



FastTrack
Scripting Host

FastTrack Scripting Host 6

User's Guide

**User's Guide
Version 1.5, August 2010**

Copyright © 2010 FastTrack Software

Table of Contents

About this manual	vi
Description	vi
Audience	vi
How the manual is organized	vi
Manual conventions	vii
Copyright notice.....	viii
Disclaimer	viii
CHAPTER 1 THE PROBLEM WITH SCRIPTING	1
For impatient Users	1
Understanding Scripting	2
What is Scripting?	2
The problem with Scripting	2
About FastTrack Scripting Host	4
What does FastTrack Scripting Host comprise?.....	4
FastTrack Scripting is Different.....	5
A typical logon Script example	5
Using VBScript	5
Using FSH	6
Comparing FSH with VBScript and PowerShell.....	7
Language comparison for Script construction	7
CHAPTER 2 INSTALLING FASTTRACK SCRIPTING HOST	9
Installing FastTrack Scripting Host.....	10
About the FSH download package.....	10
<i>Procedure – Download and Install</i>	<i>10</i>
Starting the FSH Editor for the first time	11
<i>Procedure – Test the Installation.....</i>	<i>11</i>
<i>Procedure – Troubleshoot logon Scripts</i>	<i>13</i>
FSH Licensing	14
Trial Key and License Key	14
How FSH licensing works.....	14
Privacy Statement	15
CHAPTER 3 FASTTRACK SCRIPTING HOST COMPONENTS.....	17
The FSH Engine.....	18
About the engine.....	18
Using the engine to record network logon faults	18
The FSH Editor	19
About the editor	19
Elements of the editor.....	20
Features of the editor	21
Comparing FSH Editor Features.....	22
Language comparison for Script editor features.....	22
FastTrack Logon.....	23
About FastTrack Logon.....	23
How does it work?	23

File and folder structure	23
SmartDock	25
About SmartDock	25
Why is SmartDock useful?	25
Running SmartDock.....	25
An executable program, not a service	25
Sample Scripts.....	26
Included in the download package	26
Demo Script	26
Logon Scripts.....	27
Installation Scripts.....	29
Error-handling Script	29
SmartDock Script	30
CHAPTER 4 USING THE FSH EDITOR.....	31
The FSH Editor Graphical User Interface	32
Layout	32
Debugging Window	32
Summary of elements and features	33
Editor Building Blocks	34
Operations.....	34
Commands	34
Functions	34
Collections.....	35
Conditions	35
Loops.....	36
Additional editor building blocks	36
Executing and Debugging Scripts	39
Testing a Script within the editor	39
Running Scripts outside the editor	39
Running Scripts on other computers	39
Handling Script errors	41
Debugging command line parameters	42
Encrypting Scripts	42
CHAPTER 5 BUILDING SCRIPTS.....	45
Logon Scripts.....	46
Setting up domain logon Scripts.....	46
<i>Procedure – Set up logon Scripts for a domain.....</i>	<i>47</i>
Un-hiding logon Scripts on Windows 7/Vista/Server 2008	47
<i>Procedure – Un-hide logon Scripts on one machine</i>	<i>48</i>
<i>Procedure – Un-hide logon Scripts using Group Policy.....</i>	<i>48</i>
Working with logon Scripts	49
Using Group Policy to run logon Scripts	50
<i>Procedure – Use Group Policy to run logon Scripts.....</i>	<i>50</i>
Using the company logo	50
Using startup, shutdown and logoff Scripts	50
About offline Scripts	51
About terminal services and Citrix	51
Installation Scripts.....	52
Installation leads to complexity	52
Simple install Script.....	52
<i>Procedure – Create an installation Script.....</i>	<i>52</i>
<i>Procedure – Deploy the installation Script</i>	<i>53</i>
<i>Procedure – Apply default User settings.....</i>	<i>54</i>

Common install Scripts 55
Creating an install Script for Adobe Reader 9.0 56
 Procedure – Extract Adobe Reader installation files 56
Installations without a management system 57
Installations with a management system 58
SmartDock Connectivity Tool 60
 Offline Scripts made possible 60
 Setting up SmartDock 60
 Sample Script 60
Where to from here?..... 62
 What can be done with FastTrack Scripting Host? 62
 For up-to-date information 62
INDEX 63

About this manual

Description

This User's Guide is an instruction manual describing the software product known as FastTrack Scripting Host (FSH).

The manual covers all aspects of using the software, including obtaining and installing it, creating and debugging Scripts, using built-in commands and functions and working with special features.

The essential core of the manual is all about how to write and execute Scripts in a Windows networking environment.

Audience

The manual is designed for Computer Network Administrators – people whose job it is to install and manage Microsoft Windows computers that are connected to organization networks.

Other interested parties may include Line or Department Heads, Project Managers, Business Systems Administrators and Business Analysts. Others who might benefit are systems professionals and consultants tasked with the installation and operational management of Windows operating systems within any organizational networking environment.

How the manual is organized

The manual is organized into chapters and each chapter comprises a number of topics. All content for instruction and discussion is contained in chapter topics and each chapter starts with an overview page that includes a topic index for easy navigation.

“How to” material in the manual is organized into *procedures* – things to be done with the software to make it work. Most procedures simply comprise a sequence of steps, but some are further organized into a hierarchy, where each procedure has one or more *tasks*, and each task is broken down further into a number of *steps*.

Where relevant, the procedures within each chapter are further documented in a graphical flow chart, which illustrates at a high level the tasks required and the order in which they need to be done. If there is a choice between tasks, this is also shown in the flow chart.

Continued...

About this manual, Continued

Manual conventions

Convention	Description
Bold	Indicates buttons, data, commands and other entries that must be typed exactly as shown.
<i>Italic</i>	Used for emphasis in narrative text.
Quotation marks " ... "	Indicates multi-word labels or field names in narrative text and procedures.
ALL CAPS	Used in narrative text to indicate particular keys or key combinations on the computer keyboard, e.g. CTRL+ENTER.
> ... >	Indicates a sequence of menu selections, e.g. Insert > Picture > From File....
Lucida Console font	Text that appears in a file, a Script or on-screen and should look (or be entered) exactly as shown.
→	Indicates one line in a file, a Script or on-screen that continues onto the next line in the manual simply for readability.
ShowMessage [Var Name], Message Text	Text that appears inside the FSH Editor, formatted with color and syntax highlighting.

Continued...

About this manual, Continued

Copyright notice

FastTrack Scripting Host 6 User's Guide

Copyright © 2010 FastTrack Software
First printing, April 2010 in the USA.

No part of this documentation may be reproduced in any form or by any means or used to make any derivative work (such as translation, transformation or adaptation) without the express written permission of FastTrack Software ApS.

Microsoft, MS-DOS, Windows, Windows 95, Windows 98, Windows 2000, Windows NT, Windows XP, Windows Explorer, Microsoft Office, Windows Vista, Windows 7, VBScript, PowerShell, Visual Studio and Sysprep are registered trademarks of Microsoft Corporation.

Other brand and product names may be registered trademarks or trademarks of their respective holders.

FastTrack Software ApS
Majvaenget 21
9270 Klarup
Denmark
www.fasttrackscript.com

Disclaimer

No Warranty

This technical document is provided AS IS. FastTrack Software ApS makes no warranty as to its accuracy or use. Any use of the technical documentation is at the risk of the User. FastTrack Software ApS reserves the right to make changes without prior notice.

Chapter 1

The Problem with Scripting

Overview

This document is the User's Guide for FastTrack Scripting Host, also known simply as FSH. The User's Guide provides comprehensive information on installing and using FSH, as well as including sample Scripts that can be used in almost any network.

FastTrack Scripting Host is a software utility for any environment that uses Scripts to control start-up and logon, task automation or indeed anything to make life easier for Users and Network Administrators. The software is a replacement for VBScript/WSH, .bat or .cmd files, PowerShell, KiXtart and all other Windows Scripting tools. Anything that uses Scripts today can be done faster and easier with FastTrack Scripting Host.

Users and/or Network Administrators can create domain logon Scripts, install Scripts, backup Scripts, administrative tasks and anything else Scripts might conceivably be used for, all with a comprehensive end-user graphical user interface (GUI). Writing Scripts is very easy using the GUI – there is no need for programming skills and no need to deploy anything to client computers.

Scripts written in FSH are more powerful, easier to read and are typically less than 10% of the size of the equivalent Script written in VBScript or PowerShell.

Chapter 1 discusses general problems with Scripting and explains in more detail just what FastTrack Scripting Host is all about. The chapter also compares key features of FastTrack Scripting Host with VBScript and PowerShell.

In this chapter

Topic	See Page
Understanding Scripting	2
FastTrack Scripting is Different	5
Comparing FSH with VBScript and PowerShell.....	7

For impatient Users ...

To get straight into using FSH, do this:

1. Do **Procedure – Download and Install** on page **10**.
2. Do **Procedure – Test the Installation** on page **11**.
3. Read about **FastTrack Logon** on page **23**.
4. Read about **SmartDock** on page **25**.
5. Try the **Sample Scripts** on page **26**.

Understanding Scripting

What is Scripting?

A Script is simply a set of instructions, like a program, that is executed on a computer in sequence (i.e. one line after another). Lines can be grouped into repeated sections, or loops, and they can also be grouped into blocks that are executed only if a particular condition is met.

Scripting is the process of writing or building a Script to carry out this set of instructions.

From this definition, it is clear that a Script is very similar to a program – sometimes, in fact, it is hard to tell the difference. Typically, Scripts are not *compiled*, which means they are not turned into executable (.exe) programs. Rather, Scripts are *interpreted* line-by-line in real time.

Another feature of scripting is that Scripts can nearly always trigger operating system commands. Such commands are provided with every operating system and Microsoft Windows is no exception. These commands allow computer Users to quickly carry out tasks such as mapping a network drive, copying a large number of files and discovering the status of a connection.

When combined with the capability to loop and branch (based on decisions), the execution of many operating system commands one after another creates a powerful and flexible tool for Network Administrators as well as Users.

The problem with Scripting

The increasing functionality, power and flexibility of operating systems has led to a corresponding increase in the capability of operating system commands and the scripting languages that use them, such as VBScript and PowerShell.

Scripting languages require a significant amount of time to create and test Scripts. The complexity alone required by many Scripts prohibits IT professionals from doing the things they would like to do with scripting.

Most scripting languages are now very close to programming languages and many of them are, in fact, the counterparts of real programming languages, like VBScript. This means that a Script Writer needs real programming skills to successfully build and test Scripts. The complexity involved once Scripts move beyond the simplest requirements means that large sections are copied from previous work or from other peoples' Scripts.

This complexity can make it difficult even for a programmer to create Scripts for complex tasks (and most IT Administration departments do not employ programmers). Frequently, time constraints or unforeseen problems require a Script professional to be employed to solve the immediate problem, which greatly increases the cost of implementing scripting solutions.

Continued...

Understanding Scripting, Continued

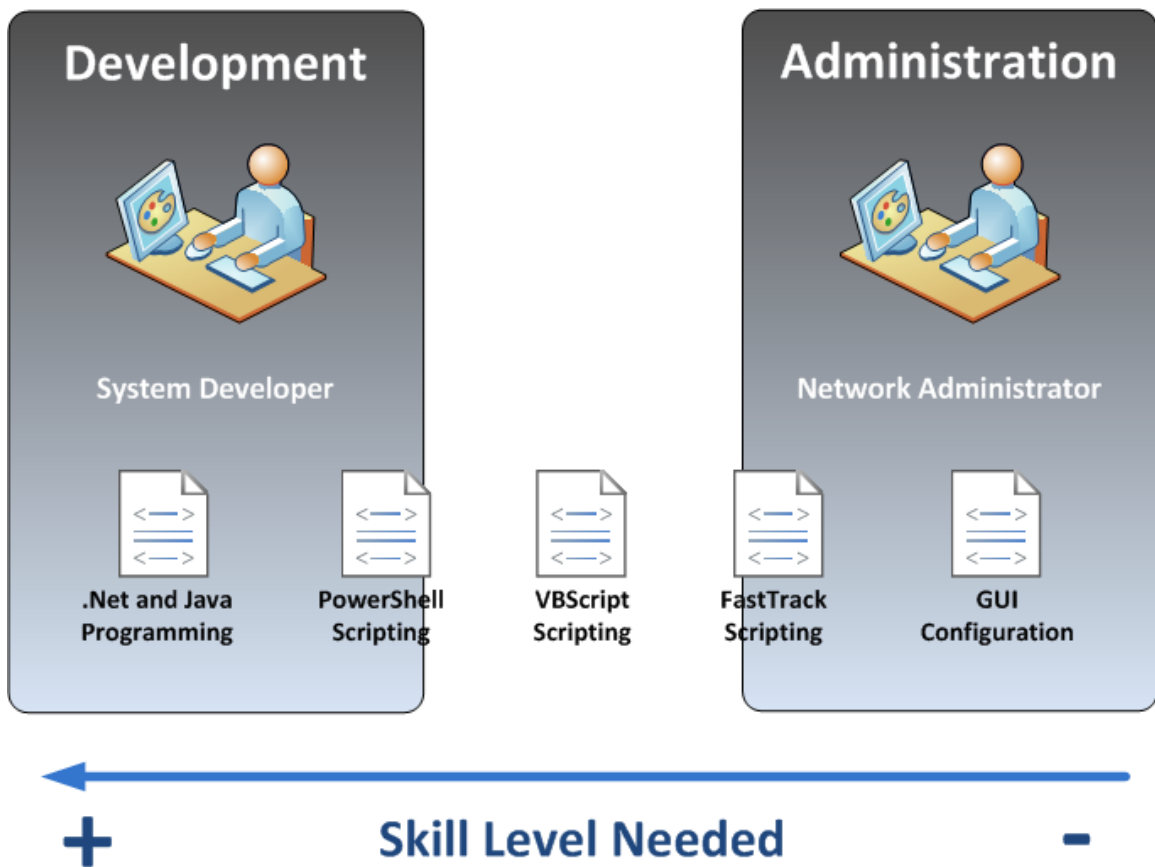
The problem with Scripting, Continued

A skills issue

FastTrack Scripting Host tries to solve one of the major problems in the world of a Network or System Administrator – scripting has always been a hybrid of *development* and *administration*.

While Microsoft moves scripting more towards actual programming, FastTrack Scripting Host goes in the opposite direction. A good Administrator is typically one who knows about infrastructure and not programming and, while FastTrack Scripting Host *is* a scripting language, it is so high-level and easy to use that it resembles configuration more than actual programming. This is shown graphically in the following diagram.

Figure 1.1 Skills needed from Administration to Development



Continued...

Understanding Scripting, Continued

About FastTrack Scripting Host

FastTrack Scripting Host (FSH) is a high-level scripting language and is currently the only real high-level scripting language available. It is targeted, but not limited to, any kind of scripting that might be needed to service clients connected to a Microsoft Windows network, such as domain logon Scripts and automatic software installations.

FastTrack Software takes a different approach with FSH. The paradigm for FSH is:

One operation → One Script line

For example, if the requirement is to synchronize files from one directory to another at logon time, the Script Writer writes *one* line in a Script using the FSH Editor.

Such a line might be as simple as this:

```
SyncDir <Source Folder>, <Destination Folder>
```

This one line, of course, triggers hundreds of lines of code behind the scenes (in the FSH Engine), but Script Writers are not concerned with any of that – they simply specify what they want to accomplish, letting the FSH Engine take care of the details.

Script Writers moving to FSH are consistently amazed at how few lines of FSH code are required to achieve some very complicated tasks. When compared with VBScript or PowerShell Scripts, the number of lines of FSH code required can be many times less (and sometimes many orders of magnitude less).

In addition, FSH Scripts are far more readable than VBScript or PowerShell Scripts and Script Writers no longer need to rely on other people's code snippets. Furthermore, experienced Script consultants are needed less frequently, or not at all. Anyone with basic IT Administration skills and knowledge of the networking environment can write a FastTrack Script to accomplish complex tasks.

What does FastTrack Scripting Host comprise?

There are five components to FastTrack Scripting Host. These are listed below by way of introduction and discussed fully in Chapter 3:

- **The FSH Engine**
The executable interpreter for FSH Scripts.
- **The FSH Editor**
An integrated development environment for creating and testing Scripts.
- **FastTrack Logon**
The access and distribution mechanism for client computers at logon time.
- **SmartDock**
A smart utility for recognizing changes to client IP address settings.
- **Sample Scripts**
Pre-written Script templates that work and can be quickly customized.

FastTrack Scripting is Different

A typical logon Script example

To illustrate the difference between FSH and other languages, this example implements the same functionality using first VBScript and then FSH.

Logon Scripts usually comprise either operating system commands entered into a simple batch file, or VBScript code. Since VBScript is more powerful than batch file commands, this example compares a typical use of VBScript and FSH to implement a logon Script that runs every time a User logs on to the network.

Use Case

Check to see if the User logging on is a member of a certain Active Directory group *SalesStaff*. If so, setup drive *I:* for the User and connect that drive to a share called *Sales* on a server called *AcmeServer*.

Assumptions

In the VBScript version, the LDAP provider is used to query the Active Directory, as this requires the least number of VBScript lines.

Using VBScript

Figure 1.2 VBScript version

```
If IsMember("SalesStaff") Then MapNetworkDrive →
    "I:", "\\AcmeServer\Sales"

Function IsMember(groupName)
    Const ADS_NAME_INITTYPE_GC = 3
    Const ADS_NAME_TYPE_NT4 = 3
    Const ADS_NAME_TYPE_1779 = 1

    '==== CONVERT GROUP NAME TO FULLY QUALIFIED NAME ====
    Set WSHNetwork = WScript.CreateObject("WScript.Network")
    Set objTrans = CreateObject("NameTranslate")
    objTrans.Init ADS_NAME_INITTYPE_GC, ""
    objTrans.Set ADS_NAME_TYPE_NT4, WSHNetwork.UserDomain & "\" & →
        groupName
    strGroupDN = objTrans.Get(ADS_NAME_TYPE_1779)
    strGroupDN = Replace(strGroupDN, "/", "\\")

    '==== GET GROUP AND USER ====
    Set objGroup = GetObject("LDAP://" & strGroupDN)
    Set objADSysInfo = CreateObject("ADSystemInfo")
    strUserDN = objADSysInfo.UserName

    '==== DETERMINE MEMBERSHIP ====
    If objGroup.IsMember("LDAP://" & strUserDN) Then
        IsMember=True
    Else
        IsMember=False
    End If
End function
```

Continued...

FastTrack Scripting is Different, Continued

Using VBScript, Continued

Figure 1.2 VBScript version, Continued

```
Function MapNetworkDrive(letter, uncPath)
    '==== DISCONNECT EXISTING DRIVE IF CONNECTED ====
    Set objNetwork = CreateObject("WScript.Network")
    Set objDrives = objNetwork.EnumNetworkDrives()
    For objDrive = 0 To objDrives.Count - 1 Step 2
        If objDrives.Item(objDrive) = letter Then
            objNetwork.RemoveNetworkDrive letter, true
        End If
    Next

    '==== MAP SHARE ====
    objNetwork.MapNetworkDrive letter, uncPath, false
End function
```

In a production environment, the VBScript version would be longer, as there is no error handling in this example, nor is there any checking that the group actually exists, to ensure that the Script can continue in case of execution error.

Using FSH

Using a FastTrack Script, the same operation requires just one line.

In the case of an error, FSH includes a general error-handler Script; errorhandler.fsh, which is explained later in this manual.

Figure 1.3 FSH version

```
If UserIsMemberOf salesStaff Then ConnectShare I:,\AcmeServer\Sales
```

Both Scripts produce the same result, which is network drive I: mapped to server and share name [\\AcmeServer\Sales](#).

However, the VBScript example executes slightly slower and stresses Active Directory, if there are many computers in the network, as it queries Active Directory over the network every time a User logs on.

The FSH example queries only against locally-cached information, which the WinLogon process has already retrieved from the domain controller.

It is assumed in the VBScript version that users are a members of SalesStaff directly and not through nesting, which would make the VBScript even more complex.

For example, if a user is a member of the group SalesEurope which is itself a member of SalesStaff, FSH resolves the group nesting automatically, as well as any other group nestings that might exist, whereas the VBScript version would be considerably longer and much more complex, if it had to recursively loop through all group nestings.

Comparing FSH with VBScript and PowerShell

Language comparison for Script construction

The example in the previous topic showed a typical VBScript and FSH construct. The comparison chart included in Figure 1.4 shows how much scripting (i.e. development work to construct a Script) is required to perform typical operations, implemented in FastTrack Scripting Host, VBScript and PowerShell.

Creating rich graphical user interfaces is not possible with VBScript or PowerShell. It is possible to create primitive forms with PowerShell, but not graphical user interfaces.

Note:































This comparison chart assumes that the Script Writer has a high level of experience using VBScript and PowerShell. If the Script Writer has less experience with these products, the comparison becomes even more favorable to FastTrack Scripting Host.

Figure 1.4 Building Scripts using FSH, VBScript and PowerShell

Script Construction - Common Task Examples

-  Requires one script line
-  Requires highly complex scripting
-  Requires multiple script lines
-  Not possible natively



































































CRUD = Create, Read, Update and Delete

	FastTrack Scripting Host	VBScript	PowerShell
CRUD registry keys and values			
CRUD 32-bit registry keys and values on 64-bit machines			
CRUD Active Directory users, computers, groups and OUs			
Query nested memberships of Active Directory groups			
Move Active Directory users, computers, groups and OUs			
CRUD local users and groups			
Query local group memberships including nested Active Directory groups			
Visual synchronization of files and folders			
Graphical menus			
Splash screens			

Continued...

Comparing FSH with VBScript and PowerShell, Continued

Language comparison for Script construction, Continued

	FastTrack Scripting Host	VBScript	PowerShell
Graphical query for user input			
Embed internet and intranet pages			
CRUD file, directory and registry permissions			
CRUD shares and share permissions			
CRUD XML files			
CRUD files and directories			
Connect shares and printers			
Query system information			
Query and control services			
Query and control running processes			
Set user settings based on installed programs			
Query or set proxy settings real-time			
Interval based execution			
Associate and unassociate file extensions			
Log events			
Send mail natively			
Zip and Unzip natively			
Perform http requests			
CRUD shortcuts			
Conditional execution on terminal servers or citrix			
Conditional execution based on operating system			
Conditional execution based on computer type			

Chapter 2

Installing FastTrack Scripting Host

Overview

Chapter 2 covers the installation and licensing of FSH. The installation process comprises the following two procedures:

- Download and Install
- Test the Installation

Each procedure is described via a series of steps, documenting exactly how to carry out the procedure. "Download and Install" includes the steps needed to obtain and apply a *key file*, either a trial key file or a license key file, which is required for correct operation of FSH.

The licensing topic explains how FSH Licensing works and the differences between a trial key file and a license key file.

In this chapter

Topic	See Page
Installing FastTrack Scripting Host	10
FSH Licensing	14

Installing FastTrack Scripting Host

About the FSH download package

The FSH download package is a compressed zip file that can be downloaded from the FastTrack website and is the primary way of obtaining FastTrack Scripting Host (see step 2 below for details).

The package is called *Fsh.zip* and it contains the setup programs (.exe and .msi) needed to install the software on an Administration computer. At the time of writing, the official name and current version of the package is **FastTrack Scripting Host 6**.

The FSH download package includes all components of the software (described in Chapter 3), but it does not include a license, which must be applied for separately. See the next topic, FSH Licensing, for more information on how licensing works.

Procedure – Download and Install

Step
1. Identify a suitable Administration computer for installation. This is the computer that will be used to run the FSH Editor to build Scripts. A workstation is fine – it does not need to be a server.
2. If a license is available (i.e. has already been purchased), log into “My Account” on www.fasttrackscript.com to download the installation package. If a purchased license is not available, then a trial key is required; use the “Download Free Trial” menu to download the installation package.
3. Extract all files from the compressed zip file. Two setup files are extracted: <ul style="list-style-type: none">- Setup.exe- Setup.msi
4. Run the setup program (setup.exe), accepting all defaults.
5. Once installation is complete, check that the following files have been installed into folder C:\Program Files\FastTrack Software\FastTrack Scripting Host\ : <ul style="list-style-type: none">- DemoScript.fsh- Editor.exe- FSH User’s Guide.pdf- FSH.exe- FTLogon.zip
6. From the desktop, start the editor by double-clicking the FSH Script Editor icon.
7. A key file installation screen is shown (as in Figure 3.1). Click the top button to install the license or trial key file.

For more information on using the editor once it is installed, see Chapter 4 Using the FSH Editor.

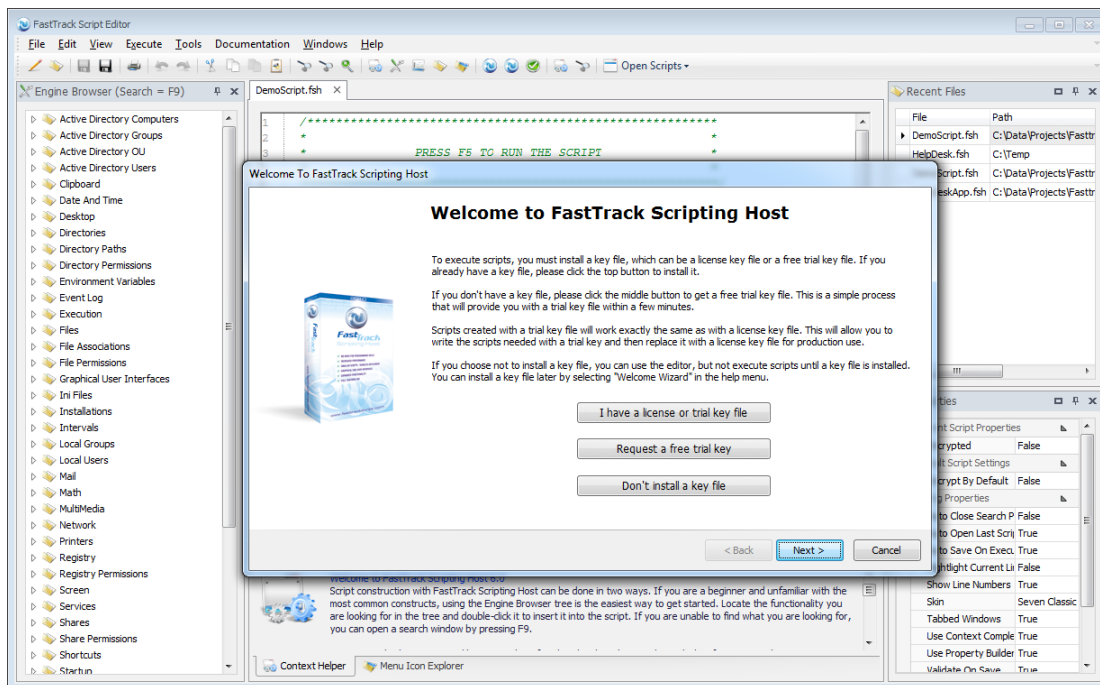
Continued...

Installing FastTrack Scripting Host, Continued

Starting the FSH Editor for the first time

Step 6 in the procedure above describes starting the FSH Editor. Figure 3.1 shows the screen displayed when the editor is started for the first time.

Figure 3.1 Starting the FSH Editor



The button clicked depends whether or not a trial or license key file is available. If so, click the top button, as described in Step 7. If not, click the middle button to obtain a 45-day trial key (no purchase needed).

Procedure – Test the Installation

This test procedure describes two tests to make sure the installation was successful and FSH Scripts will run on both the Administration computer and any other computer attached to the network.

Note:

Test 2 below describes files and folders that have not yet been explained. Refer to FastTrack Logon on page 23 for more information on FTLogon.zip.

Step

Test 1 Run a Script on the Administration computer from within the FSH Editor

1. Execute the Demo Script provided (and currently displayed in the editor) by pressing the **F5** key. The Demo Script is called **DemoScript.fsh** and it opens automatically the first time the FSH Editor is started.

Continued...

Installing FastTrack Scripting Host, Continued

Procedure – Test the Installation, Continued

Step

2. Follow the instructions given by the Demo Script. For dialog boxes that include both **OK** and **Cancel** buttons, it does not matter which one is clicked, but a selection must be made from the items listed if choosing **OK**.

The Demo Script will continue to run until all selections and dialog boxes have been displayed. There are approximately ten dialog boxes and the final one advises how much space is available on the local hard drive.

Test 2 Setup a Test User to automatically run a Logon Script

1. Identify a suitable location for storing the FastTrack Logon files to be extracted from the **FTLogon.zip** package. FTLogon.zip is one of the files installed into folder "FastTrack Software\FastTrack Scripting Host" under the program files directory during the **Download and Install** procedure.

Note that the recommended location is the domain NetLogon folder. By default, the path to any Windows domain NetLogon folder is **\\DomainServer\NetLogon**, where DomainServer is the name of the Active Directory Domain Controller.

2. Map a drive (or otherwise obtain access to) this location.
 3. Extract all files from the compressed zip file **FTLogon.zip** into the **NetLogon** location. As well as the files, a folder is extracted called **Fshbin** – make sure no additional folder is added by the extraction (or unzip) program.
 4. Copy the license file **Fsh.lic** into the **Fshbin** folder.
 5. Check the files extracted – the list of files and folders in the NetLogon folder should be:
 - **Fshbin** (folder)
 - **ErrorHandler.fsh**
 - **Fsh.exe**
 - **Fsh.lic**
 - **PostInstall.fsh**
 - **PostLogon.fsh**
 - **PostUninstall.fsh**
 - **PreLogon.fsh**
 - **SmartDock.exe**
 - **SmartDock.fsh**
 - **FTLogon.exe**
 - **ftlogon.ini**
 6. Choose a test User in the domain and modify that User's logon profile so that **FTLogon.exe** is specified in the **Logon Script** field.
 7. Logon (from a different computer) as the test User and observe the commands executed by the FSH Logon Scripts **PreLogon.fsh** and **PostLogon.fsh**.

PreLogon.fsh is executed *before* the Windows Explorer process (explorer.exe) starts on a workstation, while PostLogon.fsh is executed *after* this process starts.
 8. Use the **FSH Editor** on the Administrator computer to examine and change the FSH Logon Scripts.
 9. If the Logon Scripts do not run, carry out the troubleshooting procedure below.
-

Installing FastTrack Scripting Host, Continued

Procedure – Troubleshoot logon Scripts

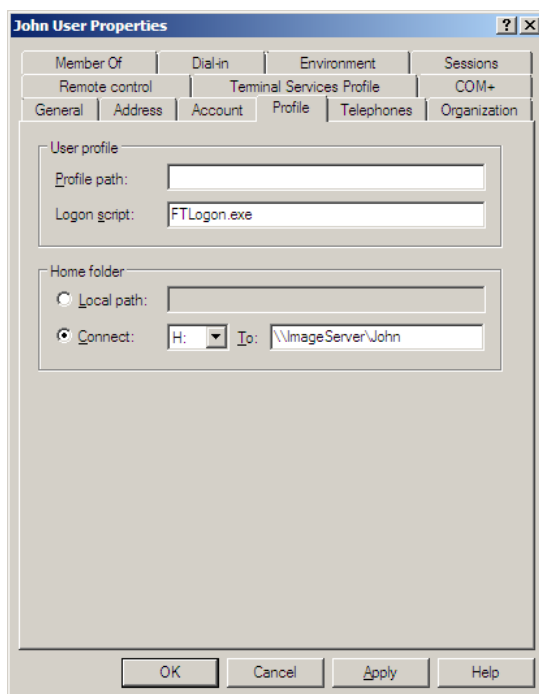
This troubleshooting procedure outlines some basic checks to carry out if logon Scripts do not execute.

Note:

FastTrack Logon is no different than other logon Scripts when it comes to file locations, how logon Scripts are specified and how they are executed. Refer to Microsoft's documentation on technet.microsoft.com for general logon Script troubleshooting.

Step

1. From the client computer, open the \\DomainName\Netlogon folder. Check that FTLogon.exe is located in the root of this folder.
2. Check that **FTLogon.exe** is spelled correctly in each User's profile and/or in Group Policy User Configuration. Also check that it is entered into the right field: the **Logon script** field on the **Profile** tab for a User in Active Directory, as in the screen shot:



3. If additional Scripts have been created to be called by logon Scripts, check their spelling (including paths if they do not reside in the default location).
 4. On Windows Vista and Windows 7 computers, make sure logon Scripts are visible when they execute. See Un-hiding logon Scripts on page 47 for more information about this.
 5. If all else fails, contact FastTrack Software by using the contact form on www.fasttrackscript.com.
-

For more information on creating, modifying and running Logon Scripts, see Chapter 4 Using the FSH Editor.

FSH Licensing

Trial Key and License Key

The FSH executable files (*fsh.exe* and *editor.exe*) cannot execute unless there is a valid license file (.lic) in the same folder as the executables. The license file can be either a *trial key* file or a *license key* file.

Trial Key message window

On a non-Administration computer (i.e. without the FSH Editor installed, but capable of running Scripts), if the engine starts and finds no .lic key file (neither trial key nor license key), it displays a window that says "Missing key file".

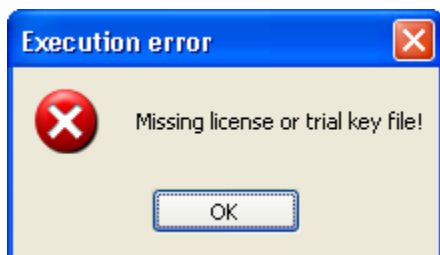
On the Administration computer (i.e. with the FSH Editor installed), the editor will not start if a key file is not found. Instead, it will show a welcome message asking for a valid key file (either a valid trial key or a valid license key file). Included in the message is an explanation of how to get a trial key file or install a license key file.

The trial message will still be there after installing a trial key, but now it will also show the company name in that window, indicating that this trial key file is specific to this organization. The days until expiration will then also be the same on all computers, as the trial key dictates the expiration.

Missing key file

If no key file can be found (neither trial key nor license key), the following message pops up on the User's screen:

Figure 3.2 Message displayed if key file Fsh.lic cannot be found



How FSH licensing works

A Network Administrator installs the setup package (editor and engine) on the Administration computer. Then *fsh.exe* and a license file are put on a domain controller and it gets replicated to all the computers on the network.

Nothing else is needed and there is no additional process on a client computer. There is also no need for a licensing server or any additional management components.

Continued...

FSH Licensing, Continued

How FSH licensing works, Continued

Call home

The network does need Internet access, however. Every time Fsh.exe is launched in licensed mode on any computer, it will check if it has already reported itself once (to prevent massive traffic) via an http request to a web page on fasttrackscript.com. This is known as "calling home". The call home procedure will not in any way effect the performance or stability of Script execution.

The call home is done by default once every two weeks. The time frame of two weeks can be changed, but will not be less than 14 days.

If it has not yet reported in, Fsh.exe makes a transparent background "report" every time a new version is run on each computer. No admin privileges are needed and nor will the Windows Vista/7 UAC warning pop-up. If the computer does not have internet access or it is not possible to reach the server, it will execute and do nothing, and then try again until it is possible.

This is included in the Terms and Conditions in the EULA, which must be accepted by all Customers prior to installation of the software.

Privacy Statement

Both FastTrack Software and Binary Research International treat the privacy of Customer information very seriously. No Customer information will be divulged to any outside party other than authorized Resellers of FSH and then solely for the purpose of follow-up with Customers on their experience with FSH.

Please refer also to the following Privacy Statements:

- FastTrack Software <http://www.fasttrackscript.com/Docs/About.aspx>
- Binary Research International http://www.binaryresearch.net/about_bri/privacy

Chapter 3

FastTrack Scripting Host Components

Overview

Chapter 3 provides an overview of the components of the FastTrack Scripting Host software. Each component is introduced, along with any sub-components or elements, and a brief description is given of the component's purpose or role.

FastTrack Scripting Host has five components:

- The FSH Engine
- The FSH Editor
- FastTrack Logon
- SmartDock
- Sample Scripts

One of the most common uses of Scripting is to write logon Scripts for Users logging on to an organization's domain. The first three components of FSH listed above make this task quick and easy.

The *FSH Editor* enables Script Writers to create and test Scripts, *FastTrack Logon* configures incoming logons for correct Script execution and the *FSH Engine* interprets and carries out the logon Script commands. Although the engine and the editor can be used anytime a Script is needed, FastTrack Logon is geared specifically towards logon Scripts.

This chapter introduces each component in its own topic. An additional topic is presented that shows a comparison of the editor features for FSH, VBScript and PowerShell.

In this chapter

Topic	See Page
The FSH Engine	18
The FSH Editor.....	19
Comparing FSH Editor Features.....	22
FastTrack Logon.....	23
SmartDock	25
Sample Scripts.....	26

The FSH Engine

About the engine

The FSH Engine is the core of FastTrack Scripting Host. It is the FSH Engine that processes and interprets Scripts, translating commands and functions into the many hundreds of lines of code needed to actually perform the tasks specified.

The engine is installed when FastTrack Scripting Host is installed. The FSH Engine is an executable file called *fsh.exe* and it runs just like any other Windows program.

A Script file can be run on a computer simply by specifying it on the command line with the engine. For example:

```
fsh.exe InstallProgram.fsh
```

The engine is *fsh.exe* and the Script file (in this example) is *InstallProgram.fsh*, which is a parameter passed to the engine.

The FSH Engine must be available to any computer that wants to process FSH Scripts. It can be easily and seamlessly deployed to all clients simply by using FastTrack Logon for logon Scripts.

Using the engine to record network logon faults

The engine has two parameters related to the way faults are handled. In general there is a distinction between Script syntax errors and runtime errors:

- Syntax errors are picked up before a Script is saved and executed. Errors are displayed to the Script Writer, so the offending lines can be corrected.
- Runtime errors are not picked up until a Script executes. The default behaviour is to stop Script execution and run the error-handler Script.

If an error-handler Script file (called *ErrorHandler.fsh*) exists in the same directory as the engine, it gets executed when a fault occurs.

The error-handler Script is especially useful with logon Scripts. Creating the single line shown in Figure 2.1 in the Script *ErrorHandler.fsh* appends all logon execution errors for all Users on a network to a log file (in this example, called *NetworkErrors.log* in shared folder *ITShare*).

This single FSH command provides a comprehensive overview in one file of all logon errors occurring on a network.

Figure 2.1 Sample line in ErrorHandler.fsh

```
AppendFile \\AcmeServer\ITShare\NetworkErrors.log, "[Date], [Time], →  
[Username], [LastError]"
```

The FSH Engine is provided in the download package.

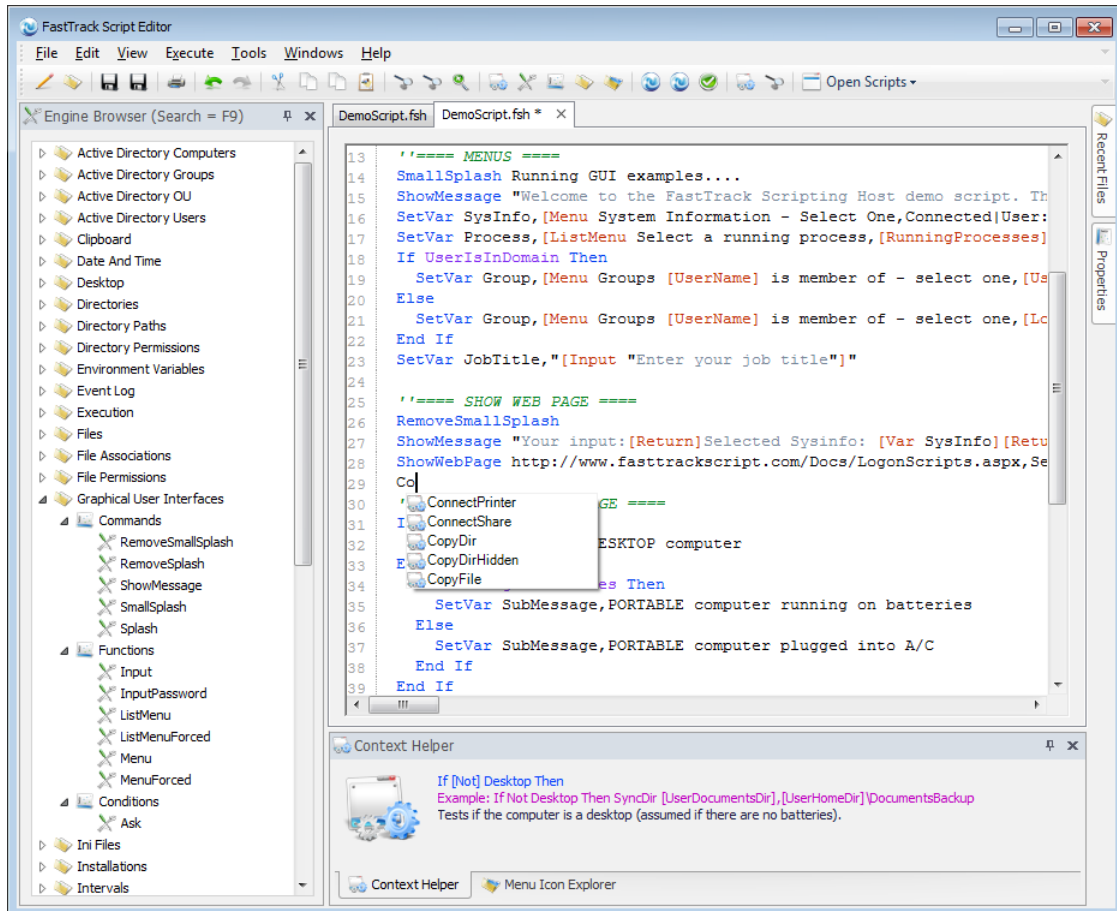
The FSH Editor

About the editor

The FSH Editor is an integrated development environment (IDE), with a standard and familiar look-and-feel, similar to Microsoft's Visual Studio. Among the features supported by the editor are keyword recognition, syntax highlighting, context-sensitive help and guided completion.

Figure 2.2 shows a typical screen view of the FSH Editor.

Figure 2.2 The FSH Editor



The first time the FSH Editor is started, a demo Script is opened, which can be run from within the editor by pressing function key F5. The demo Script showcases some of the many capabilities provided by FastTrack Scripting Host.

As well as executing a Script from within the editor using F5, a Script can be run in debug mode using F6. Running in debug mode enables a Script Writer to step through the Script and see exactly how it will be interpreted at runtime.

The FSH Editor is provided in the download package.

Continued...

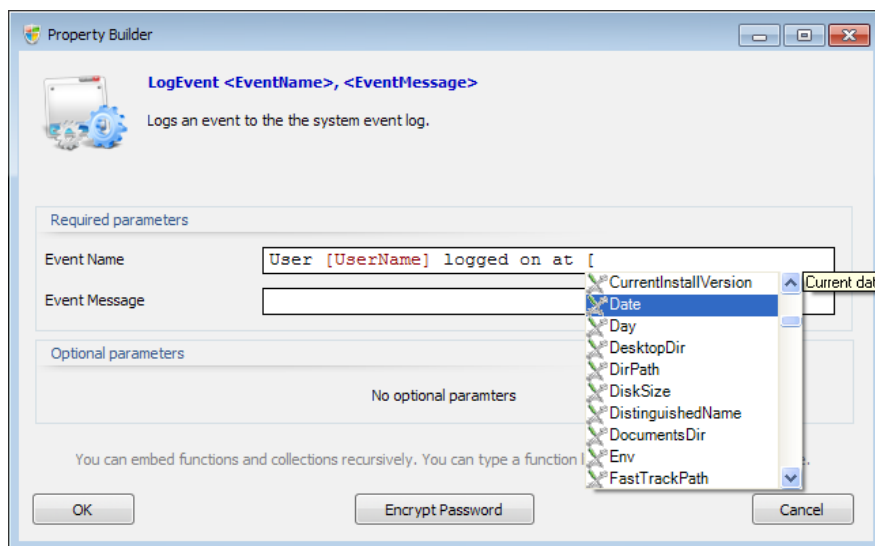
The FSH Editor, Continued

Elements of the editor

- **Script Window** – this is the main part of the editor and takes up most of the editor screen space. It appears in the top center of the screen, immediately below the toolbar.
- **Engine Browser** – the Engine Browser is a tree or hierarchy in the top left of the screen that shows all the commands, functions, collections and conditions available to be included in Scripts.
- **Property Builder** – this is a dialog window that opens to help add the correct parameters. The Property Builder is opened when an item is double-clicked in the Engine Browser tree, as shown in Figure 2.3.

The Property Builder can be disabled via the *Properties Panel*, for Script Writers who prefer to write parameters directly into the Script. If this property is set, a template is copied into the Script instead.

Figure 2.3 The FSH Editor Property Builder



- **Context Helper** – the Context Helper appears below the Script Window and shows a full explanation of the item currently under the cursor in the Script Window, or marked in the Engine Browser tree. For example, to execute a program, locate the *Run* command in the Engine Browser and double-click it to insert that command into the Script at the cursor position. Alternatively, simply click and drag it into the Script.
- **Properties Panel** – the Properties Panel is shown in the bottom right of the screen. This is where key settings for the editor, the engine and the current Script can be changed.
- **Recent Files Panel** – this panel is shown in the top right of the screen and lists the Script files most recently opened.

The FSH Editor is described in more detail in Chapter 4 Using the FSH Editor.

Continued...

The FSH Editor, Continued

Features of the editor

- **Syntax highlighting** – the editor automatically indicates syntax highlighting on commands, functions, collections, conditions and loops (which are all explained later in this document).
- **Context-sensitive help** – when moving the cursor around within a Script, the editor senses what is being done and automatically displays an explanation in the Context Helper.
- **Auto correction** – as text is entered in the Script Window, the case (i.e. capitalization) is automatically adjusted. For example, the command *splash* is automatically corrected to *Splash* with a capital S.
- **Code completion** – in Figure 2.2, at line 55 in the Script Window, *co* was typed and five commands are suggested.
- **Script validation** – when a Script is saved or executed, a simulated run is performed to validate that there are no syntactical errors in the Script, such as a missing parameter. This ensures that a mal-formed Script is not accidentally saved.

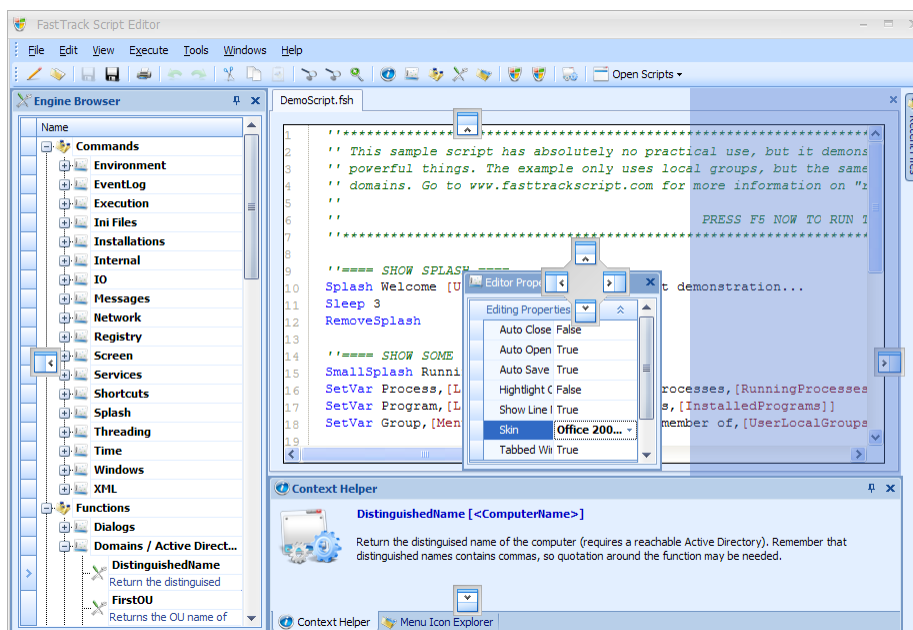
If the Script cannot pass a simulated run, the Script Writer is notified and the line where the problem was found is identified. This means that an invalid Script cannot be saved or executed.

Note:

There is a way around this check. In the Properties Panel, there is a setting *Validate Before Save*. Uncheck this setting to prevent the editor from performing the simulated run prior to saving a Script. There is also a checkmark button in the toolbox that can be clicked to check a Script before (or without) saving.

- **Customization** – the FSH Editor is completely customizable. It is possible to change the skin (i.e. look and feel) and move panels just as can be done within Visual Studio. Figure 2.4 shows the editor where the Properties Panel is about to be docked.

Figure 2.4 Docking the Properties Panel in the FSH Editor



Comparing FSH Editor Features

Language comparison for Script editor features




The FSH Editor has many more features than are provided natively with VBScript and PowerShell. FastTrack Scripting Host is a higher language scripting host than any other scripting language and the cornerstone paradigm has always been:

One operation - One script line, unmatched by any other scripting language.

When constructing and maintaining scripts with VBScript or PowerShell, it is a complex process because they are counterparts of real programming languages, and it is rarely the case that scripts are constructed completely from scratch without copying/pasting previous work or publicly available material. Often third party tools are also included in the scripts, losing control over whether or not it actually does what it's supposed to. FastTrack Scripting Host does not use any third party tools and has full error control on any operation.

Figure 2.5 compares the features of all three products.

Figure 2.5 Working with Scripts using FSH, VBScript and PowerShell

Script Editing			
	 Included out-of-the-box	 Not natively supported	
	FastTrack Scripting Host	VBScript	PowerShell
Script debugging			
Single seat script encryption			
Seamless mass-deployment script encryption			
Syntax highlighting			
Code completion			
Drag and drop scripting			
Prevent saving invalid scripts			
Real-time context help			

FastTrack Logon

About FastTrack Logon

FastTrack Logon replaces logon .bat and .cmd files, and can also replace .wsh Scripts. Users logging on see a customizable graphics screen and run a powerful set of commands, all without requiring deployment to the client computers.

All that is needed is a 50kb FastTrack Logon file *FTLogon.exe* in the domain NetLogon directory and an update of User profiles to point logon Script to this executable. FastTrack Logon then automatically distributes the FSH Engine and all Scripts to client computers for execution when Users log on. FastTrack Logon can also be configured via Group Policy and Active Directory for large numbers of Users in enterprise or department groups.

FastTrack Logon is very simple and it works for all 32-bit and 64-bit Windows operating systems; desktop operating systems (Windows NT4/2000/XP/Vista/7), server operating systems (Windows Server 2000/2003/2008) and terminal server and Citrix sessions.

How does it work?

The process is straightforward – extract all files from *FTLogon.zip* into the NetLogon share on the domain controller and point the Users' logon Scripts to FastTrack Logon. When it is executed, it will do the following on each client computer:

- Synchronize the engine *Fsh.Exe* and logon Script files to Local Settings\Application Data\FastTrack in each User's non-roaming part of the profile.
- Execute the predefined logon Scripts from this client location.
- Associate the .fsh extension to the profile copy of *Fsh.exe*.

The first time each User logs on to a client computer, about 1 MB of data is copied from the domain controller to the non-roaming part of the User's profile. After that, copying happens only when the files in the *fshbin* folder are changed on the domain controller – anything put into the *fshbin* directory on the domain controller will be synchronized to each client computer.

There is no limitation on the files and directories that can be put under *fshbin* on the domain controller. A Windows Active Directory is not a requirement; FastTrack Logon also works for domains without Active Directory.

The local copy of the files and the file association is per User, because this way no further work is required for client computers to support FSH Scripts in general in each User's context. There is no need for a per-machine deployment and the User is not required to be a local administrator. The client location of files can be customized, if they need to be placed somewhere else.

File and folder structure

The FastTrack Logon package is a compressed zip file called *FTLogon.zip*, installed by default into *<Program Files>\FastTrack Software\FastTrack Scripting Host* when FSH is installed. It contains the FastTrack Logon program *FTLogon.exe*, SmartDock and the sample Scripts.

When installation is complete, including obtaining a license file and extracting the FastTrack Logon package, the following two file and folder structures are created (assuming default installation locations).

Continued...

FastTrack Logon, Continued

File and folder structure, Continued

1. Folder <Program Files>\FastTrack Software\FastTrack Scripting Host

This structure is created after running the installation program *Setup.exe* and applying a license file:

- DemoScript.fsh
- Editor.exe
- FSH User's Guide.pdf
- Fsh.exe
- Fsh.lic
- Fsh.pk
- FTLogon.zip

2. Share name \\<Domain Server>\NetLogon

This structure is created after extracting *FTLogon.zip* into the domain NetLogon share and copying the license file into the *fshbin* folder:

- Fshbin (folder)
 - ErrorHandler.fsh
 - Fsh.exe
 - Fsh.lic
 - PostInstall.fsh
 - PostLogon.fsh
 - PostUninstall.fsh
 - PreLogon.fsh
 - SmartDock.exe
 - SmartDock.fsh
- FTLogon.exe
- ftlogon.ini

SmartDock

About SmartDock

SmartDock is a small executable program, called *SmartDock.exe*, which detects whenever the IP settings of any network adapter change on a computer. SmartDock is by default distributed automatically to clients through FastTrack Logon.

In programming jargon, an IP address settings change is known as an *event* – once such an event is detected, SmartDock can execute a Script called *SmartDock.fsh*.

SmartDock can run in any of the following scenarios:

- start-up or reboot
- resume from standby
- connection to a virtual private network (VPN)
- connection to a wireless network
- manual change of IP address settings
- any other scenario where the IP settings are changed

For any event where IP settings change, SmartDock.exe runs and triggers the execution of SmartDock.fsh.

Why is SmartDock useful?

Users of portable computers often put their computers into standby or hibernation at one location and then resume the session somewhere else, such as moving from a work network to a home environment.

With SmartDock, both Users and Network Administrators have control over what happens every time the computer is resumed or connects to a network. The SmartDock tool is a good supplement to domain logon Script for portable computers.

Running SmartDock

What actually happens when SmartDock executes (i.e. when the commands and functions in SmartDock.fsh are interpreted) is entirely up to the Script Writer.

A SmartDock Script is no different from any other FSH Script – the default name of SmartDock.fsh simply means that it is the Script executed when SmartDock.exe is triggered.

SmartDock is useful for portable computers that roam. Setting or removing proxy server settings based on whether or not the computer is on a LAN would be one thing to do instead of using the Internet Explorer auto detection mechanism. Another use could be connecting or disconnecting common shares and local printers. Think of the SmartDock Script as an offline supplement to the logon Script.

An executable program, not a service

SmartDock is not a Windows Service but a small executable file. The reason for this is simple; if it was a Windows Service, it could not run in the User's context and then it would be worthless to both Users and Network Administrators. It would be impossible to set proxy server settings, changing default printers and all the other things that are related to a particular User.

A sample SmartDock Script is included later in this chapter under *Sample Scripts*. SmartDock is provided in the download package.

Sample Scripts

Included in the download package

Several sample Scripts are provided in the download package for FSH. As well as the demo Script, these include a sample logon Script (both pre- and post- the start of the Windows explorer process), several installation Scripts, an error-handling Script and a SmartDock Script.

Demo Script

DemoScript.fsh

The DemoScript is a showcase of the capability of FastTrack Scripting Host. It includes some of the more advanced functionality such as setting variables, showing menus, accepting User selections and then carrying out tasks based on those selections.

```
/******  
*  
*          PRESS F5 TO RUN THE SCRIPT          *  
*  
******/  
  
''==== SPLASH ====  
Beep  
Splash Welcome [UserName],to a demo script  
Sleep 5  
RemoveSplash  
  
''==== MENUS ====  
SmallSplash Running GUI examples...  
ShowMessage "Welcome to the FastTrack Scripting Host demo script. _  
This script, stripped of comments, contains only 23 lines of scripting. _  
It will show you a small fraction of the functionality using the graphical _  
user interface functions. [Return] [Return]Menus can be used for _  
e.g. application launchers, printer selection and toolboxes. _  
They are very easy to use to collect user choices. The next 4 _script lines  
will collect your selections, and sum them up using the collected variables."  
SetVar SysInfo, [Menu System Information - Select One,Connected|User: [UserName], _  
PC|Computer: [ComputerName],Shield|OS Version: [OperatingSystemVersion], _  
Config|Memory: [TotalMemory]MB,NetworkFolder|IP Address: [IPAddress], _  
Save|Personal files: [UserPersonalDir],Network|CPU speed: [CpuSpeed]MhZ, _  
HardDrive|Disk size: [DiskSize]GB,  
Screen|Screen: [ScreenWidth]x[ScreenHeight]x[ScreenDepth], _  
ControlPanel|Processors: [NoProcessors]]  
SetVar Process, [ListMenu Select a running process, [RunningProcesses]]  
If UserIsInDomain Then  
    SetVar Group, [Menu Groups [UserName] is member of - select one, [UserLocalGroups]]  
Else  
    SetVar Group, [Menu Groups [UserName] is member of - select one, [LocalUserLocalGroups]]  
End If  
SetVar JobTitle, "[Input "Enter your job title"]"
```

Continued...

Sample Scripts, Continued

Demo Script, Continued

```
''==== SHOW WEB PAGE ====
RemoveSmallSplash
ShowMessage "Your input:[Return]Selected Sysinfo: [Var SysInfo][Return] _
Selected process: [Var Process][Return]Selected group: [Var Group][Return] _
Entered job title: [Var JobTitle][Return][Return] _
Next step - Showing a web page:[Return]The ShowWebPage command displays a webpage,
waiting for the user to press OK. Combined with a condition to show it only once _
per user, it can be used to e.g. display important corporate information,
displaying an intranet or internet page. This will ensure that every employee _
in the company gets the information"
ShowWebPage http://www.fasttrackscript.com/Docs/LogonScripts.aspx,Setting up _
Domain Logon Scripts with FastTrack Scripting Host

''==== SHOW END MESSAGE ====
If Desktop Then
  SetVar SubMessage,DESKTOP computer
Else
  If RunningOnBatteries Then
    SetVar SubMessage,PORTABLE computer running on batteries
  Else
    SetVar SubMessage,PORTABLE computer plugged into A/C
  End If
End If
If Ask "FastTrack Scripting Host has lots of simple conditions to determine _
even to most complex things. In this case it has been determined that _
you ran the script on a [Var SubMessage].[Return][Return]Thank you for
using FastTrack Scripting Host. Please open the "FSH User's Guide" icon in _
your start menu or go to www.fasttrackscript.com for documentation. _
[Return][Return]Would you like to see the interactive list of Fasttrack
Scripting Host functionality before exiting the demo script? _
Answering 'Yes' will run 'FSH.exe /?'." Then Run [FastTrackPath]\FSH.Exe,/?
```

Logon Scripts

Pre-Windows explorer.exe process

PreLogon.fsh

The PreLogon Script is one of two sample logon Scripts provided with FastTrack Logon. It executes prior to the Windows explorer.exe process starting up, which means Users cannot do anything on their computers until it finishes.

By default, the script shows only a splash screen and the rest is comment lines that can be uncommented and used as templates for getting started with logon Scripts. Sample logon Scripts are explained later in this document.

Continued...

Sample Scripts, Continued

Logon Scripts, Continued

PreLogon.fsh, Continued

The default name is PreLogon.fsh, but its name can be changed via FTLogon.ini.

```
''=====
''SAMPLE PRELOGON.FSH - EXECUTES BEFORE EXPLORER STARTS
''=====

''==== SHOW WELCOME SCREEN ====
splash welcome to the company network, [UserFullName]

''==== CONNECT PERSONAL SHARE - UNCOMMENT TO ENABLE ====
''ConnectShare [UserHomeDrive], [UserHomeDir], Personal data

''= CONNECT COMMON SHARES - REPLACE WITH COMPANY SPECIFIC DATA AND UNCOMMENT =
''ConnectShare J:, \\AcmeServer\CommonShare, Common data
''If UserIsMemberOf SalesStaff Then ConnectShare K:, \\AcmeServer\SalesShare, →
    Sales data

''==== PAUSE - JUST FOR DEMONSTRATION ====
sleep 5

''==== BACKUP DOCUMENTS ON PORTABLES ====
''If Portable Then
'' Splash Backing up documents for, [UserFullName]
'' IF UserOnceADay Then SyncDir [UserDocumentsDir], [UserHomeDir]\DocumentBackup
''End If

''==== INSTALL SMARTDOCK ====
''If Portable Then
'' If UserOnce Then
''     SetUserStartupItem SmartDock, [UserAppDataDir]\FastTrack\SmartDock.exe
'' End If
''End If
```

Post-Windows explorer.exe process

PostLogon.fsh

The PostLogon Script is the second sample logon Script provided with FastTrack Logon to handle the initial logging on of Users. Unlike PreLogon.fsh, it executes after the Windows explorer.exe process starts, meaning Users can start working while this Script is running. This script has no functionality by default; example use could be connecting printers or setting client time.

The default name is PostLogon.fsh, but its name can be changed via FTLogon.ini.

```
''=====
''SAMPLE POSTLOGON.FSH - EXECUTES AFTER EXPLORER STARTS
''=====

''==== SET CLIENT TIME ====
''SetTimeFromServer

''==== SLEEP FOR DEMONSTRATION PURPOSES ====
sleep 3
```

Sample Scripts, Continued

Installation Scripts

PostInstall.fsh

The PostInstall Script is provided as a sample Script in the FTLogon.zip package, but it is not directly related to FastTrack Logon. This Script is geared towards application installations and it runs automatically after an installation completes. The installation may or may not be initiated from another FSH Script.

The general idea with this Script is to log installations and/or report them to a management system.

```
''=====
''TEMPLATE POSTINSTALL.FSH - EXECUTES WHEN AN INSTALLATION IS COMPLETE
''
''Functions [CurrentInstallName] and [CurrentInstallVersion] will
''return the installation name and version.
''=====

''==== REPORT AN INSTALLATION - REPLACE WITH A COMPANY SPECIFIC SHARE WITH →
''WRITE ACCESS TO ALL USERS AND UNCOMMENT ====
''AppendFile \\MyServer\MyAdministrativeShare$\Install.log, "[Date], [Time], →
''[ComputerName], [CurrentInstallName] v[CurrentInstallVersion]"
```

PostUninstall.fsh

The PostUninstall Script is provided as a sample Script in the FTLogon.zip package, but, like PostInstall.fsh, it is not directly related to FastTrack Logon. This Script is geared towards removing software applications and it runs automatically after an un-installation completes. The un-installation may or may not be initiated from another FSH Script.

```
''=====
''TEMPLATE POSTUNINSTALL.FSH - EXECUTES WHEN AN INSTALLATION IS COMPLETE
''
''Functions [CurrentInstallName] and [CurrentInstallVersion] will
''return the installation name and version.
''=====

''==== REPORT AN UNINSTALLATION - REPLACE WITH A COMPANY SPECIFIC SHARE WITH →
''WRITE ACCESS TO ALL USERS AND UNCOMMENT ====
''AppendFile \\MyServer\MyAdministrativeShare$\Uninstall.log, "[Date], →
''[Time], [ComputerName], [CurrentInstallName] v[CurrentInstallVersion]"
```

Error-handling Script

ErrorHandler.fsh

The ErrorHandler Script runs automatically whenever another FSH Script fails. The only exception to this rule is the ErrorHandler.fsh Script itself, otherwise an infinite loop would be created.

Any valid FSH command can be used in the ErrorHandler Script, but its most common use is to record errors occurring, as in the sample Script below.

Continued...

Sample Scripts, Continued

Error-handling Script, Continued

ErrorHandler.fsh, Continued

A FastTrack Logon Script will not stop execution if an error occurs. Every time an error occurs, the errorhandler.fsh Script is inserted into the running Script, where the error occurred and then the running Script continues. If a logon Script fails three times for a user on connecting shares, the errorhandler.fsh will be executed three times. This enables the ability to get a centralized log of all execution errors.

```
''=====
''SAMPLE ERRORHANDLER.FSH - EXECUTES WHEN AN EXECUTION ERROR OCCURS
''=====

''==== REGISTER AN EXECUTION ERROR - REPLACE WITH A COMPANY SPECIFIC SHARE ->
''      WITH WRITE ACCESS TO ALL USERS AND UNCOMMENT ====
''AppendFile \\MyServer\MyAdministrativeShare$\NetworkErrors.log, "[Date], ->
''      [Time], [UserFullName], [LastError]"
```

SmartDock Script

SmartDock.fsh

The SmartDock Script executes automatically whenever the SmartDock executable *SmartDock.exe* recognizes a change of IP address setting. Both *SmartDock.exe* and *SmartDock.fsh* must be present for the Script to trigger correctly. By default the *SmartDock.fsh* Script has only comment lines, and these can be uncommented and used as a template for a SmartDock Script.

```
''SmallSplash "Setting up network, please wait..."
''If Alive AcmeProxy Then
''  SetProxyServer AcmeServer,8080
''  ConnectShare J:,\\AcmeServer\CommonShare
''  ConnectShare [UserHomeDrive],[UserHomeDir]
''Else
''  DisableProxyServer
''  DisconnectAllShares
''End If
```

Chapter 4

Using the FSH Editor

Overview

Chapter 4 describes the editor that is a key part of FastTrack Scripting Host. The FSH Editor is an integrated development environment (IDE) with a look-and-feel similar to common Microsoft IDEs, including Visual Studio.

The FSH Editor is the tool used to create and test Scripts prior to deployment. It has all the features required for building Scripts, as well as a number of powerful features that are normally found only in large, complex programming IDEs, such as syntax highlighting, code-completion and Script validation.

The chapter begins with a discussion of the graphical user interface presented to Scrip Writers, followed by an explanation of the primary building blocks for FSH Scripts – commands, functions, collections, conditions and loops.

A section on testing and debugging Scripts is also covered in this chapter, including how to run Scripts on other computers and the command-line parameters available for the FSH Engine *Fsh.exe*.

In this chapter

Topic	See Page
The FSH Editor Graphical User Interface.....	32
Editor Building Blocks	34
Executing and Debugging Scripts	39

The FSH Editor Graphical User Interface

Layout

The FSH Editor has a graphical user interface (GUI) that provides a layout similar to Microsoft's Visual Studio. This means the look-and-feel is familiar to many Microsoft developers and it follows the standards and conventions expected.

The first time the FSH Editor is started, a demo Script is opened, which can be run from within the editor by pressing function key F5. The demo Script appears in the Script Window, one of the main elements of the editor, and it showcases some of the many capabilities provided by FastTrack Scripting Host.

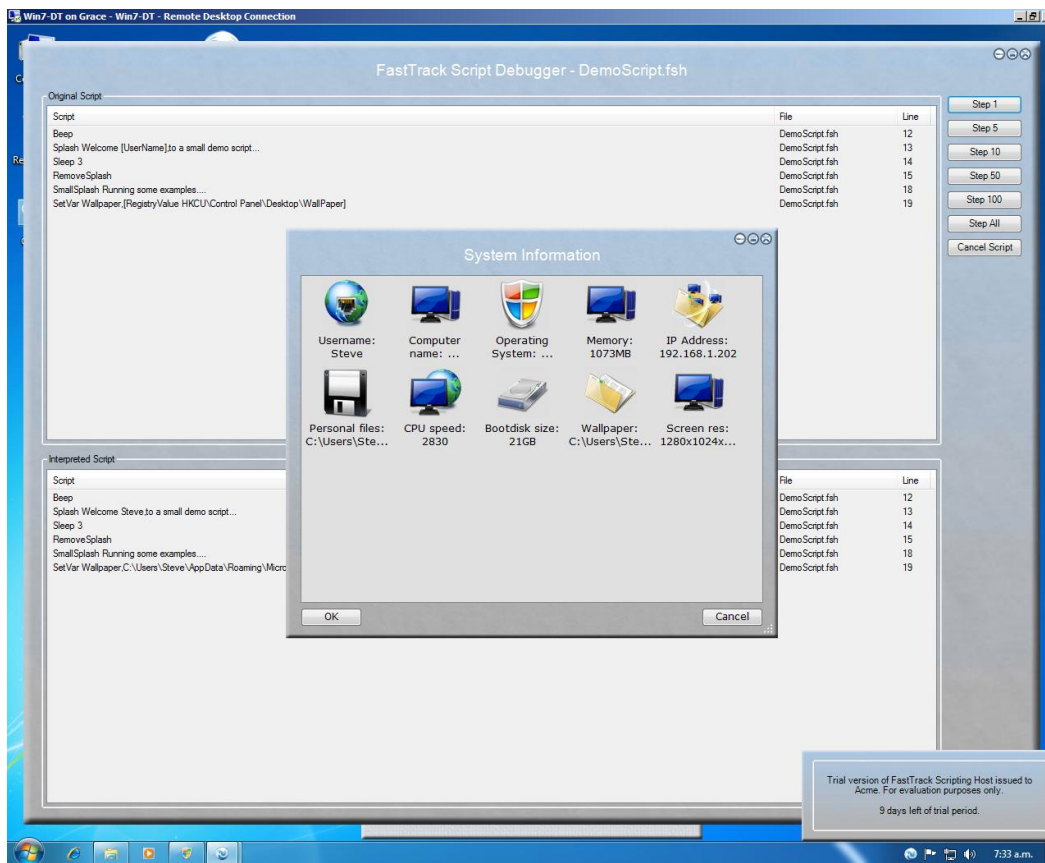
Debugging Window

As well as executing a Script from within the editor using F5, a Script can be run in debug mode using F6. Running in debug mode enables a Script Writer to step through the Script and see exactly how it will be interpreted at runtime.

The Debugger can execute the entire Script, or it can step through lines one at a time, or in groups of 5, 10, 50 or 100.

Figure 4.1 show the Debugging Window after executing the first few lines of the Demo Script.

Figure 4.1 Demo Script in the Debugging Window



Continued...

The FSH Editor Graphical User Interface, Continued

Summary of elements and features

In addition to the Debugger, the editor has a range of other capabilities that were described in Chapter 3 when the FSH Editor was introduced. The following table summarizes these elements and features.

Editor Elements	Editor Features
<p>Script Window This is the main part of the editor and takes up most of the editor screen space. It appears in the top center of the screen, immediately below the toolbar.</p>	<p>Syntax highlighting The editor automatically indicates syntax highlighting on commands, functions, collections, conditions and loops (which are all explained later in this document).</p>
<p>Engine Browser The Engine Browser is a tree or hierarchy in the top left of the screen that shows all the commands, functions, collections and conditions available to be included in Scripts.</p>	<p>Context-sensitive help When moving the cursor around within a Script, the editor senses what is being done and automatically displays an explanation in the Context Helper.</p>
<p>Property Builder This is a dialog window that opens to help add the correct parameters. The Property Builder is opened when an item is double-clicked in the Engine Browser tree, as shown in Figure 2.3.</p>	<p>Auto correction As text is entered in the Script Window, the case (i.e. capitalization) is automatically adjusted. For example, the command <i>splash</i> is automatically corrected to <i>Splash</i> with a capital S.</p>
<p>Context Helper The Context Helper appears below the Script Window and shows a full explanation of the item currently under the cursor in the Script Window, or marked in the Engine Browser tree.</p>	<p>Code completion The editor suggests keywords as soon as typing begins, based on the characters entered. The more characters entered, the more focused the list of keywords becomes.</p>
<p>Properties Panel The Properties Panel is shown in the bottom left of the screen. This is where key settings for the editor, the engine and the current Script can be changed.</p>	<p>Script validation When a Script is saved or executed, a simulated run is performed to validate that there are no syntactical errors in the Script, such as a missing parameter. This ensures that a mal-formed Script is not accidentally saved.</p>
<p>Recent Files Panel This panel is shown in the top right of the screen and lists the Script files most recently opened.</p>	<p>Customization The FSH Editor is completely customizable. It is possible to change the skin (i.e. look and feel) and move panels just as can be done within Visual Studio. Figure 2.4 shows the editor where the Properties Panel is about to be docked.</p>

This chapter continues with detailed information about other building blocks that comprise the FSH Editor, including operations, parameters, variables and labels.

Editor Building Blocks

Operations

Building blocks within the FSH Editor comprise five types of operation:

- Commands
- Functions
- Collections
- Conditions
- Loops

Commands

A *command* is an operation that "does something", such as perform a file operation or connect a share. An example is the ConnectShare command in Figure 1.2 (reproduced below exactly as seen in the editor, using syntax highlighting):

Figure 4.2 ConnectShare command

```
If UserIsMemberOf SalesStaff Then ConnectShare I:, \\AcmeServer\Sales
```

Functions

A *function* is notated with square brackets, for example [UserFullName] and it will be resolved before a command, condition or loop is executed.

The UserFullName function returns the currently logged-on User's full name registered in the domain (or Active Directory). Think of a function as a textual replacement before commands, conditions and loops are executed.

Functions can have parameters. For instance [Var Server] will return the content of a variable named "Server".

Functions can embed other functions recursively (i.e. in a number of "embeddings") and are always resolved from the most embedded level and upwards. There is no limit imposed by the FSH Editor or FSH Engine on the depth of recursion.

Consider this example from the demo Script:

Figure 4.3 SetVar function

```
SetVar File,[ListMenu Select a file,[Files [FastTrackPath]]]
```

Resolution by the engine of this command is as follows:

- First the path of the engine, [FastTrackPath], is resolved, as this is the innermost function.
- Then the result is passed to the Files collection.
- The files collection will then pass a list of files in that directory to the ListMenu function.
- The final result is a graphical menu with all the files in the directory and the selected choice goes into the "File" variable for later use.

Continued...

Editor Building Blocks, Continued

Collections

A *collection* is essentially the same as a function, except that it returns a comma-separated list of values instead of just one value. This means that a collection is the same format as a list of parameters, making collections very useful for loops or menus, as shown in Figure 4.7 below (under Loops).

Conditions

If-statements with Conditions

A condition is used with an *if-statement*. If the condition is true, the inner code is executed, otherwise it is not, as shown in Figure 4.4.

Figure 4.4 If ... Then ... Else condition

```
If Desktop Then
  ShowMessage You're on a DESKTOP computer!,Information
Else
  ShowMessage You're on a PORTABLE computer!,Information
End If
```

"Not" can always be specified after "if" for the opposite result and AND and OR operators can be specified for multiple conditions.

If there is just one command that has to be executed if the condition is true, it can be specified after "Then". In that case there is no "else" or "end if" needed.

Figure 4.5 If ... Then condition with AND

In this example, the "And" operator is used to perform a backup on a portable computer if it is plugged into A/C power (both lines are the same line).

```
If Portable And Not RunningOnBatteries Then SyncDir
[UserDocumentsDir], [UserHomeDir]\DocumentsBackup
```

If-statements with operators

"If" statements can also be used without conditions for basic comparison of text or numbers.

The operators <, > and = can be used for comparisons. For "not equal to", the "!=" operator and the "not" operator are used together.

In Figure 4.6, the result of the "FreeDiskSpace" function is compared to a number.

Figure 4.6 Using comparisons

```
If [FreeDiskSpace]<100 Then
  ShowMessage Please contact HelpDesk, diskspace is low!,Warning
End If
```

Continued...

Editor Building Blocks, Continued

Loops

Loops are ways to iterate through collections, which are comma-separated lists.

The first parameter is the name of a variable that is set during iteration and contains the actual value inside the loop. Figure 4.7 synchronizes a data directory to all Citrix servers in a farm.

Figure 4.7 Using loops

```
Loop Server, [CitrixServers]
  SyncDir C:\Data, \\[Var Server]\d$\Data
End Loop
```

Additional editor building blocks

Parameters

Parameters are always separated by commas and can always be written inside quotes. Quotes are not needed in general, but if a text contains (or the result of a function can contain) a comma, it must be quoted to tell the engine that there are no further parameters.

Figure 4.8 contains a comma inside the text, and therefore the text must be quoted to ensure that the engine will not see three parameters instead of the correct two. Syntax highlighting helps in this case, because it is clear the comma is part of the quoted text.

Figure 4.8 Using quotes with parameters

```
If [FreeDiskSpace]<5 Then ShowMessage "Your harddrive space is too low, →
  contact HelpDesk", Error
```

Variables

A variable is defined with the command "SetVar" and retrieved again with the function "Var". The example shown in Figure 4.9 presents a menu with the names of all local security groups and puts the selected one into the variable "Group" (first line).

The second line retrieves the value again and displays it in a message box.

Figure 4.9 Using variables

```
SetVar Group, [Menu Select a group, [LocalGroups]]
ShowMessage [Var Group], You selected
```

Continued...

Editor Building Blocks, Continued

Additional editor building blocks, Continued

Comments

Comments can be used anywhere, and are notated as two single quotation marks with nothing between them. Anything beyond these two characters on any line is considered a comment. Block comments for multiple lines are also possible by starting with `/*` and ending with `*/` (as in C#, C++ and a number of other languages).

Breaking lines

A single, long script line can be broken into multiple lines by ending the line with an underscore character (`_`). The next line is then parsed as belonging to the previous line. In this way, a long script line can be broken into an infinite number of lines to make it more readable in the Editor.

Go-to labels

Any line that starts with a colon is a *go-to label*. In most other scripting and programming languages, labels cannot contain spaces for parsing reasons. This is not the case with FastTrack Scripts – labels *can* contain spaces.

The reason why this is important is that in almost any case where a *go-to* is needed, spaces will be involved. The example in Figure 4.10 is a menu that synchronizes data to a selectable server.

Figure 4.10 Using go-to labels

```
SetVar Selected,[Menu Select operation, Replicate to Server 1,Replicate to →  
Server 2,Replicate to Server 3]  
if Not [Var Selected]= Then Goto [Var Selected]  
Exit  
  
:Replicate to Server 1  
SyncDir C:\Data, \\Acme\Server1\Data  
Exit  
  
:Replicate to Server 2  
SyncDir C:\Data, \\Acme\Server2\Data  
Exit  
  
:Replicate to Server 3  
SyncDir C:\Data, \\Acme\Server3\Data  
Exit
```

Continued...

Editor Building Blocks, Continued

Additional editor building blocks, Continued

Embedding other scripting languages

It is possible (and easy) to include a Script written in another scripting language into a FastTrack Script. This is done simply by issuing a Run command. The Run command uses shell execution, meaning that the executable does not need to be specified to handle a particular file type, if there is already an association to that specific scripting type.

To hand over a status from the embedded script, or an executable, exit codes can be used. In VBScript, an exit code can be optionally supplied to the VBScript code. Quit like this:

```
WScript.Quit(200)
```

Inside the FastTrack Script, this status can then be read by using the LastExitCode function:

```
Run AcmeScript.vbs
If [LastExitCode]=200 Then
    '..Do something...'
End If
```

Using Escape

Functions and collections reside inside square brackets []. If the bracket characters themselves are ever needed (for example, when writing a registry value that contains them), the *char* function must be used instead.

The [Char 91] function produces a start bracket character and [Char 93] produces an end bracket character.

Executing and Debugging Scripts

Testing a Script within the editor

The simplest way to execute and test a Script is within the editor by simply pressing F5. This runs the Script (on that computer only) and makes it very easy to observe results, debug the Script and make changes.

For more complex Scripts and harder to resolve errors, the Debugger can be started by pressing F6.

Running Scripts outside the editor

Scripts can be executed outside the FSH Editor, and on other computers, by using the FSH Engine (fsh.exe). Simply make the engine and license file available to any other computers that need to run Scripts. For example, the engine and license file could be placed at a different location – the only requirement is that client computers have access to that location.

To run Scripts outside the editor, simply double-click an fsh file and it executes, because the fsh extension is associated with FSH.exe.

Note:

Double-clicking a Script (.fsh) file on a computer that has the editor installed runs the Script – it does not open the Script inside the editor. Use File > Open inside the editor to open a Script.

If using FastTrack Logon (FTLogon.exe), the same can be done on all computers, as FastTrack Logon makes the same file association. FastTrack Logon is explained in more detail in Chapter 5 Building Scripts.

Running Scripts on other computers

To execute a Script on any other computer, all that is needed is a copy of FSH.exe and the FSH.lic file. Simply make the files available to the target computer and pass the Script as the only parameter, as shown in Figure 4.11.

Figure 4.11 Passing a Script to the FSH Engine for execution

```
FSH.Exe MyScript.fsh
```

Command line parameters

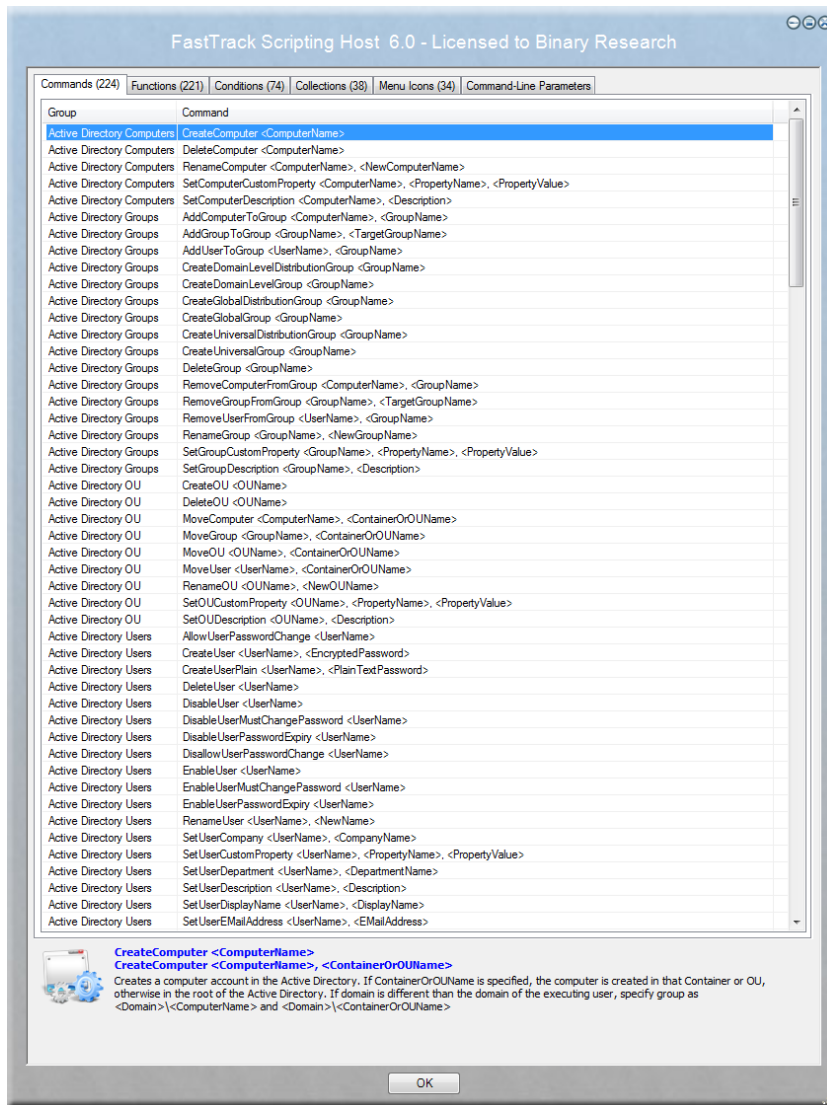
There are several parameters that can be passed to the engine on the command line. These are documented below and also on a computer that has the FSH Editor installed, by selecting Start > All Programs > FastTrack Scripting Host > Documentation > Language Reference. This selection displays the dialog box shown in Figure 4.12.

Continued...

Executing and Debugging Scripts, Continued

Running Scripts on other computers, Continued

Figure 4.12 FSH Language Reference



The last tab in this dialog box lists the command line parameters, which are:

- **/?** – Show the Language Reference dialog box
- **/d** – Debug the execution of a Script
- **/ie** – Ignore execution errors
- **/if** – Ignore Script faults
- **/n** – Do not display the Notify icon in the system tray
- **/p** – Specify a parameter to pass to the Script via the CmdParam function
- **/s <ScriptFile>** – Execute the ScriptFile. If no other parameters are passed to the engine, the /s can be omitted (as in Figure 4.11, where MyScript.fsh is the only parameter).

Continued...

Executing and Debugging Scripts, Continued

Running Scripts on other computers, Continued

Note that for security reasons, FSH follows the .net security model, to allow a User or Network Administrator to control where it may execute from. The default policy is to not allow execution of applications from network shares, so fsh.exe will not be allowed to run, unless a Network Administrator grants the intranet zone full rights on the executing machine.

For distributing to all computers in a network, The use of FastTrack Logon is highly recommended. FastTrack Logon automatically distributes the engine and logon Scripts to all computers seamlessly.

Handling Script errors

Error handling is essential in any scripting or programming. In VBScript, a Script Writer can essentially choose to ignore errors and see what happens, or stop execution with a fault. With FastTrack Scripting Host, a Script Writer has more control over errors.

If there is a file called *ErrorHandler.fsh* in the same directory as the engine (fsh.exe), it executes when a fault occurs. The Script Writer can then choose if the Script should include the error handler and continue, or simply stop execution. By choosing to continue, a Script Writer has the ability to do some logging and reporting as part of subsequent Script operations.

Figure 4.13 shows an example that could be used as the content of ErrorHandler.fsh. A Script like this is very useful when used in conjunction with FastTrack Logon.

Figure 4.13 Typical ErrorHandler.fsh

```
AppendFile \\AcmeServer\ITShare\NetworkErrors.log,"[Date], [Time], →  
[Username], [LastError]"
```

When an error occurs, for example, being unable to map a share, Script execution switches to ErrorHandler.fsh, which appends information to a global log file (in this example NetworkErrors.log) and then continues execution.

This means that a Network Administrator will see **all** execution errors on the entire network in a log file without stopping Script execution. This feature provides an extremely useful view of network issues for Administrators – all errors on all computers logged in a single file.

Continued...

Executing and Debugging Scripts, Continued

Debugging command line parameters

The engine has two command line parameters related to the way faults are handled. In general there is a distinction between Script syntax errors and runtime errors:

- Syntax errors are picked up before a Script is saved and executed. Errors are displayed to the Script Writer, so the offending lines can be corrected.
- Runtime errors are not picked up until a Script executes. The default behaviour is to stop Script execution and run the error-handler Script.

The distinction is there to allow Network Administrators to decide what behavior is needed for each type of error, which can be controlled with command line switches.

/ie

If `/ie` is specified, runtime errors will not stop Script execution and every time a runtime error occurs, the error handler Script is executed before continuing.

If this parameter is used, all functions return a blank string if they fail and conditions are not entered if the condition test fails. FastTrack Logon executes logon Scripts with `/ie` to ensure full execution.

/if

If `/if` is specified, Script faults can be ignored and the FSH Engine will continue to attempt to execute a Script.

Important:

Use `/if` with caution!. Running Scripts with the `/if` parameter is not recommended because Script faults are severe and generally fatal.

Note:

The `ErrorHandler.fsh` Script will never fail itself, because that would produce a circular, never-ending loop. If there is a runtime or Script error in the error handler Script, it will be ignored. Always test that the error handler Script works by executing it directly first.

The function `LastError` can be used to return (and therefore log) the error message that would otherwise have been shown to the User.

Encrypting Scripts

Script encryption is needed for hiding information from Users. If a Script is not encrypted, it is possible for Users to access information that is not intended for them, like installation paths and administrative domain User account names.

Passwords are already encrypted in Scripts, as all commands and functions requiring the password of a domain User must be specified in encrypted format from the editor. However, the content of a Script itself is not encrypted, unless specifically requested by the Script Writer.

To use Script encryption, it is necessary to have a public key and a private key. The public key is the license file and the private key is a key file that is made available to fully licensed Customers after purchasing a license.

The license file `fsk.lic` is the public key that is put on all computers and it allows encrypted Scripts to be executed, but not modified. Using encryption makes the Script file completely unreadable (like a binary file) once it is encrypted.

Continued...

Executing and Debugging Scripts, Continued

Encrypting Scripts, Continued

Two encryption keys

To modify and encrypt Scripts, both the license key file *fsb.lic* and the private key file *fsb.pk* are required, which means that no one outside of the Customer organization can decode the Scripts. Without both of these key files, the encrypted Scripts cannot be modified.

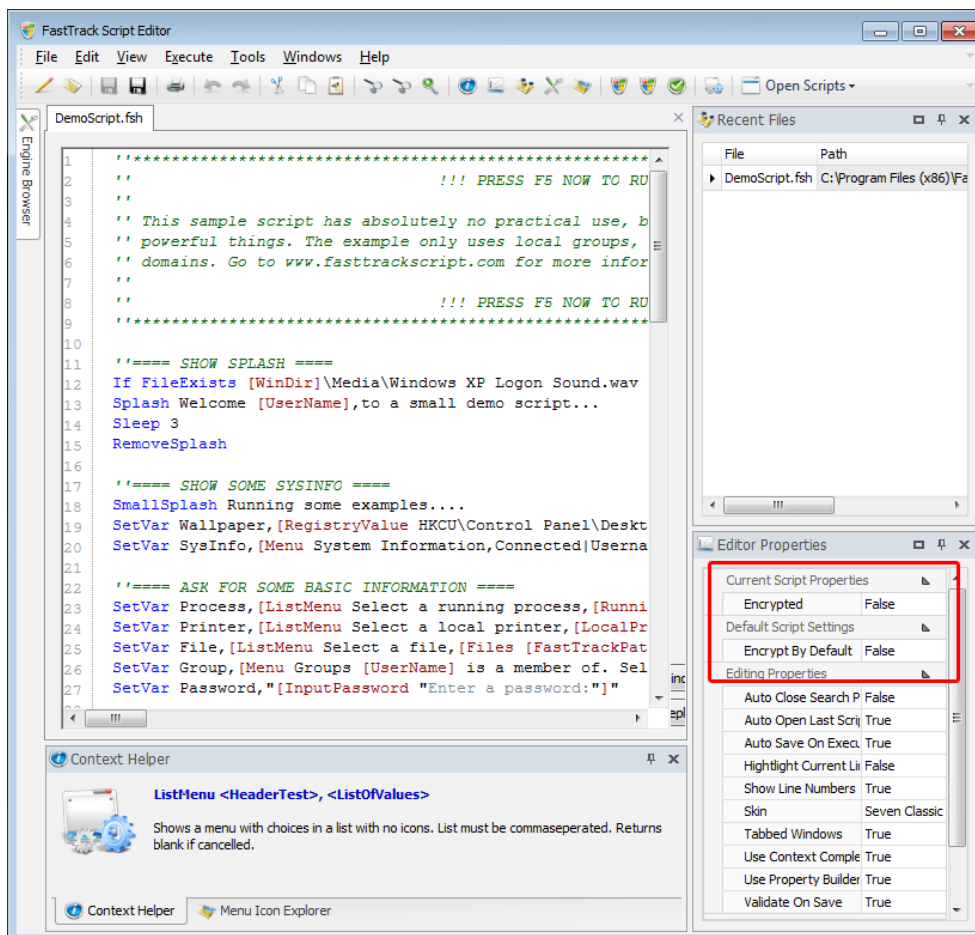
Once the license key file and the private key file have been received, put them in the same folder as the editor, which by default is *FastTrack Software\FastTrack Scripting Host* in the program files folder.

Important:

DO NOT put the private key on client computers and DO make sure it is safe from copying. In the event the private key is lost, it is possible to download it again from fasttrackscript.com, provided the same logon credentials are used as those sent at the time of purchase.

When both files are in the editor folder, Script encryption is completely transparent. Just start the editor, open a Script and select "Encryption" in the Script properties, as shown in Figure 4.14 below, and that's it. Remember that the Scripts can now execute **only** on computers that have a copy of the Customer organization's license key file.

Figure 4.14 Enabling Script encryption



Chapter 5

Building Scripts

Overview

Chapter 5 describes the creation of different types of scripts, including logon Scripts and installation Scripts. An explanation of FastTrack Logon is given, together with unpacking the downloadable zip file, the resulting file and folder structure and setting up domain logons so that they can use FastTrack Logon Scripts.

The working of the two sample logon Scripts (PreLogon.fsh and PostLogon.fsh) is explained and a procedure is given for un-hiding Script display on post-Windows Vista operating systems.

The Installation Scripts topic shows how to use FSH to build Scripts for installing application software. It also explains how to use PostInstall.fsh and Postuninstall.fsh.

Finally, this chapter covers SmartDock, a useful connectivity tool for running Scripts automatically on computers that move between networks, or that move from being on a network to not being on a network (or vice versa). SmartDock can recognize a change in IP address settings and run a Script when this event occurs.

In this chapter

Topic	See Page
Logon Scripts.....	46
Installation Scripts	52
SmartDock Connectivity Tool	60
Where to from here?	62

Logon Scripts

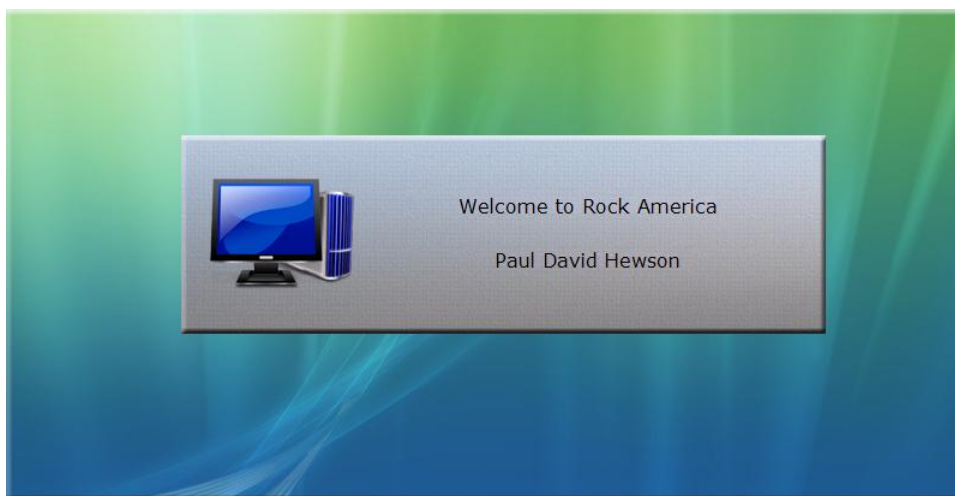
Setting up domain logon Scripts

Some companies use simple .bat files with a few "net use" commands, others use VBScript files mainly using MapNetworkDrive and other functions from the WScript.Network object. In this section, some much easier and much more scalable ways to create logon Scripts are presented using FastTrack Logon.

FastTrack Logon is a requirement for setting up logon Scripts with FastTrack Scripting Host. FastTrack Logon sets up FastTrack Scripting Host on client computers and executes a logon Script predefined by the Network Administrator.

By following this simple setup procedure, the Users' logon screens will appear as shown in Figure 5.1.

Figure 5.1 FSH Logon Script sample splash screen



FastTrack Logon will do the following seamlessly on each client computer:

- Synchronize Fsh.exe and logon Script files to Local Settings\Application Data\FastTrack in each User's non-roaming part of the profile.
- Execute the network logon Script from this client location.
- Associate the .fsh extension with the profile copy of Fsh.exe.

The first time each User logs on to a client computer, about 1 megabyte of data is copied from the domain controller to the non-roaming part of the User's profile. After that, copying occurs only when the files in the fshbin folder are changed on the domain controller.

Anything put in the fshbin directory on the domain controller will be synchronized to each client. There is no limitation as to what files and directories can be put in the fshbin folder on the domain controller for synchronization.

Continued...

Logon Scripts, Continued

Procedure – Set up logon Scripts for a domain

It is important to make sure that FastTrack Logon executes correctly on the network. The best way is to initially try it without group policies and only assign it to a few test Users. This is a safe procedure, as it does not interfere with existing logon Scripts.

Step

1. Locate the file **FTLogon.zip** in the installation folder, which by default is under **FastTrack Software\FastTrack Scripting Host** in the **Program Files** folder.
2. Unpack the full, unedited content of **FTLogon.zip** into the root of the **NetLogon** share on the domain controller (`\\%UserDomain%\NetLogon`).
3. Copy the license file **fsh.lic** file to the `\\%UserDomain%\NetLogon` folder.
4. Open the properties for any test User in the domain or active directory. Enter **FTLogon.exe** in the **Login Script** field under the **Profile** tab.
5. Make sure the logon Script for the test Users is not overruled by group policies.

That's it! Logon to any computer in the domain as the recently modified test User. Provided no changes have been made to `PreLogon.fsh`, a splash screen is displayed for five seconds saying "Welcome to the network" <UserName>, similar to Figure 5.1 above.

Note:

If there are Windows 7 and/or Vista clients, the first part of the logon Script might be hidden by default under the Windows Welcome screen for the first 30 seconds of execution. This behavior can be changed by downloading an Admin Policy file (.adm) from www.fasttrackscript.com.

The following section explains how this works and describes a procedure for downloading and installing the relevant Admin Policy.

Un-hiding logon Scripts on Windows 7/Vista/Server 2008

Computers running Windows 7/Vista/Server 2008 use a "Welcome" screen that hides a logon Script. By default, it becomes visible only if the execution takes more than 30 seconds. This means that the FSH `PreLogon.fsh` Script is not visible on these clients, unless execution takes more than 30 seconds.

Decide whether or not the commands executed by FSH should be visible. If not, then no action is needed. For example, if the logon Script is simply connecting a few shares and setting up some printers, Users typically don't need or want to know this is happening, they just want it to work.

If the FSH commands do need to be visible, a simple procedure is required on Windows 7/Vista/Server 2008 computers to make it so.

By default, Windows 7/Vista/Server 2008 show a black background once the Welcome screen disappears. During User logon, FastTrack Logon uses the wallpaper specified by the User (or Network Administrator), so the black screen can be replaced with something under User/Administrator control.

Two procedures follow. The first allows un-hiding of logon Scripts on a per-machine basis by modifying a registry key, while the second describes how to un-hide for multiple machines using Group Policy.

Continued...

Logon Scripts, Continued

Procedure – Un-hide logon Scripts on one machine

Changing Windows settings to allow the un-hiding of logon Scripts requires setting a registry key, which requires Administrator access on the local machine. When initial testing of a logon Script is done, the relevant registry key can be set manually without Group Policies to make the logon script visible on the testing client.

This procedure shows how to do this under HKEY_LOCAL_MACHINE. The Group Policy method described in the next procedure sets the same registry key for Users under HKEY_CURRENT_USER.

Step

1. On the machine to have its registry changed, start the registry by typing **regedit** in the **Run** field.
 2. Backup the registry before making any changes.
Select registry menu item **File > Export**, find a location, give the backup a name and click **Save**.
 3. Locate registry key
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System.
 4. Add a registry key named **DelayedDesktopSwitchTimeout** of type **REG_DWORD**.
Right-click in white space and select **New > DWORD (32-bit) Value**.
 5. Enter a value into this key of **0**.
Double-click the new entry and type **0** into field **Value data**.
 6. Exit the registry.
-

Procedure – Un-hide logon Scripts using Group Policy

Changing Windows settings to allow this still requires setting a registry key, but this can be done via Group Policy to apply to multiple computers. Such a thing is not possible in the User's context, which is why FastTrack Logon cannot set it during User logon. New in Windows 7 is the requirement that all policy keys in HKEY_CURRENT_USER are no longer writable in the User's context, so, for multiple computers, it can be done with Group Policies.

Step

1. Download the Welcome Screen Policy from
<http://www.fasttrackscript.com/Download/ADM.aspx>.
 2. Place the policy file in the **%windir%/inf** folder on the domain controller. Typically, this is **C:\Windows\Inf**.
 3. Backup the registry before making any changes:
 - (a) Start the registry
 - (b) Select registry menu item **File > Export**
 - (c) Find a location, give the backup a name and click **Save**
 - (d) Exit the registry
 4. Start **Group Policy** on the domain controller and browse to **User Configuration**.
To start Group Policy, type **gpmmc.msc** in the **Run...** field on the domain controller.
-

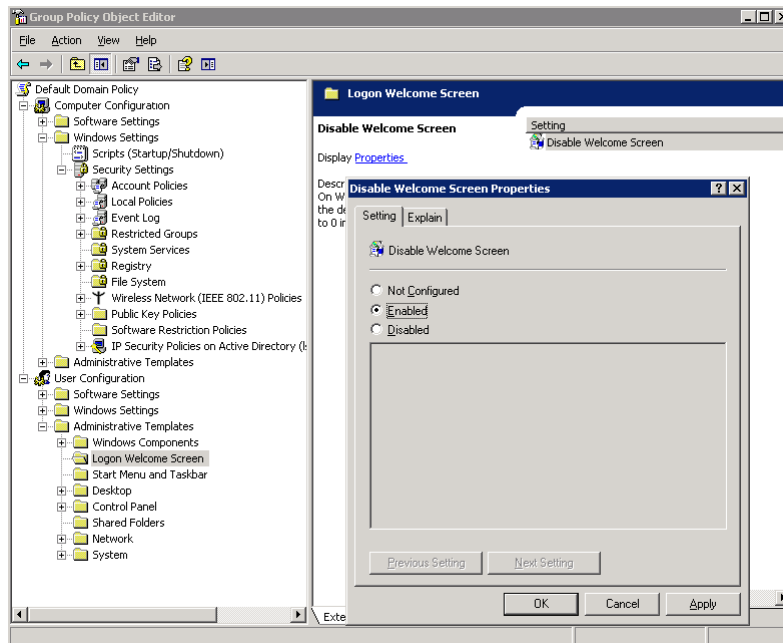
Continued...

Logon Scripts, Continued

Procedure – Un-hide logon Scripts using Group Policy, Continued

Step

5. Under **User Configuration**, right-click **Administrative Templates** and then click **Add/Remove Templates**.
6. Click **Add** and select the **CustomWelcomeScreen.adm** file just downloaded.
7. Select **Disable Welcome Screen** and select **Enable** as shown:



8. Click **OK**. Users logging-on will now see the logon Script.

Working with logon Scripts

Take a look in the fshbin folder in the NetLogon share on the domain controller. In this directory there is a template logon Script called PreLogon.fsh. This is the FSH Script that shows the splash screen (among other things).

There are actually two files that are executed by FastTrack Logon: *Prelogon.fsh* and *Postlogon.fsh*.

PreLogon.fsh is executed *before* the explorer process starts. It is not possible for the User to interfere with what is happening here, so apart from connecting shares, this is a good place to make document backups and maybe perform installations – anything that Administrators do not want Users working with before the Script completes.

PostLogon.fsh is executed *after* the explorer process starts. Anything that is not required to be completed before the User can start working can be done here to prevent the User from having to wait. This could be connecting printers and synchronizing the computer clock.

Continued...

Logon Scripts, Continued

Using Group Policy to run logon Scripts

Once the logon Scripts are created and tested, they can be assigned to all domain Users. Unless there are only a few Users in the domain, the logon Script should be assigned via Group Policy.

Procedure – Use Group Policy to run logon Scripts

Step
1. Start Group Policy on the domain controller. To start Group Policy, type gpmc.msc in the Run... field on the domain controller.
2. Browse to the Organizational Unit (OU) for the Users concerned and select a policy (or create a new one).
3. If using Windows Server 2003, select Open User Configuration > Windows Settings > Scripts (Logon/Logoff) and double-click Logon .
4. If using Windows Server 2008, select Open User Configuration > Policies->Windows Settings->Scripts (Logon/Logoff) and double-click Logon .
5. Click Add and select the file \\%UserDomain%\NetLogon\FTLogon.exe as the Logon Script name, where %UserDomain% is the actual DNS name of the domain.
6. Logon to a client computer using a suitable test User and verify the logon Script works successfully.

Using the company logo

The prelogon.fsh Script shows a splash screen by default. The Splash command can take a custom image file as the third parameter.

If the custom image file is **128 x 128** or less, the icon will be replaced by the image file specified; png format is recommended, as it supports transparency. If the image is *exactly* **537 x 165** pixels, the whole background will be replaced by the specified skin file and the logo icon will be removed.

The easiest way to get a custom image file to the clients, is simply to put it into the fshbin folder on the domain NetLogon share, as it automatically gets distributed to the clients and refers to that location. The following Script line, with an image file called *Logo.png*, produces the splash screen shown in Figure 5.1 on page 46:

```
Splash welcome to Rock America, [UserFull]Name], →  
[UserAppDataDir]\FastTrack\Logo.png
```

Using startup, shutdown and logoff Scripts

Use FastTrack Logon for startup, shutdown and logoff Scripts too, simply by copying the FastTrack Logon package to those directories on the domain controller with different PreLogon.fsh and PostLogon.fsh Scripts.

Startup and shutdown Scripts can be used for machine maintenance. However, in most cases it is a better solution to simply change the execution context in the logon Script, because the User then has some interaction and the Script can show the User what is going on.

Continued...

Logon Scripts, Continued

About offline Scripts

Offline logon Scripts are not supported on the Windows the platform, but FastTrack Software provides an application to support the execution of offline Scripts. Refer to the SmartDock section later in this chapter for more details.

About terminal services and Citrix

If using terminal services, the same logon Script is executed regardless of logon Script type. If a different logon Script is desired on terminal services, put a construct like that shown below (Figure 5.2) at the beginning of the prelogon.fsh Script to run a specific Terminal Server Script.

Figure 5.2 Using FSH Scripts with terminal services (pre-logon)

```
If TerminalServer Then
  Include TSLogonScript.fsh
  Exit
End If

'...the rest of the regular logon script
```

In the example above, a Script named TSLogonScript.fsh is created in the root of the fshbin folder that is specific to terminal services.

In the postlogon.fsh Script, execution can be stopped by using a similar construct, as shown in Figure 5.3:

Figure 5.3 Using FSH Scripts with terminal services (post-logon)

```
If TerminalServer Then Exit
```

When using Citrix, a separate logon Script can be specified, which makes it possible to take a copy of FastTrack Logon and create separate Scripts.

Installation Scripts

Installation leads to complexity

Installing a software application on a User's computer with all program defaults is seldom the end of the matter. Users almost always want varying degrees of customization to the vanilla, out-of-the-box installation. Typical settings that might be changed include:

- Default location
- Default behavior
- Default template applied and additional templates available
- Start-up window/screen

Further, there is no 1:1 relationship between computers and Users with regard to settings and customization. FSH has built-in mechanisms to handle this type of challenge easily.

Note:

This documentation assumes that FastTrack Logon is used to enable FSH Scripts as the logon Script executed by domain Users.

Simple install Script

The basic mechanisms are illustrated by working through the following scenario:

1. Create an installation Script for a simple internal Windows-based phonebook
2. Deploy it from the logon Script to everyone, using a FastTrack Logon Script
3. Apply some default User settings

Procedure – Create an installation Script

The phonebook application is, in fact, simply a group of files residing in a folder – in this case, the \Acme\Phonebook folder in the Program Files folder.

The actual Script is shown in Figure 5.4 below. It has two parts – an *uninstall* part and an *install* part. These are identified by a simple If ... Then ... Else statement.

Step

1. Start the **FSH Editor**.
 2. Enter the commands shown in **Figure 5.4**.
 3. Save the Script with a suitable name (e.g. **Install.fsh**). The Script can be saved into the standard location for installing domain applications (in this case **\\AcmeServer\Installers\$\Phonebook**).
 4. Test the Script within the FSH Editor using the **F5** key.
-

Continued...

Installation Scripts, Continued

Simple install Script, Continued

Figure 5.4 Simple installation Script

```
If [CmdParam Action]=Uninstall Then
  DeleteDir [ProgramFilesDir]\Acme\Phonebook
  DeleteFile [ProgramsDir]\Acme Phonebook.lnk
  UnregisterInstallation Acme Phonebook
Else
  SyncDir Bin, [ProgramFilesDir]\Acme\Phonebook
  CreateShortcut [ProgramsDir],Acme Phonebook, →
    [ProgramFilesDir]\Acme\Phonebook\Phonebook.exe
  RegisterInstallation Acme Phonebook,1,1
End If
```

RegisterInstallation and UnregisterInstallations are key elements, as they store the information that the application is installed or uninstalled for later query in Scripts.

Note:

Testing an installation Script may require installing and uninstalling numerous times. Remove the application in exactly the same way it was installed each time, including any registry keys created. Do not, for example, uninstall an application using Control Panel, having just installed it using an FSH Script – use an FSH Script to uninstall it as well. The reason for this is because the FSH install Script creates registry keys that must be removed during an uninstall.

Install and Uninstall – one Script or two?

It is a matter of personal preference as to whether the same Script is used for installation and uninstallation, or if two separate Scripts are used. In this example one Script is used for both situations.

If using one Script, use a command-line parameter to control which one is actually executed, (as shown in Figure 5.4). Use the /p switch to pass a command-line parameter to the Script, which is extracted with the CmdParam function. In this case, specifying "/p action=uninstall" will run an uninstallation – anything else will run the installer.

Procedure – Deploy the installation Script

For this example, assume that *all* Users need access to the phonebook application.

Step
1. From within the FSH Editor , open the FastTrack Logon Script PreLogon.fsh .
2. Add the line given in Figure 5.5 to PreLogon.fsh.
3. Save and test the updated PreLogon.fsh Script by logging-on as a domain User from another computer.

Note that this should probably be done after-hours. If necessary, disable Users from logging-on during testing.

Continued...

Installation Scripts, Continued

Simple install Script, Continued

Figure 5.5 Additional line added to PreLogon.fsh

```
If Not InstalledBuild Acme Phonebook, 1 Then RunScript →  
  \\AcmeServer\Installers$\Phonebook\Install.fsh, →  
  AcmeDom\AcmeInstall, "<encryptedpassword>"
```

The example uses binaries located at a share called \\AcmeServer\Installers\$ and uses an administrative account named AcmeInstall in the AcmeDom domain.

The <EncryptedPassword> is replaced with a string of encrypted data, which must be generated from the FSH Editor via the Tools menu.

Note:

Any command or function that can include a User name will require an encrypted password to ensure that no one can get the actual password just by looking at a Script.

Note that if UAC is enabled on Windows Vista and Windows 7 clients, the User will get a security warning when the Script changes execution context. In this case, it could be a better option to deploy the FastTrack Script with a management system, if available, and still let the logon Script set the User settings. Alternatively, a message can be displayed via *ShowMessage* before the actual execution that tells the User to approve the UAC, if they want the program installed now. If they reject it, they will be asked at every logon until they approve the installation.

Procedure – Apply default User settings

Assume the Windows Phonebook application lets the User add comments to people in the phonebook. Assume further that these comments are stored in the User's personal share.

In the example below, the logon Script maps drive O: to the personal share.

Step
1. From within the FSH Editor , open the FastTrack Logon Script PreLogon.fsh .
2. Add the lines given in Figure 5.6 to PreLogon.fsh.
3. Save and test the updated PreLogon.fsh Script by logging-on as a domain User from another computer.

Figure 5.6 Further lines added to PreLogon.fsh

```
If UserSettingsOnceABuild Acme Phonebook Then  
  MakeDir O:\PhonebookData  
  WriteRegistry HKCU\Software\Acme\Phonebook\FilePath, O:\PhonebookData  
End If
```

Continued...

Installation Scripts, Continued

Simple install Script, Continued

Every time the installed Script build is changed on the machine, the settings are applied once for every User that logs on to that machine, executing the Script.

Note:

If there are legacy Scripts that are not FastTrack Scripts, the *UserSettingsProgramOnce* setting can be used for a program that is listed in the programs list in the control panel to get the same "Run Once Per User" functionality. This is illustrated in Figure 5.7.

Figure 5.7 Handling legacy Scripts

```
If UserSettingsProgramOnce Adobe Reader Then Include →  
Settings\ReaderSettings.fsh
```

This would find "Adobe Reader 9" in the case of Adobe Reader 9. If Adobe Reader is later upgraded, the Once mechanism will run again for every User, when the text changes in the programs list.

Common install Scripts

The previous example deployed some flat files, which is rarely the case. In almost every case, an installation package is provided and this package needs to be wrapped in a Script.

The only criteria for the install Script is that it can run without User intervention and preferably also without any GUI, as a logon Script will usually display some installation GUI with a splash screen.

Almost all installers can actually run quietly or silently. Most installers today use MSI (Microsoft Installer), which allows the installation to run itself quietly. The next example used MSI to illustrate how this works in FSH.

Using MSI to package application installs

If an installer cannot be run without User intervention, search the internet for that installer or application. In most cases there is in fact a hidden switch to run it unattended.

However, if this is not the case, the installation needs to be repackaged into (for instance) an MSI package that *can* be run quietly. There are hundreds of these products available, but Microsoft's "SMS Installer" is free and it is based on the "Wise Installer" engine.

Note:

Repackaging has to be the last resort. Repackaging is not an effective way of solving the problem, because it requires a common starting point as described below.

Let's say an application installer sets 100 registry keys and on the snapshot machine, it captures 98, meaning two are not in the new package. This is probably because those two are almost always set, but if they are missing on a few of the machines to which the package is deployed, the installation is not complete.

The same applies to system components, which is even more serious if one is missing. Fortunately this problem seldom occurs, because unattended installation is built into all common installers.

Continued...

Installation Scripts, Continued

Creating an install Script for Adobe Reader 9.0

Sometimes the actual installation files are wrapped inside a single .exe file, in which case it will be necessary to extract the real installation files.

Adobe Reader has been selected in this case, because it is representative of many situations whereby the downloaded file is just one .exe file. To get the actual installation files in this case, follow this procedure:

Procedure – Extract Adobe Reader installation files

Let's assume the client computers already have Microsoft Installer installed (if they don't, Script it and install it on all machines), so the MSIEXEC program (Microsoft Installer) can be called directly.

Step
1. Run the .exe file. Note that it starts by unpacking the actual installation files.
2. When the installation prompts for start, locate the real installation files. In this case they are located under Local Settings\Application Data\Adobe\Reader 9.0\Setup Files in the logged-on User's profile.
3. Copy these files and close the installer. This removes the original files.
4. In the copy of the files, notice that the Reader9 directory is the actual installer and that it is in fact MSI.
5. Wrap the Adobe Reader installer files in a Script, as shown in Figure 5.8 . In this example, the files were made available in a \Bin folder.
6. Save the Script as Install.fsh .

So scripting an installation of Adobe Reader is no different than all other MSI packages such as Microsoft Office.

Figure 5.8 Wrapping Adobe Reader install files in a Script

```
If [CmdParam Action]=Uninstall Then
  Run MSIExec.exe,/X Bin\AcroRead.MSI /QN /NoRestart
  If [LastExitCode]=0 Then UnregisterInstallation Acrobat Reader
  If [LastExitCode]=3010 Then UnregisterInstallation Acrobat Reader
Else
  Run MSIExec.exe,/I Bin\AcroRead.MSI /QN /NoRestart
  If [LastExitCode]=0 Then RegisterInstallation Acrobat Reader,9.0,1
  If [LastExitCode]=3010 Then RegisterInstallation Acrobat Reader,9.0,1
End If
```

Use MSIEXEC return codes to determine success. MSIEXEC returns the following:

- 0 for success
- 3010 for success, but reboot required.

Note:

This is also the case for any other MSI package, like Microsoft Office, making this example a good MSI template.

As in the first example, it is possible to deploy the install Script to everyone from the logon Script and apply overruled default settings.

Continued...

Installation Scripts, Continued

Creating an install Script for Adobe Reader 9.0, Continued

In the case of Adobe Reader, the language of the GUI, for instance, is a per-u\User registry key, which can also be set from the logon Script.

In most cases, the Script will contain more tasks than just running the installer. For example, it might delete the desktop shortcuts, but the Script shown in Figure 5.8 can be used as a generic template for MSI packages.

Installations without a management system

Common applications can be installed with just one line in the logon Script as shown in the previous example.

For applications that are to be installed on specific computers, this can be achieved by using an XML file in the same folder as the logon Script to control the actual installations, as shown in Figure 5.9.

Figure 5.9 XML code to control targeted installations

```
<?xml version="1.0"?>
<Acme>
  <Applications>
    <AcrobatReader Name="Acrobat Reader"
InstallPath="\\AcmeServer\Install$\AcrobatReader\Install.fsh">
      <ACMEPC0054 Install="Yes"/>
      <ACMEPC0134 Install="Yes"/>
      <ACMEPC0334 Install="Yes"/>
    </AcrobatReader>
    <Office Name="Microsoft Office"
InstallPath="\\AcmeServer\Install$\Office\Install.fsh">
      <ACMEPC0111 Install="Yes"/>
      <ACMEPC0855 Install="Yes"/>
    </Office>
  </Applications>
</Acme>
```

Then, in the logon Script, it is possible to write just a few lines to handle it, as in Figure 5.10:

Figure 5.10 XML code called from logon Script

```
Loop Application, [XMLSubNodes Applications.xml,Acme/Applications]
  If [XMLAttribute Applications.xml, Acme/Applications/[var →
    Application]/[ComputerName], Install]=Yes Then
    SetVar Name, [XMLAttribute Applications.xml, Acme/Applications/[var →
      Application], Name]
    If Not Installed [Var Name] Then
      SetVar Path, [XMLAttribute Applications.xml,Acme/Applications/[var →
        Application], InstallPath]
      SmallSplash "Installing [Var Name] ... please wait!"
      RunScript [Var Path], AcmeDom\AcmeInstall, "<encryptedpassword>"
    End If
  End If
End Loop
```

Continued...

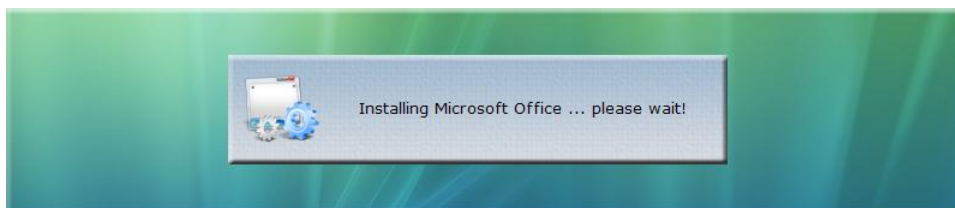
Installation Scripts, Continued

Installations without a management system, Continued

The Script looks through the XML file for nodes under the "Application" node, and then looks to see if the current computer needs the application and has it installed or not.

In this case, if ACMEPC0111 does not have Microsoft Office installed when a User logs on, they will see the following splash screen (Figure 5.11) while the installation runs unattended in the background in an administrative User context.

Figure 5.11 Splash screen shown during targeted installation



Important:

It is very easy to handle uninstallations in the same way, by simply removing computers from the XML file, but great care is needed. If the XML file becomes corrupted, or otherwise malformed, it may effectively uninstall everything on all computers!

Once this way of distributing the software is established, it becomes very easy to create an intranet application whereby Users can request certain software. Once an application for software is approved, all the intranet application needs to do is to insert one line into the XML file.

Installations with a management system

If the organization has a management system, simply package the installation Script and binaries into a distribution package and deploy it with the management system instead of using the logon Script. The logon Script must still take care of the per-User settings part of the process, as the management system will not be able to handle this.

If some applications are installed with the management system and some via the logon Script, it might be convenient to report to the management system, that an application has been installed. This can be done with the post-installation Script (*PostInstall.fsh*).

If there is a file named "PostInstall.fsh" or "PostUninstall.fsh" in the executing directory, they are executed any time the Script runs into *RegisterInstallation* or *RegisterUninstallation*. This is to enable Network Administrators to inform the management system about installations.

These Scripts can also be used for anything that needs to happen when an application is installed or uninstalled.

Continued...

Installation Scripts, Continued

Installations with a management system, Continued

Inside the Script, write whatever is needed to inform the management system about the installation. This would typically be creating an XML file in a specific format on a specific location. The name of the application and the version can be extracted with the functions *CurrentInstallName* and *CurrentInstallVersion* in the post-installation Script.

That said, it may be simpler to ignore the option of using a management system for installations, depending on the size of the network. Management systems are expensive and tend not to offer much functionality, but obviously it depends on the requirements of the environment.

In a small-to-medium sized company, consider if a management system is actually needed, if FSH is handling all installations. It is possible to write a few Script lines in the logon Script to collect basic inventory data in a common XML file, which can then be imported into a spreadsheet such as Microsoft Excel for reporting.

SmartDock Connectivity Tool

Offline Scripts made possible

SmartDock is a small .exe file that detects whenever the computer gets an IP address of any network adapter. Detection of such an event automatically executes a FastTrack Script. This means that when the computer is booted, resumes from standby/hibernation, connects to VPN, a wireless network or any other scenario where the computer gets an IP address, it triggers a Script execution.

The SmartDock tool is a good supplement to domain logon Script for portable computers. Often Users of portable computers put the computer into stand-by or hibernation at work and then resume the session somewhere else, or vice versa. With this tool, Users and Administrators have control over what happens every time the computer is resumed or connects to a network.

A solution for use with proxy servers

If using a proxy server, this tool is must. Instead of relying on the quirky IE auto detection mechanism, it is possible to control when the proxy should be set or removed and to ensure there are not dead network share connections.

Often a SmartDock Script will be a subset of the actual logon Script, as much of the same functionality is required when a computer resumes at the company network.

Setting up SmartDock

SmartDock is not a Windows Service but a small .exe file. The reason for this is that if it was a Windows Service, it would not run in the User's own context, making proxy server settings, default printer settings and other things that are related to the User impossible to set.

SmartDock is included in the FastTrack Logon package and is replicated to a local directory on the client. The easiest way to use it is to include a line in the logon Script that will ensure it always runs when the User logs on, as shown in Figure 5.12.

Figure 5.12 Including the SmartDock trigger in a logon Script

```
If Portable Then SetUserStartupItem SmartDock, →  
[UserAppDataDir]\FastTrack\SmartDock.exe
```

This line makes SmartDock execute any time the User logs on, regardless of whether or not the computer is on LAN. Since the Script file(s) and the SmartDock executable are replicated to a local directory with FastTrack Logon, it will also execute without LAN access.

Sample Script

Smartdock executes a Script named *SmartDock.fsh* in the same directory as the *SmartDock.exe* program. In the FastTrack Logon folder *fshbin* there is already a template Script. All lines are commented out in the sample Script; they are reproduced un-commented in Figure 5.13 below.

Continued...

SmartDock Connectivity Tool, Continued

Sample Script, Continued

Figure 5.13 Sample SmartDock.fsh Script

```
SmallSplash "Setting up network, please wait..."
If Alive AcmeProxy Then
    SetProxyServer AcmeServer,8080
    ConnectShare J:, \\AcmeServer\CommonShare
    ConnectShare [UserHomeDrive], [UserHomeDir]
Else
    DisableProxyServer
    DisconnectAllShares
End If
```

In this example, if the server AcmeProxy is reachable, the proxy server is set and two shares are connected.

If the server is not reachable, all shares are disconnected and the proxy server is disabled to make sure that the User can actually browse the internet outside the company environment.

It is also possible to detect whether the User is on a LAN or not by checking the IP scope or using the *Alive* condition on a server that exists in the company network. The Script can be expanded with a variety of other options specific to an organization's needs.

Where to from here?

What can be done with FastTrack Scripting Host?

FastTrack Scripting Host can be used for anything that can be done with other scripting languages, but it can also do much more. For example, it can be used for:

- Logon Scripts – With very few Script lines it is possible to handle location-based connection of shares and printers, apply User settings, do cleanups, installations, etc. One can also get a centralized log of all execution errors by using the Error Handler Script, which gives a detailed insight into what goes on "out there".
- Offline logon Scripts – With the free SmartDock application, a Script can be executed when Users are offline and handle proxy servers, shares, etc.
- Installations – It is possible to Script all installations with simple FastTrack Scripts or just wrap existing Scripts. The logon Script can then be used to solve the classic User settings problem.
- Administrative tasks – Tasks such as replicating data between servers or backing up files with synchronization can be achieved.
- Active Directory tasks – any operation on an object of the type user, computer, group or organizational unit (OU) can be scripted with a single script line.
- Modifying ACLs – permissions on files, directories, shares and registry values can be scripted with a single script line.
- App launchers – Small Scripts can be written to handle launching of User applications based on Script logic.
- Terminal services – Specific logon Scripts can be made that log Users off, if they attempt to use a certain Citrix application on client locations where Administrators don't want them to run it from. Information on client IP addresses and User information is also available.

For up-to-date information

Please refer also to www.fasttrackscript.com for up-to-date information, release notes and frequently asked questions.

Index

A

Adobe Reader 9 71, 73
 Alive 80

B

Building Blocks 44

C

Citrix 47, 66, 81
 CmdParam 51, 68, 73
 Collections 46
 command line parameters 54
 Command line parameters 50
 commands 2, 14, 20, 24, 26, 31, 42, 44, 54, 58, 59, 67
 operating system 2, 6
 Commands 44
 Comments 48
compiled 2
 Conditions 46
 If-statements 46
 context-sensitive help 22
CurrentInstallName 77
CurrentInstallVersion 77

D

Debugging 40, 50, 54
 demo Script 22, 33, 40, 44
 domain logon 58
 download package 12, 21, 22, 32, 33

E

Editor.exe 12, 30
 Elements 24, 42
 Context Helper 24, 26, 42
 Engine Browser 24, 42
 Properties Panel 24, 26, 42
 Property Builder 24, 42
 Script Window 24, 26, 40, 42
 EncryptedPassword 69
 Encrypting Scripts 54
 encryption keys *See* Encrypting Scripts
 Errors
 logon errors 20
 Runtime errors 20, 54
 Script errors 53
 Escape 49
 executable file 20, 31

F

FastTrack Script 4, 7, 69, 78
 Features 26, 28, 42
 Auto correction 26, 42
 Code completion 26, 42
 Context-sensitive help 26, 42
 Customization 26, 42
 Script validation 26, 42
 Syntax highlighting 26, 42, 47
 File and folder structure 29
 FSH Editor 4, 19, 22, 24, 25, 26, 28, 39, 40, 42, 43, 50, 69
 editor 19, 22, 24, 26, 28, 40, 42, 50
 FSH Engine 4, 19, 20, 21, 50, 54
 engine 20, 42, 50, 53, 54
 Fsh.exe 17, 30, 58
 Fsh.lic 30
fshbin 58, 63, 66, 79
 Fshbin 14, 30
 FTLogon.exe 14, 29, 30, 50, 59, 64
 ftlogon.ini 14, 30
 Functions 44

G

Go-to labels 48
 Group Policy 61, 64
 guided completion 22

H

HKEY_CURRENT_USER *See* registry key

I

IDE 22
 Installation
 Download and Install 12, 14
 Test the Installation 13, 15
 installation process 13
 Installation Scripts 67
interpreted 2, 22, 31, 40

K

keyword recognition 22

L

Language comparison 8, 28
 license key 56
 Licensing 12, 16
 logon Script 6, 31, 33, 58, 59, 63, 64, 65, 66, 67, 69, 71, 74, 75, 76, 77, 78, 81

Logon Scripts	58	PreLogon.fsh ..	14, 30, 35, 36, 37, 59, 63, 64, 68, 69
Loops	47	SmartDock.fsh.....	See SmartDock
M		Script encryption	See Encrypting Scripts
management system	69, 75, 76, 77	scripting	
Microsoft Installer	See MSI	languages	2, 48, 81
MSI.....	71, 73, 75	Scripting.....	2, 8
MSIEXEC.....	See MSI	Skills	3
N		SmartDock4, 14, 19, 29, 30, 31, 32, 33, 37, 66, 78, 80,	
NetLogon	14, 59, 63, 64	81	
O		SmartDock.exe.....	31, 78
offline Scripts	See SmartDock	SmartDock.fsh	30
Operations	44	startup, shutdown and logoff	64
P		synchronize.....	4
Parameters	47	Syntax errors.....	20, 54
PowerShell	2, 4, 8, 19, 28	syntax highlighting.....	22, 26, 42, 44
private key	56	T	
proxy server	31, 32, 78, 80	terminal services.....	66
R		U	
RegisterInstallation	68, 73, 76	Un-hiding logon Scripts.....	15, 59
registry key	61	UnregisterInstallations	68
Repackaging.....	71	UserSettingsProgramOnce.....	71
return codes	73	V	
Run Once Per User.....	See UserSettingsProgramOnce	Variables	47
S		VBScript	2, 4, 6, 7, 8, 19, 28, 53, 58
Sample Scripts	5, 19, 32, 33	Visual Studio	22, 26, 40, 42
DemoScript.fsh	12, 13, 30	W	
ErrorHandler.fsh	14, 20, 30, 36, 53, 54	Windows Service	31, 78
PostInstall.fsh	14, 30, 36, 76	WinLogon.....	7
PostLogon.fsh	14, 30, 35, 63, 64	X	
PostUninstall.fsh	14, 30, 36, 76	XML.....	75, 76, 77