Automatic Synchronization of 4D Software

By Tim Penner, Technical Services Team Member, 4D Inc.

Technical Note 09-32

Table of Contents

Abstract

It can be time consuming for some developers to update a large number of client software if their database is running interpreted because each client must be manually updated. However, when a new update of 4D v11 SQL becomes available, it is imperative that the developer be able to quickly deploy updates of 4D software.

This Technical Note and the accompanying scripts were developed by 4D, Inc. to help developers facilitate an efficient update of the 4D software (aka 4D in remote mode) on their machines.

Introduction

This Technical Note and the accompanying scripts were developed by 4D, Inc. to help developers facilitate an efficient update of the 4D software (aka 4D Client) on their machines.

Currently it can be time consuming for some developers to update a large number of client software if their database is running interpreted because each client must be updated manually. However, when a new update of 4D v11 SQL becomes available, it is imperative that the developer be able to quickly deploy updates of 4D software.

If your database currently runs interpreted, you will be unable to take advantage of 4D's built-in automatic client update feature. The suggested approach is to move to a compiled database to make full use of the built in automatic update feature of built application.

The solution provided in this Technical Note is being provided in the meantime to ensure that developers spend the minimum amount of time installing client updates.

Much effort has been made to ensure that the included script files are robust and fit the needs of most developers. However it is important to acknowledge that there may be unforeseen issues encountered when testing and deploying this solution. This solution does not rely on any 4D technology and would therefore be possible to implement in any environment. The expectation is that the developer will take ownership of these files and update them as needed. Thorough testing is highly recommended.

Design

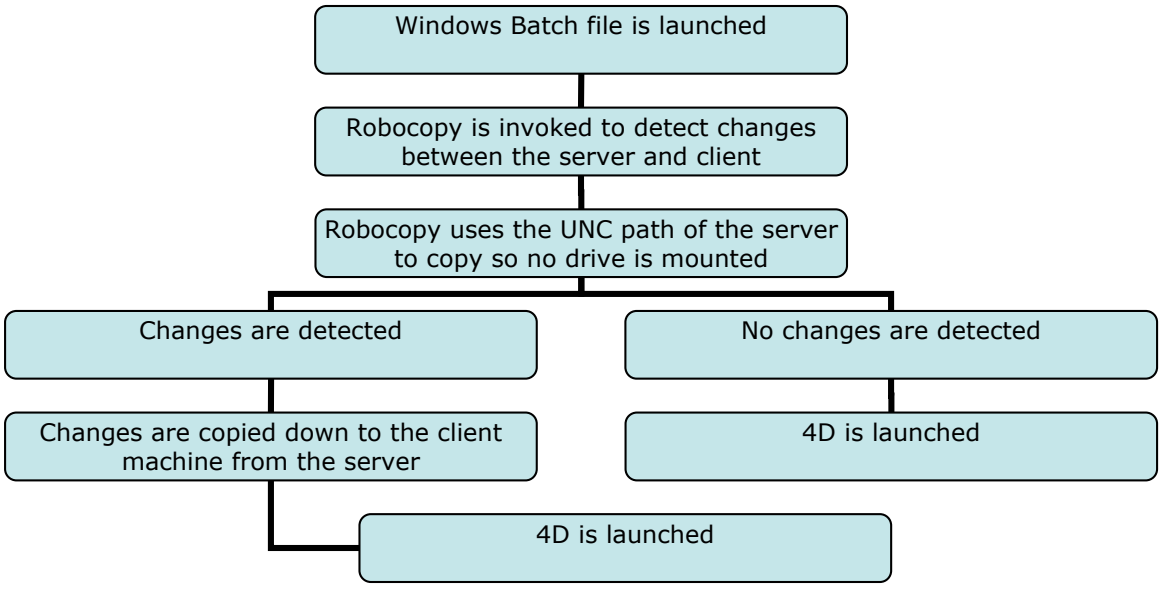
The solution provided involves synchronizing a user's copy of 4D with a remote location every time 4D is launched. 4D is launched via a batch/script file.

This solution is based on the following technologies:

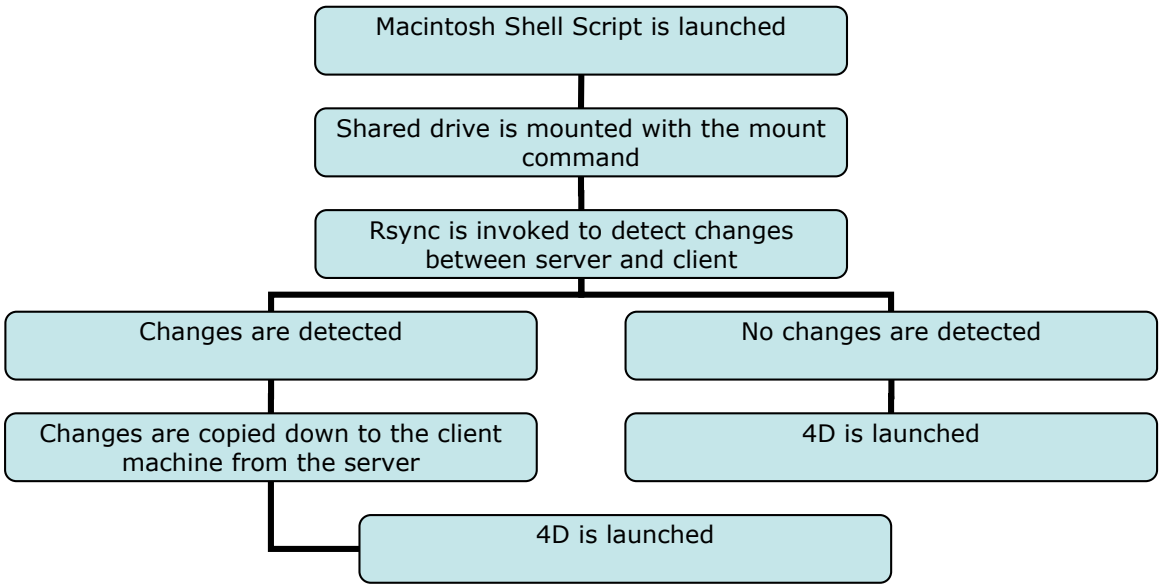
- A network share (Samba or CIFS) must contain a copy of the current 4D software.
- On Windows, users must launch a batch file that synchronizes 4D with the network share.
  - The batch file uses "robocopy" for synchronization, which is provided by Microsoft.
- On Mac OS X, users must launch a script file that synchronizes 4D with the network share.
  - The script file uses "rsync" for synchronization, which is provided by Apple.
- Effort has been made to ensure that the script files are robust and report necessary errors.

When the batch/script file is launched, it checks for any changes to the shared copy of 4D and copies them to the client's machine. To update the remote version of 4D, you need only to update the shared copy. The next time the batch/script file executes, it will synchronize and then launch 4D.

Here is a general overview of how the Windows batch file works:



Here is a general overview of how the Macintosh Shell Script works:



Installation

This section covers the various steps required for installation

Server-side installation

In all cases, a network share must be created to house the current version of 4D. All Windows users must have access to this share. For Mac OS X it is recommended to create a single user name and password combo for all users. This information must be added to the script file, see below.

Note: *The 'mount' command on Mac OS X has limited options and *appears* to be unable to handle user names with spaces, therefore it is recommended to create a user name with no spaces to access the share from Mac.*

Modifying the Script for each platform

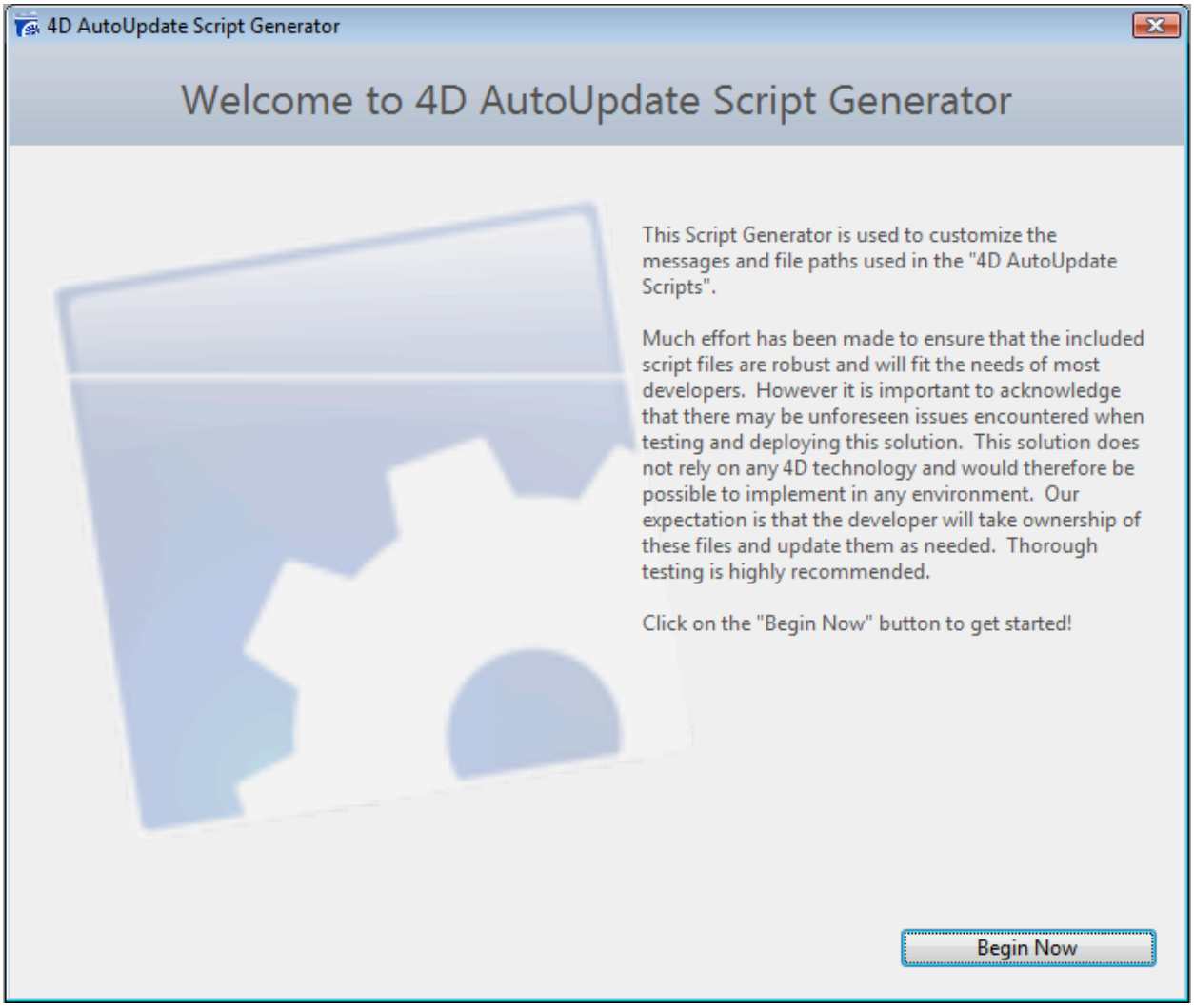
Each script file contains variables for both paths to files as well as error messages. Please feel free to edit these as needed.

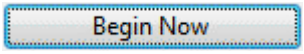
Script Generator

A script generator has been built and accompanies this Technical Note. The script generator can be used to quickly and easily modify the runtime variables and generate the scripts without needing to manually open and edit the scripts themselves. This should prove to be much easier than following

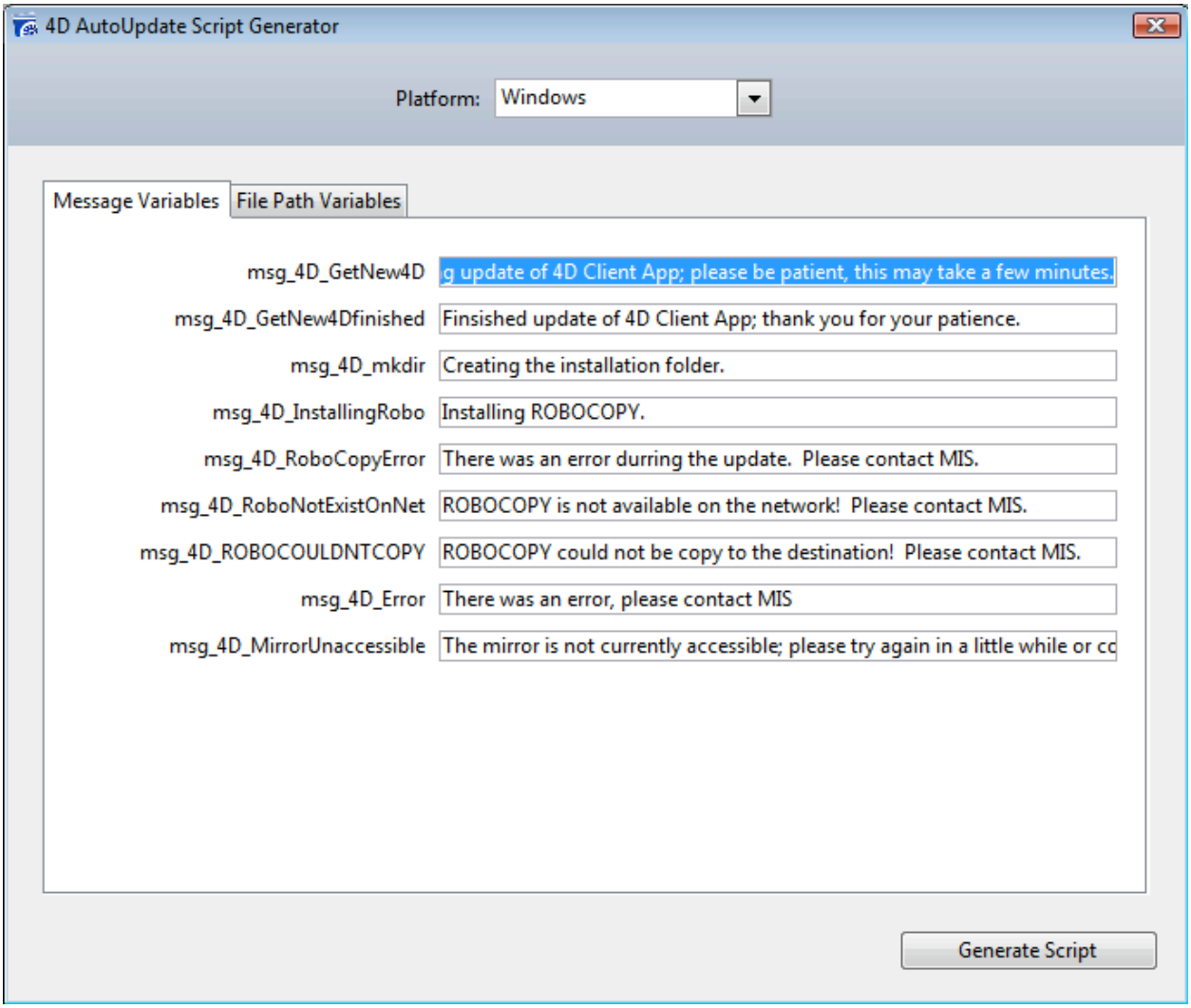
the steps in the Manual Script Modifications section; however both are covered in this document.

To get started with the Script Generator, first launch the Script Generator application. Upon launching the Script Generator application, you are presented with the following Welcome Screen:

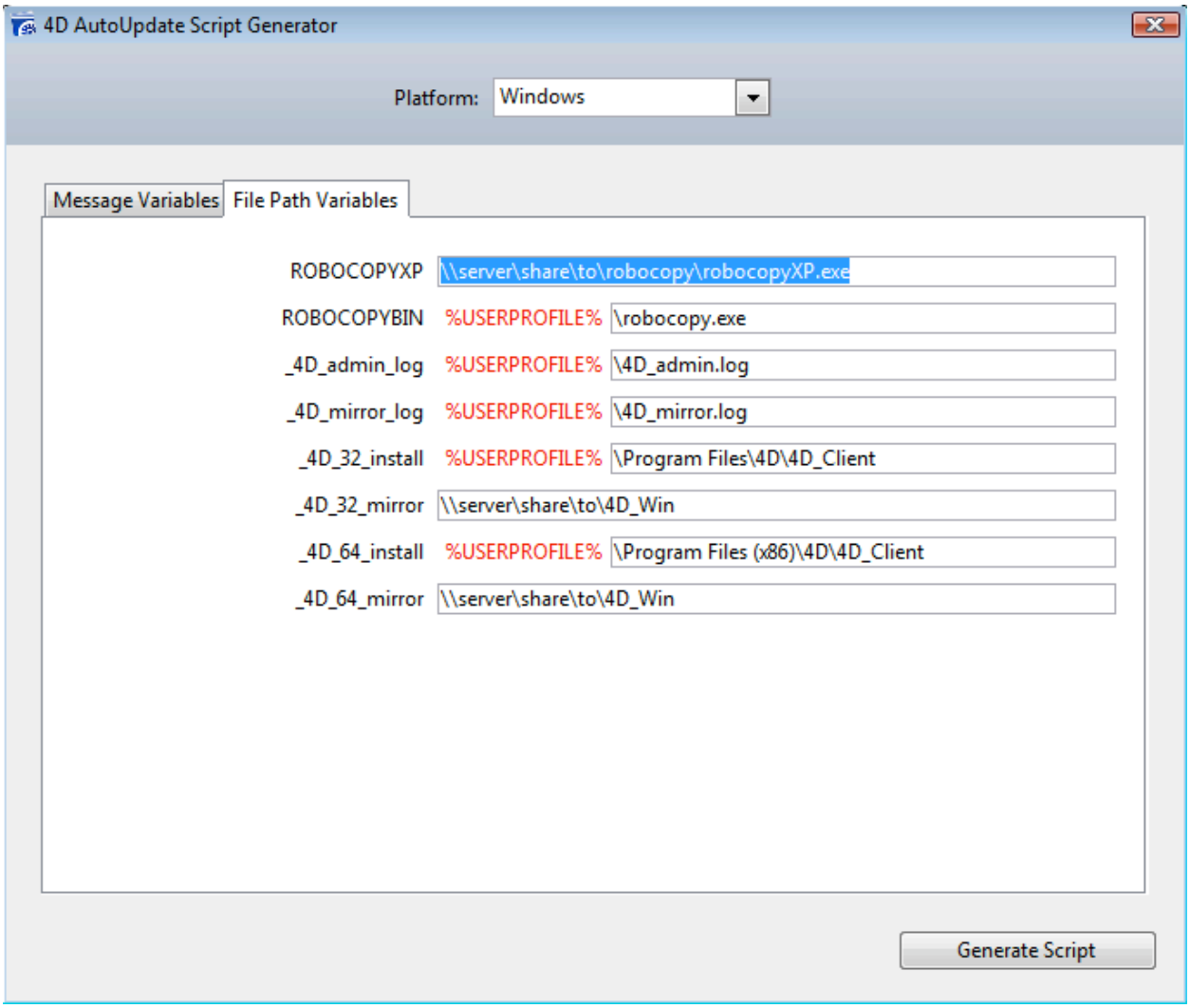


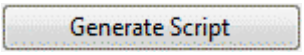
Click on the  button to get started.

Next you are presented with the "***Message Variables***" page for the Windows Script:



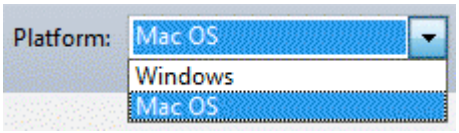
Select the "***File Path Variables***" tab when done with the message declarations. Doing so presents you with the File Paths for the Windows Script:



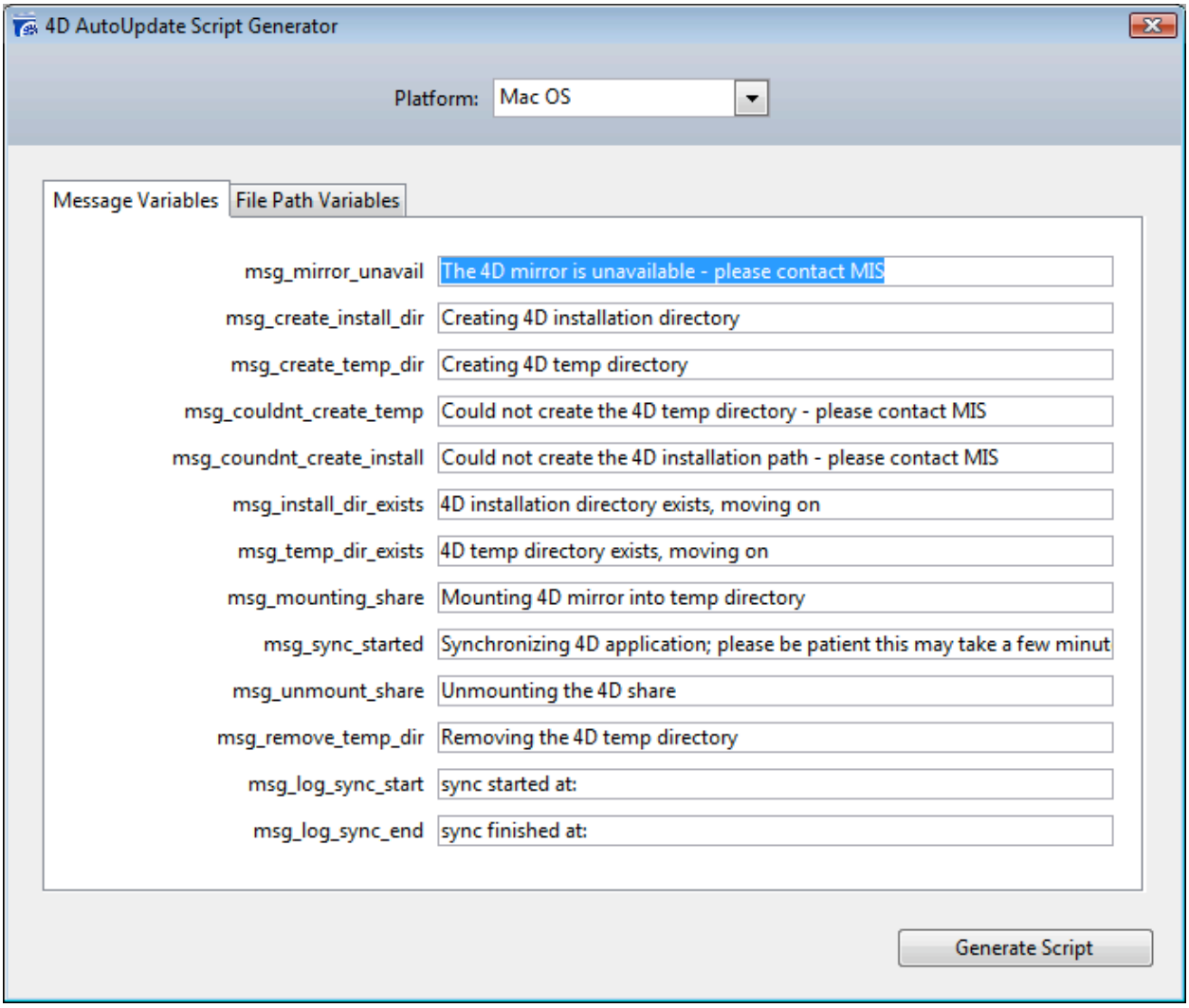
When done filling in the information, clicking on the 

button generates the Windows Script.

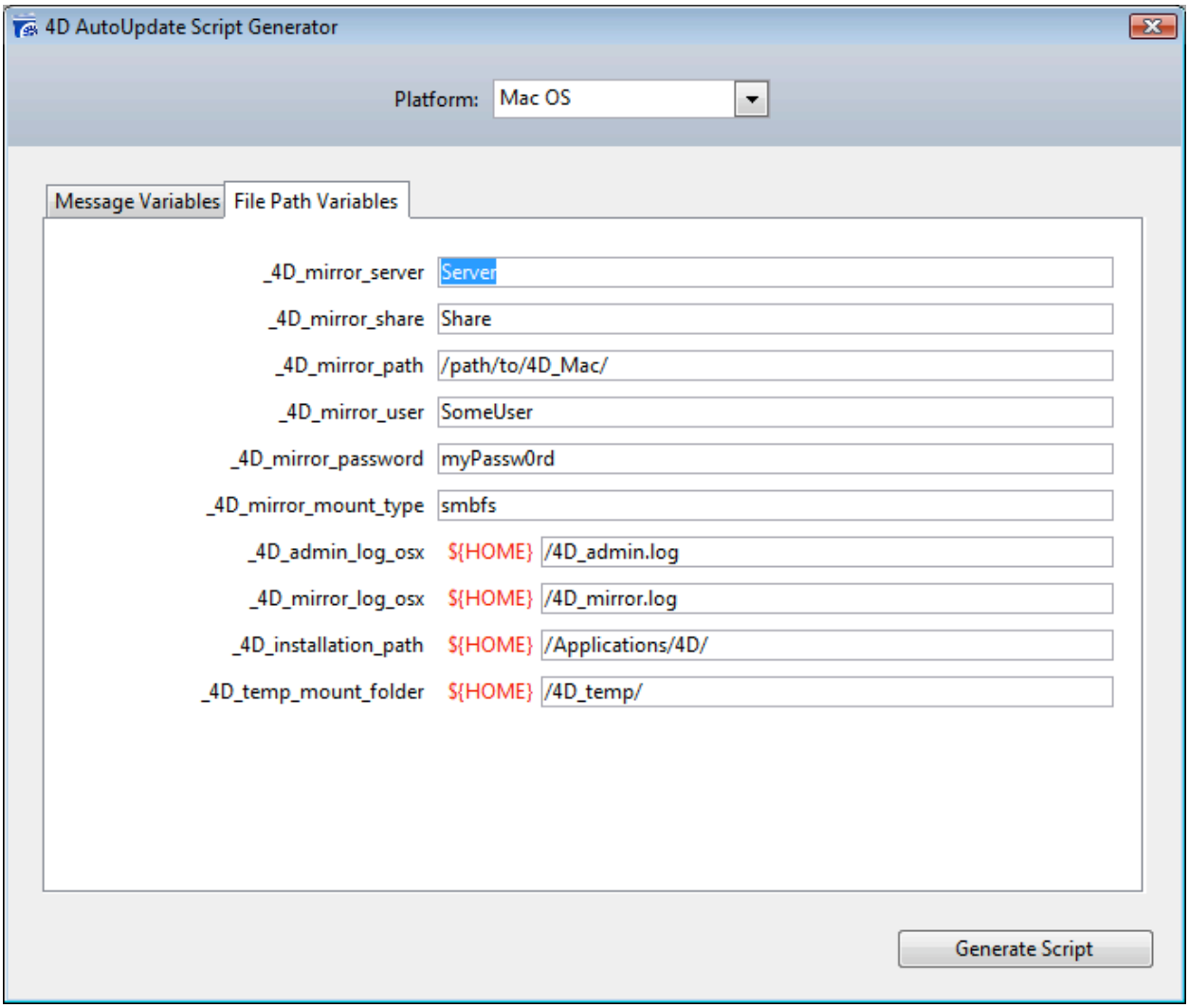
To start modifying a Macintosh script, simply select ***Macintosh*** from the Platform drop down menu at the top:

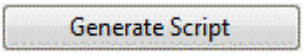


This presents you with the "***Message Variables***" page for the Macintosh Shell Script:



When done with the ***Message Variables*** select the File Paths tab. Doing so presents you with the "***File Path Variables***" page for the Macintosh Shell Script:



When done filling in the information, clicking on the 

button generates the Macintosh Shell Script.

If you have been following along to this point and have already generated the scripts using this Script Generator you can skip over the following section (***Manual Script Modifications***) and go directly to "***Client Side Installation***" section.

Manual Script Modifications

Some developers may find it more powerful to manually edit the scripts as opposed to using the Script Generator. For this reason the following sections covers how to manually modify the scripts.

Windows Script Modifications

The windows script is named Sync_4DClient.bat and is located in the Scripts/Win/ directory of this package.

Because Robocopy is not included with all versions of Windows, it is included with this package and it is recommended that you also place a copy on a network share. In this way the script automatically copies it to the user's machine.

1) Setup a shared location on the network that holds the currently used version of 4D such as a SAMBA / CIFS share (standard windows share). The Windows script expects that the user running the script has access to the shared drive. For example:

```
\\server\share\path\to\4D_Win
```

2) Place the currently deployed version of 4D in remote mode in the share from step 1.

3) Setup a shared location on the network to store the robocopy.exe application. For example:

```
\\server\share\path\to\robocopy
```

4) Place the copy of robocopy.exe in the share from step 3.

5) Modify the ROBOCOPYXP variable on line 19 of the Windows Sync_4DClient.bat script to equal the share from step 3. For example:

```
set "ROBOCOPYXP=\\server\share\path\to\robocopyXP.exe"
```

You may also modify the following variables:

a. ROBOCOPYBIN on line 20 is the location robocopy.exe will be stored if not found; make sure your users can write to this folder. For example:

```
set "ROBOCOPYBIN=%USERPROFILE%\robocopy.exe"
```

b. _4D_admin_log on line 21 is the log kept every time the script is executed; make sure your users can write to this folder. The example uses user profiles so it would work with non-admins:

```
set "_4D_admin_log=%USERPROFILE%\4D_admin.log"
```

c. _4D_mirror_log on line 22 is the log generated from robocopy.exe; make sure your users can write to this folder. The example uses user profiles so it would work with non-admins:

```
set "_4D_mirror_log=%USERPROFILE%\4D_mirror.log"
```

d. _4D_32_install on line 25 is the location on the 32 bit client machine that the 4D application will be installed; make sure your users can write to this folder. The example uses user profiles so it would work with non-admins:

```
set "_4D_32_install=%USERPROFILE%\Program Files\4D\4D_Client"
```

e. _4D_32_mirror on line 26 is the location to find the 4D Client on the network; this should be set to the share from step 1. For example:

```
set "_4D_32_mirror=\\server\share\path\to\4D_Win"
```

f. _4D_64_install on line 29 is the location on the 64 bit client machine that the 4D application will be installed; make sure your users can write to this folder. The example uses user profiles so it would work with non-admins:

```
set "_4D_64_install=%USERPROFILE%\Program Files (x86)\4D\4D_Client"
```

g. _4D_64_mirror on line 30 is the location to find the 4D Client on the network; this should be set to the share from step 1. For example:

```
set "_4D_64_mirror=\\server\share\path\to\4D_Win"
```

6) There are a bunch of error messages defined at the top; feel free to modify the error messages to better suit your environment/needs.

7) The script is currently setup to have robocopy display the file sync progress both on the screen and in the log; this is so that the users can know that the script is working; without showing the progress on the screen the users may think the script is hanging. If you would like to remove the progress display from the screen simply remove the /tee parameter from lines 106 and 107.

Macintosh Script Modifications

The Macintosh shell script is named Sync_4DClient.command and is located in the Scripts/Mac/ directory of this package. The shell script has the username and password hard coded into the script; *please avoid spaces in the username*.

Note: *It has been observed that the mount can fail on Mac OS X 10.5 Leopard if the user is already connected to the same share; this does not seem to be a problem on Mac OS X 10.4 Tiger.*

1) Setup a shared location on the network that holds the currently used version of 4D such as a SAMBA / CIFS share (standard Windows share). For example:

```
\\server\share\path\to\4D_Mac
```

2) Place the currently deployed version of 4D (client) in the share from step 1

3) Modify the following variables in the Sync_4DClient.command script

   a. _4D_mirror_server on line 8 is the name or IP address of the server that is hosting the 4D client share. For example:

   ```
   export _4D_mirror_server="Server"
   ```

   b. _4D_mirror_share on line 9 is the name of the share that is hosting the 4D client share. For example:

   ```
   export _4D_mirror_share="Share"
   ```

   c. _4D_mirror_user on line 10 is the username used for mounting the share (please avoid spaces in the name). For example:

   ```
   export _4D_mirror_user="SomeUser"
   ```

   d. _4D_mirror_password on line 11 is the password used for mounting the share. For example:

   ```
   export _4D_mirror_password="myPassw0rd"
   ```

   e. _4D_mirror_mount_type on line 12 is the mount type used for the share created in step 1. For example:

   ```
   export _4D_mirror_mount_type="smbfs"
   ```

   f. _4D_mirror_path on line 16 is the path to the 4D Remote application (using the share as the root. For example:

   ```
   export _4D_mirror_path="/path/to/4D_Mac"
   ```

   g. _4D_installation_path on line 17 is the location that the 4D client application will be installed to; make sure your users can write to this folder. The example uses user profiles so it would work with non-admins:

   ```
   export _4D_installation_path=${HOME}"/Applications/4D/"
   ```

h. _4D_temp_mount_folder on line 18 is the location that the share is mounted to; make sure your users can write to this folder. The example uses user profiles so it would work with non-admins:

```
export _4D_temp_mount_folder=${HOME}"/4D_temp/"
```

i. _4D_admin_log on line 19 is the location of the admin log in case you should need to check on log from the last sync; make sure your users can write to this folder. The example uses user profiles so it would work with non-admins:

```
export _4D_admin_log=${HOME}"/4D_admin.log"
```

j. _4D_mirror_log on line 20 is the location of the rsync.log file that logs times and dates of all files transferred in case you should need to go back and check on this information; make sure your users can write to this folder. The example uses user profiles so it would work with non-admins:

```
export _4D_mirror_log=${HOME}"/4D_mirror.log"
```

4) There are a bunch of error messages defined at the top; feel free to modify the error messages to better suit your environment/needs.

Client-side installation

After you have setup the shared network location, and modified the scripts as noted above, the next step is to push these scripts out to the client machines. The scripts do not need to be located in any particular place. They can be placed on the desktop of the users, on a network share, even on a USB key. So long as the user running the script has a connection to the server share that hosts the updated 4D software, and the other caveats mentioned are taken care of (no spaces in the user name, mount on Tiger fails if you are already connected to the share), they should work okay.

Usage

It is important to note that your users will no longer launch 4D directly. They should be launching the script or batch file. To decrease confusion, you may want to change the icon of the batch/script file to be something your users recognize. Better yet, create a shortcut/alias to the file and change its name and icon. This way the update is transparent for your users.

If you are already using 4DLINK files to launch 4D, you need to modify the batch files to specify the 4DLINK file information.

You will need to go to each user machine 1 time to install the batch or script file.

If you wish to deploy a new version of 4D, simply copy the new version into the appropriate network share. The next time your users launch the batch/script file, it automatically updates to the new version and of course launches 4D.

Note that the update process generates two log files:

- **4D_admin.log –** This is a log of the actions of the script file.
- **4D_mirror.log –** This is a log file generated by the sync'ing utility on each OS.

The logs (by default) are created in the root of the user's profile directory.

Enhancements

These are just some ideas you may want to implement:

- On Mac OS X it's possible to "wrap" a script file as an application, so you don't see a terminal window when it runs. See Platypus for more information: http://www.sveinbjorn.org/platypus
- You could extend to batch/script files to be self updating. Using Scheduled Tasks (Win) or CRON jobs (Mac) you could have the OS check for updated versions of the script files.

Conclusion

This Technical Note described an approach for updating 4D remote software. A Macintosh shell script and a Windows Batch file were given as examples. This approach should allow the 4D developer to efficiently update the 4D remote software in a timely manner when new updates become available.