 6](#_Toc13230945)

[SQL Server Permissions (Central Repository) 6](#_Toc13230946)

[Connectivity and Port Requirements 6](#_Toc13230947)

[MonitorDomain.ps1 7](#_Toc13230948)

[Parameters 7](#_Toc13230949)

[Active Directory Objects 7](#_Toc13230950)

[Scheduling 8](#_Toc13230951)

[General Notes on the Data Model 8](#_Toc13230952)

[Primary Keys and Unique Keys 8](#_Toc13230953)

[Active, dbAddDate, dbLastUpdate attributes 9](#_Toc13230954)

[Object Creation Time Stamp versus dbAddDate 9](#_Toc13230955)

[Use of Data for Auditing 10](#_Toc13230956)

# What’s New

March 11, 2015

Version 1.0

First release.

# Background

This script (MonitorDomain.ps1) is used to selectively extract Active Directory objects and synchronize them to a SQL Server database.

The primary purpose of the tool is to provide answers to the following types of questions:

1. How many computers do I have in my Active Directory environment, right now?
2. How many computers did I have last year?
3. How many users do I have in Active Directory and what groups are they members of?

This script is not intended to replace Active Directory, nor does is provide a full audit capability for Active Directory. It is useful for creating SQL-based reports and for reconciling objects in Active Directory with other database-hosted systems (for example, System Center Operations manager).

# Setup Requirements and Permissions

## Server Installation Requirements

Operating System:

Windows Server 2012

Windows Server 2012 R2

Windows Server 2016

Powershell Version: 4.0, 5.1

Software Requirements:

Active Directory Powershell Module

## Domain and Server Permissions

### Active Directory Domain Permissions

No special permissions are required for Active Directory. The Service account used to schedule the check process just needs to be a member of the domain (or have an established trust identity if the domain being checked is in a separate forest).

### SQL Server Permissions (Central Repository)

There are predefined roles for each schema used for the checks to be successfully performed. The service account must be a member of the pre-defined roles in order to successfully update the database.

### Connectivity and Port Requirements

Currently, port and network requirements are assessed to be:

Port 9389 : Active Directory Web Services

Port 1433 : SQL Server (or non-default port if configured)

## MonitorDomain.ps1

MonitorDomain attempts to enumerate objects in Active Directory and load these into corresponding tables in the database. This can be useful for correlating users in Active Directory with users in SQL Server and discovering new computers which may have been added to the Domain (which can then be added to the list of computers to check). Each object checked is listed below. Note that not all attributes for every object are returned.

### Parameters

#### adDomain

The Active Directory domain to which the process connects. This is best supplied in the form: <domain>.<com>. Alternate domains can be connected to as long as the user is defined (ie, a trust relationship exists).

#### adObjectType

Currently this parameter accepts a string which must contain characters in the set {domain, forest, computer, user, group, groupmember, site, subnet}. Multiple objects may be checked in a single execution.

#### syncType

The synchronization process can be set to **Full** or **Incremental** (the default value is **Full**). A Full synchronization will retrieve all of the objects of the specified class in Active Directory. An Incremental synchronization will only retrieve those objects that have been changed since the last synchronization was performed.

If an Incremental update is specified, the process first checks the [ad].[SyncStatus] table to determine when the last incremental update was performed. It then uses the date/time of **start** of the last incremental update as the basis for the whenChanged filter when retrieving objects from AD.

If the process cannot find an entry corresponding to that object, then it will automatically escalate to a Full synchronization.

Active objects in the database are not updated to “Inactive” during Incremental updates. These are only updated after a Full update.

### Active Directory Objects

#### Computers

All AD computer objects are loaded into the database into table [ad].[Computer].

#### Users

All AD user objects are loaded into the database into [ad].[User].

#### Groups

All AD groups are loaded into the database into [ad].[Group].

#### GroupMembers

All AD groups are enumerated and them members of those groups are then listed and synchronized to the database.

**NOTE:** The default group for a user is not listed in the MemberOf property for that group. This is a known issue and will be addressed at a future date.

**WARNING:** The process for enumerating and updating user group membership relationships can be VERY time consuming in a large environment with many users and many group members.

#### Sites

All AD sites are loaded into the database into [ad].[Site].

#### Subnets

All AD subnets are loaded into the database into [ad].[Subnet].

# Scheduling

The key to being able to answer these questions effectively is to schedule the various scripts to run at appropriate and predictable intervals. The following tables should serve as a guideline for the frequency with which checks should be performed:

#### Active Directory Checks Schedule (Sample)

| **Abbr** | **Frequency** | **Comment** |
| --- | --- | --- |
| Users | Daily |  |
| Computers | Daily |  |
| Groups | Daily |  |
| Group Membership | Daily\* |  |
| Sites and Subnets | Weekly |  |
| Domain | Daily |  |
| Forest | Daily |  |

(\*) Group Membership can be a very lengthy process depending on the number of groups and the number of members in each group. This process can potentially take hours for very large environments.

# General Notes on the Data Model

## Primary Keys and Unique Keys

For Active Directory objects (users, computers, groups, sites and subnets), the Unique index is on the objectGUID column (which derives its value from the ObjectGUID in Active Directory). In addition, there is a Unique Key index on the columns Domain and Name. This can potentially cause issues when, for example, a computer is deleted from AD and then re-added with the same Name. To deal specifically with this issue, there is a table object called [ad].[DeletedObject]; this is a generic entity for Users, Computers, Groups, Sites and Subnets. In the Upsert stored procedures for each of these objects, there’s an initial check to see if a duplicate object exists with a different [objectGUID]. If a duplicate exists, meta data for the object is added to [ad].[DeletedObject], the object is removed from the entity table and the new record is inserted.

Objects which are deleted and re-added with a different Distinguished Name will not cause an issue; they will get added and the old record will persist (but it should be marked InActive).

Objects that have their Distinguished Names updated (such as when the object is moved from one Organizational Unit to another) will not cause an issue. The base GUID for the object will not change and the database will be updated to reflect the new DN.

## Active, dbAddDate, dbLastUpdate attributes

Each entity contains three common attributes: Active (Boolean), dbAddDate and dbLastUpdate. The state of these three attributes is related to whether an object currently exists, when it was first discovered and when it was last updated.

An entity with an Active attribute of 1 means that -- as of the value in dbLastUpdate – that entity was found to exist in the environment. An entity with an Active attribute of 0 means that that entity was inactivated (as of dbLastUpdate) and ***not found*** during the last check process.

As part of the check process, each routine will first “Inactivate” (set the Active flag to 0) for all objects (of the same object type) belonging to the parent of that object. For example, before enumerating the Logical Volumes on a computer, all existing Logical Volumes, ***for that computer*** are first “Inactivated”. As each volume is then enumerated, the value of the flag is set back to 1.

If a previously discovered object has been removed from a computer (for instance, following the above example, if a Logicial Volume was removed from a computer) then the Active flag (which had been set to 0) will not get set back to 1. Effectively, the application has “discovered” that this object has been delete

In most cases, this process is very fast and the user won’t notice if an object momentarily “goes away”. However, administrators must take care not to schedule reports during mass discovery operations (for example, you should take care not to schedule reports against the database during a period when the disk entities are being updated).

The value of inactivating (versus hard deleting a record) is that we retain the history of the environment from the time that an object was first discovered, even after that object has been retired from the environment. In addition, it can help identify certain Active Directory objects which persist, even though the object has been disabled.

## Object Creation Time Stamp versus dbAddDate

In some cases, entities have their own “Creation time stamp” (such as the Active Directory attribute “whenCreated”). Always keep in mind that the time an object is created does not necessarily correlate with the time that an object is discovered by the application. In cases where these is no creation time stamp attribute, **you must not infer too much information from dbAddDate**. This attribute simply describes when the entity was first added to the database. Only after you have established a set schedule for running the various processes (and have been running it for a while) can you begin to infer things like: “well, the computer wasn’t here last week, and it’s here now: therefore it must have been added in the last week.”

Bear in mind that this process/tool was **never intended for real time monitoring.** In most cases, the major checks were only scheduled to be run daily or at most twice daily.

## Use of Data for Auditing

It’s important to point out another key limitation of this application: **although this tool can be useful for auditing the current state of your environment, it is not a proper auditing solution.** The tool lacks the necessary transactional component(s) -- not to mention the necessary security controls -- to be used in an auditing role. Nevertheless, it can be used to quickly answer questions such as:

1. What groups is this user a member of currently?
2. What groups was this user a member of last week?
3. When was this computer added to the Domain?
4. What groups is this computer currently a member of?