ConfigMgr Startup Script 1.65

What's new in 1.6.5
- Added auto discovery of local built-in admin group name for non-english systems
- Updated client version comparison to account for SP1 CU3 which uses a single zero for the second part of the major version instead of a double zero
- Modified CCMSetupParameter option to account for ccmsetup parameters that do not have values like /NoCRLCheck

What's new in 1.6.1
- Not much ☺ Just a small bug fix to add three undefined constants: MSG_CHECKLOCALADMIN_FINDLOCALGROUP", "MSG_CHECKLOCALADMIN_CHECKACCOUNT", and "MSG_CHECKLOCALADMIN_FINISH"

What's new in 1.6
- Added ability to specify ccmsetup parameters (lie source and BITSPriority) using the CCMSetupParameter XML element
- Updated AutoHotfix feature to automatically detect OS Architecture and then select either the i386 or x64 subfolder of the directory specified. This only works is the AgentVersion major value is 5.

What's new in 1.57
- Added ability to specify multiple accounts to add to the local admins group, comma separated

What's new in 1.56
- Added Extra error checking and logging to the client cache checking functionality

What's new in 1.55
- Added the last error message encountered to the error log file and updated naming format for error log file include the FQDN of the client system instead of just the short name

What's new in 1.54
- Updated WMI namespace to check back to root\cimv2 instead of root\ccm

What's new in 1.53

- Added a check to the beginning of the script so that it does not run if started in WinPE
- Added local admin check to beginning of script – no sense in running the script if the user doesn't have local admin permissions
- Added ability to run an external script to check WMI further/in-depth and possibly fix WMI

What's new in 1.52

- Fixed bug when checking for the ConfigMgr client agent

What's new in 1.51

- Added additional error checking during service checks
- Fixed bug in AutoHotfix Sub
- Added error checking for error file creation and deletion
- Added date/time stamps to beginning and end of execution and elapsed time
- Added creation of the error file for WMI issues
- Added last result tracking to the registry to help maintain state and determine whether we should try to delete the error file

What's new in 1.50

- Added error descriptions to log file in addition to error codes
- Fixed bug where if no admin was specified in the configuration file, the script always created the error file
- Added Check client assignment message to beginning of CheckAssignment Sub
- Added use of multiple PATCH properties
- Added auto-hotfix ability
- Added startup delay
- Added check for existence of ccmsetup before attempting to run it

What is a startup script?

A startup script runs during a system's initial boot up; it is applied to a system using a group policy. Startup scripts run under the context of the local computer's SYSTEM account.

Why use a Startup Script for ConfigMgr?

To check configuration settings and the state of services that the ConfigMgr client agent depends on for successful operation as well as install the client agent if it is not install or functioning properly.

Why use a Startup Script instead one of the built-in methods to install the client agent?

A startup script avoids most common DNS issues, firewall issues, and other connectivity issues that are common when a central system attempts to touch all of the clients. It also adds the health/configuration checks mentioned above to check for (and correct some) dependencies as well as report on systems still having issues.

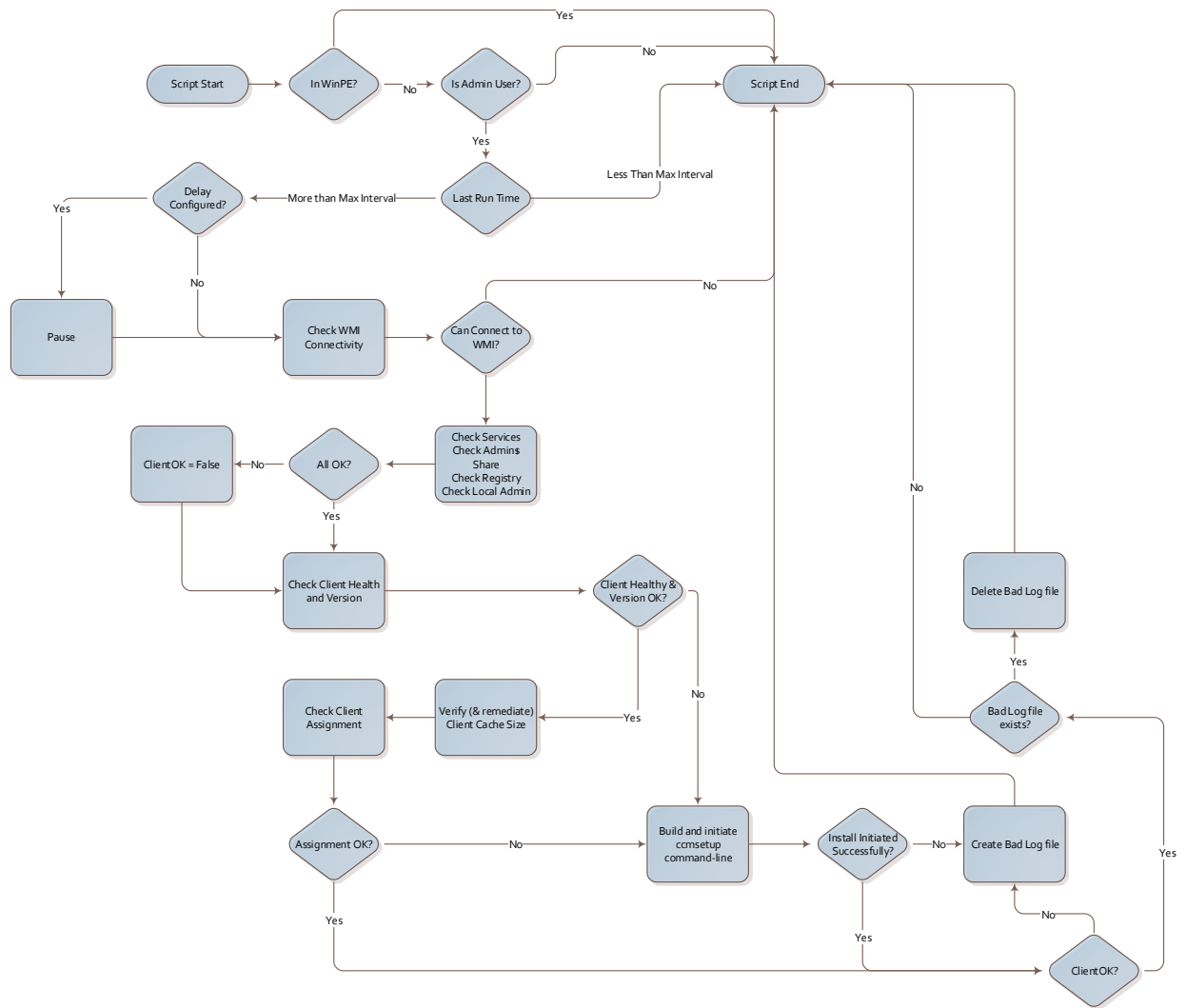# What about the other community scripts available?

This script was inspired by those other scripts, however, in reviewing the other scripts that I had at my disposal, I had a hard time recommending them for use at my customers. This was mainly because I didn't know exactly what the scripts did. A lot of time and effort were put into the other scripts, but they were not straight-forward and tended to resemble spaghetti; they were also somewhat difficult to configure, decipher, and troubleshoot because of the spaghetti code. Thus, this script was written from scratch with simplicity in mind and to be configurable using a separate configuration file. The checks that it performs are also configurable and can therefore be adapted to most (if not all) environments.

What this script does

- Checks to ensure WMI connectivity
- Checks the running and startup state of identified services
- Remediates running and startup state of identified services
- Checks for the Admin$ share and enables admin shares if they are disabled in the registry
- Check for identified registry values
- Sets registry values if they do not exist or do not match
- Adds a specified account to the local administrators group
- Checks for an installation of the client agent
- Checks the version of client agent
- Checks to ensure the ConfigMgr agent is set to Automatic startup and is running
- Installs the client agent if it is not installed or if the version is old
- Checks the client agent's cache size and resets it if needed
- Checks the client agent's site assignment and reinstalls the client agent if not assigned (which implies some type of client corruption)
- Drops a file into a shared folder location if there are any uncorrectable issues found for manual review
- Logs all activity to the client

What this script does not do

Fix WMI. Fixing WMI is a "big" task that I could not hope to ever tackle; however, incorporating a WMI fix script is an easy thing to do using the new WMIScript and WMIScriptOptions options. See the **Using WMIDiag** section below for more details on using this script.

Script Start → In WinPE? — Yes → Script End
In WinPE? — No → Is Admin User?
Is Admin User? — No → Script End
Is Admin User? — Yes → Last Run Time
Last Run Time — Less Than Max Interval → Script End
Last Run Time — More than Max Interval → Delay Configured?
Delay Configured? — Yes → Pause
Delay Configured? — No → Check WMI Connectivity
Pause → Check WMI Connectivity
Check WMI Connectivity → Can Connect to WMI?
Can Connect to WMI? — No → Script End
Can Connect to WMI? → Check Services / Check Admin$ Share / Check Registry / Check Local Admin
Check Services Check Admin$ Share Check Registry Check Local Admin → All OK?
All OK? — No → Client OK = False
All OK? — Yes → Check Client Health and Version
Client OK = False → Check Client Health and Version
Check Client Health and Version → Client Healthy & Version OK?
Client Healthy & Version OK? — Yes → Verify (& remediate) Client Cache Size
Client Healthy & Version OK? — No → Build and initiate ccmsetup command-line
Verify (& remediate) Client Cache Size → Check Client Assignment
Check Client Assignment → Assignment OK?
Assignment OK? — No → Build and initiate ccmsetup command-line
Assignment OK? — Yes → Client OK?
Build and initiate ccmsetup command-line → Install Initiated Successfully?
Install Initiated Successfully? — No → Create Bad Log file
Install Initiated Successfully? — Yes → Client OK?
Create Bad Log file → Bad Log file exists?
Client OK? — No → Create Bad Log file
Client OK? — Yes → Bad Log file exists?
Bad Log file exists? — Yes → Delete Bad Log file
Bad Log file exists? — No → Script End
Delete Bad Log file → Script End

What's Configurable?

The script behavior and options are configurable using an accompanying xml configuration file. There are four different valid elements that can be used in the configuration file.

Option

This element specifies an option used during the script. The value for the given option is specified using the value of the element and the name of the option is specified using the **Name** attribute.

If no value is specified for an option or an option is not specified at all in the configuration file, the option is not set and either a default value is used or the activity associated with that option is not performed.

Attributes

Name – Specifies the name of the option

Valid Options

The following are the available options and their associated functionality:

LocalAdmin

No default value. If not specified, nothing is added to the local administrators group.

Defines the account or group to add to the local administrators group.

To specify multiple accounts or groups, comma-separate them.

Note the use of a forward-slash instead of a back-slash.

Examples:

```
<Option Name="LocalAdmin">domain/paul</Option>

<Option
Name="LocalAdmin">domain/paul,domain/bob,domain/HelpDesk</Option>
```

SiteCode

No default value.

Defines the site code to use during script initiated client agent installation.

Example: `<Option Name="SiteCode">Auto</Option>`

CacheSize

Default value = 5120

Defines the Cache Size to specify during script initiated client agent installation and to set all clients to.

Example: `<Option Name="CacheSize">8120</Option>`

AgentVersion

Default value = 4.00.6487.2000

Defines the minimum client agent version. Systems with version less than this value will have the client agent install triggered.

Example: `<Option Name="AgentVersion">4.00.6487.2000</Option>`

MinimumInterval

Default value = 12

The minimum number of hours between the script running, i.e., if the script starts before this number of hours has passed since the last successful run, the script will simply exit without performing any activity.

Example: `<Option Name="MinimumInterval">24</Option>`

ClientLocation

No default value. If this value does not exist, no script initiated client agent install will be attempted even if called for.

Defines the location on the network to initiate ccmsetup from specified as a UNC. This shared path must be accessible by client computer accounts and should contain at least ccmsetup from the client directory on the site server – if you plan to use the /source ccmsetup switch, this directory should actually contain the entire contents of the client directory. Typically, the best choice for permissions is to add share and NTFS read permissions for the domain based Domain Computers group.

Example: `<Option Name="ClientLocation">\\sccmpri\Client</Option>`

*It is also possible to place ccmsetup.exe in the **netlogon** share on your domain controllers and call it from there:*

`Option Name="ClientLocation">\\domainfqdn\Client</Option>`

or

`Option Name="ClientLocation"> %LOGONSERVER%\Client</Option>`

MaxLogFile

Default value = 2048 (KB)

Defines the maximum size of the script's log file. Once this size is reached, the log file is renamed with a .old extension and a new log file is created.

Example: `<Option Name="MaxLogFile">1024</Option>`

ErrorLocation

No default value. If this value is not specified, a bad file will not be created.

Defines the UNC location to place bad sentinel files.

Example: `<Option Name="ErrorLocation">\\sccmpri\BadLogs</Option>`

AutoHotfix

No default value. Specifies the directory to search for hotfixes to be added to the ccmsetup command-line.

Example: `<Option Name="AutoHotfix">\\cm1\Client\i386\Hotfix</Option>`

Delay

No default value. If specified, pauses the script for the specified number of seconds.

Example: <Option Name="Delay" >5</Option>

WMIScript

No default value. If specified, calls the designated VBScript from the same directory as this script – presumably to check and fix WMI.

Example: `<Option Name="WMIScript" >WMIDiag.vbs</Option>`

WMIScriptAsynch

Default value = 1.

Because scripts like WMIDiag may actually take a while to run, they can be set to execute asynchronously so that the main script will exit and not hold up the login process. If set to 0 (so that the script runs synchronously, then the startup script will recheck WMI connectivity after the called WMI Script exits to see if WMI is functioning again.

Example: `<Option Name=" WMIScriptAsynch" >0</Option>`

WMIScriptOptions

No default value. If specified, pauses the script for the specified number of seconds. A single replaceable parameter exists for this option: %logpath%. This string/parameter, if found, is automatically replaced by the current log file path used by this startup script.

Example: `<Option Name=" WMIScriptOptions" >sms</Option>`

InstallProperty

This element specifies a command-line install property to be used for client agent installation initiated by the script. Only agent installation MSI public properties are valid; i.e., those in all upper case. Examples include SMSMP, SMSSLP, and FSP.

The parameter itself is specified using the **Name** attribute and the value of the parameter is specified using the value of the element.

Valid properties are listed at http://technet.microsoft.com/en-us/library/bb680980.aspx (for 2007) and http://technet.microsoft.com/en-us/library/gg699356.aspx (for 2012) under the *Client.msi Properties* section.

Attributes
Name – Specifies the name of the property

Example
```
<Parameter Name="FSP">fsp.domain.com</Parameter>
```

➕ CCMSetupParameter

This element specifies a command-line install parameter to be used for client agent installation initiated by the script. Only ccmsetup.exe installation parameters are valid; i.e., those prefixed by a forward slash. Examples include source and BITSPriority.

Valid properties are listed at http://technet.microsoft.com/en-us/library/bb680980.aspx (for 2007) and http://technet.microsoft.com/en-us/library/gg699356.aspx (for 2012) under the *CCMSetup.exe Command-Line Properties* section.

Attributes
Name – Specifies the name of the property

Examples
```
<CCMSetupParameter Name="BITSPriority">HIGH</Parameter>

<CCMSetupParameter Name="NoCRLCheck" />
```

ServiceCheck

This element specifies local services on the client to check and remediate if desired.

Attributes
Name – Specifies the name of service. This should be short name of the service and not the display name.

State – The expected state of the service. Valid values are "Running" and "Stopped".

StartMode – The expected start mode of the service. Valid values are "Auto" and "Disabled"

Enforce – Specifies whether to enforce the expected State and StartMode on the service. Valid values are "True" and "False".

Example
```
<ServiceCheck     Name="BITS"     State="Running"     StartMode="Auto"
Enforce="True" />
```

RegistryValueCheck

This element specifies registry values to check and remediate.

Attributes

Key – The registry key where the value to be checked exists. Ensure the key paths are prefixed with HKLM.

Value – The name of the value to check.

Expected – The expected value.

Enforce – Specifies whether to enforce the expected value.

Type – The type of value expected. Valid values are "REG_SZ" and "REG_DWORD".

Hotfixes

Hotfixes can be added the ccmsetup command-line if a client install is called for using the following methods.

PATCH InstallProperty

To manually specify a single or a multiple hotfixes, use the PATCH InstallProperty. Not that patches (in the form of MSPs) specified during installation must be fully qualified. To specify multiple patches, separate them using a semi-colon.

Examples:

```
<InstallProperty Name="PATCH">\\cm1\client\i386\hotfix\05-
KB2276865\sccm2007ac-sp2-kb2276865-x86.msp</InstallProperty>

<InstallProperty Name="PATCH">\\cm1\client\i386\hotfix\05-
KB2276865\sccm2007ac-sp2-kb2276865-x86.msp;
\\cm1\client\i386\hotfix\09-KB2509007\sccm2007ac-sp2-kb2509007-
x86-enu.msp</InstallProperty>
```

PATCH*x* InstallProperty

To manually specify multiple hotfixes, use multiple PATCH InstallProperties each with a unique character appended to PATCH. The major advantage of using this method is readability in the XML file. Also, when a single hotfix is specified, the script checks to make sure the file exists before it is added to the command-line to be run. Each PATCH InstallProperty is added to the command-line in the order it appears in the configuration file.

Examples:

```
<InstallProperty Name="PATCH1">\\cm1\client\i386\hotfix\05-
KB2276865\sccm2007ac-sp2-kb2276865-x86.msp</InstallProperty>
```

```
<InstallProperty Name="PATCH2">\\cm1\client\i386\hotfix\09-
KB2509007\sccm2007ac-sp2-kb2509007-x86-enu.msp</InstallProperty>
```

Note that using either PATCH or PATCH*x* is not recommended for use with ConfigMgr 2012 client installation because the 2012 client has two versions, one for each OS architecture: x64 and x86. The hotfixes for the 2012 client are also OS architecture specific but there is no way to specifically differentiate between the two OS architectures using the PATCH property. Instead, use the Auto Hotfix Discovery for ConfigMgr 2012 feature discussed next.

Auto Hotfix Discovery for ConfigMgr 2007

When this option is specified, the script searches a specified folder and all sub-folders for hotfixes (actually any .msp files) and automatically appends them to the command-line.

Example:

```
<Option Name="AutoHotfix">\\cm1\Hotfix</Option>
```

Do not use the client directory on the site server for this. Instead, copy the hotfixes (including their parent folders) to an alternate location and share out this location ensuring that you grant the appropriate NTFS and share permissions – if you are using this script as a startup script, then you should grant Domain Computers read permissions. The reasons for this recommendation are as follows:

- This directory typically has its NTFS permissions reset during site resets.
- This directory contains all client hotfixes installed on the server; however, you don't always want or need to install them all on clients. Some updates have of course been superseded by others – like 977203 – and others add little to no value – like 978754. Copying (or not copying) the desired updates to this alternate location allows you to easily include (or exclude) updates without changing anything in the original client folder or updating the configuration file.
- Some hotfixes have a certain install order (sad but true: http://blog.configmgrftw.com/?p=163). The script enumerates and adds hotfixes to the command-line in the order that they appear in the folder structure. Thus, by simply renaming the folders directly inside the folder specified for this option, you can easily control the hotfix installation order also without changing anything in the original client folder or updating the configuration file.

    For example, the following is an example folder structure, each directly containing the MSP:

Each folder is prefixed with a two digit number to enforce the sort order. Using a one-digit number may suffice if you do not have more than 9 hotfixes. A prefix is used because without it, the newer KB2xxxxxx folders would sort before the older KB9xxxxx folders and potentially cause issues.

Note that the order depicted here was not tested and is merely shown to present an example for naming the folders.

If desired, the PATCH InstallProperty (or multiple PATCH InstallProperties) can be used in combination with auto hotfix discovery. Hotfixes specified using the PATCH InstallProperty will be added to the command-line first (as above) in the order they appear in the configuration file. If auto discovery discovers a hotfix already added to the command-line (by file name no full path), it will not add the hotfix to the command-line again.

⊕ Auto Hotfix Discovery for ConfigMgr 2012

The script determines if it is deploying a 2012 agent based upon the value of the AgentVersion option specified in the configuration file. If the major version is 5, then it detects the OS architecture.

Based on this detection, the script will automatically append either i386 or x64 as appropriate to the directory specified and then only search that sub-directory (recursively) for hotfixes (actually any .msp files) and automatically appends them to the command-line. Thus, if you specify \\cm1\Hotfix for this option, the script will recursively search either \\cm1\Hotfix\i386 or \\cm1\Hotfix\x64 based upon the OS architecture for hotfixes (in the form of .msps).

Example:

```
<Option Name="AutoHotfix">\\cm1\Hotfix</Option>
```

Do not use the client or hotfix directory on the site server for this. Instead, copy the hotfixes (including their i386 and x64 parent folders) to an alternate location and share out this location ensuring that you

grant the appropriate NTFS and share permissions – if you are using this script as a startup script, then you should grant Domain Computers read permissions.

The Script Log File

The script automatically generates a cumulative log file of all its activity and actions named ConfigMgrStartup.vbs.log. If the system does not have a ConfigMgr client agent installed, this log file is placed in the system temp directory (usually C:\Windows\Temp). If the system does have the client agent installed, the log file will be stored in ConfigMgr client agent log directory. This log file is in a format readable by Trace32.

The Error Log

If the script encounters an error during processing that requires intervention, it will create or append to a file in the configured error log location. This file will be named for the client system and contain the date and time that the script finished running along with the last error message encountered. This share location must be manually created and must have full control NTFS and share permissions for the client computer accounts – this is usually best done using the Domain Computers domain security group.

Reasons for this error file to be created include the following:

- ServiceCheck set to enforce but script received error when attempting to start or stop service or set the service start type
- RegistryValueCheck set to enforce but script received an error when attempting to set value
- Script cannot enable admin shares in the registry
- Script cannot admin the specified account to the local administrators group
- Client agent installation initiation fails

If the script finishes successfully without any of the above conditions, then this file will be deleted from the error log location.

Using WMIDiag

The recommended script for the WMI Script functionality is the WMI Diagnosis Utility (http://technet.microsoft.com/en-us/library/ff404265.aspx) that was recently updated to version 2.1 (http://www.microsoft.com/en-us/download/details.aspx?id=7684). This script doesn't actually fix any WMI issues – as mentioned, that is a "big" task – instead it runs a whole series of tests to try to identify them.

To use call WMI, perform the following:

1. Download WMIDiag 2.1 (linked above) and extract the vbs.
2. Place WMIDiag.vbs in the same directory that you store this startup script in as it will be called from there.

3. Add the following two options to the XML configuration file:

```
<Option Name="WMIScript" >WMIDiag.vbs</Option>

<Option Name=" WMIScriptOptions" >sms BaseNamespace=root\ccm
OldestLogHistory=14 LogWMIState  LogFilePath=%logpath%</Option>

<Option Name="WMIScriptAsynch" >1</Option>
```

There are a handful other options documented in the Word document accompanying the extracted WMIDiag script including the ability to send e-mail. Read through the document for complete details on what the WMIDiag script does and does not do.

Note: WMIDiag is a very complete script and takes a while to run even scoped to check just the root\ccm namespace; running it synchronously is not recommended.

Other Info

To help with troubleshooting, the script adds a handful of event to Windows Application Event log on the client where it is run. These events have WSH listed as their source and provide some simply script flow information.

Implementation

Create a configuration file using the attached sample as a guide and the information above for specific details.

To implement this script in your environment, simply create a GPO and specify this script as a startup script. For the command-line parameter specify /config:*yourconfigfile.xml*. I always store my scripts with the GPO itself.