

# TrueUpdate™

**The Dynamic Software Update Solution**

## **User's Guide**

Copyright © 2001—All Rights Reserved

**IndigoRose**  
SOFTWARE DESIGN CORP.

<http://www.indigorose.com>  
[info@indigorose.com](mailto:info@indigorose.com)

### **Proprietary Notice**

The software described in this document is a proprietary product of Indigo Rose Software Design Corporation and is furnished to the user under a license for use as specified in the license agreement. The software may be used or copied only in accordance with the terms of the agreement.

Information in this document is subject to change without notice and does not represent a commitment on the part of Indigo Rose Software Design Corporation. No part of this document may be reproduced, transmitted, transcribed, stored in any retrieval system, or translated into any language without the express written permission of Indigo Rose Software Design Corporation.

### **Trademarks**

TrueUpdate, Setup Factory, Visual Patch and the Indigo Rose logo are trademarks of Indigo Rose Software Design Corporation. All other trademarks and registered trademarks mentioned in this document are the property of their respective owners.

### **Copyright**

Copyright © 2001 Indigo Rose Software Design Corporation.  
All Rights Reserved.

---

# Table of Contents

<b>Introduction .....</b>	<b>13</b>
What is TrueUpdate? .....	13
How Does TrueUpdate Work?.....	13
The TrueUpdate Design Environment .....	14
<i>The Client Configuration Utility</i> .....	14
<i>The Server File Editor</i> .....	14
TrueUpdate Technology .....	15
<i>Integrates Easily</i> .....	15
<i>Reduces Support Costs</i> .....	15
<i>Is Lightweight and Stand-alone</i> .....	16
<i>Uses Standard Internet Protocols</i> .....	16
<i>Puts You In Control</i> .....	16
Document Conventions .....	17
Other Resources .....	19
<b>Key Concepts .....</b>	<b>21</b>
Design Time .....	21
Run Time .....	21
Variables.....	22
<i>Built-in Variables</i> .....	22
<i>Custom Variables</i> .....	22
CRC Values .....	24
The TrueUpdate Client .....	25
Client Side .....	25
Server Side .....	25
The Server File .....	25
Server File Locations .....	26
Version Identifiers .....	26
Actions .....	27
The Update Process.....	28

<b>Getting Started .....</b>	<b>29</b>
A Few Tips Before you Begin.....	29
<i>Consider a Dry Run or Two</i> .....	29
<i>Map Out Your Course</i> .....	29
<i>Don't be Afraid to Just Follow the Radiator Cap</i> .....	29
Building the TrueUpdate Client.....	30
Building (or Updating) the Server File.....	33
<i>Setting Up the First Version</i> .....	33
<i>Setting Up the Second Version</i> .....	33
Publishing the Server File.....	34
Integrating the Client into your Software .....	35
Different File Patching Methods.....	36
<b>The Client Configuration Utility .....</b>	<b>39</b>
Projects .....	40
<i>Starting a New Project</i> .....	40
<i>Opening an Existing Project</i> .....	40
<i>Saving the Current Project</i> .....	41
<i>Reopening a Recent Project</i> .....	41
Product Information .....	41
<i>Setting Product Information</i> .....	42
Server File Locations .....	42
<i>Adding Local Server File Locations</i> .....	44
<i>Adding HTTP Server File Locations</i> .....	45
<i>Adding FTP Server File Locations</i> .....	46
<i>Removing Server File Locations</i> .....	47
<i>Changing the Search Order for Server File Locations</i> .....	48
<i>Setting Conditions for Server File Locations</i> .....	48
Interface Settings .....	49
<i>Selecting the Default Interface Style</i> .....	51
<i>Showing and Hiding Screens</i> .....	51

<i>Changing the Image Used on a Wizard Screen</i> .....	52
<i>Previewing Screens</i> .....	54
<i>Using a Custom Icon in the Title Bar</i> .....	54
<i>Disabling Support for User Configuration</i> .....	55
<i>Changing User Configuration Settings</i> .....	56
Messages .....	61
<i>Editing Messages</i> .....	62
<i>Adding Languages</i> .....	63
<i>Removing Languages</i> .....	63
<i>Setting The Default Language</i> .....	64
<i>Exporting Messages to a Language File</i> .....	64
<i>Translating Messages</i> .....	65
Variables .....	66
<i>Assigning a Value to a Variable</i> .....	68
<i>Reading a Value from the Registry</i> .....	70
<i>Reading a Value from an INI File</i> .....	71
<i>Removing Variables</i> .....	72
<i>Changing the Order that Variables are Assigned In</i> .....	72
Building the Client .....	73
<i>Changing the Output Folder</i> .....	74
<i>Changing the Client Executable Filename</i> .....	74
<i>Enabling Debug Mode in the Client Executable</i> .....	74
<i>Requiring a Debug Mode Password</i> .....	76
<i>Building the Client</i> .....	76
General Preferences .....	77
<i>Setting the Temporary Build Folder</i> .....	77
<i>Setting the Default Output Folder</i> .....	77
<i>Choosing Startup Options</i> .....	78
Language Preferences .....	79
<i>Setting Default Language Files</i> .....	79
Update Preferences .....	81

## **User's Guide**

---

<i>Automatically Checking for New Versions of TrueUpdate</i> .....	81
<i>Hiding the Update Interface Until a New Version is Available</i> .....	82
<i>Setting How Often TrueUpdate Checks for Updates</i> .....	82
<i>Configuring the TrueUpdate Connection Settings</i> .....	82
User Tools.....	82
<i>Configuring User Tools</i> .....	82
<b>The Server File Editor .....</b>	<b>85</b>
Server Files .....	86
<i>Creating a New Server File</i> .....	86
<i>Opening an Existing Server File</i> .....	86
<i>Saving the Current Server File</i> .....	86
<i>Reopening a Recent Server File</i> .....	87
<i>Publishing the Current Server File</i> .....	87
<i>Password Protecting the Current Server File</i> .....	89
<i>Viewing and Editing Server File Properties</i> .....	91
<i>Validating the Current Server File</i> .....	91
Publish Sites.....	92
<i>Adding a Local Publish Site</i> .....	93
<i>Adding an FTP Publish Site</i> .....	94
<i>Removing a Publish Site</i> .....	96
<i>Editing a Publish Site</i> .....	96
Versions Bar .....	97
<i>Adding a Version</i> .....	98
<i>Removing a Version</i> .....	99
<i>Copying an Existing Version</i> .....	99
<i>Changing the Version Search Order</i> .....	100
<i>Labeling a Version</i> .....	101
<i>Enabling the More Information Button</i> .....	102
<i>Providing an Update Size</i> .....	104
<i>Making Notes</i> .....	105

Version Identifiers .....	105
<i>Reading a Value from the Registry</i> .....	106
<i>Reading a Value from an INI File</i> .....	108
<i>Comparing File Versions</i> .....	109
<i>Comparing CRC Values</i> .....	111
<i>Using OR Operators</i> .....	112
<i>Adding Comments</i> .....	114
<i>Rearranging the Version Identifiers</i> .....	115
<i>Cutting, Copying and Pasting Version Identifiers</i> .....	115
<i>Removing Version Identifiers</i> .....	116
<i>Adding Indentation to Version Identifiers</i> .....	116
<i>Removing Indentation from Version Identifiers</i> .....	117
Actions .....	117
<i>Adding Actions</i> .....	118
<i>Editing Actions</i> .....	120
<i>Changing the Order Actions are Performed In</i> .....	120
<i>Adding Indentation to Actions</i> .....	120
<i>Removing Indentation from Actions</i> .....	121
<i>Cutting, Copying and Pasting Actions</i> .....	121
<i>Removing Actions</i> .....	122
<i>Exporting Actions</i> .....	122
<i>Importing Actions</i> .....	122
Reports .....	123
<i>Generating a Report</i> .....	123
General Preferences .....	124
<i>Setting the Temporary Build Folder</i> .....	124
<i>Changing the Location of the Publish Sites Data File</i> .....	125
<i>Choosing Startup Options</i> .....	125
Log File Preferences .....	126
<i>Logging Server File Editor Activity</i> .....	127
<i>Selecting which Events to Log</i> .....	127

## User's Guide

---

<i>Viewing the Log File Contents</i> .....	127
<i>Refreshing the Log File Tab</i> .....	127
<i>Emptying the Log File</i> .....	127
Environment Preferences .....	128
<i>Changing the Version Identifiers/Actions List Colors</i> .....	128
<i>Changing the Indent Size</i> .....	129
<i>Changing the Double-click on Publish Site Options</i> .....	129
Update Preferences .....	130
<i>Automatically Checking for New Versions of TrueUpdate</i> .....	131
<i>Hiding the Update Interface Until a New Version is Available</i> .....	131
<i>Setting How Often TrueUpdate Checks for Updates</i> .....	131
<i>Configuring the TrueUpdate Connection Settings</i> .....	131
User Tools .....	132
<i>Configuring User Tools</i> .....	132
<b>Variables</b> .....	<b>133</b>
What Are Variables? .....	133
<i>Built-in Variables</i> .....	133
<i>Custom Variables</i> .....	134
What Can You Do With Variables? .....	135
<i>Inserting Variables</i> .....	136
<i>A Little Common Sense Never Hurts</i> .....	137
Naming Variables .....	138
Defining Variables with Server File Actions .....	139
Using Variables in Expressions .....	140
<b>Expressions</b> .....	<b>143</b>
What Are Expressions? .....	143
Where Can You Use Them?.....	143
<i>On Condition tabs</i> .....	143
<i>In IF and WHILE actions</i> .....	144

---

## Table of Contents

<i>On Assign Value screens</i> .....	145
Values .....	147
Operators.....	149
<i>Table of Operator Precedence and Associativity</i> .....	150
<i>Parentheses</i> .....	151
<i>Logical (Boolean) Operators</i> .....	151
<i>Relational Operators</i> .....	152
<i>Arithmetic Operators</i> .....	153
<i>String Operators</i> .....	153
<i>Version Operators</i> .....	154
Notes .....	154
Syntax Rules .....	155
<b>Actions .....</b>	<b>159</b>
<b>Handling Errors .....</b>	<b>163</b>
Built-in Error Handling (The On Error Tab).....	163
<i>Setting the User Notification Options</i> .....	164
<i>Setting the Action Taken After an Error Occurs</i> .....	166
Custom Error Handling (Using Actions) .....	167
<i>Checking %LastErrorNum%</i> .....	167
<i>Using Continue at label</i> .....	168
<b>Integrating TrueUpdate Into Your Application .....</b>	<b>171</b>
The Integration Two-Step.....	172
<i>Step 1: Adding the Client Files</i> .....	172
<i>Step 2: Triggering TrueUpdate</i> .....	173
Command Line Options .....	174
ShellExecute.....	175
Source Code.....	175
<i>Calling ShellExecute from Visual Basic</i> .....	176
<i>Calling ShellExecute from C/C++</i> .....	176

*Opening the Connection Settings Screen* ..... 177

**Command Line Options ..... 179**

*/BYPASS* ..... 179

*/C:filename* ..... 180

*/CONFIGURE* ..... 180

*/DEBUG* ..... 181

*/DEBUG:password* ..... 181

*/DF:filename* ..... 181

*/L:#* ..... 182

*/P:#* ..... 183

*/R:#* ..... 183

*/S:#* ..... 184

**Glossary ..... 185**

**Built-in Variables ..... 195**

**Contact Info ..... 203**

Corporate Headquarters ..... 203

Sales ..... 203

Technical Support ..... 204

*Before You Contact Our Support Department* ..... 204

*Limitations of Technical Support* ..... 205

**Minimum System Requirements ..... 207**

TrueUpdate Design Environment ..... 207

TrueUpdate Client Executable ..... 207





---

## Chapter 1

# Introduction

---

Welcome to the TrueUpdate User's Guide. This User's Guide is designed to explain important concepts and help familiarize you with the features available in TrueUpdate.

## What is TrueUpdate?

TrueUpdate is a comprehensive solution for developers who want to integrate automated updating capabilities into their software products. It provides a sophisticated framework that can determine what updates are required and automatically retrieve and apply the necessary patches or installation files via the Internet, intranet or LAN.

Basically, TrueUpdate allows your users to continually update your software, automatically and on demand. In a few mouse clicks they can find out whether a new version is available, download it and install it automatically. Or, you might design your software to launch TrueUpdate silently whenever your program is executed, notifying the user as soon as an update becomes available. This way, your users will always have the latest version of your software installed.

## How Does TrueUpdate Work?

TrueUpdate consists of two separate components: a client-side executable that runs on the user's system, and a server-side data file located somewhere else (typically on a web site or FTP server on the Internet).

The TrueUpdate client contains a list of locations where the server file can be found. At run time, the client executable downloads the server file from one of these locations, and then uses the information in the server file to guide the rest of the update process.

The server file contains *version identifiers* and *actions*. The version identifiers enable the client to distinguish one version of your software from another. They consist of criteria such as the value of a specific registry key or INI file entry, the version from a file's resource information, or the CRC value of a specific file. For instance, if your installer

## **Chapter 1**

---

writes your software's version number to the registry, you could use a version identifier to compare the value at that registry key with a specific version number. If the values matched, that version would be identified.

Once a version has been identified, the TrueUpdate client performs a series of actions associated with that version. For instance, if an update is available, the actions might instruct the client to download and then execute a patch file.

From the user's perspective, the whole process is seamless and smooth. TrueUpdate connects to the Internet and performs the update for them—as silently or interactively as the developer desires.

## **The TrueUpdate Design Environment**

The TrueUpdate design environment consists of two separate programs: the Client Configuration Utility, and the Server File Editor.

### ***The Client Configuration Utility***

The Client Configuration Utility is used to customize the client-side "update.exe" program that will coordinate the update from the user's machine. This is where you can configure the look and feel of the update process—for instance, you can edit messages, select which screens are hidden or shown, and choose between a standard "wizard" style interface or a series of streamlined dialog windows. The Client Configuration Utility is also where you specify product information such as the name of your software, your company name, and your copyright notice.

The most important setting in the Client Configuration Utility is the list of Server File locations. This tells the client executable where to download the server file from. Note that this means you need to know where your server files are going to be located before you can generate the TrueUpdate client.

### ***The Server File Editor***

The Server File Editor is used to edit the server-side update information file (or "server file" for short). This is where you define the version identifiers that will be used to identify which version of your software is installed on the user's system. It's also where you

specify the actions you want performed when a particular version of your software is detected on the user's system.

## **TrueUpdate Technology**

Take advantage of technology developed by a team with over ten years experience building professional install tools for developers. The TrueUpdate brand is a recognizable mark of quality that you can proudly display as an assurance to your customers that your software updates are in the hands of the experts.

### ***Integrates Easily***

TrueUpdate is designed to minimize the time it takes to add automated update capabilities to your software. As a separate executable, the TrueUpdate technology is easy to integrate into your software, without complicating your build process with additional compile-time dependencies—and without limiting your ability to control how your users will initiate the update process.

TrueUpdate's industry-leading ease of use is a very important feature—what good is it to save on support costs if you have to spend more on development as a result? In order to maximize your return on investment, you need an automated update solution that can be implemented with a minimum of effort. TrueUpdate technology doesn't tie up your development team with difficult implementation details. It was designed from the ground up to be flexible, easy to use, and easy to integrate.

---

With TrueUpdate technology, you can spend more time developing and marketing your product, and less time managing its distribution.

---

### ***Reduces Support Costs***

Automating the update process for your users can save you valuable time and expense. It reduces support costs by making it easier for your users to keep their software up to date—giving your tech support department fewer legacy support issues to deal with. It reduces your marketing costs, because you won't have to resort to expensive ad campaigns to keep your users abreast of new versions. It reduces production costs by allowing you to maintain a larger inventory with less concern over obsolescence, relying

## Chapter 1

---

on the TrueUpdate technology to get your users up to the latest release immediately after installing the product.

---

No other product provides the same combination of industry-leading ease of use, cutting edge features, and absolute flexibility as TrueUpdate.

---

### ***Is Lightweight and Stand-alone***

Written completely in optimized C and C++ code, the TrueUpdate client is small, weighing in around 600K in size. It's also completely self-contained—the TrueUpdate client has no external dependencies, so you don't have to distribute any extras to make it work. Unlike some products, it doesn't require the Java runtime, Internet Explorer, Visual Basic runtime, or any other multi-megabyte runtime engines and dependencies.

---

Compact and efficient, TrueUpdate technology is everything you need to get the job done.

---

### ***Uses Standard Internet Protocols***

TrueUpdate uses readily available client-server technologies, rather than the proprietary servers required by some other update products. By making use of affordable and trusted protocols such as HTTP and FTP, organizations of any size can deploy TrueUpdate enabled software without the need for specialized and costly hardware and software platforms.

---

TrueUpdate is built on the trusted, dependable standards you already rely on.

---

### ***Puts You In Control***

With TrueUpdate, there's no need to relinquish control over the reliability of your update process. Unlike services that lock you into using their servers, with TrueUpdate, *you* decide where your update files are hosted. You decide on the level of redundancy. You're in control of your update files, patches and web servers. There's no need to rely on the uncertain future of a "free" update service, or to wait helplessly during downtimes you are powerless to resolve.

---

TrueUpdate technology puts *you* in control of the update process for your software.

---

# Document Conventions

Throughout this User's Guide, we've presented different kinds of information in the following ways:

## Start Menu

Items in the start menu are presented in bold, italicized text and separated by -> symbols, like so:

***Start -> Programs -> TrueUpdate -> Important Notes***

To access this item, you would click on the ***Start*** button, click on ***Programs*** in the Start menu to open the Programs menu, click on ***TrueUpdate*** in the Programs menu to open the TrueUpdate menu, and click on the ***Important Notes*** menu item.

## TrueUpdate Program Menus

Items in the TrueUpdate program menus are presented in bold text and separated by | symbols, like so:

**Settings | Product Info**

To access this item, you would click on the **Settings** menu at the top of the TrueUpdate Client Configuration Utility's program screen to open the Settings menu, and then click on the **Product Info** item in that menu.

## Paths and filenames

Directory paths and filenames are presented in a monospace font, like so:

```
C:\Program Files\TrueUpdate
```

This path refers to the TrueUpdate folder within the Program Files folder on your C: drive.

## Chapter 1

---

### Tips, Notes, Warnings and Cross-References

We've used special formatting to set some information apart from the rest of the text:

#### TIP



Tips provide helpful information, such as alternative ways of accessing a feature or shortcuts for advanced users.

#### NOTE



Notes provide extra information to clarify points discussed in the text.

#### IMPORTANT



Warnings alert you to important information.

#### SEE ALSO



This is how we highlight links to information online, in the Command Reference or on other pages in this User's Guide.

### Screens

The names of TrueUpdate screens are presented in italics:

The *Shell Operations* screen

### Buttons

The names of TrueUpdate buttons are presented in bold text, like so:

The **Publish Now** button

When applicable, button names are followed by a thumbnail of the button image between parentheses:

The **Add Location** button (  )

### Fields

The names of fields and other GUI elements like check boxes and drop-down lists are presented in bold text, like so:

Select the **Use custom icon** check box

## Other Resources

If you ever have any questions about the software that aren't answered by this User's Guide, there are several other resources available to you:

### Command Reference and Online Help

The Command Reference is included in HTML Help format. It is accessible from both **Start -> Programs -> TrueUpdate -> TrueUpdate Help** and from the **Help** menu within the product. This is a comprehensive reference for all the TrueUpdate screens, options and properties.

#### TIP



Most screens within the development environment contain a **Help** button, which will automatically open the appropriate topic in the Command Reference.

### Web Site

The TrueUpdate web site is a great place to learn about the product. It is located at <http://www.indigorose.com/trueupdate/>

The web site is where you will find:

- Knowledge Base articles
- Tutorials

## **Chapter 1**

---

- Answers to frequently asked questions
- Information about the latest version, bug fixes, etc.

### **Forums**

We maintain a number of popular web-based discussion and support forums at <http://www.indigorose.com>. These forums are an ideal place to meet with other users of Indigo Rose products. Members frequently discuss the usage of products, share tips and tricks, exchange ideas and much more.

### **Technical Support**

For information on tech support please see page 204.

---

## Chapter 2

# Key Concepts

---

This chapter explains important concepts and terminology that you must understand in order to become proficient with TrueUpdate.

### TIP



If you're already familiar with our other software products, you might want to jump directly to the explanation of client side on page 25.

## Design Time

Design time refers to the process of designing your updates using the TrueUpdate design environment. The TrueUpdate design environment consists of two separate programs: the Client Configuration Utility, and the Server File Editor. Everything you do with these two TrueUpdate components, from the moment you start either program, is considered work done "at design time."

## Run Time

Run time refers to when the actual client executable is run. This is where the interface you customized during design time with the Client Configuration Utility is presented to the user, and the actual work of downloading the server file and updating the user's software is performed.

### TIP



There are several ways the TrueUpdate client can be started. It could be launched when the user selects an "update" item in the Start menu. Or your software could be designed to run the client executable automatically every time your software is started. How the update process is initiated for your software is entirely up to you. (See page 173 for more examples.)

# Variables

Variables are special named "containers" for values that change. (The word "variable" comes from the changing nature of the values that variables represent.)

There are two kinds of variables in TrueUpdate: built-in variables, and custom variables.

### ***Built-in Variables***

Built-in variables are used to represent values that might differ between systems, like the location of the user's temp directory (%TempDir%) or the width of the user's display screen in pixels (%ScreenWidth%). These variables serve as pre-defined constants that you can use in the paths and conditional expressions within your update.

There are also built-in variables for things like your company name (%CompanyName%) and copyright notice (%Copyright%). These variables are automatically defined to match the information you provide in the various fields on the Product Info tab of the Client Configuration Utility. You only have to change these values in one place (in this case, on the Product Info tab), and wherever those variables are used in your update, the new values will be shown automatically.

Finally, there are built-in variables you can use to get information about the last action that was performed (%LastCommand%), and in the case of an error, to get information about the type of error that occurred (%LastErrorNum%, %LastErrorMsg% and %LastErrorDetails%).

#### **SEE ALSO**



See *Built-in Variables* in the Command Reference or on page 195 of this User's Guide for a complete list of the built-in variables.

### ***Custom Variables***

In addition to the built-in variables automatically provided by TrueUpdate, you can define your own custom variables. Custom variables can be used to get information from the Registry or INI files, such as the path where a given software component is installed on the user's system. For instance, you could use a custom variable to query the Registry for information that was put there by your application's installer.

You can also assign values to custom variables and use them as constants. For instance, you could set a variable like %FaxNumber% to your company's fax number, and then use that variable in messages throughout your update instead of the number itself. That way, you would only have to change the number in one place if ever your fax number changes.

### TIP



You can define custom variables on the Variables tab of the Client Configuration Utility, and you can also define them using many of the actions in the Server File Editor. (See page 134 for information on when to use either program.)

Variable names in TrueUpdate should always begin and end with a percentage sign. At design time these variable names serve as placeholders, marking the places where the values will go once those values become known. Wherever you use a variable like %CompanyName% in TrueUpdate, the user will see the value that you specified for your company name in the Client Configuration Utility, i.e. some text like "Foobar Widgets and Gadgets Corp."

### NOTE



Normally, the user never sees the variable names—just the values they represent.

(The exception is when a variable gets used before a value is assigned to it. In that case, the name of the variable is shown where the value would have appeared.)

Variables can be used in conditions, perhaps serving as "flags" that control whether certain parts of the update are done. You can even make the way your update operates depend on the values of specific variables.

### SEE ALSO



For information on how to define variables in the Client Configuration Utility, see page 66. For information on how to define variables using server file actions, see page 139.

# CRC Values

CRC values are calculated using an algorithm known as the Cyclic Redundancy Check, or "CRC" for short. Basically, this involves generating a 32-bit number (or "CRC value") based on the contents of a file. If the contents of a file change, its CRC value changes as well. This allows the CRC number to be used as a "checksum" in order to identify whether or not the file has changed. It also allows you to distinguish between different versions of a file by comparing its CRC value to the CRC values of the originals.

A file doesn't have to change much for its CRC value to be different. In fact, if even just one bit in a file changes, the CRC value for that file will change as well. If all you did was change one letter in a `readme.txt` file between version 1 and version 2, the CRC value for that `readme.txt` file would be completely different.

CRC values can be calculated for any type of file.

### **More on CRC values:**

The basic idea of the CRC algorithm is to treat all the bits in a file as one big binary number, and then divide that number by a standard value. The remainder from the division is the CRC value.

You can think of this value as being like a fingerprint for each file. Unlike human fingerprints, however, it isn't impossible for two files to have the same CRC-32 value. TrueUpdate uses an industry-standard CRC-32 algorithm which generates CRC values that are 32 bits in length. This means that one in every 4,294,967,296 files could have the same CRC "fingerprint."

Although the chances of any two files having the same CRC value are incredibly small, the CRC value alone isn't enough to guarantee an accurate identification. If you need to be *absolutely* sure, check the CRC in addition to other information about the file, such as the size of the file in bytes and its location on the user's system.

## The TrueUpdate Client

The TrueUpdate client is the program that performs or "coordinates" the update from the user's system. The TrueUpdate client consists of two files: a client executable (named `update.exe` by default), and its accompanying data file (named `update.cli` by default).

At run time the client connects to the Internet (or to an intranet or LAN), downloads a TrueUpdate server file, and carries out the instructions found in that server file. Those instructions determine how the update is performed, but it's the TrueUpdate client that actually performs them.

## Client Side

In TrueUpdate, the term "client side" refers to the user's system, where the TrueUpdate client is run.

## Server Side

In TrueUpdate, the term "server side" refers to the other end of the client-server interaction. For instance, when downloading the server file, "server side" is wherever the server file is being downloaded from. This could be a web site on the Internet, or it could be a hard drive in a computer on your LAN—or even a local hard drive in your system. (Usually server side will refer to your web or FTP server.)

## The Server File

The TrueUpdate server file is a special data file containing lists of *version identifiers* and *actions* for each version of your software. The version identifiers tell the TrueUpdate client how to identify a particular version of your software, and the actions tell the client how to bring a particular version of your software up to date.

The server file is designed to give you easy access to the version identifiers and actions at any time. Because the server file is hosted at "your end" of the client-server relationship,

## Chapter 2

---

you can change the update process for any version of your software without having to distribute anything to your users.

The server file is downloaded by the client early in the update process. The information in the server file is encrypted and compressed, so the server file is both secure and very small—typically less than 1 kilobyte in size.

### TIP



Because the server file is so small, you could check for updates silently every time the user runs your software, and the user wouldn't even notice—until an update is available and they're treated to an automated update.

The server file is created and edited using the TrueUpdate Server File Editor.

## Server File Locations

The server file needs to be hosted where the client executable can access it from the user's system at run time. Each place where the server file is made available for download is known as a *server file location*.

There can be multiple locations for each server file—for instance, you might want to mirror the server file at three web sites and an FTP server, to ensure that your users are always able to access it. Server files can be hosted on web servers, FTP servers, or even on networked hard drives.

The list of server file locations is stored in the TrueUpdate client. You can add and remove locations to and from the list using the Client Configuration Utility.

## Version Identifiers

TrueUpdate uses *version identifiers* to identify which version of your software is installed on the user's system. Each version identifier acts as a criterion that must be met in order for a version to be identified. You can provide as many version identifiers as you need to

uniquely identify a version of your software. Each version of your software will therefore have a list of version identifiers associated with it.

At run time, the client goes through each list of version identifiers until it finds a list that matches what's on the user's system. A version is only considered "identified" when all the criteria listed for that version are met—in other words, when all the version identifiers for that version "match."

If none of the version identifiers match, the user is shown an error message informing them that the product is not properly installed on their system.

### TIP



Always have your installation program write your software's version number to the registry. That way you can identify each version with a single identifier that compares the registry key's value to the appropriate version number.

Version identifiers are stored within the server file, and can be added or removed on the Version Identifiers tab of the Server File Editor.

### SEE ALSO



For more information on version identifiers, see page 105.

## Actions

The TrueUpdate client performs the update by carrying out the *actions* listed in the server file. Actions are just instructions that tell the client executable what to do once a version is identified.

The actions are version-specific; there's a separate list of actions for each version. The actions for a given version are only carried out when that version has been identified on the user's system.

## Chapter 2

---

A wide variety of actions are available. In addition to Internet functions like HTTP downloads and FTP transfers, you can perform file operations, open and close programs, send email, submit data to web forms and CGI scripts, work with variables, read and write to the Registry and INI files, manipulate strings, and show custom dialog boxes to the user. Special control structure actions provide advanced features like IF blocks, WHILE loops, labels, and GOTO jumps. Optional design actions allow you to insert maintenance-friendly comments and whitespace, too.

Actions are stored in the server file, and can be added or removed on the Actions tab of the Server File Editor.

### SEE ALSO



For more information on how to use actions, see page 117.

## The Update Process

The TrueUpdate client connects to the Internet, intranet or LAN and downloads the server file. It uses the version identifiers in the server file to determine what version is installed on the user's system, and then carries out the list of actions associated with that version.

---

## Chapter 3

# Getting Started

---

TrueUpdate boils down to four basic steps: building the client, building the server file, putting the server file where the client can access it, and integrating the client into your software.

## A Few Tips Before you Begin

### *Consider a Dry Run or Two*

Building an update is kind of like building a puzzle; it's a lot easier to see how the pieces fit together if you've already seen what the end result will look like. When building a puzzle, you can sneak peeks at the picture on the box; when you're designing your first update, you might find yourself missing that kind of "big picture" at times. If you feel a bit confused at first, rest assured that things will make a lot more sense once you've completed the process of building an update.

### *Map Out Your Course*

Designing an update requires a bit of thought. You need to know where your server files will be located, and what user names and passwords you'll need for any FTP or web servers you plan to use. Exactly what information you need depends on the specific requirements of your update, but the more information you can gather before you start, the better.

### *Don't be Afraid to Just Follow the Radiator Cap*

Feel free to jump between the Client Configuration Utility and the Server File Editor at will. You don't have to start off in one or the other; it makes just as much sense to build the server file first as it does to start by configuring the client. There are some areas where the two programs overlap—for instance, you can define variables in both programs—and at first it might be difficult to know which program you should use. This User's Guide will try to make it clear what each program does, but if all else fails, feel free to experiment and go with what works.

# Building the TrueUpdate Client

At first glance, building the client might seem like the easiest step. The Client Configuration Utility is easy to use, and there aren't that many settings you need to change—for most situations, the default settings will work fine.

Keep in mind that once you've distributed the client to your users, it's more difficult to change the client than the server file. Unlike the server file, which you can change at any time without distributing any files to your users, the only way to change the TrueUpdate client files is to replace them with new ones on the user's system. Building the client requires some forethought and planning if you want to avoid having to distribute a new client in the future.

### IMPORTANT



Try to anticipate all the changes you'll need to make to the client, *before* you distribute the client with your software. Your goal should be to never have to update the client once it's built.

It's not impossible to update the client once it's in the field, but it's easier if you don't have to.

One of the things you should determine before building the client is how you're going to identify your versions. Bear in mind that you're limited by what's available to you on the Version Identifiers tab of the Server File Editor: built-in variables, values in the Registry or INI files, the version or CRC value of a file (which you must have a path to), and any custom variables that you defined in the Client Configuration Utility.

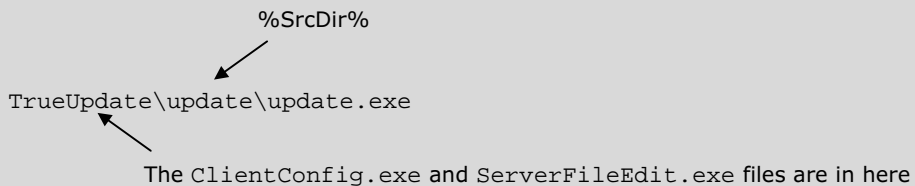
In order to check the version or CRC value of a file, you need to know where the file will be located. If the file is located in the same folder as the TrueUpdate client, you can just use the built-in variable %SrcDir%. But if the file is located in a different folder, you'll need to write the path to the Registry (or an INI file) as part of your installation, and read the path in from the Registry (or INI file) using a custom variable. Remember: the only custom variables that exist when the version identifiers are tested are the ones that you define in the Client Configuration Utility. If your version identifiers require any custom variables, you must define them in the Client Configuration Utility before you distribute the client to your users.

**TIP**

Always have your installation program write your software's installation or "application" directory to the Registry (or an INI file).

**Identifying a version (a real-life example)**

Our programming team wanted to identify the version of TrueUpdate using the version resource information in the `ClientConfig.exe` and `ServerFileEdit.exe` files. Because the TrueUpdate client is run from a sub-directory, the programming team wasn't able to use `%SrcDir%` as the path to those two files. (`%SrcDir%` would hold the path to the client, and not to the TrueUpdate program folder, which is one level "above" the sub-directory where `update.exe` is found.)



The programming team decided to write the location of the TrueUpdate folder to the Registry with Setup Factory when TrueUpdate is installed. Then they defined a custom variable named `%AppDir%` in the Client Configuration Utility to get that path from the Registry at run time. Finally, they used `%AppDir%` for the path to each file in the version identifiers they defined in the Server File Editor.

Another way to solve this would be to distribute `update.exe` in the TrueUpdate folder instead, so the two target files could be accessed via `%SrcDir%`.

## Chapter 3

---

To build the TrueUpdate client:

1. Modify the product info on the Product Info tab (see page 41).
2. Add the server file locations to the Product Info tab (see page 42).

### NOTE



The server file doesn't have to actually be at the server file location yet...you just need to know where it will be located. For instance, if the server file is going to be hosted on a web server, you only need to know what the URL of the file will be.

The client needs the server file in order to be able to accomplish anything, so the Client Configuration Utility won't let you generate the client files until you provide at least one server file location.

3. *Optional:*  
Configure the default settings for the client's interface on the Interface tab (see page 49).
4. *Optional:*  
Use the Messages tab to customize the interface text or load additional language files into the client (see page 61).
5. Define any custom variables your update will need early in the update process (up to the point where a version has been identified) on the Variables tab (see page 66).
6. Generate the client files (see page 73).

---

## Building (or Updating) the Server File

### *Setting Up the First Version*

Setting up the server file the first time is very easy, since you don't have to identify any older versions of your software. (There's no point identifying any versions distributed without the TrueUpdate client, since those versions won't be initiating the update process. The only exception is if you distribute the client separately.)

To set up the current version in the server file:

1. Name the version (see page 101).
2. Add version identifiers to identify the version (see page 105).
3. Add the "Latest Version" action from the "Status" category (see page 117). This will inform the TrueUpdate client that this is the latest version.

Once those three steps are done, the server file is ready to be published.

### *Setting Up the Second Version*

Setting up the second version in the server file requires a bit more effort, because now you actually have an older version of your software with TrueUpdate technology in the field, and a new version to update it to.

To set up this second edition of the server file:

1. Prepare the files that will be used to update your software.
2. Add a new version to the server file (see page 98).
3. Add version identifiers to identify the new version (see page 105).
4. Add the "Latest Version" action from the "Status" category for this new version (see page 117). This will inform the TrueUpdate client that this is the latest version.

## Chapter 3

---

5. Remove the "Latest Version" action from the Actions tab for the previous version (see page 122).
6. Add whatever actions are necessary to bring the previous version up to date (see page 117).

Once those steps are done, the new server file is ready to replace the old server file—in other words, the server file is once again ready to be published.

### TIP



Things become easier once you have the second version in the server file, because you can use an existing version as a template. Simply copy the second version and paste it back onto the versions bar. Then, rename this copy to create a "new" version, and modify its actions and identifiers. This way you don't have to build a new version from scratch.

## Publishing the Server File

Once you've built the server file, you can publish it. Publishing the server file simply means making it available at a location where the client can access it via HTTP, FTP, or LAN.

### SEE ALSO



For more information on publishing the server file, see page 87.

Publishing the server file doesn't involve distributing anything to your users; it simply changes what the TrueUpdate client will download when it checks for an update. Because you can publish a new version of the server file at any time, you're free to make changes to the server file whenever you want.

## Integrating the Client into your Software

Once you've built the TrueUpdate client, you need to integrate it into your software. Integrating the client simply means including it with your software distribution and providing one or more ways for your users (or your software) to initiate an update.

Integrating TrueUpdate can be as simple as putting an item on the Start menu to call the client executable, or it can be as sophisticated as having your application run the executable silently every 15 days. You could provide an item in your application's Help menu to initiate an update, or even a button on your application's toolbar. The level of integration is entirely up to you.

### NOTE



The client program is simply an executable that can be run normally from Windows.

To integrate the TrueUpdate client into your software:

1. Add the client files to your software distribution. The two client files you need to add are named `update.exe` and `update.cli` by default.
2. *Optional:*  
Provide some means for your users to initiate an update. For instance, you could use your preferred installation software to add a link in the Start menu to call the client executable.
3. *Optional:*  
Call the client executable directly from your application, either automatically or in response to some kind of user action.

### SEE ALSO



For more information on integrating the client, see page 171.

# Different File Patching Methods

TrueUpdate provides you with a flexible framework for updating your software. Within that framework, you're free to use any system you want in order to actually bring your software up to the latest version.

Here are some of the different methods you can use with TrueUpdate to bring your software up to date:

### **Full-history patch files**

Full-history patch files are able to bring any number of older versions of your software up to date with a single executable. Using full-history patch files makes designing updates easy. With full-history patches, you only ever need two versions in the server file: one for all the "old versions" that need to be updated, and one for the "current release." You also only really need two actions: one action to download the full-history patch, and another action to execute it.

You can build full-history patch files with Indigo Rose's Visual Patch.

### **Incremental or "binary" patch files**

Incremental or "binary" patch files can only update one version of your software at a time. Usually they're used to bring one version of software up to the next version. In order to bring an older version completely up to date, you need to download and apply several patches in sequence. TrueUpdate makes applying binary patch files much easier for your users, because they don't have to worry about acquiring the right patch files and running them in the correct order. The actions you define for TrueUpdate will handle this often difficult update process for them.

### **Full setup executables (setup.exe)**

You can use a full setup executable to update your software—essentially distributing a new full version of the product to your users. You could even have TrueUpdate uninstall the older version by running the uninstaller set up by your installation program.

You can create a setup executable by using Indigo Rose's Setup Factory.

**Zip files**

You can download the updated files in a compressed zip archive, and use TrueUpdate's built-in zip support to extract them.

**Simple download actions**

You can just download the new files individually and overwrite the old ones, perhaps backing up the old ones first. (You could even compress the backups using TrueUpdate's built-in zip support.)



---

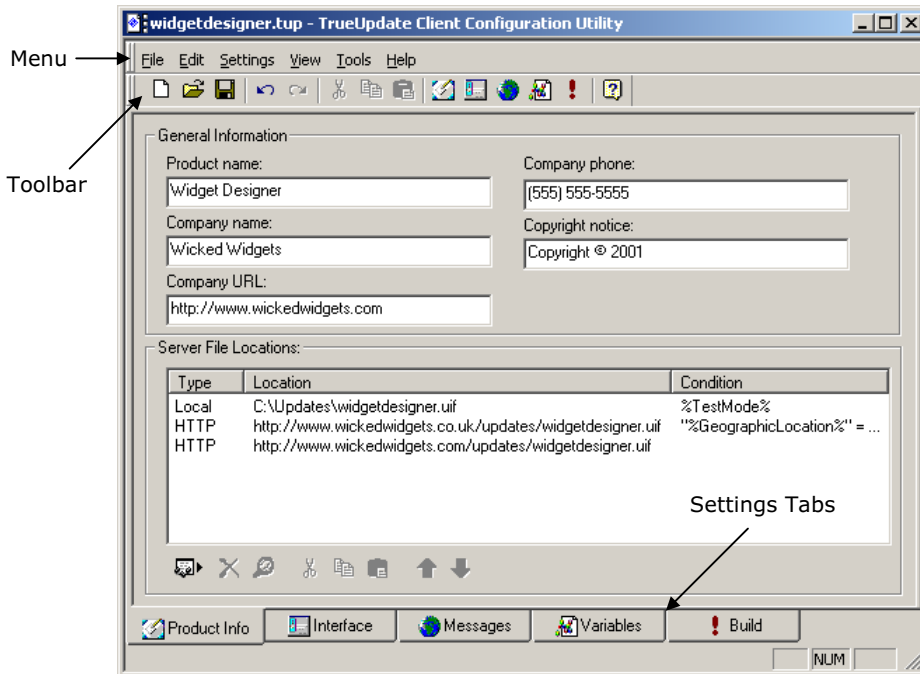
## Chapter 4

# The Client Configuration Utility

---

The TrueUpdate Client Configuration Utility is used to build the client executable and its accompanying data file. You use the Client Configuration Utility to specify general product information, configure the default interface settings, edit the messages that will be shown to the user, define custom variables, and generate the TrueUpdate client.

This chapter will help you get familiar with the TrueUpdate Client Configuration Utility as quickly as possible. It will provide you with a quick tour of the Client Configuration Utility's interface, and introduce the tasks you can perform with this tool.



The TrueUpdate Client Configuration Utility

# Projects

The Client Configuration Utility uses project files to store your work between sessions. These project (.tup) files contain the same information that goes into the TrueUpdate client when it's built.

Think of the project files as saving the work you do in the Client Configuration Utility at design time. When you load a project file into the Client Configuration Utility, you're returning the Client Configuration Utility to the same state it was in when that project file was saved.

### **Why use project files?**

The settings for the TrueUpdate client are stored in a special data file. When the Client Configuration Utility generates this data file, it is encrypted and compressed, which is a "one-way" process for security reasons—once you generate a client data file, you (or someone else with access to TrueUpdate) can't load it back into the Client Configuration Utility to modify it. The only way to make changes to a client data file is to generate a new one. This is why the Client Configuration Utility uses project files to store your work between sessions.

### ***Starting a New Project***

To start a new project, select **File | New** from the menu or press the Ctrl+N hotkey. The new project replaces the current project in the Client Configuration Utility. If you've made any changes to the current project you will be asked if you'd like to save the changes before continuing.

### ***Opening an Existing Project***

To open an existing project, select **File | Open** from the menu or press the Ctrl+O hotkey. This project will replace the current project in the Client Configuration Utility. If you've made any changes to the current project you will be asked if you'd like to save the changes before continuing.

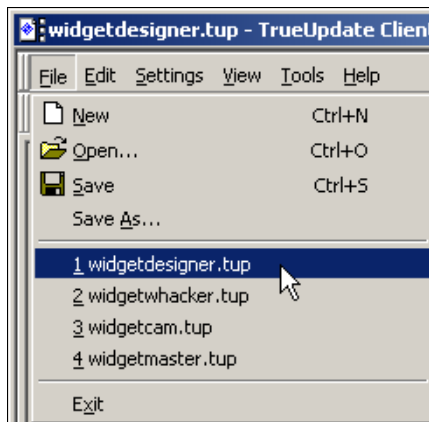
## Saving the Current Project

To save the current project, select **File | Save** from the menu or press the Ctrl+S hotkey.

To save the project with a different filename, select **File | Save As**.

## Reopening a Recent Project

You can access the four most recently saved project files directly from the **File** menu.



Opening a recent project file

### TIP



The name of the current project file is displayed in the TrueUpdate Client Configuration Utility's title bar.

## Product Information

The TrueUpdate client contains some general information about your product, including your product name, company name, web site address, phone number and copyright notice. This information is stored in the client so it can be used throughout the update process. (If it was stored in the server file, it couldn't be used at the very beginning, before the server file is downloaded.) For instance, this allows the client to display the name of your product on the *Welcome* screen.

## Chapter 4

---

The product information is made available throughout the project in the form of five built-in variables: %ProductName%, %CompanyName%, %CompanyURL%, %CompanyPhone%, and %Copyright%.

General Information	
Product name: <input type="text" value="Widget Designer"/>	Company phone: <input type="text" value="(555) 555-5555"/>
Company name: <input type="text" value="Wicked Widgets Inc."/>	Copyright notice: <input type="text" value="Copyright © 2001"/>
Company URL: <input type="text" value="http://www.wickedwidgets.com"/>	

The General Information section of the Product Info tab

### ***Setting Product Information***

You can edit the product information on the Product Info tab of the Client Configuration Utility. To get to the Product Info tab, select **Settings | Product Info** from the menu, press the Ctrl+1 hotkey, or click on the Product Info tab at the bottom of the Client Configuration Utility screen. Then, simply edit the text in the General Information section to match the values that apply to your product.

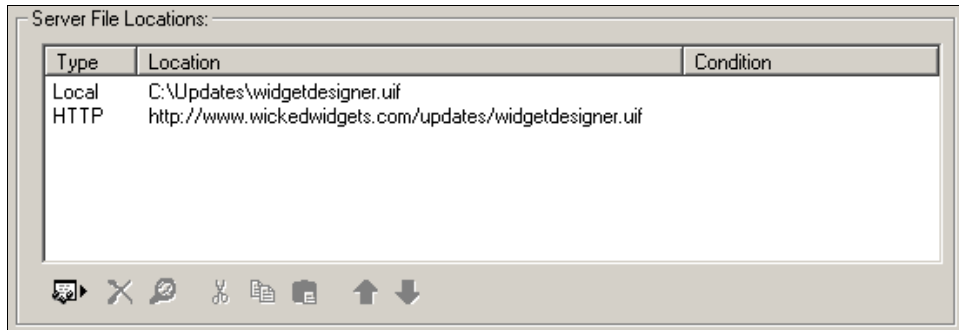
## **Server File Locations**

The list of server file locations is the most important setting in the Client Configuration Utility. It tells the TrueUpdate client where to look for the server file at run time. The client will attempt to download the server file from each location in the list in turn, until it either successfully downloads a server file, or runs out of locations to try.

### **NOTE**



The list of server file locations is so important, you can't generate the client data file if the list is empty. You must always provide at least one server file location in the list.



The list of server file locations on the Product Info tab

There are three types of server file locations you can add: Local, HTTP, and FTP.

### Local Server File Locations

A local server file location is a fully qualified path to the server file on a system the user has access to. For instance, you could have the TrueUpdate client "download" the server file from a hard drive on your company's local area network, or even from a hard drive or floppy disk on the user's system.

### HTTP Server File Locations

An HTTP server file location is a URL to the server file on a web server. The TrueUpdate client will download the server file from the web site using the HTTP protocol.


### FTP Server File Locations

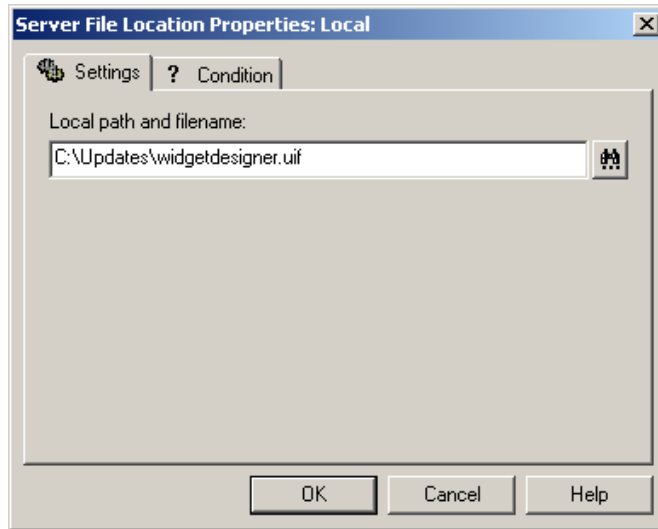
An FTP server file location consists of all the information needed to log into an FTP server account and retrieve a server file using the FTP protocol.

You can edit the list of server file locations on the Product Info tab of the Client Configuration Utility. To get to the Product Info tab, select **Settings | Product Info** from the menu, press the Ctrl+1 hotkey, or click on the Product Info tab at the bottom of the Client Configuration Utility screen.

### ***Adding Local Server File Locations***

To add a local server file location:

1. Press the **Add Location** button (  ) and select "Local" from the list that pops up. This will open the *Server File Location Properties: Local* screen.



The Server File Location Properties: Local screen

2. Enter the path to the server file in the **Local path and filename** field. You can use hard coded paths like `c:\foo\mydir\myupdate.uif`, or you can use UNC paths like `\\server\directory\filename.uif`. You can also use built-in path variables like `%SysDir%` to access files on the user's system.


#### **NOTE**

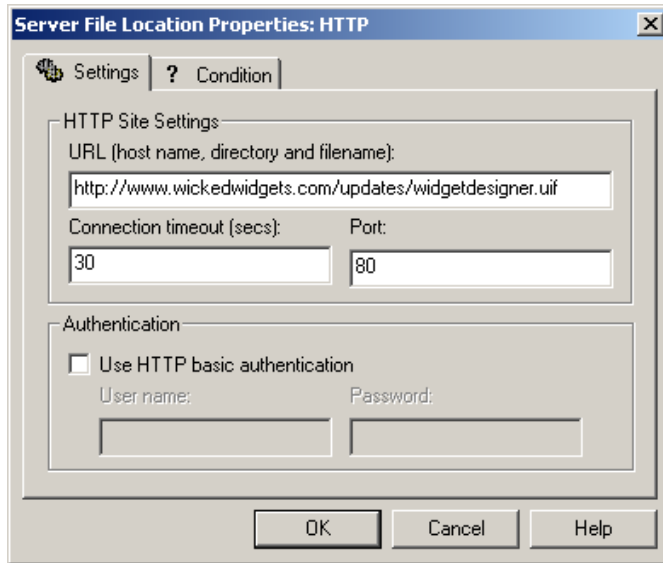


Remember that the TrueUpdate client will be running on the user's system. A path like `c:\foo\mydir\myupdate.uif` will access the user's `c:` drive—not yours.

## Adding HTTP Server File Locations

To add an HTTP server file location:

1. Press the **Add Location** button (  ) and select "HTTP" from the list that pops up. This will open the *Server File Location Properties: HTTP* screen.




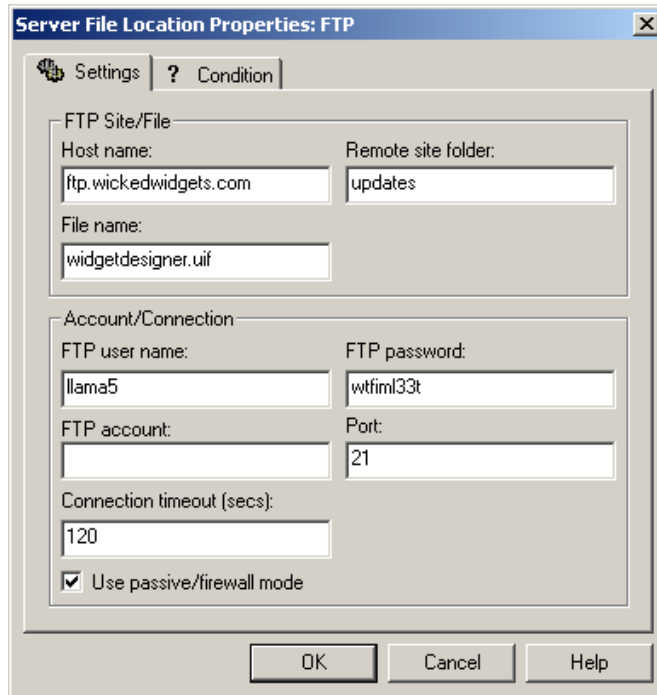
The Server File Location Properties: HTTP screen

2. Enter the URL to the server file in the **URL** field.
3. Adjust the **Connection timeout** and **Port** values as required.
4. If HTTP basic authentication is required to download the server file, select the **Use HTTP basic authentication** check box, and provide the required account information in the **User name** and **Password** fields.

### *Adding FTP Server File Locations*

To add an FTP server file location:

1. Press the **Add Location** button (  ) and select "FTP" from the list that pops up. This will open the *Server File Location Properties: FTP* screen.



The screenshot shows a dialog box titled "Server File Location Properties: FTP". It has two tabs: "Settings" and "Condition". The "Settings" tab is selected. The dialog is divided into two main sections: "FTP Site/File" and "Account/Connection".

**FTP Site/File section:**

- Host name: ftp.wickedwidgets.com
- Remote site folder: updates
- File name: widgetdesigner.uif

**Account/Connection section:**

- FTP user name: llama5
- FTP password: wtfim133t
- FTP account: (empty)
- Port: 21
- Connection timeout (secs): 120
- Use passive/firewall mode

At the bottom of the dialog are three buttons: "OK", "Cancel", and "Help".

The Server File Location Properties: FTP screen

2. Enter the name or IP address of the ftp server in the **Host name** field.
3. Enter the path to the server file in the **Remote site folder** field
4. Enter the name of the server file in the **File name** field.


5. If the client will need to use a user name and password to access the FTP server, you can change the values in the **FTP user name** and **FTP password** fields.

The **FTP account** field lets you specify an account name for servers that require it. (Some servers require a login name, password, and account name instead of just a login name and password.) Normally, though, you won't need to enter anything in this field.

6. You can adjust the **Connection timeout** and **Port** values as required.
7. If the FTP server you're connecting to doesn't support passive mode connections, you can clear the **Use passive/firewall mode** check box. Passive mode is required if you're connecting to the Internet through a proxy server. Most servers support passive mode connections, though, so it's generally best to leave this setting enabled.

### ***Removing Server File Locations***

To remove a server file location:

1. Select the location you want to remove in the Server File Locations list.
2. Press the **Remove Location** button (  ) to remove that location from the list.



#### **TIP**



You can hold a Ctrl or Shift key down to select multiple server file locations. Pressing the **Remove Location** button removes all the locations that are currently selected.

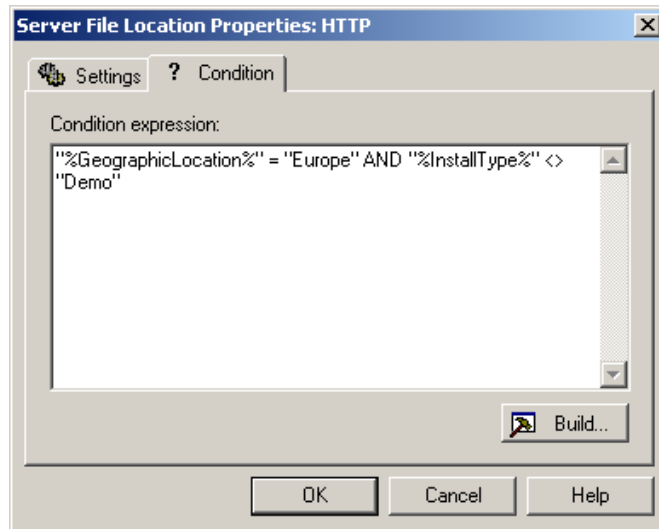
### ***Changing the Search Order for Server File Locations***

The TrueUpdate client will try to download the server file from each location in the Server File Locations list in turn, starting with the location at the top of the list. You can change the order that the locations are tried by changing the order of the server file locations in the list. To do so:

1. Select the server file location that you want to move.
2. Use the **Move Up** (  ) and **Move Down** (  ) buttons or the Ctrl+Up and Ctrl+Down hotkeys to reposition the server file location in the list.

### ***Setting Conditions for Server File Locations***

You can set conditions for a server file location, and the location will only be used if the conditions are met. Conditions are defined on the Conditions tab of the *Server File Location Properties* screens.



A condition for an HTTP server file location

#### **SEE ALSO**



For more information on how to define conditions, see page 143.

## Interface Settings

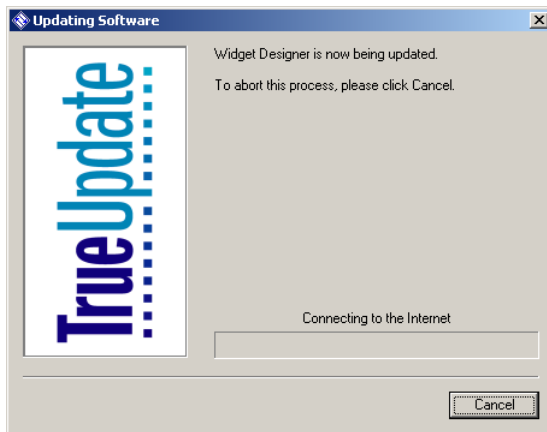
The TrueUpdate client can display two different interface styles: a "wizard" interface similar to the interface used by professional install software, and a "dialog" interface consisting of small dialog windows.

### Wizard

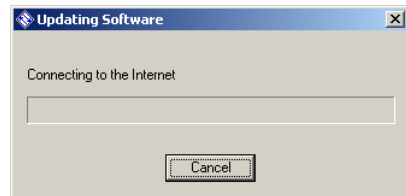
The wizard interface has room for longer messages and allows you to display an image on the left-hand side of the interface.

### Dialog

The dialog interface is small and simple, presenting a streamlined interface to the user.



The Wizard interface



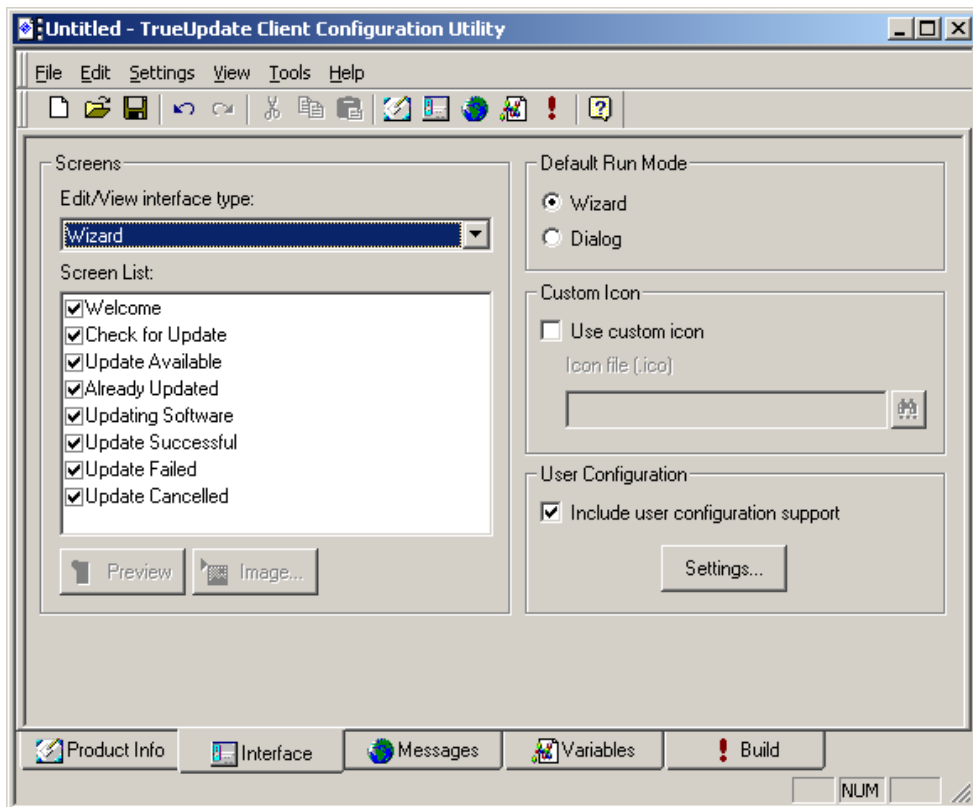
The Dialog interface

The wizard and dialog interfaces are both made up of eight screens. The screens are displayed in a set order, but you can select which screens get shown, and you have full control over the textual messages displayed on them.

You can modify the interface settings on the Interface tab of the Client Configuration Utility. To get to the Interface tab, select **Settings | Interface** from the menu, press the

## Chapter 4

Ctrl+2 hotkey, or click on the Interface tab at the bottom of the Client Configuration Utility screen.



The Interface tab of the Client Configuration Utility

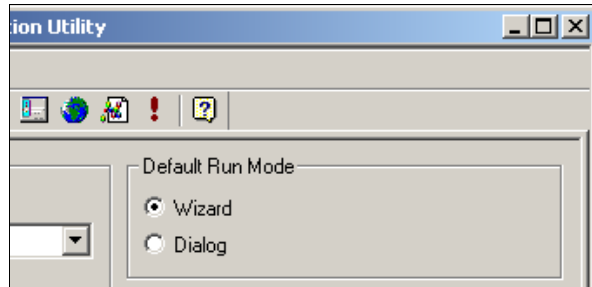
### NOTE



Most of the interface settings can be overridden at run time using the client executable's command line options. The Interface tab of the Client Configuration Utility allows you to configure how the client will appear by default, when no command line options are used.

## Selecting the Default Interface Style

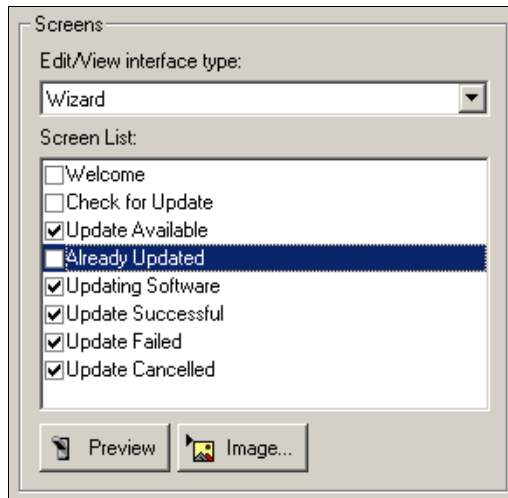
To set the interface style that will be used by default, select either the "Wizard" or the "Dialog" option in the Default Run Mode section of the Interface tab.



The Default Run Mode section of the Interface tab

## Showing and Hiding Screens

You can enable or disable individual screens by selecting or clearing the appropriate check box in the **Screen List** on the Interface tab. Only screens with a check mark will be displayed during the update process.

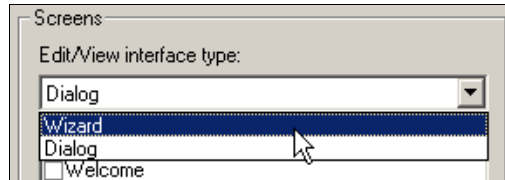


The Screen List

## Chapter 4

---

There are two separate screen lists—one for the screens that make up the wizard-style interface, and one for the screens that make up the dialog-style interface. You can switch between the two screen lists by selecting either "Wizard" or "Dialog" in the **Edit/View interface type** drop-down list.



Switching to the Wizard screen list

### NOTE




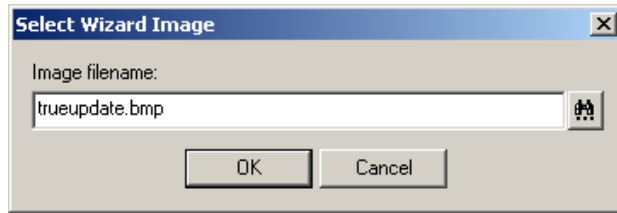
Switching between the interface styles with the drop-down list has no effect on which interface is used by default. The default interface displayed at run time is determined by the **Default Run Mode** option.

### ***Changing the Image Used on a Wizard Screen***


You can use a custom image on each of the screens in Wizard mode.

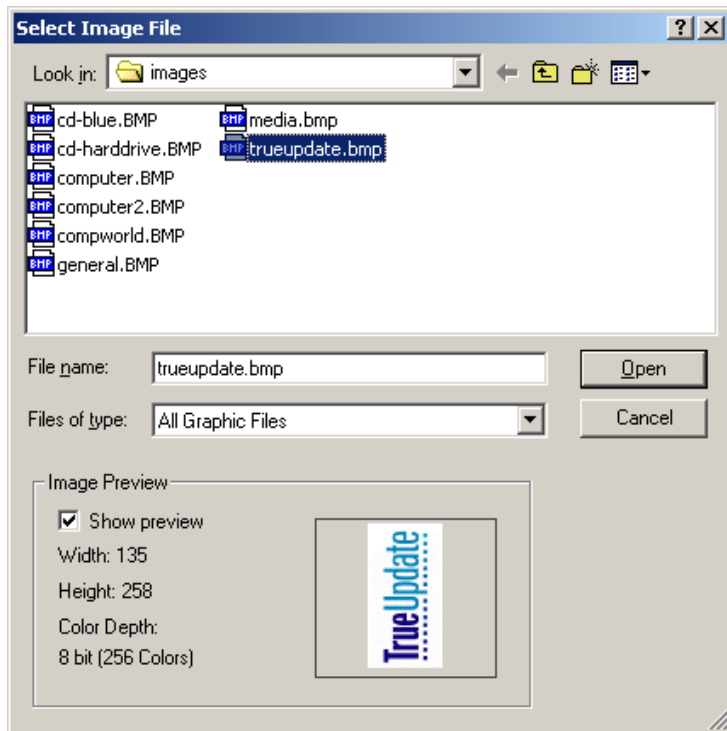
To select a custom image for a screen:

1. Switch to the Wizard screen list by selecting "Wizard" in the **Edit/View interface type** drop-down list.
2. Select the name of the screen whose image you want to change in the **Screen List**.
3. Press the **Select Image** button (  ) to display the *Select Wizard Image* screen.



The Select Wizard Image screen

You can change the image by entering the full path and filename of the image you want to use, or you can click on the **Browse** button (  ) to select a file using the *Select Image File* screen.




The Select Image File screen

## Chapter 4

---


You can use any BMP, JPG, PCX, PNG, TGA or uncompressed TIF file for the image. If the image is larger or smaller than 135x258 pixels, the image will be shrunk or stretched to match those dimensions.

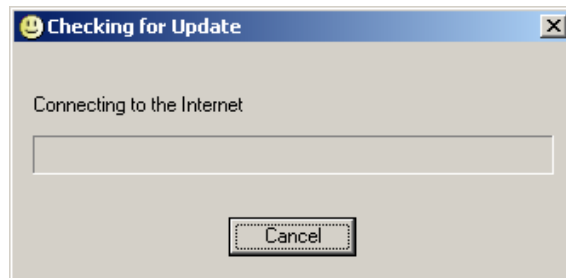
### ***Previewing Screens***

You can preview an individual Wizard or Dialog interface screen by selecting its name in the Screen List and pressing the **Preview** button (  ).

### ***Using a Custom Icon in the Title Bar***

To specify a custom icon for the title bar of the wizard or dialog interface screens:

1. Enable the custom icon feature by selecting the **Use custom icon** check box.
2. Enter the full path and filename of the icon ( `.ico` ) file you want to use, or click on the **Browse** button (  ) to select a file using the *Select Icon File* screen.



Example of a custom icon in use

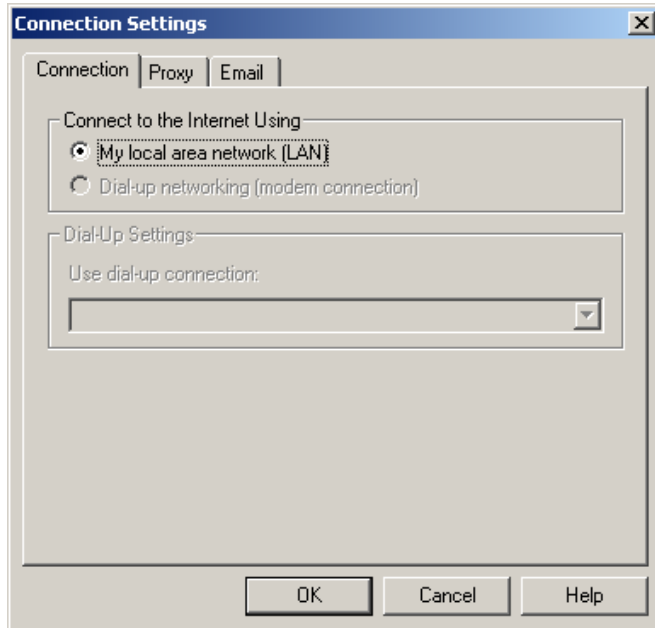
#### **NOTE**



TrueUpdate supports standard `.ico` files containing 16-color, 16x16-pixel and 16-color, 32x32-pixel bitmap images.

## ***Disabling Support for User Configuration***

At design time, you have the option to disable the user's ability to configure their TrueUpdate connection settings. Normally these settings determine how TrueUpdate will connect to the Internet from the user's system: via a LAN, using a dial-up service, through a proxy server, using passive mode during FTP transfers, etc. The settings are stored on the user's system, so the user only has to configure them once.



The TrueUpdate Connection Settings screen

When support for user configuration is disabled, the user is unable to configure the network settings. TrueUpdate simply assumes that a connection to the Internet already exists whenever the client is run. If there are any settings on the user's machine (perhaps from another product that also uses TrueUpdate), the client will ignore them.

### IMPORTANT

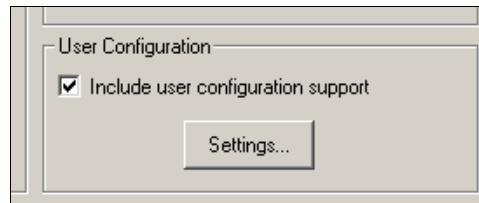


You should only disable TrueUpdate's user configuration support if you're certain that it won't be required—for instance, if the client will only be used on your company's internal LAN and none of the users connect through a proxy.

Disabling support for user configuration prevents TrueUpdate from connecting to the Internet from behind a proxy server, and from initiating a dial-up connection if required.


Normally this option should be enabled so the user can configure the proxy settings if they need to.

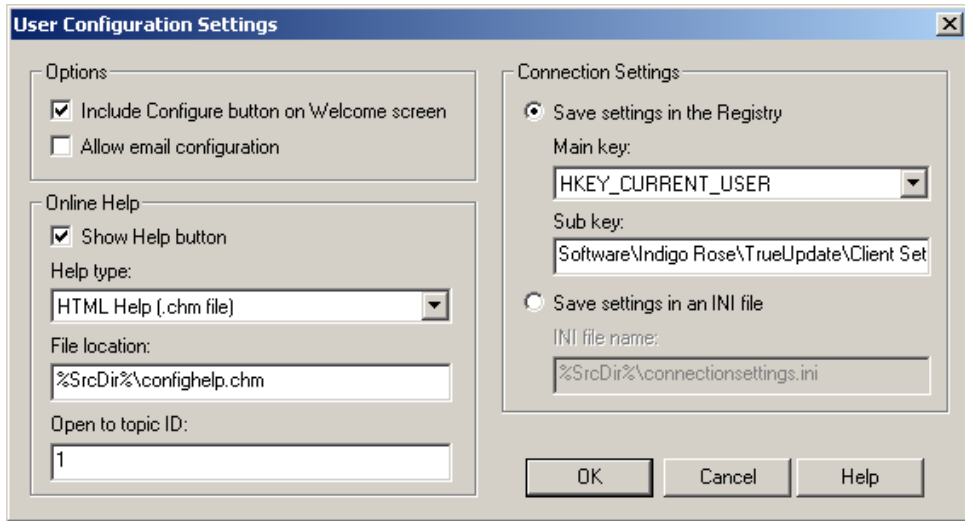
To disable user configuration support, clear the **Include user configuration support** check box.



The User Configuration section of the Interface tab

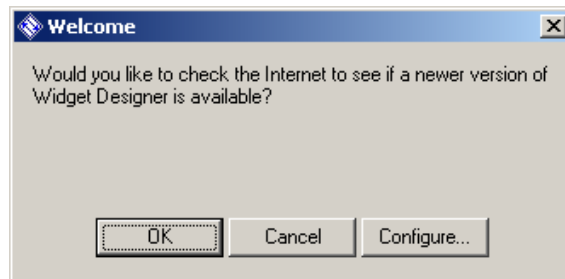
### ***Changing User Configuration Settings***

When support for user configuration is enabled, you can configure the support options on the *User Configuration Settings* screen. To access the *User Configuration Settings* screen, press the **Settings** button (  ) in the User Configuration section of the Interface tab.



The User Configuration Settings screen

The **Include Configure button on Welcome screen** check box determines whether a **Configure** button will be shown on the *Welcome* screen at run time.



The Configure button on the Welcome screen

Assuming the *Welcome* screen isn't hidden, the user can use the **Configure** button to access the *Connection Settings* screen and configure their TrueUpdate connection settings.

## Chapter 4

---

The **Allow email configuration** check box determines whether an Email tab will be included on the *Connection Settings* screen.

The Email tab allows users to provide an SMTP server address and port, and their email address. The information they provide is then stored in built-in variables that you can use in server file actions during the update process.

### NOTE



The email settings don't actually do anything on their own—they just provide values for three built-in variables: %UserEmailAddress%, %UserSMTPPort% and %UserSMTPServer%.



The screenshot shows a dialog box titled "Connection Settings" with three tabs: "Connection", "Proxy", and "Email". The "Email" tab is selected. Inside the dialog, there are two main sections. The first section is labeled "SMTP Mail Server" and contains two text input fields: "Server address:" with the value "mail.widgetlovers.com" and "Port:" with the value "25". The second section is labeled "Options" and contains one text input field: "Your email address:" with the value "cool\_user@widgetlovers.com". At the bottom of the dialog, there are three buttons: "OK", "Cancel", and "Help".

The optional Email tab of the Connection Settings screen

The **Show Help button** check box determines whether a **Help** button will appear on the *Connection Settings* screen.

The following options are available on the *User Configuration Settings* screen when the **Show Help button** check box is selected:

### Help type

This drop-down list allows you to choose between using an HTML help (.chm) file, a WinHelp (.hlp) file, or any other file type (such as .doc or .txt).

### File location

This field allows you to specify the path and filename of the help file you want to use.

### NOTE



When **Help type** is set to Other File Type, the client performs a shell "Open" on this file.

### Open to topic ID / Command line arguments

When **Help type** is set to either HTML help or WinHelp, the **Open to topic ID** field allows you to specify the topic ID you want the **Help** button to open to. For instance, you could have the **Help** button open topic ID "245" in your help file.

When **Help type** is set to Other File Type, the **Command Line arguments** field allows you to specify command line arguments. This is mainly useful if the file you specified in the **File location** field is an executable.

### TIP



You can use variables in the **File location** and **Open to topic ID** fields to provide help in multiple languages.

For instance, you could define a variable named %HelpFileLocation% and another named %HelpTopicID% on the Variables tab of the Client Configuration Utility. Then, using conditions, you could make the value assigned to either variable depend on what language is being used on the user's system. (The built-in variable %SysLanguage% would come in handy here.)

### Saving Connection Settings

The user's connection settings can either be stored in the Registry or an INI file. Storing settings in the Registry makes it easier to share them between products on one system; storing settings in an INI file makes it easier for the user to move them (especially between systems). Generally, if you want the user's connection settings to be portable, use an INI file; otherwise, use the Registry for the connection settings.

The choice is largely a matter of developer preference, however, so the decision is ultimately up to you. Either storage method will work as well as the other, so use whichever method fits better with the rest of your product.

To store the connection settings in the Registry, select the **Save settings in the Registry** option, and set the **Main key** drop-down list and the **Sub key** field to point to the Registry key where you want the settings to be stored.

To store the connection settings in an INI file, select the **Save settings in an INI file** option, and set the **INI file name** field to the path and filename of the INI file you want the settings to be stored in.

#### TIP



If you have multiple products that use TrueUpdate, you can share the user's connection settings between all of them. That way the user only has to configure the settings once for all of your products.

The opposite is also true—you can store the connection settings for any of your products in a different location to make them product-specific.

## Messages

TrueUpdate features fully customizable text and error messages. Every text message presented to the user can be edited using the Client Configuration Utility.

Each message is associated with a unique *message ID*. For instance, the message ID for the text used on all the **Next** buttons is MSG\_BUTTON\_NEXT. You can change the text of all the **Next** buttons by changing the text associated with that message ID. For example, you could translate all the **Next** buttons to French by changing the text to read "Prochain >" instead of "Next >". Or you might want to change the English text to read "Continue" instead.

TrueUpdate uses the message ID prefix to determine whether a message belongs to the dialog or wizard interface style. Message IDs that begin with MSG\_ are shown on the dialog interface, and message IDs that begin with MLMSG\_ are shown on the wizard interface. MSG\_ is short for "message" and MLMSG\_ is short for "multi-line message."

*Languages* are collections or sets of messages assigned to a specific language ID. A separate set of messages is stored in the client for each language you want the client to be capable of displaying. At run time, the client detects which language is being used on the user's system, and presents the corresponding set of messages. If there are no messages defined for the user's language, the messages from the default language are used.

Between projects, each set of messages is stored in a separate language (.lng) file. You add the messages associated with a language to your project by adding the appropriate language file.

### IMPORTANT

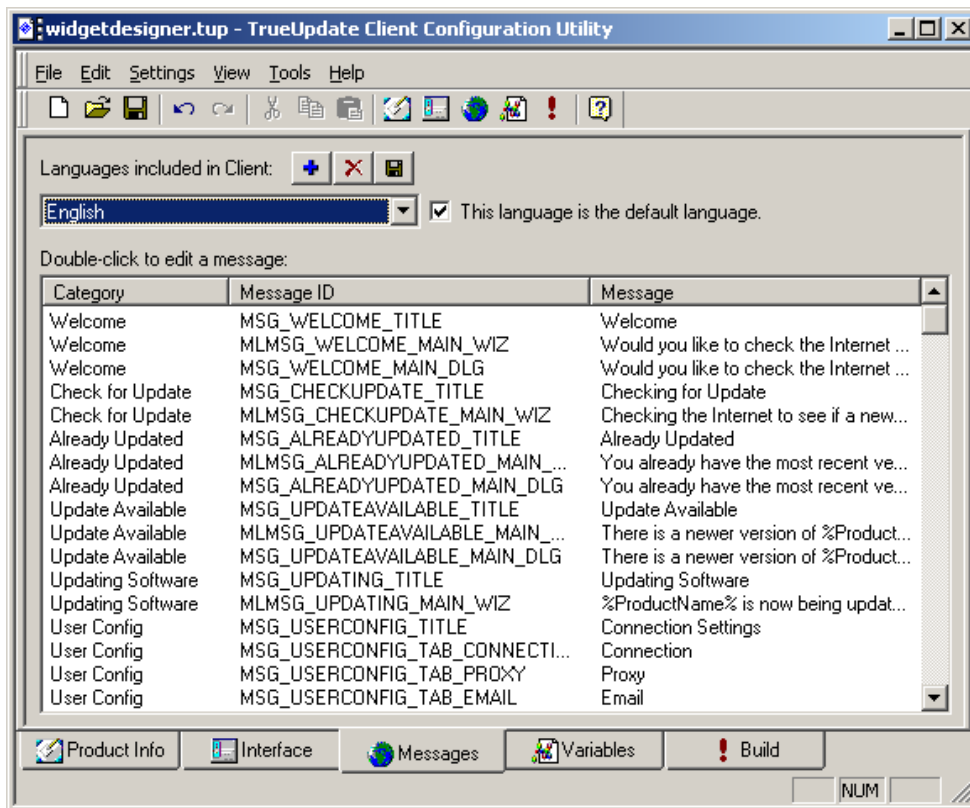


Adding a language file adds a copy of the messages in that file to the project. Editing those messages in the Client Configuration Utility has no effect on the original messages in the language file.

There can be one set of messages in a project for each primary language ID recognized by Windows.

## Chapter 4

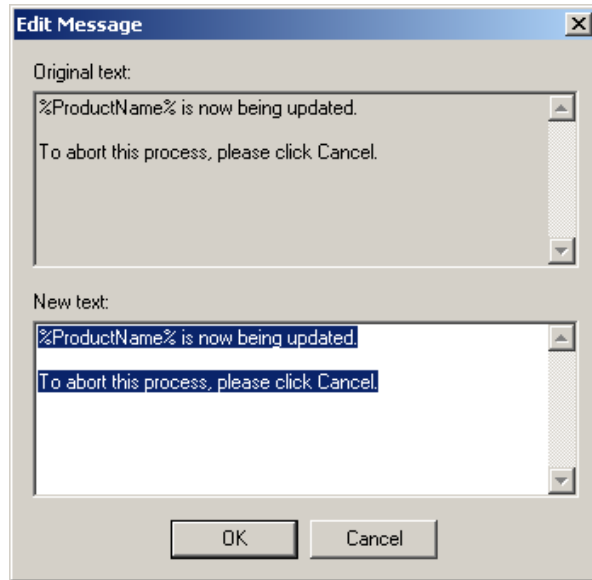
You can edit the messages on the Messages tab of the Client Configuration Utility. To get to the Messages tab, select **Settings | Messages** from the menu, press the Ctrl+3 hotkey, or click on the Messages tab at the bottom of the Client Configuration Utility screen.



The Messages tab of the Client Configuration Utility


### Editing Messages

To edit a message, double-click on the message you want to edit in the list of messages on the Messages tab. This will open the *Edit Message* screen where you can edit the selected message.



The Edit Message screen

## Adding Languages

To add another language to the current project, press the **Add Language** button (  ) at the top of the Messages tab, and select the language ( .lng ) file that contains the language you want to add.

### NOTE



If you attempt to add a language that is already listed in the Client Configuration Utility, you'll be asked to confirm that you want to overwrite the existing messages with the contents of that language file.


## Removing Languages

To remove a language from the current project:

1. Select a language in the drop-down list on the Messages tab.

## Chapter 4

---

2. Press the **Remove Language** button (  ) at the top of the Messages tab.  
All of the messages associated with the selected language will be removed from the current project.

### ***Setting The Default Language***

The default language is used when the language on the user's system isn't found among the languages included with the client. Or, to put it another way, the default language is used when the client doesn't have a language that matches the user's localization settings.

For instance, let's say your TrueUpdate client includes messages in English, French, German and Spanish, with English being marked as the default language. If the user's system is localized to Japanese, the default English messages would be used.

To set the default language:


1. Select a language in the drop-down list on the Messages tab.
2. Select the **This language is the default language** check box to make this language the default language.



The default language check box

### ***Exporting Messages to a Language File***

To export messages into a new language file:

1. Use the drop-down list to select the language whose messages you want to export.
2. Press the **Export** button (  ).

3. Provide a name for the language (.lng) file that will contain the messages.

### ***Translating Messages***

There are two methods you can use to create a new file.

Method 1:

1. Make a copy of the language file you want to translate and name it according to the new language, i.e. copy "English.lng" to "German.lng".
2. Open the new language file in a text editor and change the language ID to the appropriate ID for the new language. For example, if you were translating the English messages to German, you would change the line that reads:

```
LanguageID=9
```


to:

```
LanguageID=7
```

A complete list of language IDs can be found in the  
C:\Program Files\TrueUpdate\languages\langids.ini file.

3. Translate the messages in your text editor or load the new language file into the Client Configuration Utility and do it there.

Method 2:

1. Select the language you want to translate *from* in the drop-down list on the Messages tab.
2. Translate all the messages by editing them.
3. Use the **Export** button (  ) to export the translated messages into a new language file.

4. Use a text editor to change the language ID in the language file you exported. For example, if you were translating the English messages to French, you would change the line that reads:

```
LanguageID=9
```

to:

```
LanguageID=12
```

### TIP



Use the list of Windows language IDs found in the `langids.ini` file to determine what language ID corresponds to the language you've translated the messages to. (The `langids.ini` file is located in `C:\Program Files\TrueUpdate\languages` by default.)

## Variables

The Variables tab of the Client Configuration Utility allows you to define custom variables that will be available from the start of the update process.

### IMPORTANT



Although you can also define variables using server file actions, those actions aren't performed until after the version is identified on the user's system. Any variables that are needed *before* the actions are performed must be defined in the Client Configuration Utility.

For instance, any custom variables you need for version identifiers in the Server File Editor can only be defined in the Client Configuration Utility.

You can use these variables as constants in messages—including the messages shown on the *Welcome* and *Check for Update* screens before the server file is downloaded. (Any variables you define with server file actions can't be used in the first two screens because the server file hasn't been downloaded yet.)

You can also use these variables to add conditions to the server file locations. The TrueUpdate client will only attempt to download the server file from locations whose conditions are met. This allows you to specify server file locations that are only used in specific situations. For instance, you could check the registry for a value set by your installation program indicating which geographic region your user resides in, and use that information to determine which server file locations to check.

### SEE ALSO



For more information on conditions, see page 143.

Of course, you can use the variables that you define in the Client Configuration Utility throughout the update process. You can even use them in the actions you define using the Server File Editor.

### The secret life of variables

There is a common "pool" for all of the variables in TrueUpdate, whether they're defined in advance with the Client Configuration Utility or defined on the fly by actions in the server file. In fact, the only difference between the two kinds of variables is where you happened to define them.

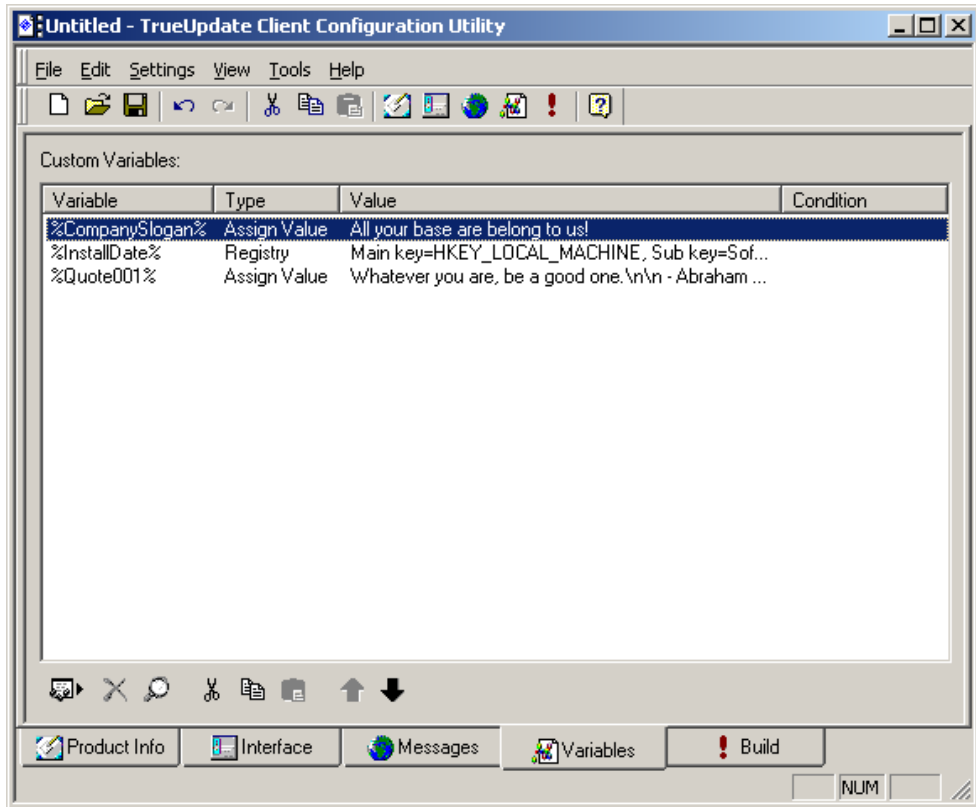
This isn't so strange when you realize that the whole update process is coordinated by the client executable. All the instructions you set in the Client Configuration Utility and the Server File Editor end up being carried out by the `update.exe` in the end.

### TIP



You can change the value of an existing variable simply by assigning a new value to it.


You can add, remove and edit variables on the Variables tab of the Client Configuration Utility. To get to the Variables tab, select **Settings | Variables** from the menu, press the Ctrl+4 hotkey, or click on the Variables tab at the bottom of the Client Configuration Utility screen.

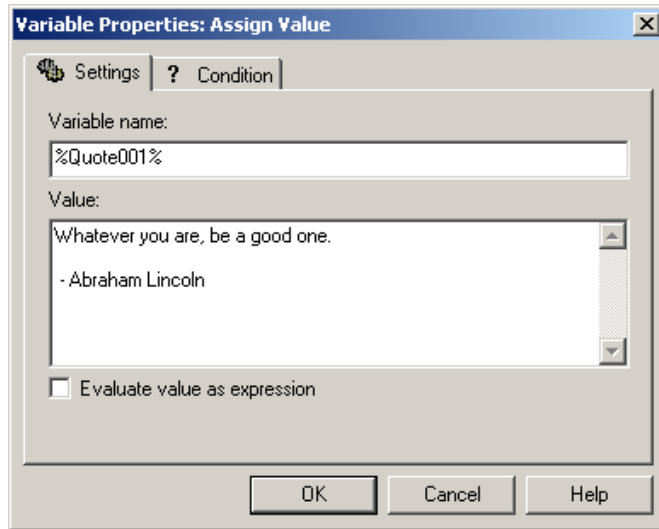


The Variables tab of the Client Configuration Utility

### ***Assigning a Value to a Variable***

To add an Assign Value variable:

1. Press the **Add Variable** button (  ) and select "Assign Value" from the list that pops up. This will open the *Variable Properties: Assign Value* screen.



The Variable Properties: Assign Value screen

2. Enter a name for the variable in the **Variable name** field. The variable name should begin and end with a percentage sign.
3. Enter the value you want the variable to have in the **Value** field.
4. If you want the value to be evaluated as an expression, select the **Evaluate Value as expression** check box. The variable will contain the result of the **Value** field after it is evaluated as an expression. For instance, if you entered the expression "5 + 3" in the **Value** field, the variable would contain the result, "8".


**SEE ALSO**

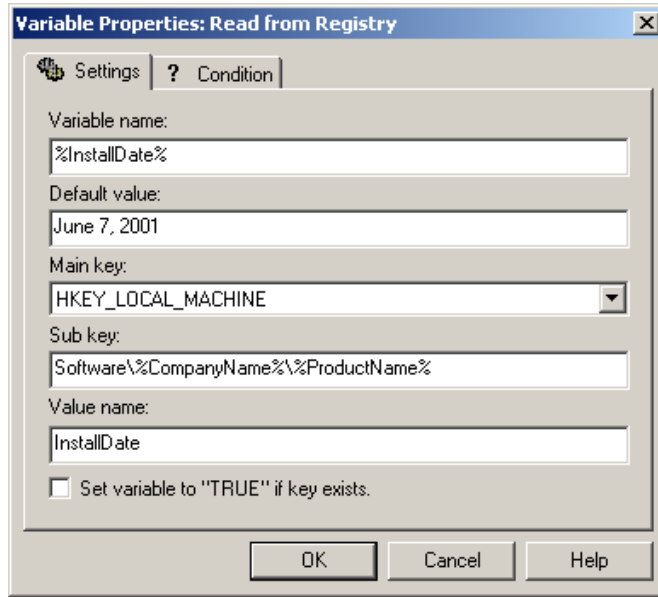


For more information on expressions, see page 143.

### Reading a Value from the Registry

To add a Read from Registry variable:

1. Press the **Add Variable** button (  ) and select "Read from Registry" from the list that pops up. This will open the *Variable Properties: Read from Registry* screen.



The Variable Properties: Read from Registry screen


2. Enter a name for the variable in the **Variable name** field. The variable name should begin and end with a percentage sign.
3. Enter a default value in the **Default value** field. This is the value that will be assigned to the variable if the registry key doesn't exist.
4. Select the main key (such as HKEY\_LOCAL\_MACHINE) where the registry value can be found using the **Main key** drop-down list.
5. Enter the path to the registry key in the **Sub key** field.

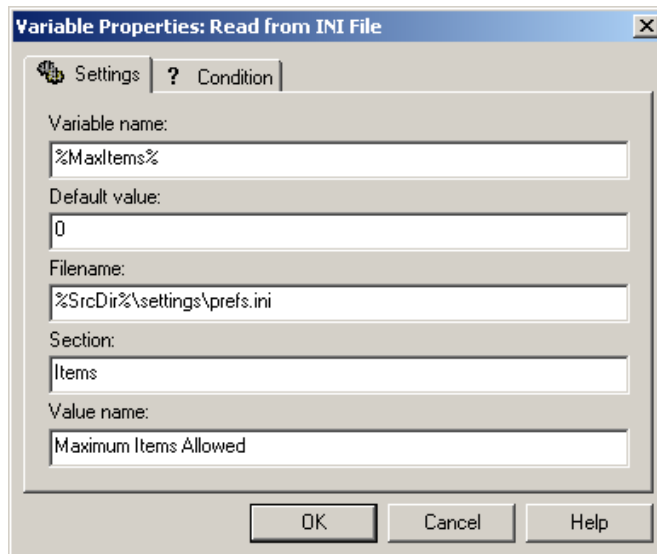
6. Enter the name of the registry value you want to read in the **Value name** field.
7. If you just want to find out whether the registry key exists, select the **Set variable to "TRUE" if key exists** check box. When this check box is selected, the variable named in the **Variable name** field will be set to "TRUE" if the key exists. If the key doesn't exist, the variable will be set to the value you entered in the **Default value** field.

If you want to copy the value at this registry key into the variable, leave the **Set variable to "TRUE" if key exists** check box unselected.

### Reading a Value from an INI File

To add a Read from INI File variable:

1. Press the **Add Variable** button (  ) and select "Read from INI File" from the list that pops up. This will open the *Variable Properties: Read from INI File* screen.



The screenshot shows a dialog box titled "Variable Properties: Read from INI File". It has a "Settings" tab selected. The fields are filled with the following values:

- Variable name: %MaxItems%
- Default value: 0
- Filename: %SrcDir%\settings\prefs.ini
- Section: Items
- Value name: Maximum Items Allowed


Buttons at the bottom: OK, Cancel, Help.

The Variable Properties: Read from INI File screen

2. Enter a name for the variable in the **Variable name** field. The variable name should begin and end with a percentage sign.
3. Enter a default value in the **Default value** field. This is the value that will be assigned to the variable if the desired value name isn't found in the INI file.
4. Enter the path and filename of the INI file you want to read the value from in the **Filename** field.
5. Enter the name of the INI file section where the value can be found in the **Section** field.
6. Enter the name of the value you want to read in the **Value name** field. The value associated with this value name will be assigned to the variable named in the **Variable name** field.

### *Removing Variables*

To remove a variable from the Custom Variables list:

1. Select the variable definition that you want to remove.
2. Press the **Remove Variable** button (  ) or use the Delete hotkey to remove the selected variable definition from the list.

#### **TIP**





You can hold a Ctrl or Shift key down to select multiple variable definitions. Pressing the **Remove Variable** button removes all the variable definitions that are currently selected.

### *Changing the Order that Variables are Assigned In*

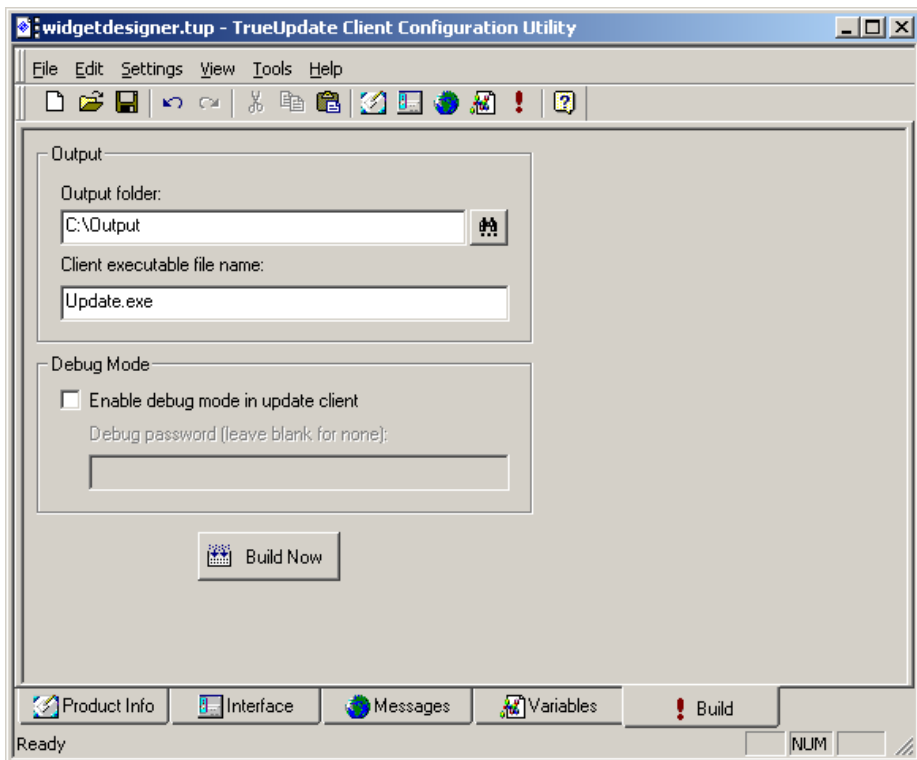
The order your custom variables are assigned in can be important if any of the variables refer to each other. You'll want to ensure that each variable is assigned a value before any other variables reference it.

You can change the order that the variables are assigned in by changing the order of the variable definitions in the Custom Variables list. To do so:

1. Select the variable definition that you want to move.
2. Use the **Move Up** (  ) and **Move Down** (  ) buttons or the Ctrl+Up and Ctrl+Down hotkeys to reposition the variable definition in the list.


## Building the Client

You can generate the TrueUpdate client on the Build tab of the Client Configuration Utility. To get to the Build tab, select **Settings | Build** from the menu, press the Ctrl+5 hotkey, or click on the Build tab at the bottom of the Client Configuration Utility screen.



The Build tab of the Client Configuration Utility

### ***Changing the Output Folder***

You can control where the TrueUpdate client will be generated by editing the path in the **Output Folder** field. Alternatively, you can click on the **Browse** button (  ) to select a folder using the *Select Folder* screen.

#### **TIP**



You can set the default output folder for new projects in the Client Configuration Utility preferences (see page 77).

### ***Changing the Client Executable Filename***

You can change the name that will be given to the client executable and its accompanying data file by editing the value in the **Client Executable Filename** field.

#### **NOTE**



The name of the client executable's data file is based on the name you provide for the client executable, but with a `.cli` extension instead of `.exe`. For instance, if you name the client executable `MyUpdate.exe`, its accompanying data file will be named `MyUpdate.cli`.

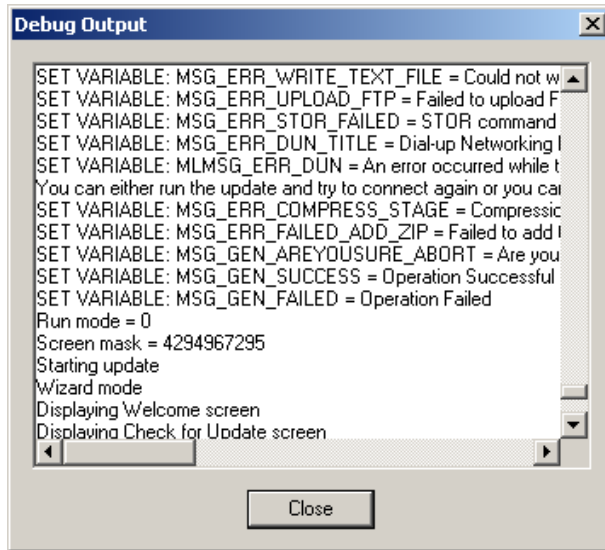
### ***Enabling Debug Mode in the Client Executable***

The TrueUpdate client can optionally be built with support for a special debug mode in the client executable. While in debug mode, the client displays a real-time list of its activities on a *Debug Output* screen. This information can be very helpful when building and debugging an update.

#### **NOTE**



Because of the large amounts of information that are written to the *Debug Output* screen, the client may appear to run slower in debug mode.



The Debug Output screen

At the end of the update process, the debug information is written to a log file. By default, this file is created in the same directory as the client executable, with the same base name and a `.log` extension. For instance, if the client executable is `C:\update.exe`, the log file will be saved as `C:\update.log`. (You can specify your own path and filename for the log file by using the `/DF` command line option—see page 181.)

**TIP**



Debug mode log files can be helpful when trying to solve any tech support issues your users might encounter. You could even have your users generate a log file and send it to you, to see exactly what your update is encountering on their system.

To enable debug mode, select the **Enable debug mode in client executable** check box.

Any clients you build when debug mode is enabled will recognize a special `/DEBUG` command line option. The `/DEBUG` option tells the client executable to run in debug mode. If debug mode wasn't enabled when a client was built, the `/DEBUG` option will have no effect on it.

### ***Requiring a Debug Mode Password***

You can optionally protect the debug mode with a password. This allows you to distribute a TrueUpdate client with debug mode enabled, and control which users have access to the information that debug mode provides.

#### **NOTE**



The log file generated by the client in debug mode is a plain, unencrypted text file. The debug mode password doesn't protect this log file—it only prevents unauthorized users from running the client in debug mode.

To require a debug mode password, simply enter one into the **Debug password** field on the Build tab of the Client Configuration Utility.


Any clients you build after specifying a password will only run in debug mode if the user provides that password with the `/DEBUG:password` command line option. The `/DEBUG` option has no effect if the incorrect password is used.

#### **NOTE**



To remove a debug mode password, simply clear the text from the **Debug password** field.

### ***Building the Client***

When you're ready to build the TrueUpdate client, press the **Build Now** button (  ). The client executable and its accompanying data ( `.cli` ) file will be generated in the output folder. If a custom icon ( `.ico` ) file was specified, it will be copied to the output folder as well.

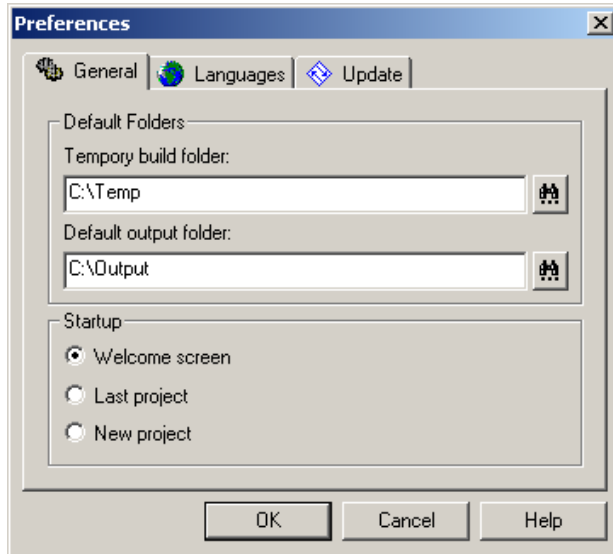
#### **TIP**



Once you've built the client, you can proceed to integrate it into your software application. The details of the integration are up to you, and may require some programming, depending on the level of integration you desire. More information on how to integrate TrueUpdate into your software can be found in chapter 10.


## General Preferences

You can set general preferences for the Client Configuration Utility on the General tab of the *Preferences* screen. To access the *Preferences* screen, select **Edit | Preferences** from the menu.




The General tab of the Preferences screen

### Setting the Temporary Build Folder

The Client Configuration Utility uses a temporary folder while generating the TrueUpdate client files. You can change the folder that is used for this purpose by editing the path in the **Temporary build folder** field, which is located on the General tab of the *Preferences* screen. You can also press the **Browse** button (  ) to browse for a path using the *Select Folder* screen.

### Setting the Default Output Folder

To change the default output folder that will be used each time you start a new project, edit the path in the **Default output folder** field on the General tab of the *Preferences* screen. You can also press the **Browse** button (  ) to browse for a path using the *Select Folder* screen.

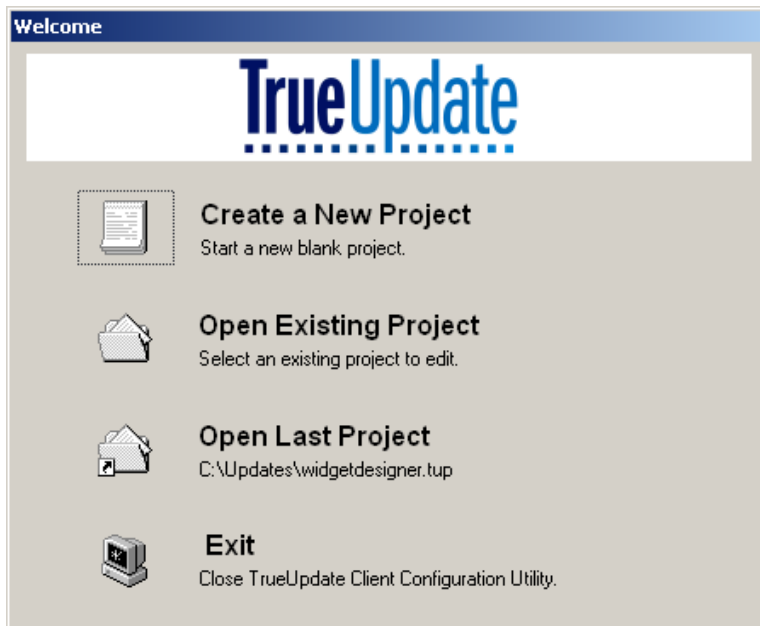
### NOTE



You can change the output folder for the current project on the Build tab of the Client Configuration Utility.

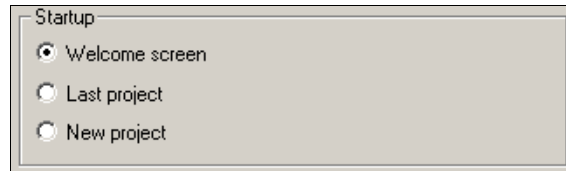
### ***Choosing Startup Options***

When you start the Client Configuration Utility, it opens a welcome screen by default to let you quickly start a new project, open an existing project, or open the last project you had open. If you prefer, you can set the Client Configuration Utility to automatically create a new project or open the last project instead of displaying this welcome screen.



The Client Configuration Utility's welcome screen

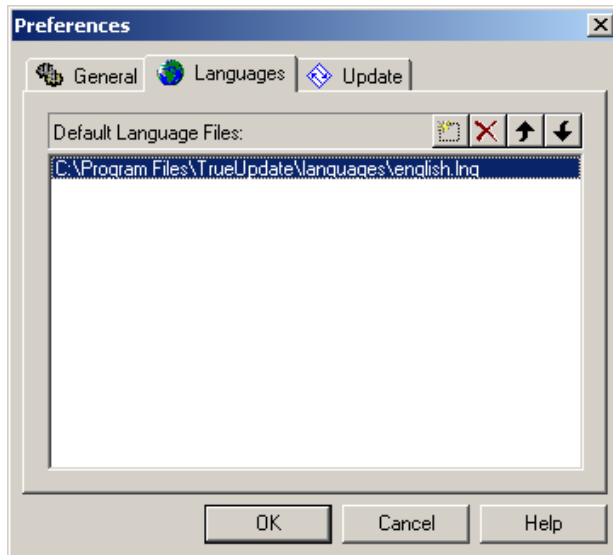
You can choose between the three options in the Startup section on the General tab of the *Preferences* screen.



The Startup section on the General tab of the Preferences screen

## Language Preferences

You can set language preferences for the Client Configuration Utility on the Languages tab of the *Preferences* screen. To access the *Preferences* screen, select **Edit | Preferences** from the menu.



The Languages tab of the Preferences screen

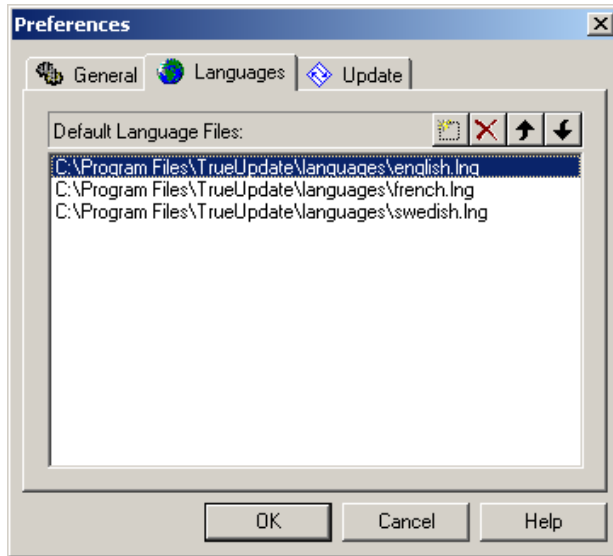
### ***Setting Default Language Files***

To change the list of languages that are added by default to every new project, edit the **Default Language Files** list on the Languages tab of the *Preferences* screen. The first language in this list will also be marked as the default language—in other words, the



## Chapter 4


---



default language check box on the Messages tab will be selected for the first item in the **Default Language Files** list by default.



These three languages will be added to every new project

To add a language to the list, press the **Insert Language** button (  ) or use the Insert hotkey. Then, enter the path to the language ( .lng ) file you want to add, or press the **Browse** button (  ) to browse for a language file.

To remove a language from the list, select the language you want to remove and press the **Remove Language** button (  ) or use the Delete hotkey.

To change the order of the languages in the list, select the language that you want to move, and then use the **Move Up** (  ) and **Move Down** (  ) buttons to reposition the language in the list.

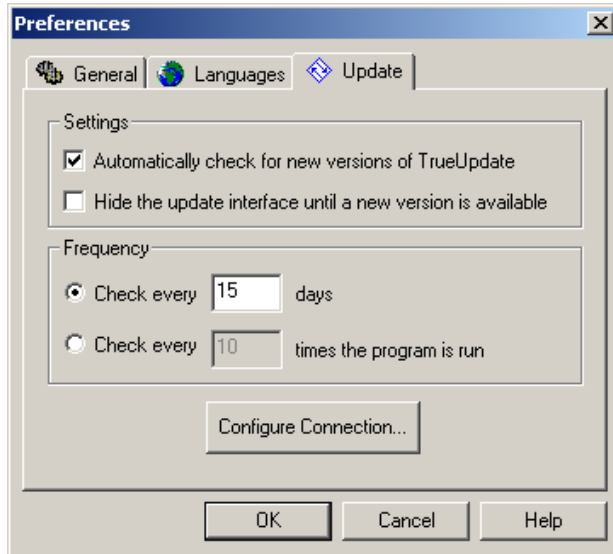
### NOTE



You can add languages to the current project on the Messages tab of the Client Configuration Utility.

## Update Preferences

You can set update preferences for the TrueUpdate software on the Update tab of the *Preferences* screen. To access the *Preferences* screen, select **Edit | Preferences** from the menu.



The Update tab of the Preferences screen

### NOTE



These settings are shared between the Client Configuration Utility and the Server File Editor.

### ***Automatically Checking for New Versions of TrueUpdate***

The TrueUpdate development software can automatically check the Internet for updates. Using the same technology that you can integrate into your software, TrueUpdate will check the Indigo Rose web site for new versions of itself.

To enable this feature, just select the **Automatically check for new versions of TrueUpdate** check box.

### ***Hiding the Update Interface Until a New Version is Available***

The **Hide the update interface until a new version is available** check box lets you control whether the TrueUpdate client interface is shown each time TrueUpdate checks the Internet for a new version. If the check box is selected, the client interface is only shown when a new version is available and an update is performed.

### ***Setting How Often TrueUpdate Checks for Updates***

The Frequency section of the Update tab allows you to specify how often TrueUpdate checks the Internet for new versions of itself. You can have TrueUpdate check every X number of times the Client Configuration Utility or Server File Editor is started, or have it check (upon starting TrueUpdate) if a minimum number of days have elapsed since the last time a check was performed.

### ***Configuring the TrueUpdate Connection Settings***

You can configure the connection settings TrueUpdate will use when it checks for an update by pressing the **Configure Connection** button.

## **User Tools**

You can add custom menu items to the Client Configuration Utility's **Tools** menu to start other programs that you use often during the design process. For instance, you might want to add a menu item to launch your favorite text editor.

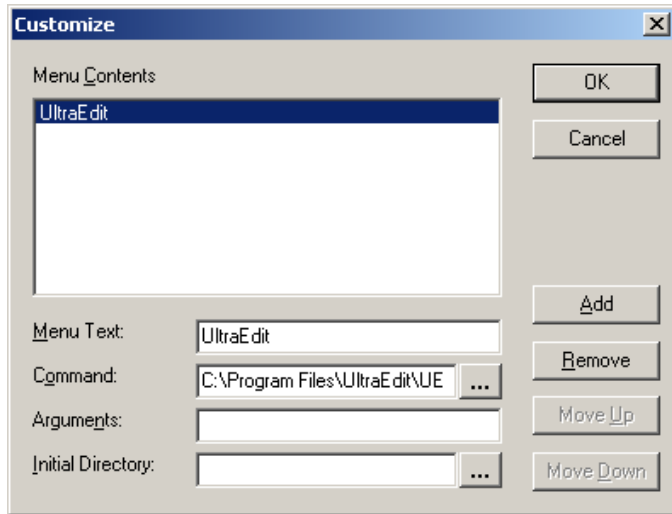
#### **NOTE**



Any custom tools you add to the Client Configuration Utility's **Tools** menu will also show up in the Server File Editor's **Tools** menu.

### ***Configuring User Tools***

To configure the custom items that will show up in the **Tools** menu, select **Tools | Configure User Tools** from the menu to open the *Configure User Tools* screen.



The Configure User Tools screen

You can use this screen to add tools to the menu, remove them from the menu, and change the order of the tools in the menu.



---

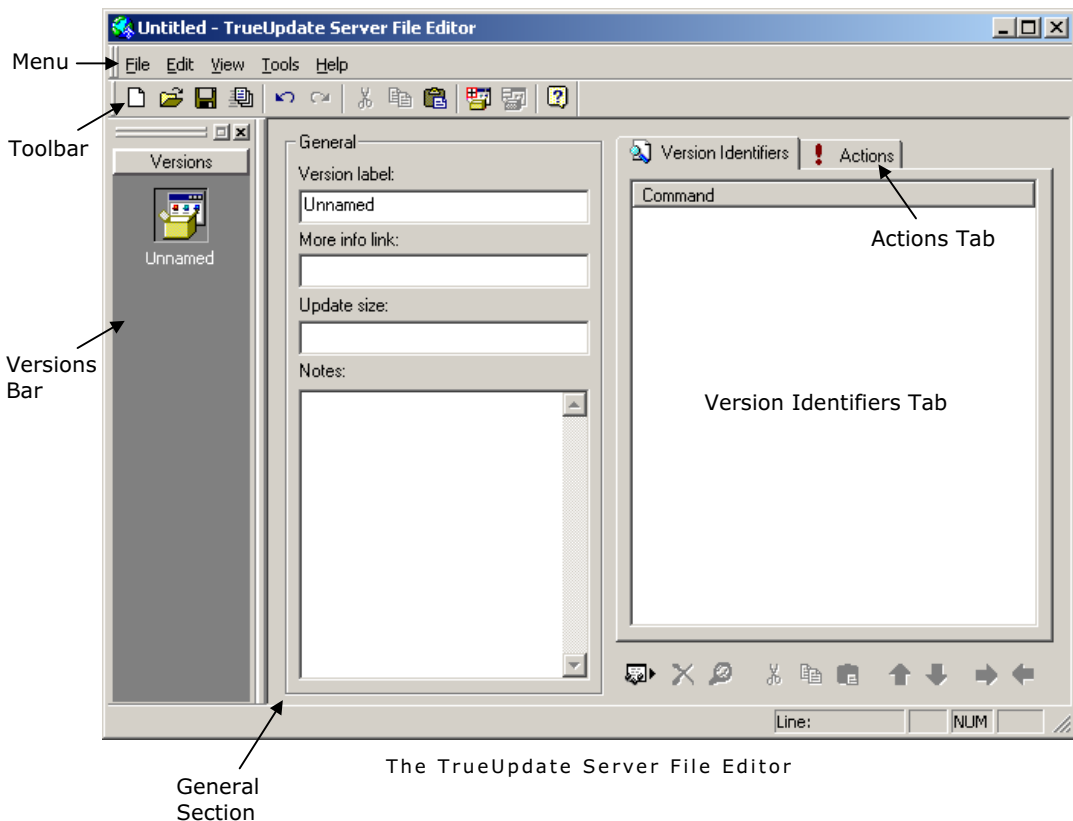
## Chapter 5

# The Server File Editor

---

The TrueUpdate Server File Editor is used to build and edit TrueUpdate server files. You use the Server File Editor to define the version identifiers that will identify specific versions of your product on the user's system, as well as the actions that can be performed for each version that is found.

This chapter will help you get familiar with the TrueUpdate Server File Editor as quickly as possible. It will provide you with a quick tour of the Server File Editor's interface, and introduce the tasks you can perform with this tool.



# Server Files

The TrueUpdate server file is a special data file containing a list of version identifiers and actions for each version of your software. Each list of version identifiers tells the TrueUpdate client how to identify a particular version of your software. Once a version is identified, the TrueUpdate client performs the actions that are listed for that version in the server file. By performing actions like downloading and executing patches or installation files, the TrueUpdate client can bring your user's software up to date.

### TIP



The name of the current server file is displayed in the TrueUpdate Server File Editor's title bar.

## *Creating a New Server File*

To create a new server file, select **File | New** from the menu or press the Ctrl+N hotkey. The new file replaces the current file in the Server File Editor. If you've made any changes to the current server file you will be asked if you'd like to save the changes before continuing.

## *Opening an Existing Server File*

To open an existing server file, select **File | Open** from the menu or press the Ctrl+O hotkey. This file will replace the current file in the Server File Editor. If you've made any changes to the current file you will be asked if you'd like to save the changes before continuing.

## *Saving the Current Server File*

To save the current server file, select **File | Save** from the menu or press the Ctrl+S hotkey. To save the file with a different filename, select **File | Save As**.

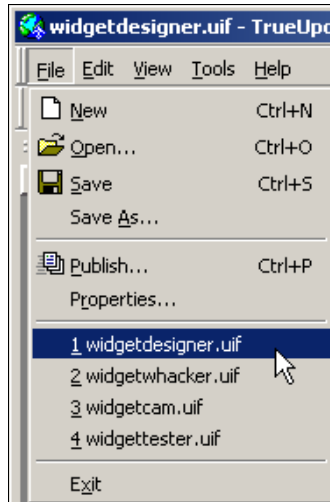
### NOTE



Server files have a .uif extension, which is short for "update information file."

## Reopening a Recent Server File

You can access the four most recently saved server files directly from the **File** menu.




Opening a recent server file

## Publishing the Current Server File

Publishing the server file means making it available to the TrueUpdate client at one or more of the server file locations. In other words, publishing the server file is simply putting it where the TrueUpdate client will be able to access it. Usually this involves uploading the server file to a web site or FTP server.

You can publish the server file using the Server File Editor, or, if you prefer, you can upload it manually using the FTP client of your choice. (If the server file is just going to be accessed across your LAN, you can "publish" it by copying it to the appropriate shared directory on the network.)

To publish the current server file:

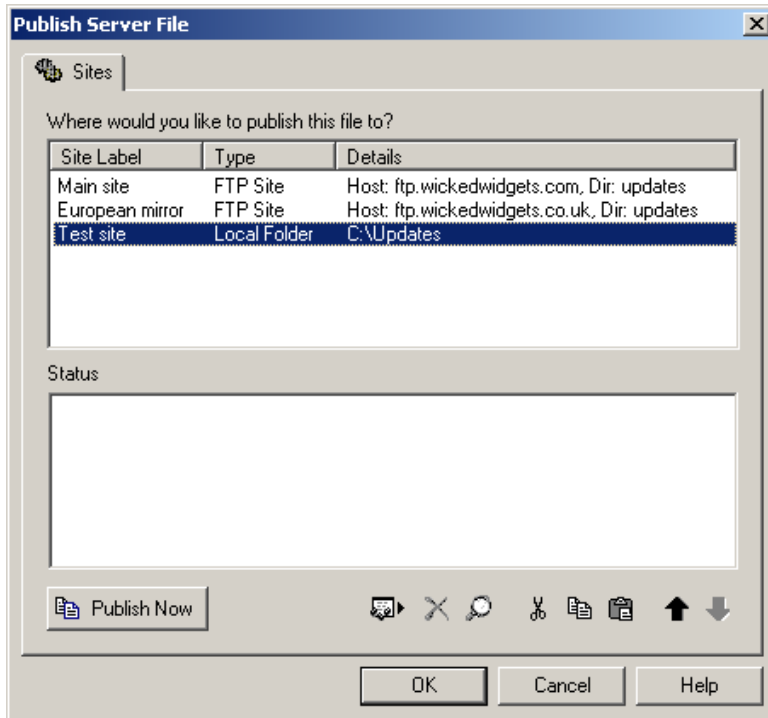
1. Select **File | Publish** from the menu, press the **Publish** button (  ) or press the Ctrl+U hotkey to save the current server file and open the *Publish Server File* screen.

### NOTE




You must save the server file before you can publish it. If you haven't saved the server file yet, you will be asked if you want to save it first.

The server file must also pass a validation test before it can be published. If any errors are found, the *Validation Report* screen will appear to explain why the server file can't be published. You'll need to correct the errors listed on that screen in order to successfully publish the server file.




The Publish Server File screen

2. If you don't have any publish sites listed on the *Publish Server File* screen, add one by pressing the **Add Publish Site** button (  ) or by pressing the Insert hotkey.

**SEE ALSO**

? See *Adding a Local Publish Site* on page 93 and *Adding an FTP Publish Site* on page 94 for more information on adding publish sites.

3. Select the site you want to publish the server file to and press the **Publish Now** button (  ).
4. Read the text in the **Status** display to verify that the file was published successfully. The text at the bottom of the **Status** display should read "Publish successful."



Example of a successful publish attempt

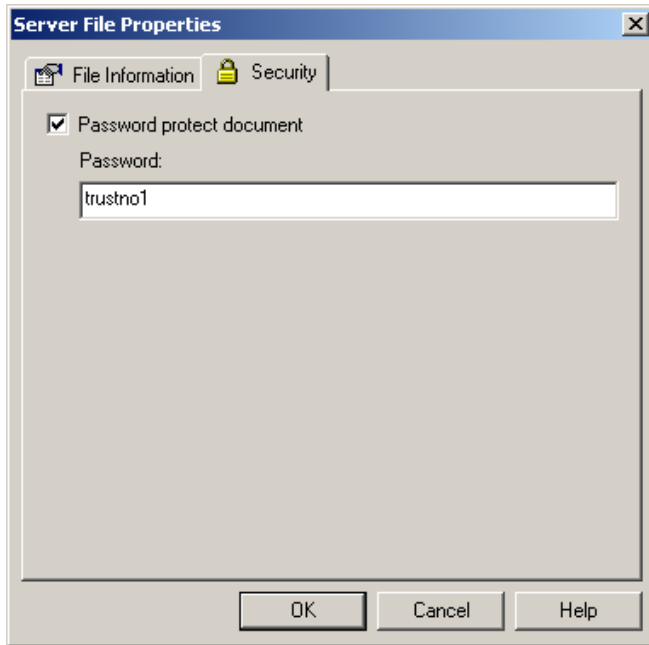
5. Press the **OK** button to exit the *Publish Server File* screen.

### ***Password Protecting the Current Server File***

Once a server file has been password protected, it cannot be opened without entering the correct password.

To password protect the current server file:

1. Select **File | Properties** from the menu to open the *Server File Properties* screen.
2. Select the Security tab.
3. Select the **Password protect document** check box.



The Security tab of the Server File Properties screen

4. Enter a password in the **Password** field.

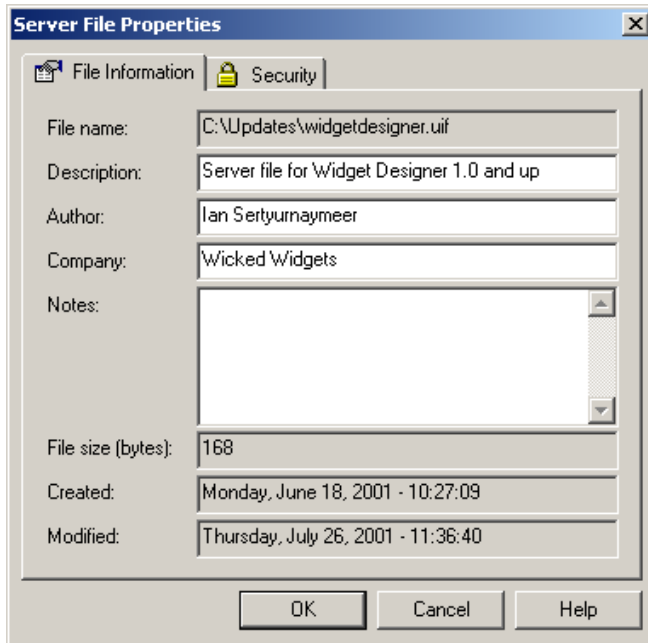
### IMPORTANT

! Remember this password! You won't be able to open the server file without it.

5. Press the **OK** button to close the *Server File Properties* screen.
6. Select **File | Save** or use the Ctrl+S hotkey to save the server file with the new password.

## Viewing and Editing Server File Properties

You can view and edit the server file properties by selecting File | Properties from the menu to open the *Server File Properties* screen.



The File Information tab of the Server File Properties screen

## Validating the Current Server File

Validating a server file involves testing it for obvious logic errors on the Version Identifiers and Actions tabs. The server file must pass a series of validation tests before it can be published.

You can initiate the server file validation tests at any time by selecting **Tools | Validate File** from the menu. TrueUpdate will perform the validation tests, and then open the *Validation Report* screen to inform you of the result. If any errors are found, they will be listed on the *Validation Report* screen.

# Publish Sites

A publish site is simply a place that you can upload server files to using the Server File Editor. You can have as many publish sites as you want; generally, you'll want to set up one publish site for every server file location that you'll be uploading a server file to.

The list of publish sites is shared between server files—all of the publish sites you add in the Server File Editor are stored in a single *publish sites data file*. By default this data file is `C:\Program Files\TrueUpdate\Data\pubsites.dat`, but you can change its location and filename on the General tab of the Server File Editor's *Preferences* screen.

### SEE ALSO



For more information on changing the path to the publish sites data file, see page 125.


There are two types of publish sites you can add: Local and FTP.

#### Local Publish Sites

A local publish site is a fully qualified path to a directory on a system you have access to. Local publish sites are typically used to test your TrueUpdate server file during development.

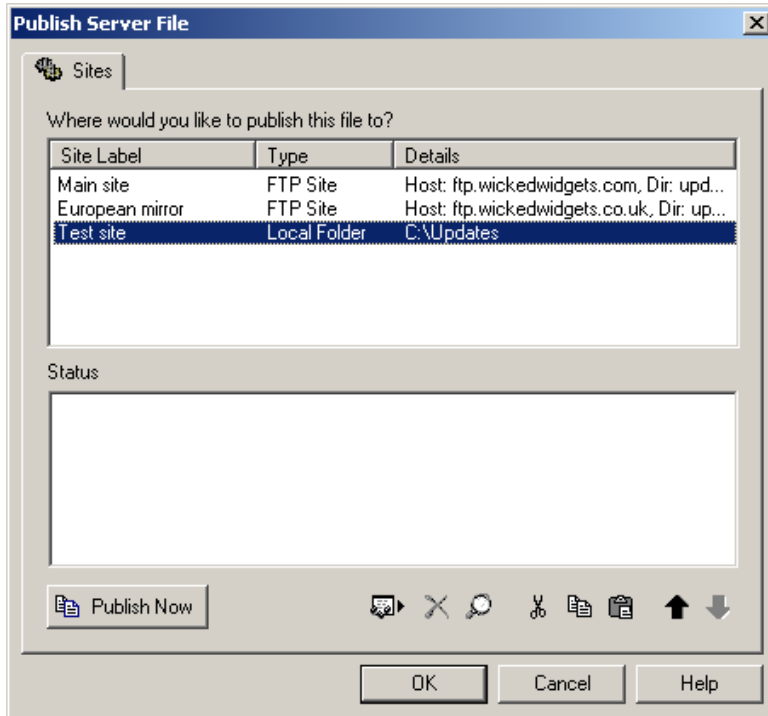
#### FTP Publish Sites

An FTP publish site consists of all the information needed to log into an FTP server account and upload a server file using the FTP protocol.

You can configure both kinds of publish sites on the *Publish Server File* screen. To get to the *Publish Server File* screen, select **File | Publish** from the menu, press the **Publish** button (  ), or press the Ctrl+U hotkey.

**NOTE**

The current server file is automatically saved and validated before the *Publish Server File* screen is opened. If you haven't saved the server file yet, you will be asked if you want to save it first. If any errors are found in the validation test, the *Validation Report* screen will appear; you'll need to correct the errors before you can access the *Publish Server File* screen.

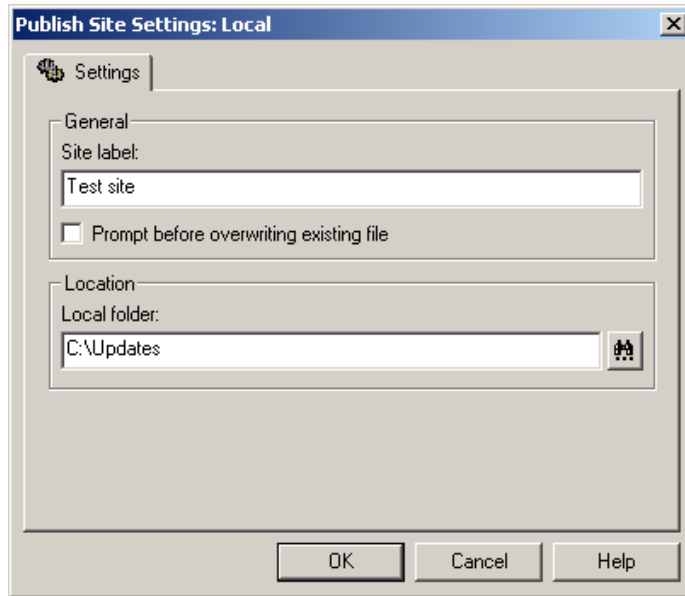


The Publish Server File screen

### ***Adding a Local Publish Site***

To add a local publish site:

1. Press the **Add Publish Site** button (  ) and select "Local" from the list that pops up. This will open the *Publish Site Settings: Local* screen.



The Publish Site Settings: Local screen

2. Enter a name for this local publish site in the **Site label** field. This name is only used to identify the publish site in the list on the *Publish Server File* screen; you can label the publish site with any name you want.
3. Select the **Prompt before overwriting existing file** check box if you want to be asked whether to overwrite the existing server file when a file with the same name exists at this publish site.
4. Enter the path to the folder you want the server file to be published to in the **Local folder** field. You can use a hard coded path like `c:\foo\mydir`, or you can use a UNC path like `\\servername\directory`.

### ***Adding an FTP Publish Site***

To add a local publish site:

1. Press the **Add Publish Site** button (  ) and select "FTP" from the list that pops up. This will open the *Publish Site Settings: FTP* screen.

**Publish Site Settings: FTP**

Settings

General

Site label:  
Main site

Prompt before overwriting existing file

FTP Site

Host name: ftp.wickedwidgets.com Remote site folder: updates

Account/Connection

FTP user name: llama5 FTP password: wtfiml33t

FTP account: Port: 21

Connection timeout (secs): 20

Use passive/firewall mode

OK Cancel Help

The Publish Site Settings: FTP screen

2. Enter a name for this FTP publish site in the **Site label** field. This name is only used to identify the publish site in the list on the *Publish Server File* screen; you can label the publish site with any name you want.
3. Select the **Prompt before overwriting existing file** check box if you want to be asked whether to overwrite the existing server file when a file with the same name exists at this publish site.
4. Enter the name or IP address of the FTP server in the **Host name** field, and the destination path for the server file in the **Remote site folder** field.


5. If the Server File Editor will need to use a user name and password to access the FTP server, you can change the values in the **FTP user name** and **FTP password** fields.

The **FTP account** field lets you specify an account name for servers that require it. (Some servers require a login name, password, and account name instead of just a login name and password.) Normally, though, you won't need to enter anything in this field.

6. You can also adjust the **Connection timeout** and **Port** values as required.
7. If the FTP server you're uploading to doesn't support passive mode connections, you can clear the **Use passive/firewall mode** check box. Passive mode is required if you're connecting to the Internet through a proxy server. Most servers support passive mode connections, though, so it's generally best to leave this setting enabled.


### ***Removing a Publish Site***

To remove a publish site from the list:

1. Select the site you want to remove.
2. Press the **Remove Publish Site** button (  ) or press the Delete hotkey.

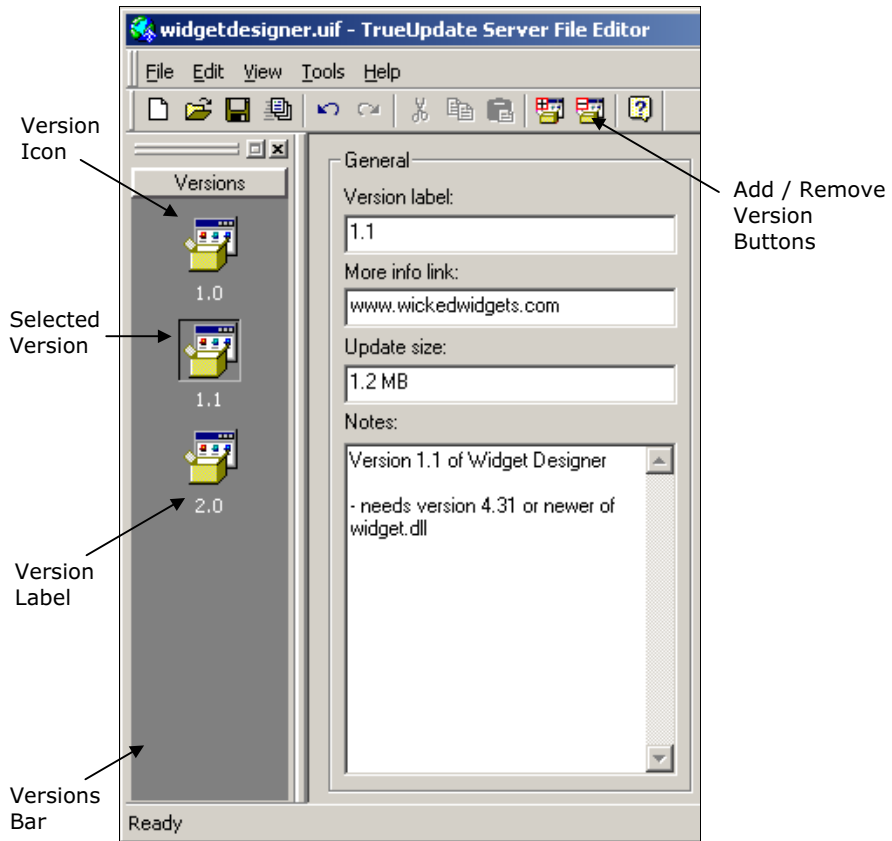
### ***Editing a Publish Site***

To modify a publish site:

1. Select the site you want to edit.
2. Press the **Properties** button (  ) or press the Ctrl+P hotkey. This will open the appropriate *Publish Site Settings* screen where you can edit the settings for the site you selected.

## Versions Bar

The versions bar allows you to organize everything in the server file according to what version (or versions) it deals with.



Three versions on the versions bar

Each icon on the versions bar represents one version (or group of related versions) of your product. A label appears under each icon to identify the version it represents.

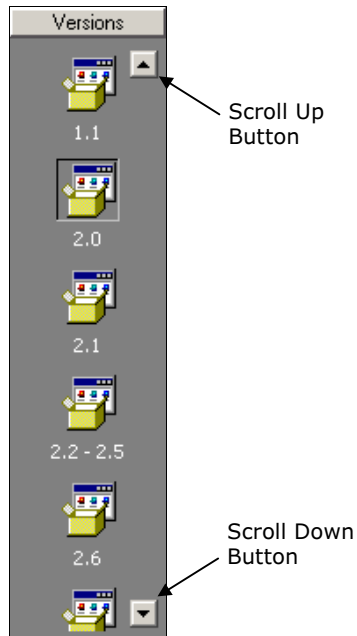
Each version on the bar has its own list of version identifiers and actions. You can edit these lists by selecting an icon on the bar, and then selecting either the Version Identifiers or the Actions tab on the right-hand side of the Server File Editor.

## Chapter 5

---

You can also edit the label, more info link, and update size string for the currently selected version in the General section of the main Server File Editor screen.

As you add versions to a server file, you can add more versions than can fit on the versions bar. When this happens, you can use the **Scroll Up** (▲) and **Scroll Down** (▼) buttons to scroll the versions bar up or down.



The scroll up and scroll down buttons

### ***Adding a Version***

To add a version, select **Edit | Add Version** from the menu, press the **Add Version** button (📄+) or use the Ctrl+K hotkey.

#### **NOTE**



The number of versions you can add to the Server File Editor is limited only by your computer's memory.


**TIP**



If you use a full-history patching product like Indigo Rose's Visual Patch, you only really ever need two entries on the versions bar...one for the current version, and one for everything else. The "everything else" entry just downloads the full-history patch, which can update any version to the latest release, and runs it.

## ***Removing a Version***

To remove a version:

1. Select the version's icon in the versions bar.
2. Select **Edit | Remove Version** from the menu, press the **Remove Version** button (  ) or use the Ctrl+L hotkey.

**TIP**



If you make a mistake, you can undo this action by selecting **Edit | Undo** from the menu, or by using the Ctrl+Z hotkey.

## ***Copying an Existing Version***

You can copy an existing version (along with all its version identifiers and actions) and paste it back onto the versions bar as a new version. To do so:

1. Select the version you want to copy on the versions bar.
2. Select **Edit | Copy** from the menu or press Ctrl+C to copy the version into the clipboard.
3. Select **Edit | Paste** from the menu or press Ctrl+V to paste the version from the clipboard back onto the versions bar.

The new version will be added to the end of the versions bar.

### *Changing the Version Search Order*

The order of the icons on the versions bar determines the version search order that the client will follow.

At run time, the client needs to identify what version of your product is installed on the user's system. It performs this "version search" by applying the version identifiers for each version in the server file. The order of the icons on the versions bar determines the order that the version identifiers are applied in.

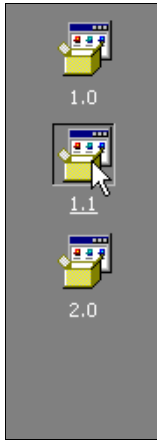
The identifiers for the version at the top of the versions bar are applied first. If the version on the user's system doesn't meet the criteria set by those identifiers, the identifiers for the next version on the bar are applied. The version search continues until either a matching set of identifiers is found, or the last version on the versions bar is reached.

#### **NOTE**

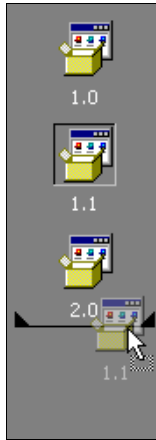


The order you use for the icons depends on how you design your version identifiers. If you use a full-history patching tool like Indigo Rose's Visual Patch, you'll want the client to try identifying the current version first. That way, you can identify all the other (older) versions with another version on the versions bar. This "catch-all" version would have no version-specific identifiers at all so that any previous version would match. In this case, the current version would have to be searched for first, to prevent the "catch-all" version from falsely identifying an up-to-date release as an older version.

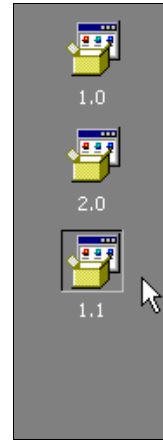
You can rearrange the order of the version icons by using the left mouse button to drag an icon to another position in the versions bar.



Press the left mouse button



Drag the version icon



Release the left mouse button

Dragging version 1.1 to the bottom of the versions bar

### ***Labeling a Version***

Each icon on the versions bar has a version label that you can change. You can label your versions with any text you find helpful.

To edit a version label:

1. Select an icon on the versions bar.
2. Edit the text in the Version label field. You can use any text you want in the label, and the label can be as long as you want.

#### **NOTE**



The version labels are only used at design time; your users will never see the labels you assign in the Server File Editor.

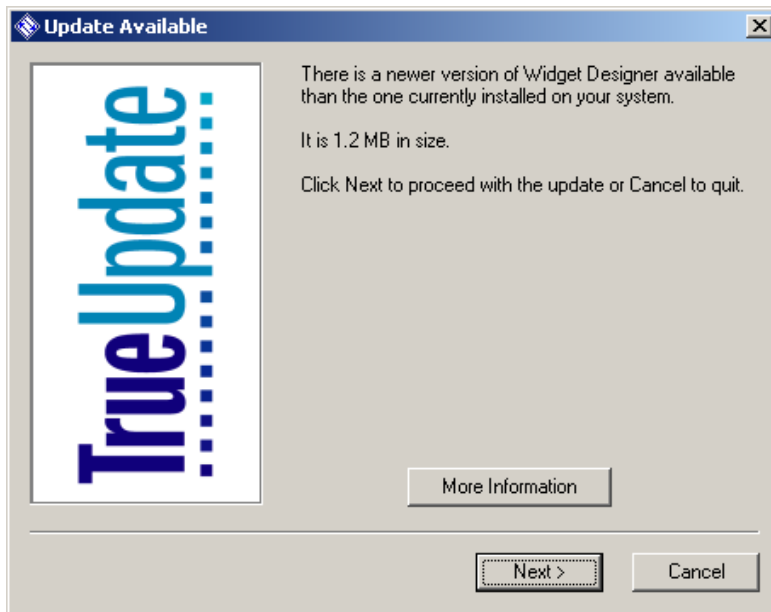
### TIP



You should use descriptive names for your labels. "Casey" and "Finnigan" are perfectly good names but they are probably less appropriate than something like "1.2," "old stuff" or "latest release."

### ***Enabling the More Information Button***

The **More Information** button allows you to provide additional information to your users when a new version is available to them. Your users can click on the **More Information** button to open a web page—or any other kind of document they have access to—with any information you want to provide. For instance, a user with access to your LAN could use the button to open a Word document located on one of your file servers.



The Update Available screen sporting a More Information button

The **More Information** button appears on the *Update Available* screen when a version is found and its **More info link** field in the Server File Editor is not empty. In other words, if you enter a URL or path in the **More info link** field for a given version in the Server File Editor, a **More Information** button will appear on the *Update Available* screen when that

version is found on the user's system. Whether you provide a **More Information** button for a given version is entirely up to you.

**TIP**

You could make the **More Information** button link to your home page instead, and rename the button from "More Information" to something like "Visit our web site." (You can change the button text by editing the MSG\_BUTTON\_MOREINFO message in the Client Configuration Utility.)

To enable the **More Information** button:

1. Select a version on the versions bar. (Each **More Information** button is version-specific.)
2. Enter the URL to a web page, or the path to a document the user has access to, in the **More info link** field.

**NOTE**

If you use a URL, the http:// prefix is optional. For instance, [www.indigorse.com](http://www.indigorse.com) and <http://www.indigorse.com> both work equally well.

**IMPORTANT**

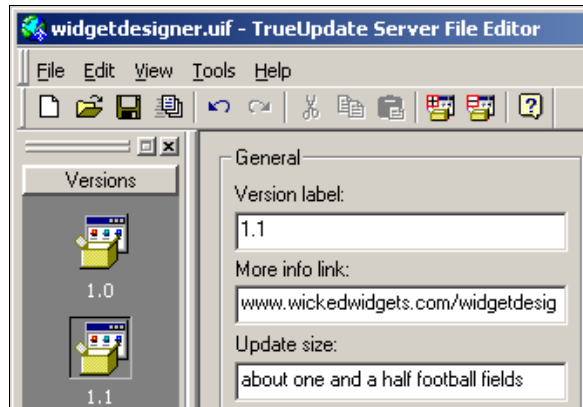
The **More Information** button only appears on the *Update Available* screen. The user won't be able to use the **More Information** button if the *Update Available* screen is not shown.

**TIP**

You can use a different "more info" URL for each version on the versions bar.

### *Providing an Update Size*

The **Update size** field allows you to specify the text that appears on the *Update Available* screen to tell the user the size of the file (or files) that the TrueUpdate client will download.



Entering a size in the Update size field

Any text you enter in the **Update size** field is assigned to a built-in variable named %UpdateSize% at run time. This variable is used in the message that gets shown on the *Update Available* screen.

#### **NOTE**



The value you enter in the **Update size** field is for informational purposes only. It has nothing to do with disk space calculations, progress bars, etc.

To provide an update size:

1. Select an icon on the versions bar. (There is a separate update size for each version.)
2. Enter the text you want the user to see on the *Updates Available* screen in the **Update size** field.

**TIP**



You can use any number format you want for the update size—whether you use "512 kilobytes" or "512 KB" or "0.5 MB" is entirely up to you.

## ***Making Notes***

You can enter notes about the currently selected version in the **Notes** field of the Server File Editor.

**NOTE**



These notes are only used at design time; your users will never see the notes you enter in the Server File Editor.

To enter notes about a version:

1. Select the version's icon on the versions bar. (Notes are version-specific.)
2. Enter your comments or remarks in the **Notes** field.

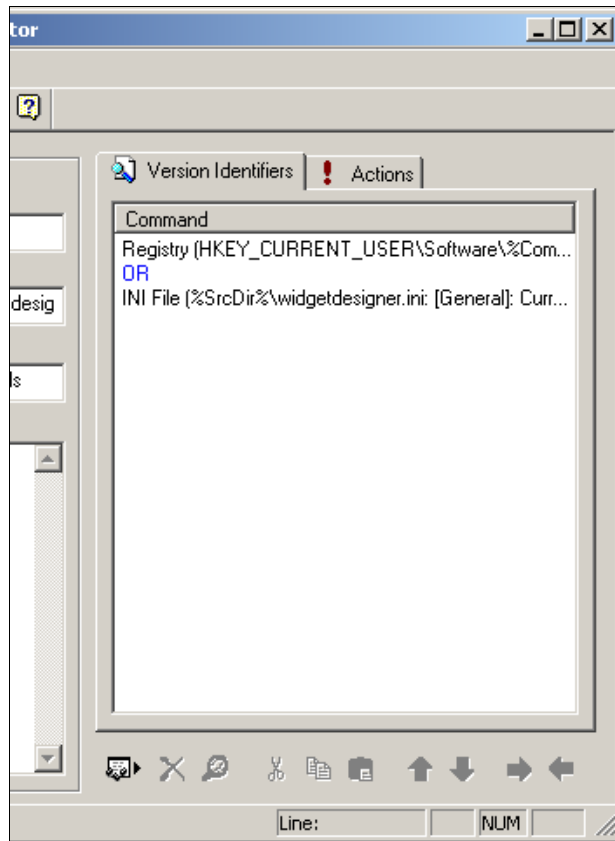
## **Version Identifiers**

Version identifiers are special rules or criteria used to identify a version of your software installed on the user's system. Each version identifier acts as a criterion that must be met in order for a version to be identified. You can provide as many version identifiers as you need to identify any given version. A version will only be considered "identified" when all of the criteria set by the version identifiers are met.

There are four different ways you can identify a version: by reading a value from the Registry, reading a value from an INI file, comparing file versions, and comparing CRC values.

You can edit the list of version identifiers for the currently selected version on the Version Identifiers tab of the Server File Editor.


To get to the Version Identifiers tab, select **View | Version Identifiers** from the menu, press the Ctrl+1 hotkey, or just click on the Version Identifiers tab.



The Version Identifiers tab

### ***Reading a Value from the Registry***

To identify a version by reading a value from the Registry:

1. Press the **Add Version Identifier** button (  ) and select "Read from Registry" from the list that pops up. This will open the *Version Identifier Properties: Read from Registry* screen.

**TIP**

You can also select the Version Identifiers tab and press the Insert hotkey, or right-click on the Version Identifiers tab and select **Add** from the context menu.

The screenshot shows a dialog box titled "Version Identifier Properties: Read from Registry". It has a close button in the top right corner. The dialog contains the following fields and controls:

- Main key:** A dropdown menu with "HKEY\_CURRENT\_USER" selected.
- Sub key:** A text box containing "Software\%CompanyName%\%ProductName%".
- Value name:** A text box containing "Version".
- Operator:** A dropdown menu with "Equal to (=)" selected.
- Compare to:** A text box containing "1.1".

At the bottom of the dialog are three buttons: "OK", "Cancel", and "Help".

The Version Identifier Properties: Read from Registry screen

2. Select the main key (such as HKEY\_LOCAL\_MACHINE) where the registry value can be found using the **Main key** drop-down list.
3. Enter the path to the registry key in the **Sub key** field.
4. Enter the name of the registry value you want to read in the **Value name** field. If you leave the **Value name** field blank, the Registry key's *Default* value will be read.
5. Select the operator you want to use in the **Operator** drop-down list. This operator will be used to compare the value in the Registry with the value you provide in the **Compare to** field.
6. Enter the value you want to compare the Registry value to in the **Compare to** field.


### TIP

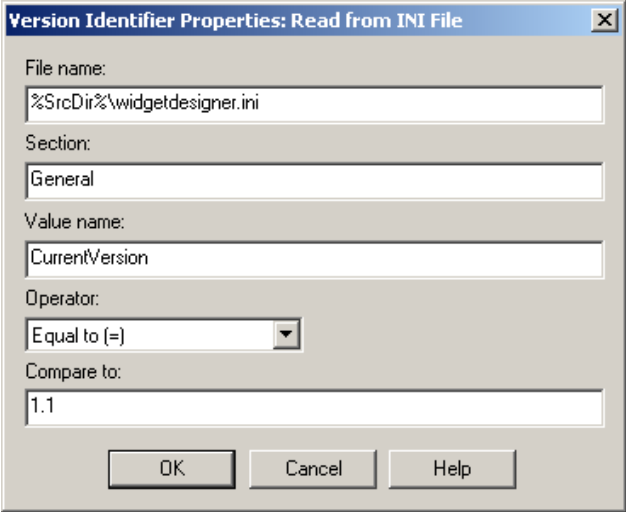


Always have your installation program write your software's version number to the registry. That way you can identify a version with a single identifier that compares the registry key's value to the appropriate version number.

### ***Reading a Value from an INI File***

To identify a version by reading a value from an INI file:

1. Press the **Add Version Identifier** button (  ) and select "Read from INI File" from the list that pops up. This will open the *Version Identifier Properties: Read from INI File* screen.



The screenshot shows a dialog box titled "Version Identifier Properties: Read from INI File". It contains the following fields and controls:

- File name:** A text box containing the path `%SrcDir%\widgetdesigner.ini`.
- Section:** A text box containing the text "General".
- Value name:** A text box containing the text "CurrentVersion".
- Operator:** A dropdown menu currently set to "Equal to (=)".
- Compare to:** A text box containing the text "1.1".
- At the bottom, there are three buttons: "OK", "Cancel", and "Help".

The Version Identifier Properties: Read from INI File screen

2. Enter the path and filename of the INI file in the **Filename** field.

3. Enter the name of the INI file section in the **Section** field. This is the section (e.g. "[General]") of the INI file where the value can be found.

```
[General] ← section  
CurrentVersion=1.0
```

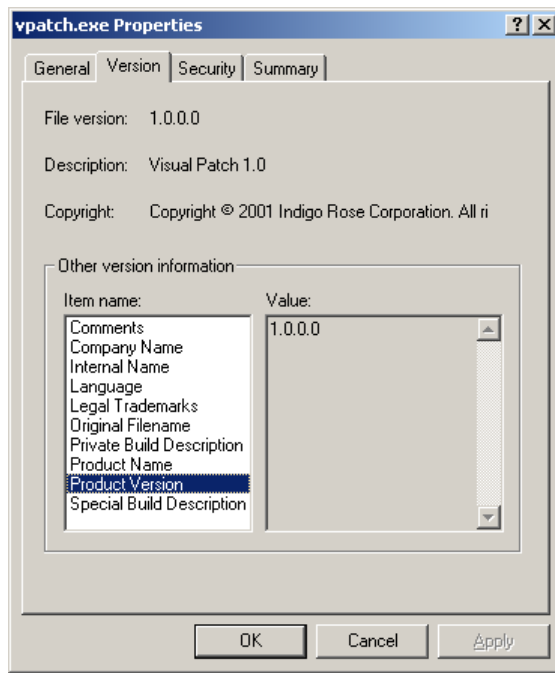
4. Enter the name of the value you want to read in the **Value name** field. The value associated with this name in the INI file will be compared to the value you enter in the **Compare to** field.

```
[General]  
CurrentVersion=1.0 ← value  
                   ↙  
                   value name
```

5. Select the operator you want to use in the **Operator** drop-down list. This operator will be used to compare the value in the INI file with the value you provide in the **Compare to** field.
6. Enter the value you want to compare the INI file value to in the **Compare to** field.


### ***Comparing File Versions***

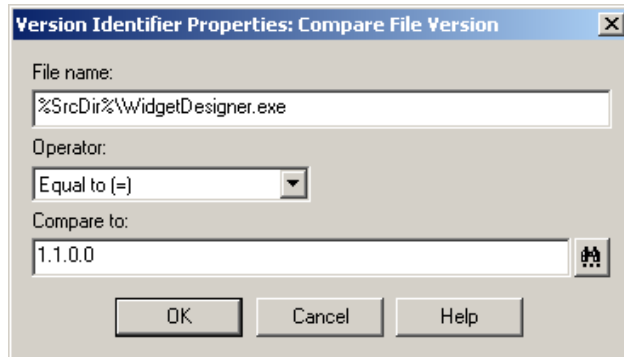
An executable file can contain version information embedded in the resource section of its file header. This is the same information you can access in Windows by right-clicking on an executable file (such as an EXE, DLL, VxD, etc.) and selecting **Properties**.




The version information for Visual Patch 1.0

To identify a version using a file's version information string:

1. Press the **Add Version Identifier** button (  ) and select "Compare File Version" from the list that pops up. This will open the *Version Identifier Properties: Compare File Version* screen.




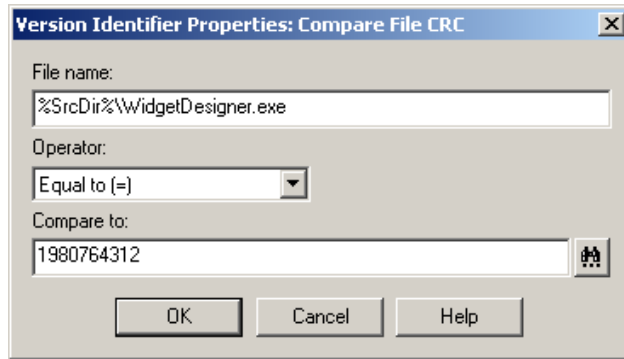
The Version Identifier Properties: Compare File Version screen

2. Enter the path and filename of the file that has the version resource information in the **File name** field. This can be any executable file (EXE, DLL, DRV, OCX etc.) that has a version information string in its resource section.
3. Select the operator you want to use in the **Operator** drop-down list. This operator will be used to compare the version information in the file with the value you provide in the **Compare to** field.
4. Enter the value you want to compare the version information to in the **Compare to** field. You can also press the **Browse** button (  ) to read the version information from a file using the *Read File Version* screen.


## Comparing CRC Values

To identify a version using a file's CRC value:

1. Press the **Add Version Identifier** button (  ) and select "Compare File CRC" from the list that pops up. This will open the *Version Identifier Properties: Compare File CRC* screen.



The Version Identifier Properties: Compare File CRC screen

2. Enter the path and filename of the file whose CRC value you want to calculate in the **Filename** field.
3. Select the operator you want to use in the **Operator** drop-down list. This operator will be used to compare the CRC value of the file with the value you provide in the **Compare to** field.
4. Enter the value you want to compare the file's CRC value to in the **Compare to** field. You can also press the **Browse** button (  ) to select a file using the *Calculate File CRC* screen, and its CRC value will be entered in the **Compare to** field automatically.

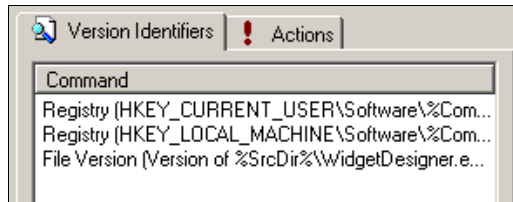
### SEE ALSO



For more information on CRC values, see page 24.

## Using OR Operators

Normally, consecutive identifiers on the Version Identifiers tab are ANDed together. In other words, they form a group of version identifiers that must be satisfied as a whole. Whenever two or more version identifiers are listed one after the other, they form one big multi-part criterion that must be met.



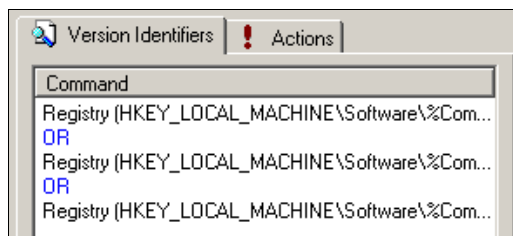
These three identifiers are ANDed together—the version will only be identified if all three identifiers are met

You can separate identifiers into multiple groups by using OR operators. Each group will consist of one or more version identifiers that are automatically ANDed together.

For example, let's say you want to use the same actions to update versions 1, 2 and 3 of your software. You could set up three separate versions on the versions bar, but any changes you made to the actions for one version would have to be duplicated for the other two—increasing the potential for human error.

A better solution would be to use a single icon on the versions bar to represent all three versions (perhaps labeling it "1-3"). This way, you can share one set of actions between the three versions, and only have one set of actions to maintain. All you need is a way to tell TrueUpdate to perform those actions if version 1, 2 or 3 are identified.

Assuming you can identify each version with a single Read from Registry identifier, you would simply separate the three Read from Registry identifiers with OR operators. Instead of one group of three identifiers, you would now have three groups, each consisting of a single identifier.




These three identifiers are ORed together—the version will be identified if any of these identifiers are met

### NOTE



You can separate groups consisting of multiple identifiers with OR operators as well.


To add an OR operator:

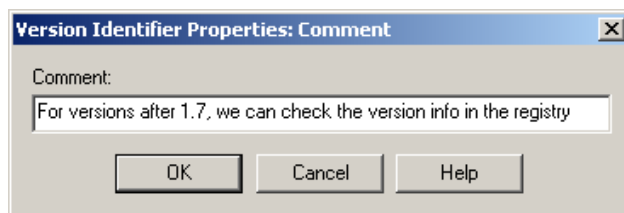
1. Select the line you want the OR operator to be inserted at.
2. Press the **Add Version Identifier** button (  ) and select "OR Operator" from the list that pops up.

### ***Adding Comments***

You can add comments on the Version Identifiers tab to make your lists of identifiers easier to maintain (and easier to understand). Comments are shown in a different color, and have no effect on the update process.

To add a comment:

1. Select the line you want the comment to be inserted at.
2. Press the **Add Version Identifier** button (  ) and select "Comment" from the list that pops up. This will open the *Version Identifier Properties: Comment* screen.



The Version Identifier Properties: Comment screen

3. Enter your comment in the **Comment** field.



**TIP**

You can use blank comments to add whitespace between identifiers on the Version Identifiers tab.

### ***Rearranging the Version Identifiers***

You can move individual version identifiers up and down on the Version Identifiers tab relative to other version identifiers. This is primarily useful when arranging the identifiers into groups separated by OR operators.

To change the order of the version identifiers for the currently selected version:

1. Select the version identifier you want to move.
2. Use the **Move Up** (  ) and **Move Down** (  ) buttons or the Ctrl+Up and Ctrl+Down hotkeys to reposition it in the list.

**NOTE**

The order of the version identifiers doesn't change the version search order. To change the order that versions are searched for, you need to reposition the icons on the versions bar.

### ***Cutting, Copying and Pasting Version Identifiers***

You can cut, copy and paste version identifiers within the same version, between versions, and even between server files. (When you cut or copy version identifiers, they are placed in the Windows clipboard. You can open a different server file or even switch to another instance of the Server File Editor to copy identifiers from one server file to another.)

To cut a version identifier and place it in the clipboard:

1. Select the version identifier you want to cut.
2. Select **Edit | Cut** from the menu, or use the Ctrl+X hotkey.

## Chapter 5

---

To copy a version identifier and place it in the clipboard:

1. Select the version identifier you want to copy.
2. Select **Edit | Copy** from the menu, or use the Ctrl+C hotkey.

To paste a version identifier from the clipboard onto a Version Identifiers tab:

1. Select the line on the Version Identifiers tab where you want the new identifiers to be inserted.
2. Select **Edit | Paste** from the menu, or use the Ctrl+V hotkey.


### TIP



You can hold a Ctrl or Shift key down to select multiple version identifiers when preparing to cut or copy them.

## ***Removing Version Identifiers***

To remove a version identifier from the Version Identifiers tab:

1. Select the version identifier you want to remove.
2. Press the **Remove Version Identifier** button (  ), use the Delete hotkey, or right-click on the version identifier and select **Remove** from the context menu.

## ***Adding Indentation to Version Identifiers***


You can add indentation to version identifiers to help make the list of identifiers easier to read. This is primarily useful to help set groups of identifiers apart when using OR operators.

### NOTE




Indentation has no effect on how the version identifiers are applied at run time.

To indent a block of version identifiers:

1. Select the version identifiers you want to indent.
2. Press the **Increase Indent** button (  ), use the Ctrl+H hotkey, or right-click on the version identifiers and select **Increase Indent** from the context menu.

### ***Removing Indentation from Version Identifiers***

To remove a level of indentation from a block of version identifiers:

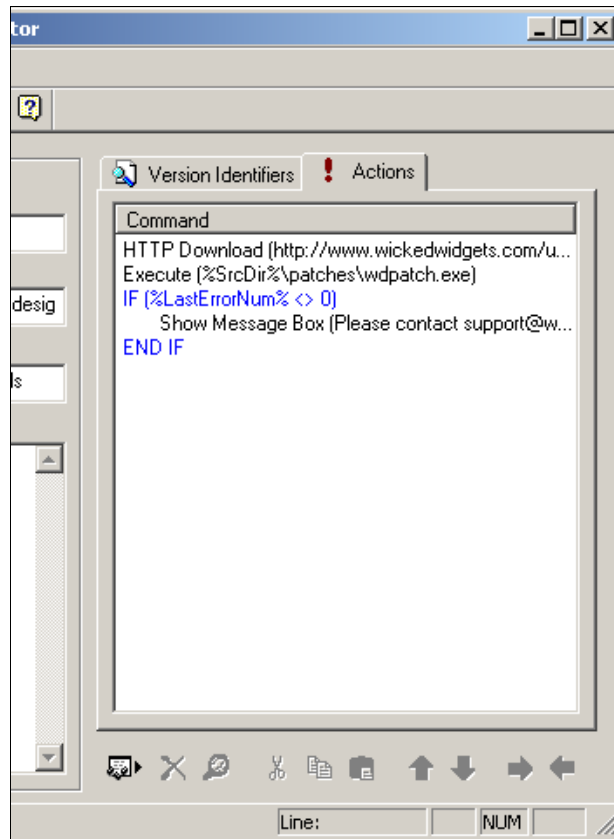
1. Select the version identifiers that you want to remove a level of indentation from.
2. Press the **Decrease Indent** button (  ), use the Ctrl+G hotkey, or right-click on the version identifiers and select **Decrease Indent** from the context menu.

## **Actions**

Actions are the instructions that tell the TrueUpdate client what to do once a version is identified. There is a separate list of actions for each version in the server file.

A wide variety of actions are available. In addition to Internet functions like HTTP downloads and FTP transfers, you can perform file operations, open and close programs, send email, submit data to web forms and CGI scripts, work with variables, read and write to the Registry and INI files, manipulate strings, and show custom dialog boxes to the user. Special control structure actions provide advanced features like IF blocks, WHILE loops, labels, and GOTO jumps. Optional design actions allow you to insert maintenance-friendly comments and whitespace, too.


You can edit the list of actions for the currently selected version on the Actions tab of the Server File Editor. To get to the Actions tab, select **View | Action** from the menu, press the Ctrl+2 hotkey, or just click on the Actions tab.

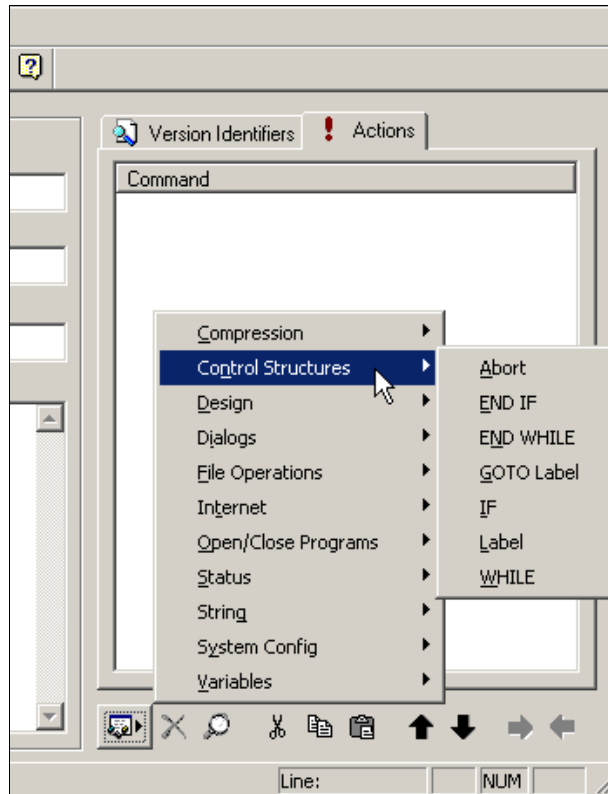


The Actions tab

### ***Adding Actions***

To add an action:

1. Select the line on the Actions tab where you want the action to be inserted.
2. Press the **Add Action** button (  ), use the Insert hotkey, or right-click and select **Add** from the context menu. This will open the list of action categories right next to the **Add Action** button.



The list of actions for the Control Structures category


3. Select the category that the action you want to add belongs to.
4. Select the action you want to add from the list that pops up. This will open the appropriate *Action Properties* screen.
5. Edit the properties for the action.

**SEE ALSO**

? For more information on the actions you can add, see *Actions* in the Command Reference. For a complete list of the actions, see page 159 in this User's Guide.

### ***Editing Actions***



To edit an action's properties:

1. Select the action you want to edit.
2. Press the **View Properties** button (  ), use the Ctrl+P hotkey, or right-click and select **Properties** from the context menu. This will open the *Action Properties* screen for the action you selected.
3. Edit the properties for the action.

### ***Changing the Order Actions are Performed In***

Actions are performed in the same order as they're listed on the Actions tab. The action at the top of the Actions tab is performed first, and the action at the bottom of the tab is performed last.

To change the order of the actions for the currently selected version:

1. Select the action you want to move.
2. Use the **Move Up** (  ) and **Move Down** (  ) buttons or the Ctrl+Up and Ctrl+Down hotkeys to reposition it in the list.

### ***Adding Indentation to Actions***

You can add indentation to help make the list of actions easier to read. This is especially useful to help set blocks of actions apart when using actions like IF and END IF.


#### **NOTE**



Indentation has no effect on how the actions are performed at run time. Actions have the same effect whether they're indented or not.


To indent a block of actions:

1. Select the actions you want to indent.

2. Press the **Increase Indent** button (  ), use the Ctrl+H hotkey, or right-click on the actions and select **Increase Indent** from the context menu.

## ***Removing Indentation from Actions***

To remove a level of indentation from a block of actions:

1. Select the actions you want to remove a level of indentation from.
2. Press the **Decrease Indent** button (  ), use the Ctrl+G hotkey, or right-click on the actions and select **Decrease Indent** from the context menu.

## ***Cutting, Copying and Pasting Actions***

You can cut, copy and paste actions within the same version, between versions, and even between server files.

### **NOTE**



When you cut or copy actions, they are placed in the Windows clipboard. You can open a different server file or even switch to another instance of the Server File Editor to copy actions from one server file to another.

To cut an action and place it in the clipboard:

1. Select the action you want to cut.
2. Select **Edit | Cut** from the menu, or use the Ctrl+X hotkey.

To copy an action and place it in the clipboard:

1. Select the action you want to copy.
2. Select **Edit | Copy** from the menu, or use the Ctrl+C hotkey.

To paste an action from the clipboard onto the Actions tab:

1. Select the line on the Actions tab where you want the action to be inserted.

2. Select **Edit | Paste** from the menu, or use the Ctrl+V hotkey.


### TIP



You can hold a Ctrl or Shift key down to select multiple actions when preparing to cut or copy them.

## ***Removing Actions***

To remove an action from the Actions tab:

1. Select the action you want to remove.
2. Press the **Remove Action** button (  ), use the Delete hotkey, or right-click on the action and select **Remove** from the context menu.

## ***Exporting Actions***

You can build a library of often-used actions by exporting them to TrueUpdate Actions Archive (.tua) files. This also makes it easier to share action "scripts" with other users.

To export actions from the Actions tab:

1. Select the actions you want to export.
2. Select **Edit | Export Actions** from the menu, use the Ctrl+E hotkey, or right-click on the actions and select **Export Actions** from the context menu.
3. Enter a name for the TrueUpdate Actions Archive (.tua) file on the *Save As* screen and press the **Save** button.

## ***Importing Actions***

To import actions from a TrueUpdate Actions Archive (.tua) file:

1. Select the line on the Actions tab where you want the actions to be inserted.

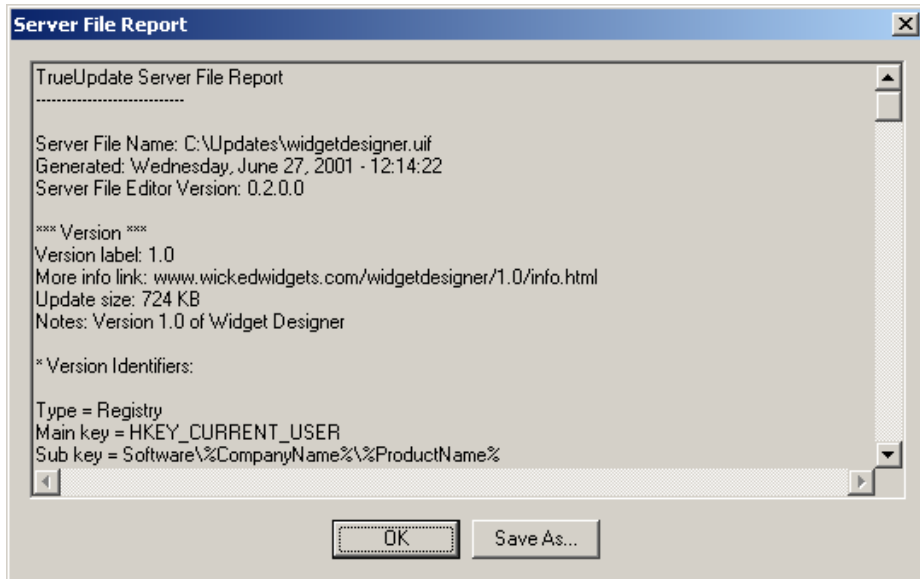
2. Select **Edit | Import Actions** from the menu, use the Ctrl+T hotkey, or right-click on the actions and select **Import Actions** from the context menu.
3. Use the *Open* screen to select the TrueUpdate Actions Archive (.tua) file containing the actions you want to import.

## Reports

This feature allows you to generate a report with detailed information on each version in the server file, including all version identifiers and actions.

### *Generating a Report*

You can generate a report on the current server file at any time by selecting **Tools | Generate Report** from the menu. This will open the *Server File Report* screen.

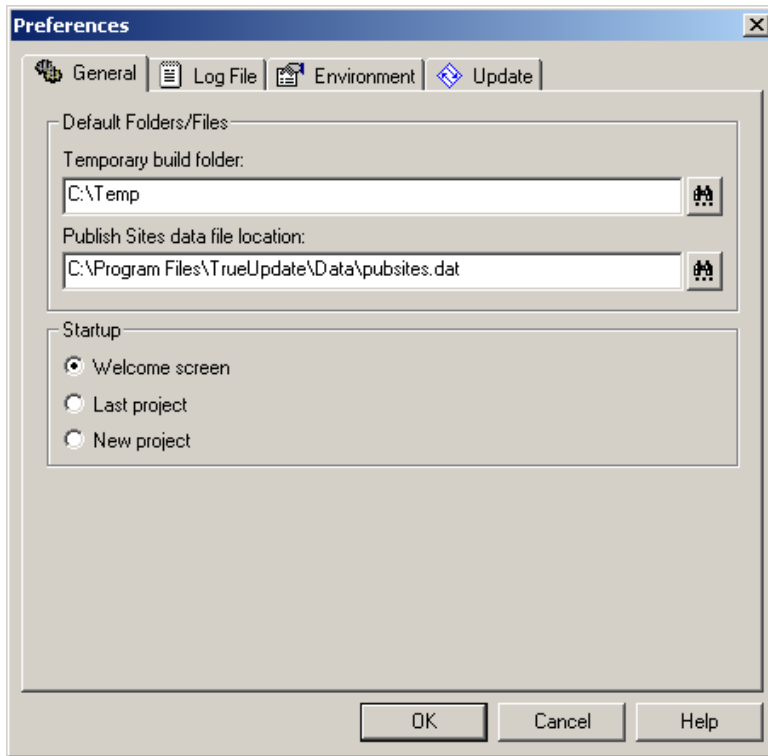


The Server File Report screen

To save the report, press the **Save As** button.


## General Preferences

You can set general preferences for the Server File Editor on the General tab of the *Preferences* screen. To access the *Preferences* screen, select **Edit | Preferences** from the menu.




The General tab of the Preferences screen

### Setting the Temporary Build Folder

The Server File Editor uses a temporary folder while performing operations on TrueUpdate server files. You can change the folder that is used for this purpose by editing the path in the **Temporary build folder** field, which is located on the General tab of the *Preferences* screen. You can also press the **Browse** button (  ) to browse for a path using the *Select Folder* screen.

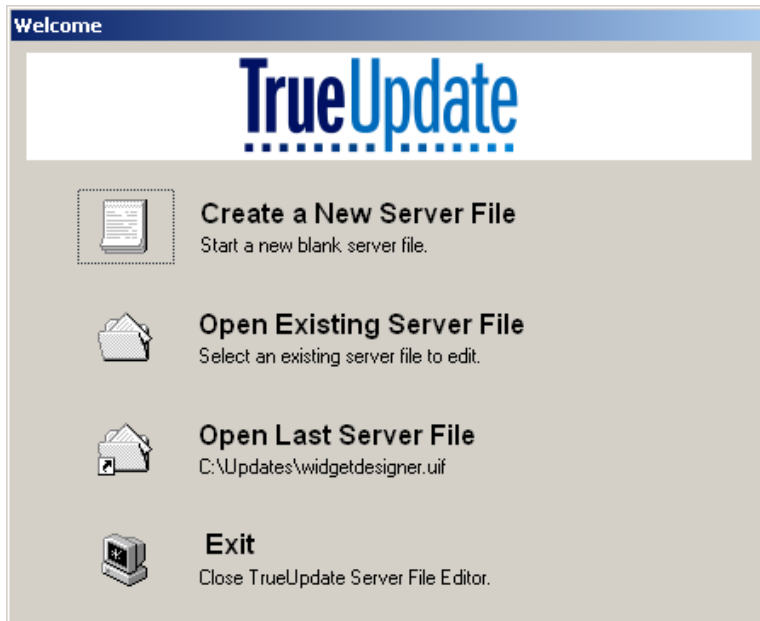
## Changing the Location of the Publish Sites Data File

The list of publish sites is stored in a single data file that is shared between instances of the Server File Editor. You can use the *Preferences* screen to select a different data file or change the location where the data file is stored. For instance, in a corporate environment, you might want to share the data file by storing it on a file server.

To select a different data file, edit the path and filename in the **Publish Sites data file location** field on the General tab of the *Preferences* screen. You can also press the **Browse** button (  ) to browse for the file using the *Select Folder* screen.

## Choosing Startup Options

When you start the Server File Editor, it opens a welcome screen by default to let you quickly start a new server file, open an existing server file, or open the last server file you had open. If you prefer, you can set the Server File Editor to automatically create a new server file or open the last server file instead of displaying this welcome screen.

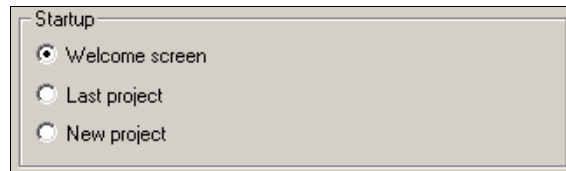


The Server File Editor's welcome screen

## Chapter 5

---

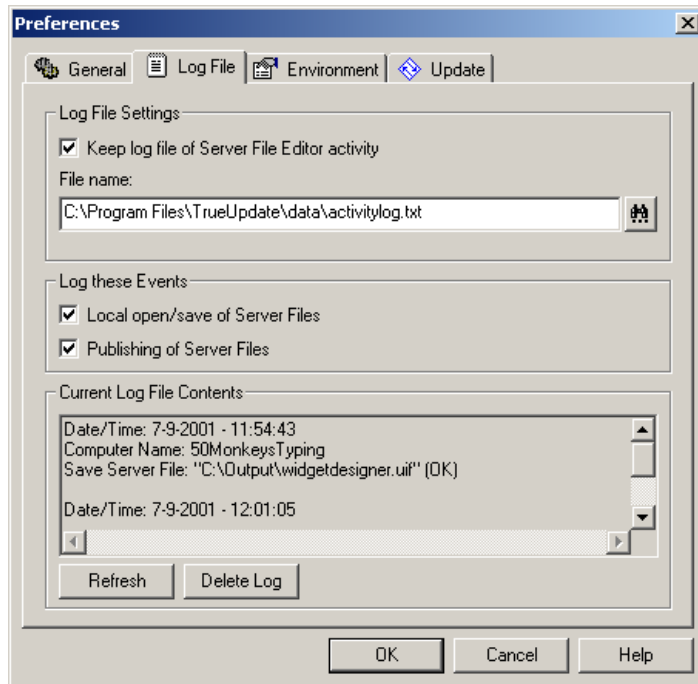
You can choose between the three options in the Startup section on the General tab of the *Preferences* screen.



The Startup section on the General tab of the Preferences screen

## Log File Preferences

You can set log file preferences for the Server File Editor on the Log File tab of the *Preferences* screen. To access the *Preferences* screen, select **Edit | Preferences** from the menu.



The Log File tab of the Preferences screen

## ***Logging Server File Editor Activity***

To keep a log of Server File Editor activity, select the **Keep log file of Server File Editor activity** check box on the Log File tab of the *Preferences* screen. You can specify the location and name of the log file in the **File name** field.

### **NOTE**



The log file retains information on all of the server files opened, saved and published with the Server File Editor; it isn't specific to each server file.

## ***Selecting which Events to Log***

To choose which events are logged, select or clear the **Local open/save of Server Files** check box and **Publishing of Server Files** check box.

## ***Viewing the Log File Contents***

You can view the log file contents on the Log File tab of the *Preferences* screen, or you can load the log file into a text editor or word processor and read it there.

## ***Refreshing the Log File Tab***

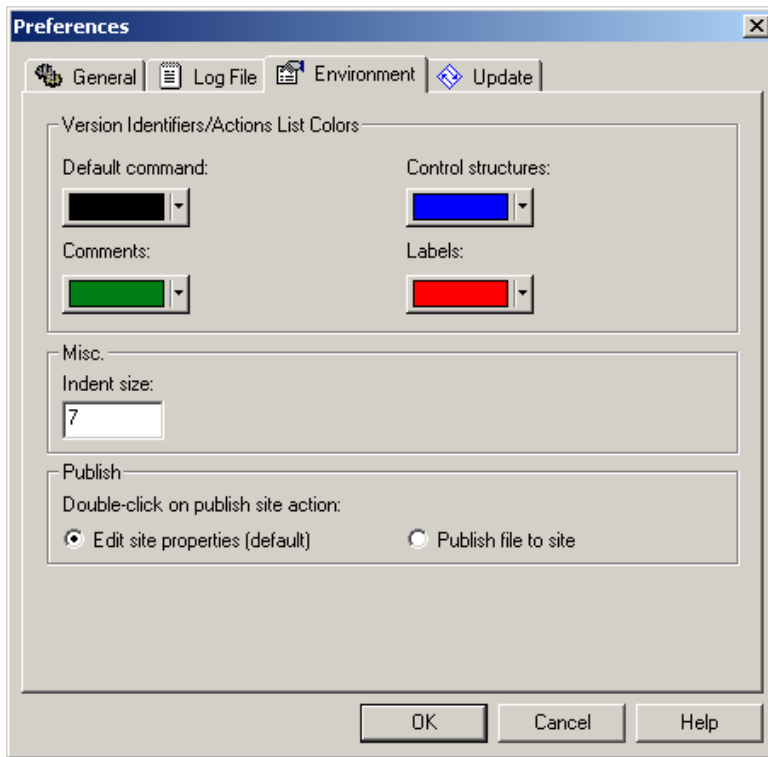
The **Refresh** button updates the log file contents on the Log File tab of the *Preferences* screen. You can use this to refresh the Log File tab after making changes to the log in an external text editor.

## ***Emptying the Log File***

When the log file grows too large, or if you want to reset the log, you can empty the log file by pressing the **Delete Log** button on the Log File tab of the *Preferences* screen. This will clear the log file (all its current contents will be deleted).

## Environment Preferences

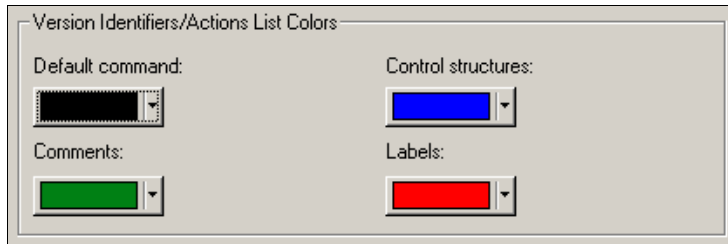
You can set development environment preferences for the Server File Editor on the Environment tab of the *Preferences* screen. To access the *Preferences* screen, select **Edit | Preferences** from the menu.



The Environment tab of the Preferences screen

### ***Changing the Version Identifiers/Actions List Colors***

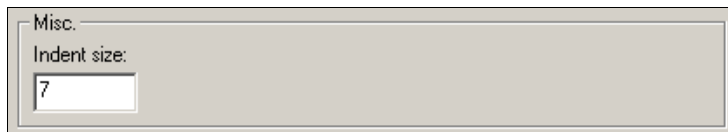
You can change the colors that are used for the control structures, comments, labels and default commands (which are basically "everything else") on the Version Identifiers and Actions tabs. To do so, just click on one of the four color choosers on the Environment tab of the *Preferences* screen, and select a different color from the list that appears.



The four color choosers on the Environment tab of the Preferences screen

### Changing the Indent Size

You can change the indent size used on the Version Identifiers and Actions tabs. To do so, enter the width in spaces for each level of indentation in the **Indent size** field on the Environment tab of the *Preferences* screen.

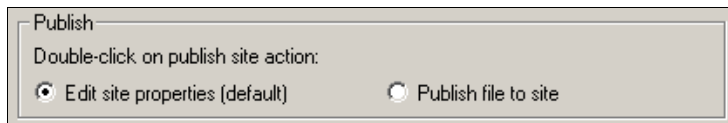


The Indent size field

### Changing the Double-click on Publish Site Options

By default, double-clicking on a publish site on the *Publish Server File* screen opens the appropriate *Publish Site Settings* screen for that site. If you find that you publish server files more often than you edit publish site settings, you might want to change this default setting.

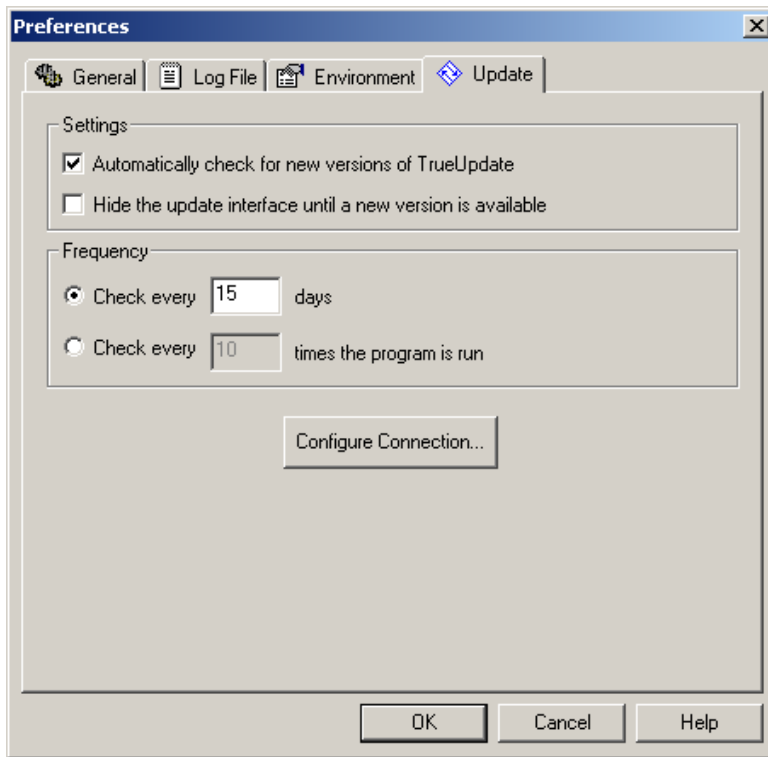
You can choose which action is performed by setting the **Double-click on publish site action** option to either **Edit site properties (default)** or **Publish file to site** on the Environment tab of the *Preferences* screen.



The Double-click on publish site action options

## Update Preferences

You can set update preferences for the TrueUpdate software on the Update tab of the *Preferences* screen. To access the *Preferences* screen, select **Edit | Preferences** from the menu.



The Update tab of the Preferences screen

**NOTE**



These settings are shared between the Client Configuration Utility and the Server File Editor.

### ***Automatically Checking for New Versions of TrueUpdate***

The TrueUpdate development software can automatically check the Internet for updates. Using the same technology that you can integrate into your software, TrueUpdate will check the Indigo Rose web site for new versions of itself.

To enable this feature, just select the **Automatically check for new versions of TrueUpdate** check box.

### ***Hiding the Update Interface Until a New Version is Available***

The **Hide the update interface until a new version is available** check box lets you control whether the TrueUpdate client interface is shown each time TrueUpdate checks the Internet for a new version of itself. If the check box is selected, the client interface is only shown when a new version is available and an update is performed.

### ***Setting How Often TrueUpdate Checks for Updates***

The Frequency section of the Update tab allows you to specify how often TrueUpdate checks the Internet for new versions of itself. You can have TrueUpdate check every X number of times the Client Configuration Utility or Server File Editor is started, or have it check (on starting TrueUpdate) if a minimum number of days have elapsed since the last time a check was performed.

### ***Configuring the TrueUpdate Connection Settings***

You can configure the connection settings TrueUpdate will use when it checks for an update by pressing the **Configure Connection** button.

## User Tools

You can add custom menu items to the Server File Editor's **Tools** menu to start other programs that you use often during the design process. For instance, you might want to add a menu item to launch your favorite text editor.

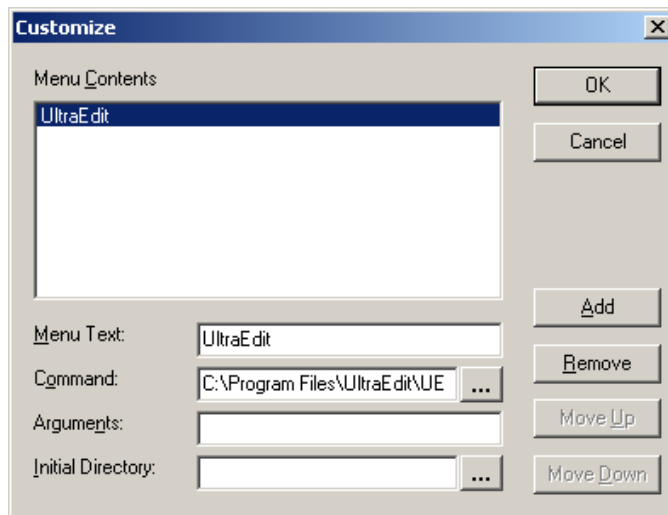
**NOTE**



Any custom tools you add to the Server File Editor's **Tools** menu will also show up in the Client Configuration Utility's **Tools** menu.

### Configuring User Tools

To configure the custom items that will show up in the **Tools** menu, select **Tools | Configure User Tools** from the menu to open the *Configure User Tools* screen.



The Configure User Tools screen

You can use this screen to add tools to the menu, remove them from the menu, and change the order of the tools in the menu.

---

## Chapter 6

# Variables

---

## What Are Variables?

In TrueUpdate, variables are special named "containers" for values that change. They're used to store information that won't be known until run time (like the location of the user's system directory, or the result of a Yes/No dialog presented to the user), or that you want to be able to change in one place at design time (serving the same purpose as constants in programming languages like C++).

All variable names in TrueUpdate begin and end with a percentage sign. At design time these variable names serve as placeholders, marking the places where the values will go once those values become known.

### SEE ALSO



For more information on variables, see page 22.

There are two kinds of variables in TrueUpdate: built-in variables, and custom variables.

### ***Built-in Variables***

TrueUpdate has a number of built-in variables that are already defined for you. They include:

- Directory and path variables such as %SrcDir% (the directory where the TrueUpdate client is being run from), %WinDir% (the windows directory), %SysDir% (the windows\System directory), etc.
- System variables such as %OS% (the Operating System), %ScreenWidth% (the width of the user's display screen in pixels), %SysLanguage% (the numeric ID for the language being used on the user's system), etc.

- Product information variables such as %ProductName% (the name of your product), %CompanyName% (the name of your company), etc. (These variables are defined in the General Information section of the Product Info tab of the Client Configuration Utility.)
- Time and date variables such as %JulianDate% (the number of days since midnight on January 1, 4713 B.C.), %CurrentMinute% (the current minute as set on the user's system), %CurrentYear% (the current year as set on the user's system), etc.
- Action-related variables such as %LastCommand% (the last action that was performed), %LastErrorNum% (a code indicating whether an error occurred while the last action was being performed), etc.

### SEE ALSO



For a complete list of the built-in variables, see *Built-in Variables* in the Command Reference, or see page 195 of this User's Guide.

## ***Custom Variables***

Custom variables are entirely up to you, and are specific to each project. You can define custom variables in several places within TrueUpdate, including:

### **The Variables tab of the Client Configuration Utility**

On the Variables tab you can define variables to read Registry values and INI file settings, or you can assign values to variables directly. These variables come into effect at the beginning of the update process, right after the client is executed.

The variables you define in the Client Configuration Utility require some planning on your part, because you can't change them without distributing a new client to your users. You can override them using server file actions, but those actions aren't performed until the server file is downloaded and a version has been identified. If the variables you defined in the Client Configuration Utility are being used before the server file is downloaded (e.g. in server file location conditions, or in a message

shown on the *Welcome* or *Check For Update* screens), the server file actions will happen too late to have any effect.

### **The Action Properties screens in the Server File Editor**

You can define variables using many of the actions in the Server File Editor. These variables only come into effect as each action is performed.

The variables you define in the Server File Editor offer you the most flexibility, as they are version-dependent and can be changed, added or removed easily without having to update the TrueUpdate client at all. Their only limitation is that they can't be used at the beginning of the update process—they only come into effect after the server file is downloaded and a version is identified.

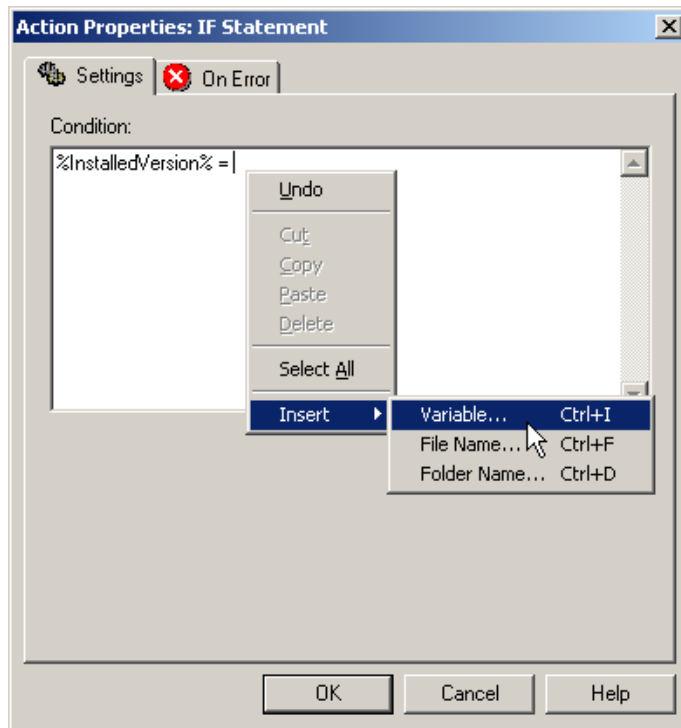
## **What Can You Do With Variables?**

Variables offer a lot of flexibility. You can use variables in the following ways:

1. You can use them to set up Boolean expressions in IF and WHILE actions in the Server File Editor.
2. You can use them in expressions to set conditions for individual server file locations and variable definitions in the Client Configuration Utility.
3. You can use variables in any of the messages that you define using the Client Configuration Utility.
4. You can use them to receive information that is read from the Registry, an INI file or a text file.
5. You can write their values to the Registry, to an INI file or to a text file using server file actions.
6. You can use them wherever you need to provide a path.
7. Basically, anywhere that you can enter text, you can use variables.

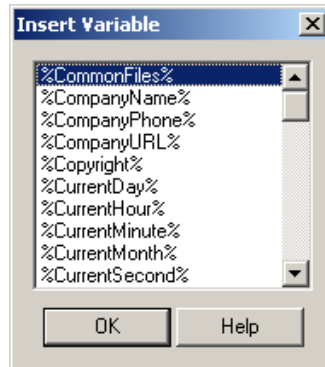
### *Inserting Variables*

You can easily insert variables where applicable by right-clicking on an edit field and selecting **Insert | Variable** from the context menu.



The Insert | Variable item in the right-click context menu

This will open the *Insert Variable* screen with all the built-in variables listed. Just select the variable you want to insert, and click OK.



The Insert Variable screen

**TIP**

Any custom variables you define in the Client Configuration Utility or Server File Editor will automatically show up on the *Insert Variable* screen of that program. In other words, variables you define in the Client Configuration Utility will show up on the Client Configuration Utility's *Insert Variable* screen, and variables you define in the Server File Editor will show up on the Server File Editor's *Insert Variable* screen.

***A Little Common Sense Never Hurts***

When defining and using variables, be sure to think about when and where they will be used, and what their values will be at that point in time. For example, when you define a variable on the Actions tab, it is only available to the actions that are listed below it on the tab. If you try to use the variable in another action higher "up" on the Actions tab, the variable won't have a value yet, and your update probably won't work properly.

**SEE ALSO**

For more information on defining variables in the Client Configuration Utility, see page 66. For information on the various actions, many of which allow you to define variables, see *Actions* in the Command Reference.

# Naming Variables

You can name your custom variables anything you like, but there are a few guidelines to follow:

1. All variable names should begin and end with a percentage sign (%).
2. Variable names are not case sensitive – the names %MyVar% and %myvar% both refer to the same variable.
3. Don't use the same variable name twice unless you mean to.

Using a name twice is fine if you're defining a variable in one place, and using it in another. Just be careful not to unintentionally give the same name to two different variables if they're being used to represent two different things.

Of course, it's okay to define a variable twice if you *want* its value to change. In that case, the first value will only be in effect until the second one "overwrites" it.

4. Don't use a built-in variable name for one of your custom variables unless you know what you're doing.

For example, calling a variable %SysDir% would override the built-in variable with the same name. Take time to familiarize yourself with the built-in variable names so you don't use any of them by accident.

5. Try to use meaningful names.

For example, if you read your product's installation date from the Registry, naming the variable %Greegleborg452sx% probably isn't a good idea. Use a variable name like %InstallDate% instead.

6. Be consistent.

If you start out with %UserName% and %UserAge%, don't name your third variable %WhereTheUserGoesToWork%. A name like %UserCompanyName% will be easier to remember when you find yourself asking "what did I name that variable, again?"

7. When in doubt, double-check.

A common mistake is to name a variable one thing in one part of TrueUpdate, and start calling it something else later. If you're not sure what name you used, double-check by going back to the screen where you defined the variable.

## Defining Variables with Server File Actions

Many of the actions in TrueUpdate produce results that must be stored in variables. For example, the Read from Registry action reads a value from the Registry, and assigns that value to the variable of your choice. The value that is read from the Registry is the action's result, and the variable is where that result is stored.

Each of these actions has a field where you can provide the name of the variable you want the result to be stored in. If you provide the name of an existing variable, the result will overwrite that variable's contents. If you provide a new variable name, a new variable will be created automatically. The action's result will be stored in the variable you provide, regardless of whether the variable already exists.

Essentially, wherever a result needs to be stored, you can create a new variable "on the fly" by simply providing a variable name that isn't already being used in your update.

### SEE ALSO



For more information on the various actions, many of which allow you to define variables, see *Actions* in the Command Reference.

# Using Variables in Expressions

Before an expression is evaluated, any variable names in it are replaced by the values they represent. If these values contain spaces (or other value-delimiting characters) they can cause the expression to produce unexpected results, and even generate errors at run time.

### SEE ALSO



For more information on expressions, see page 143.

For instance, let's say we want to concatenate (i.e. join) the string " is a nice person" to the string contained in the variable %UserName%. We could use the expression:

```
%UserName% + " is a nice person"
```

...which works fine if the variable %UserName% contains a single word like *Mark*. But if %UserName% contains a space like, say, *Mark Smark*, the expression becomes:

```
Mark Smark + " is a nice person"
```

...which fails, because *Mark* and *Smark* are seen as two separate values with no operator between them.

In order to avoid these problems, it's a good idea to surround variable names with quotation marks when using them in expressions. Any value between quotes is always interpreted as a single value, regardless of any spaces or other delimiting characters it might contain. By putting quotes around variables, each variable's contents are guaranteed to be seen as a single value.

So, if we rewrite our example expression as:

```
"%UserName%" + " is a nice person"
```

...we end up with:

```
"Mark Smark" + " is a nice person."
```

...which will not cause any problems.

**NOTE**



Variables only need to be quoted in two places: in expressions, and in the command line arguments for an execute command. You don't have to put quotes around variables anywhere else.

**Value-delimiting characters**

Delimiting characters separate values in expressions. In TrueUpdate, spaces and all the operators except AND, OR and MOD will delimit or "break" a word into two parts.

In order to be seen as a single value, a variable will need to be surrounded by quotation marks if it contains spaces or any of the following characters:

+ - \* / = > < ( )



---

## Chapter 7

# Expressions

---

## What Are Expressions?

An expression is any valid combination of *values* and *operators* that resolves to a single result. For instance, in the expression  $1 + 2$ , "1" and "2" are values, "+" is the add operator, and the resulting value would be 3.

You can use expressions to perform calculations, compare values, set conditions and make decisions at run time.

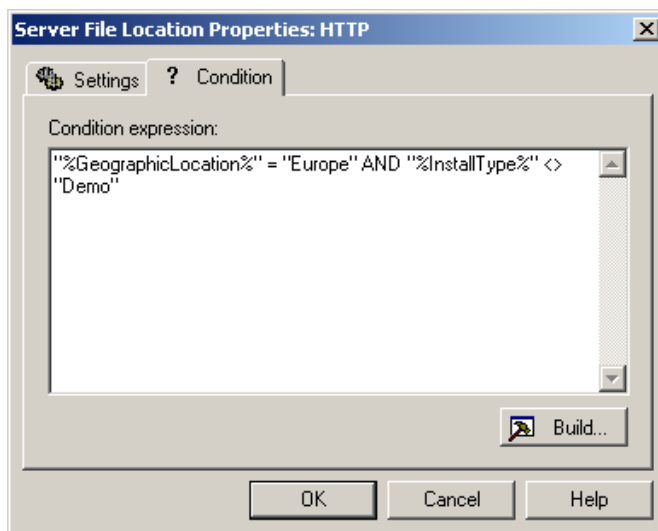
## Where Can You Use Them?

There are three places where expressions can be used in TrueUpdate: on Condition tabs, in the IF and WHILE actions, and on Assign Value screens.

### ***On Condition tabs***

You can enter expressions on the Condition tabs which are found on the various *Server File Location Properties* and *Variable Properties* screens in the Client Configuration Utility.

Before any server file location or variable definition is used, the expression on its Condition tab is evaluated, and the result is interpreted as a Boolean value. If the result is *true* (which in TrueUpdate means that the result is any non-zero integer), the condition has been met. If the result is *false* (which means the result is the number 0 or a non-numeric string), the condition has not been met. Each server file location or variable definition will only "work" if its condition is met—in other words, if the expression on its Condition tab evaluates to a true value, the server file location will be used or the variable assignment will be made.



An expression on the Condition tab for a HTTP server file location

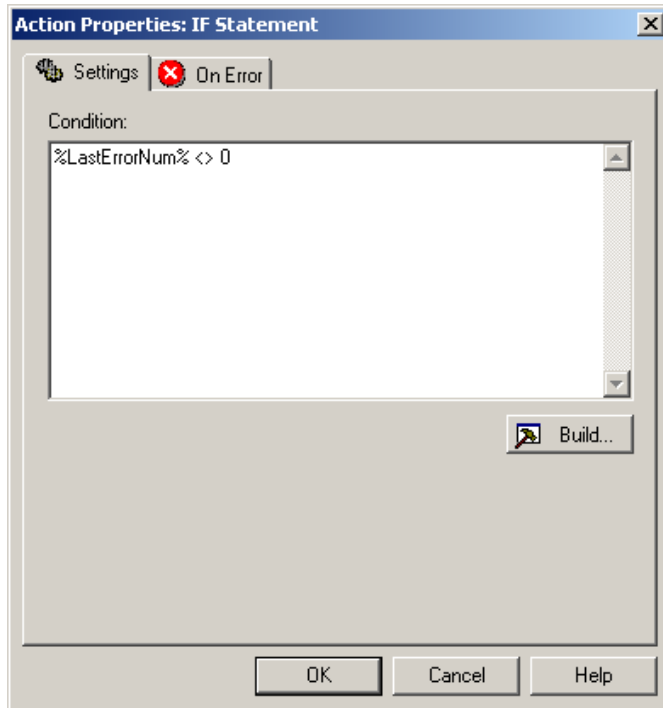
### ***In IF and WHILE actions***

You can use expressions to set up IF and WHILE actions in the Server File Editor. The IF action causes every action between it and the next matching END IF action to be performed only if its expression evaluates to a true result (any non-zero integer). The WHILE action continues performing every action between it and the next matching END WHILE action until its expression evaluates to a false result (either the number 0 or any non-numeric string).

```
IF (%UserName% = "Joe Blow")
  These actions are only performed if the variable...
  ...%UserName% contains the string "Joe Blow"
  Show Message Box (Hello Joel)
END IF
```

Example of IF and END IF actions in use

You can enter expressions in the **Condition** field on any *Action Properties: IF Statement* or *Action Properties: WHILE* screen.

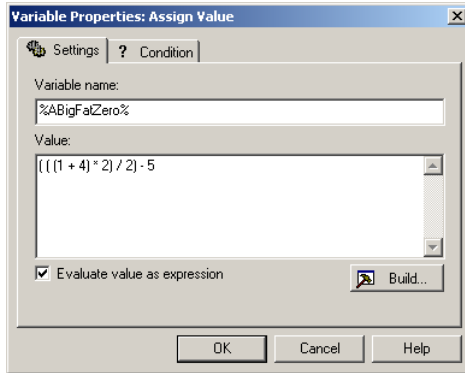


An expression in the Condition field for an IF action

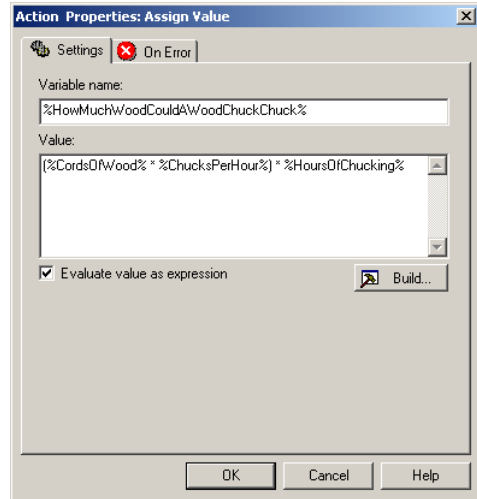
### ***On Assign Value screens***

You can also use expressions instead of values on *Variable Properties: Assign Value* screens in the Client Configuration Utility, and on *Action Properties: Assign Value* screens in the Server File Editor. Both of these screens have a special **Evaluate value as expression** check box that determines how the text in the **Value** field is interpreted.

When the **Evaluate value as expression** check box is selected, anything you type into the **Value** field is evaluated at run time, and the result of the evaluated expression is then assigned to the variable. When the check box is not selected, whatever you type in the **Value** field is assigned to the variable without being evaluated first.



In the Client Configuration Utility




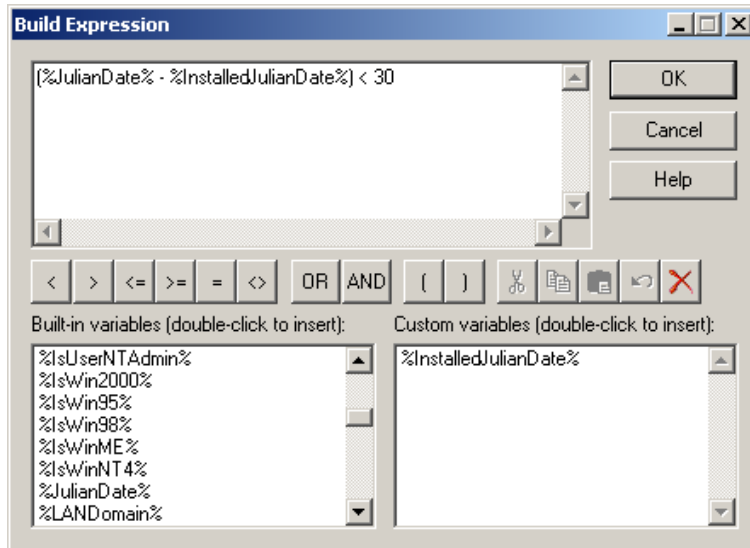
In the Server File Editor

The two Assign Value screens where you can use expressions

### TIP



On any screen where you can enter an expression, you can press the **Build Expression** button (  ) to open the *Build Expression* screen.



The Build Expression screen

## Values

There are four kinds of values you can use in TrueUpdate expressions: integers, strings, versions and variables.

### Integers

Integers consist of whole numbers, like 1, -32516 and 475. They can be written with or without quotation marks, i.e. "341" and 341 are equivalent.

### Strings

Strings consist of any sequence of characters surrounded by quotation marks, like "Wilbur", "the 25 happiest days of summer" and "hello world".

### Versions

Versions are special strings made up of integer numbers separated by periods, like "4.0", "1.03" and "0.2.0.0".

### Variables

You can use variable names as placeholders for integers, strings or versions in an expression. Before an expression is evaluated, any variable names in the expression are converted to the values they represent.

#### TIP



Expressions are essentially just strings that are interpreted differently by TrueUpdate. This means that you can store an expression in a variable, the same way you would store any other kind of string. You can evaluate the string by using the variable in an expression without quotes around it (so it isn't interpreted as a single string value).

For example, you could store the string `5 * 2` in a variable named `%a%`, and then use that variable in the expression `%a% + 1`, which would expand to `5 * 2 + 1` and evaluate to 11.

#### IMPORTANT



You should always surround variable names with quotation marks whenever they're used as values in expressions, in case they're assigned values that contain spaces. This way, a variable like `"%fullname%"` becomes a single string like `"Joe Blow"` instead of being interpreted as two string values like `"Joe"` and `"Blow"`, which would generate an error.

Expressions can also be used as "values" in a more complex expression. For instance, in the expression `(1 + 2) - 3`, the sub-expression `(1 + 2)` is evaluated first, resulting in the value 3. Then the expression `3 - 3` is evaluated, resulting in the final value 0.

#### How values are stored

All values are stored internally as strings. It's the content of the strings that determines how they're interpreted at run time.

A string containing only numeric characters is considered an integer when it's used in an expression. A string consisting of numbers and "internal" periods is considered a version. Anything else is assumed to just be a string.

## Operators

Operators are used in expressions to perform specific actions on one or more *operands*, generating a single return value or result. Operands are simply the values or expressions that each operator operates on. For instance, in the expression  $1 + 2$ , the values "1" and "2" are the operands, and "+" is the symbol for the add operator.

Operators are said to have *precedence*, which is a way of describing the rules that determine which operations in a series of sub-expressions get performed first. A simple example would be the expression  $1 + 2 * 3$ . In TrueUpdate, the multiply (\*) operator has *higher precedence* than the add (+) operator, so this expression is equivalent to  $1 + (2 * 3)$ . In other words, the sub-expression  $2 * 3$  is performed first, and then  $1 + 6$  is performed, resulting in the final value 7.

You can override the natural order of precedence by using parentheses. For instance, the expression  $(1 + 2) * 3$  resolves to 9. The parentheses make the whole sub-expression  $1 + 2$  the left operand of the multiply (\*) operator. Essentially, the sub-expression  $1 + 2$  is evaluated first, and the result is then used in the expression  $3 * 3$ .

Operators are also said to have *associativity*, which is a way of describing which sub-expressions are performed first when the operators have equal precedence. In TrueUpdate, most operators are *left associative*, which means that whenever two operators have the same precedence, the operation on the left is performed first. (The only exceptions are the unary plus, the unary minus, and the logical NOT operators, all three of which are right-associative.)

The left-associativity of the subtract (-) operator is why the expression  $10 - 5 - 2$  resolves to 3 instead of 7. It's interpreted as  $(10 - 5) - 2$ , and not  $10 - (5 - 2)$ .

### ***Table of Operator Precedence and Associativity***

The following operators can be used in TrueUpdate expressions.

Notes: A **unary** operator takes a single value.

A **binary** operator takes two values.

A **right-associative** operator operates on the value to its right.

An **infix** operator operates on one value to its left and one value to its right.

(All infix operators in TrueUpdate are left-associative.)

In order of precedence, from highest to lowest:

	<b>Name:</b>	<b>Notes:</b>	<b>Precedence level:</b>
(	open parenthesis		8 ( <i>highest</i> )
)	closed parenthesis		8
+	unary plus	unary, right-associative	7
-	unary minus	unary, right-associative	7
!	logical not	unary, right-associative	7
*	multiply	binary, infix	6
/	divide	binary, infix	6
<b>MOD</b>	modulus	binary, infix	6
+	add	binary, infix	5
-	subtract	binary, infix	5
<	less than	binary, infix	4
<=	less than or equal	binary, infix	4
>	greater than	binary, infix	4
>=	greater than or equal	binary, infix	4
=	equal	binary, infix	3
!= or <>	not equal	binary, infix	3
<b>AND</b>	Boolean AND	binary, infix	2
<b>OR</b>	Boolean OR	binary, infix	1 ( <i>lowest</i> )

## Parentheses

Parentheses are used to group sub-expressions and override the rules for precedence and associativity. Anything between an open parenthesis and a closed parenthesis is resolved first, before the rest of the expression is evaluated.

For instance, in the expression  $(5 + 2) * 3$ , the part between parentheses is performed first, and the result (7) is then used as a value in the larger expression  $7 * 3$ . If the parentheses were omitted, the expression  $5 + 2 * 3$  would resolve to 11 instead.

You can "nest" parentheses to form more complex expressions. Nesting just means using parenthetical expressions inside other parenthetical expressions. For example, the expression  $-((2 + 4) * 2)$  resolves to -12.

### TIP



Use parentheses whenever you can to help make your expressions easier to read.  $10 + (2 * 5)$  requires less thought to interpret than  $10 + 2 * 5$ .

## Logical (Boolean) Operators

Logical operators are used to combine the results of Boolean expressions. A Boolean expression is just like any other expression, but its result is evaluated to either true (any non-zero value) or false (0). In TrueUpdate, true and false are represented by 1 and 0.

There are three logical operators in TrueUpdate: AND, OR and !.

<b>AND</b>	And	Returns 1 (true) if both of its values are true. Returns 0 (false) otherwise.	<i>Example: 23 AND 0 resolves to 0</i>
<b>OR</b>	Or	Returns 1 (true) if either of its values are true. Returns 0 (false) if both of its values are false.	<i>Example: 5 OR 0 resolves to 1</i>
<b>!</b>	Not	Returns the Boolean opposite of its value, which is 1 (true) if its value is false, and 0 (false) if its value is true.	<i>Example: !5 resolves to 0</i>

### Relational Operators

Relational operators are used to compare two values. Relational expressions resolve to a Boolean (true/false) value: a true result resolves to 1, and a false result resolves to 0. For example,  $23 > 4$  is true because 23 is greater than 4, so this expression resolves to 1. "orange" < "apple" is false because the string "orange" is not alphabetically lower than "apple", so this expression resolves to 0.

There are seven relational operators in TrueUpdate: =, >, >=, <, <=, <> and !=.

---

=	Equal	Returns 1 (true) if the value on its left is equal to the value on its right. Returns 0 (false) if its two values are not the same. <i>Example: 4 = 4 resolves to 1</i>
>	Greater than	Returns 1 (true) if the value on its left is greater than the value on its right. Returns 0 (false) otherwise. <i>Example: 5 &gt; 20 resolves to 0</i>
>=	Greater than or equal	Returns 1 (true) if the value on its left is greater than or equal to the value on its right. Returns 0 (false) otherwise. <i>Example: 5 &gt;= 5 resolves to 1</i>
<	Less than	Returns 1 (true) if the value on its left is less than the value on its right. Returns 0 (false) otherwise. <i>Example: 5 &lt; 20 resolves to 1</i>
<=	Less than or equal	Returns 1 (true) if the value on its left is less than or equal to the value on its right. Returns 0 (false) otherwise. <i>Example: "bart" &lt;= "lisa" resolves to 1</i>
<>	Not equal	Returns 1 (true) if the value on its left is not equal to the value on its right. Returns 0 (false) if both values are the same. <i>Example: "1.4.7" &lt;&gt; "1.4.8" resolves to 1</i>
!=	Not equal	Same as the <> operator above. <i>Example: 3 != 3 resolves to 0</i>

---

## Arithmetic Operators

There are seven arithmetic operators in TrueUpdate: plus, minus, +, -, \*, / and MOD.

+	Unary plus	Indicates a positive value.  <i>Example: +7 resolves to 7</i>
-	Unary minus	Forms a negative value.  <i>Example: -4 resolves to -4</i>
+	Add	Used to add two values together.  <i>Example: 2 + 3 resolves to 5</i>
-	Subtract	Used to subtract one value from another.  <i>Example: 150 - 120 resolves to 30</i>
*	Multiply	Used to multiply one value by another.  <i>Example: 5 * 3 resolves to 15</i>
/	Divide	Used to divide one value by another. Performs integer division—any remainder is discarded.  <i>Example: 21 / 2 resolves to 10</i>
<b>MOD</b>	Modulus	Returns the remainder after an integer division is performed.  <i>Example: 21 MOD 2 resolves to 1</i>

## String Operators

There are two special string operators in TrueUpdate: + and \*.

+	Add (concatenate)	Concatenates two strings, i.e. appends one string to another.  <i>Example: "hello" + "world" resolves to helloworld</i>
*	Multiply (repeat)	Repeats a string a given number of times.  <i>Example: "Apple" * 3 resolves to AppleAppleApple</i>

## Chapter 7

---

### Version Operators

There are two special version operators in TrueUpdate: + and -.

---

+	Add	Adds two versions together. <i>Example: "1.3.2" + "1.7" resolves to 2.10.2</i>
-	Subtract	Subtracts each element of one version from the corresponding element of another. Negative results are "truncated" to zero. <i>Example: "2.0.5" - "1.9" resolves to 1.0.5</i>

## Notes

1. Anything between quotes (""), including whitespace, is considered a single string value.
2. Non-quoted strings are delimited by whitespace and any of the operators *except* MOD, AND and OR. For instance, sandy is seen as the single value "sandy" and not the expression "s and y".
3. You can perform string concatenation with the add operator ('+').
4. You can compare strings alphabetically using the <, <=, >, >=, =, != and <> operators.
5. String comparisons are case insensitive. "Hello world" is equal to "hElIo WoRlD" in TrueUpdate.
6. You can "multiply" strings with an integer value. (2 \* "Ha" resolves to HaHa)
7. Stand-alone strings resolve to 0 (false) when a Boolean result is expected. For instance, the expression "hello" resolves to 0 (false) when used alone in an IF statement.

8. Strings resolve to 0 (false) when used with the Boolean infix operators AND and OR. For instance, (1 and "hello") is equivalent to (1 and 0) and resolves to 0 (false).
9. Strings made up of only numbers and internal periods (periods that aren't at the very beginning or end of the string) are interpreted as versions. You can compare versions using <, <=, >, >=, =, != and <>, and you can perform "corresponding element addition" and "corresponding element subtraction" on them using + and -.
10. There is no real number support in TrueUpdate. Numbers with decimals are interpreted as version strings. Although you can perform a form of addition on the version strings, you must not confuse this with adding real numbers. For instance, 1.1 + 1.9 is 2.10 ("the 10<sup>th</sup> revision of version 2"), not 3.0.
11. The division operator performs integer division only—any remainder is thrown away.

## Syntax Rules

The following rules describe the various syntax checks that are performed on all expressions in TrueUpdate:

1. Each '(' must have a matching ')' and vice-versa.

Good:        ((%a%))

Bad:         ((%a%))

2. No empty parentheses. '()' is not allowed.

Good:        (%BelovedGazeInThineOwnHeart%)

Bad:         ()

## Chapter 7

---

3. No open parenthesis immediately after a closed parenthesis. ')(' isn't allowed.

Good:        (%foo%) > (%bar%)  
(2 > 3) OR ("tree" <= "tree-house")

Bad:         (%foo%)(%bar%)  
(2 > 3)("tree" <= "tree-house")

4. Each closed parenthesis must follow a value or a closed parenthesis. Only '<value>)' and '))' are allowed.

Good:        (%foo% = (%bar% + "hello"))  
              ((1 < 2) AND (2 < 3))

Bad:         (%foo% = )  
              3 \* (5 + )

5. Only open parentheses and right-associative operators are allowed before the first value in an expression. An expression can't begin with a binary infix operator or a closed parentheses.

Good:        !7  
              !(!(!0))

Bad:         \* %foo%  
              !( / 4)

6. You can't have two values in a row without an operator between them. '<value><value>' isn't allowed.

Good:        %foo% = %bar%

Bad:         %foo% %bar%  
              "apple" "jack"

7. You can't have a value immediately before an open parenthesis. '<value>(' isn't allowed.

Good: "hello" + ("wo" + "rld")

Bad: "hello" ("wo" + "rld")

8. You can't have a value immediately after a closed parenthesis. ')<value>' isn't allowed.

Good: (2 + 4) + "th day violation"  
(%count% + 1) > 5

Bad: (2 + 4) "th day violation"  
(%count% + 1) 5

9. All right-associative operators must be followed by a value, with no other operators between the right-associative operator and its value *except* for open parentheses and other NOT (!) operators. In other words, the NOT operator is allowed to repeat, but not the unary plus or minus...you can have '!!!<value>', but not '++++<value>'.

Good: !((-(-1)))  
!!!!1  
+(-(-2))

Bad: !+-%foo%  
!!-6  
-!2  
--(2)  
+-5

## Chapter 7

---

10. Every infix operator must have a value to the left of it.

Good:        %foo% + %bar%  
              4 \* 5 AND 2 + 3

Bad:         / 12  
              AND 26  
              \* 5 + 2

11. The last token in an expression must be a value or a closed parenthesis. It can't be an infix operator, right-associative operator or open parenthesis.

Good:        2 + 3  
              (5 > 2)

Bad:         2 +  
              5 >

---

## Chapter 8

# Actions

---

Actions are the instructions that tell TrueUpdate what to do once a version is identified. You will build a separate list of actions for each version in a server file.

### SEE ALSO

? For more information on using actions, see page 117. For detailed information on each action, including settings and error codes, please consult the Command Reference.

The following actions are available in TrueUpdate:

<b>Action</b>	<b>Category</b>	<b>Description</b>
<b>Abort</b>	<i>Control Structures</i>	Aborts the update process.
<b>Assign Value</b>	<i>Variables</i>	Assigns a value to a variable.
<b>Blank Line</b>	<i>Design</i>	Allows you to insert a blank line on the Actions tab. Blank lines have no effect on run-time operation.
<b>Close Program</b>	<i>Open/Close Programs</i>	Locates and closes a running program.
<b>Comment</b>	<i>Design</i>	Allows you to insert a comment on the Actions tab. Comments have no effect on run-time operation.
<b>Copy Files</b>	<i>File Operations</i>	Copies files.
<b>Create Directory</b>	<i>File Operations</i>	Creates a directory on the user's system.

---

## Chapter 8

<b>Action</b>	<b>Category</b>	<b>Description</b>
<b>Delete Files</b>	<i>File Operations</i>	Deletes files on the user's system.
<b>Download File FTP</b>	<i>Internet</i>	Downloads a file from an FTP site.
<b>Download File HTTP</b>	<i>Internet</i>	Downloads a file from a web site.
<b>END IF</b>	<i>Control Structures</i>	Terminates an IF/END IF block.
<b>END WHILE</b>	<i>Control Structures</i>	Terminates a WHILE/END WHILE block.
<b>Execute File</b>	<i>Open/Close Programs</i>	Runs an executable or batch file on the user's system.
<b>Find String</b>	<i>String</i>	Searches for one string within another and determines the location of the match if found.
<b>Generate Random Value</b>	<i>Variables</i>	Generates a random integer value between two numbers, or generates a random string based on a mask (e.g. a serial number).
<b>GOTO Label</b>	<i>Control Structures</i>	Jumps directly to a label on the Actions tab.
<b>IF</b>	<i>Control Structures</i>	Begins an IF/END IF block. The actions in this block will only be performed if the condition in the IF action resolves to a non-zero (true) result.
<b>Label</b>	<i>Control Structures</i>	Marks a position in the list of actions that can be reached using a GOTO Label action.
<b>Latest Version</b>	<i>Status</i>	Tells the client this is the latest version and the <i>Already Updated</i> screen should be shown. This action should always be on an Actions tab by itself.
<b>Left String</b>	<i>String</i>	Creates a new string from the left-most x characters of an existing string.

<b>Action</b>	<b>Category</b>	<b>Description</b>
<b>Length of String</b>	<i>String</i>	Counts the number of characters in a string.
<b>Mid String</b>	<i>String</i>	Creates a new string consisting of a number of characters starting from a given position in an existing string.
<b>Modify INI File</b>	<i>System Config</i>	Sets an INI file value, deletes an INI file value, or deletes an INI file section.
<b>Modify Registry</b>	<i>System Config</i>	Creates or deletes a Registry key or value.
<b>Move Files</b>	<i>File Operations</i>	Moves files between directories on the user's system.
<b>Open Document</b>	<i>Open/Close Programs</i>	Opens, plays, prints or performs other actions on a document using its associated viewer.
<b>Read File Association</b>	<i>Variables</i>	Assigns the path of the executable associated with a file extension (e.g. ".txt") to a variable.
<b>Read File Information</b>	<i>Variables</i>	Determines a file's version, its CRC value, its size in bytes, or whether it exists.
<b>Read from INI File</b>	<i>Variables</i>	Reads a value from an INI file and stores it in a variable.
<b>Read from Registry</b>	<i>Variables</i>	Reads a value from the Registry and stores it in a variable.
<b>Read Text File</b>	<i>Variables</i>	Reads the contents of a text file into a variable.
<b>Remove Directory</b>	<i>File Operations</i>	Removes (deletes) a directory on the user's system. The directory must be empty in order to be removed.
<b>Rename File</b>	<i>File Operations</i>	Renames a file on the user's system.

## Chapter 8

---

<b>Action</b>	<b>Category</b>	<b>Description</b>
<b>Right String</b>	<i>String</i>	Creates a new string from the right-most x characters of an existing string.
<b>Send Email</b>	<i>Internet</i>	Sends an email message using the SMTP protocol.
<b>Show Message Box</b>	<i>Dialogs</i>	Displays a standard Windows message box to present information to the user.
<b>Submit to Web</b>	<i>Internet</i>	Submits data to a web site and stores the response. Allows you to perform a POST or GET to a web script or program just as you would from an HTML form.
<b>Unzip Files</b>	<i>Compression</i>	Extracts all the files from a zip file to a directory on the user's system.
<b>Upload File FTP</b>	<i>Internet</i>	Uploads a file to an FTP server.
<b>WHILE</b>	<i>Control Structures</i>	Begins a WHILE/END WHILE block. The actions in this block will be performed while the condition in the WHILE action resolves to a non-zero (true) result.
<b>Write to Text File</b>	<i>String</i>	Writes a string to a text file.
<b>Yes/No Message Box</b>	<i>Dialogs</i>	Displays a standard Windows message box to get a Yes/No response from the user. The response is stored in a variable that you specify.
<b>Zip Files</b>	<i>Compression</i>	Adds files to a new or existing zip file on the user's system.

---

---

## Chapter 9

# Handling Errors

---

Even the most well-designed updates can run into situations that generate errors. For instance, connection problems might prevent an FTP download from succeeding, or an INI file you need to read from may have been deleted by a careless user.

TrueUpdate gives you two ways to respond to such errors:

### **Built-in error handling (The On Error tab)**

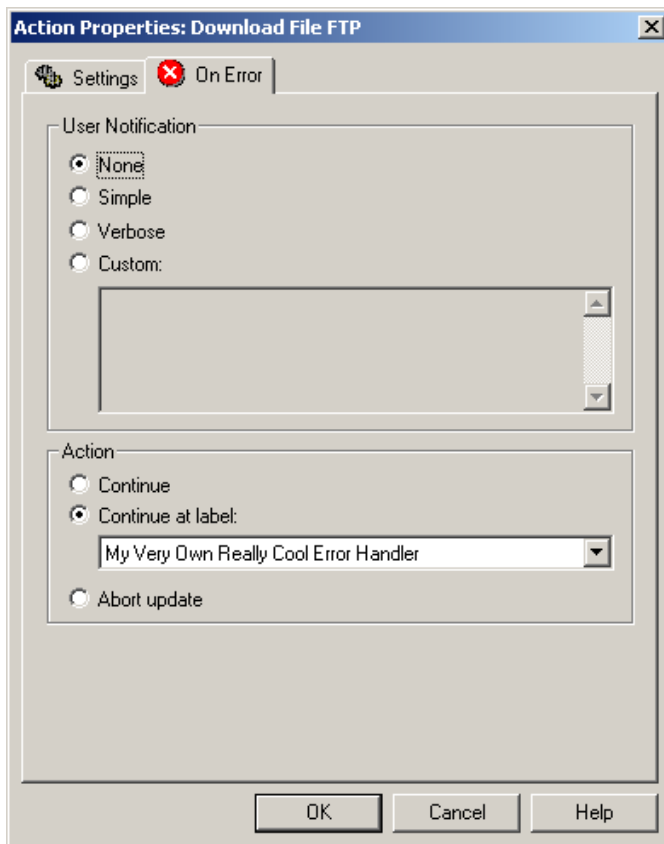
You can use the standard error handling features built into each action, which you can configure using the On Error tab of the action's properties screen.

### **Custom error handling (Using actions)**

You can use your own combination of actions to respond to the error. These actions could be contained in an IF/END IF block that checks built-in variables like %LastErrorNum% to determine whether an error occurred. Or, the actions could be assigned a label and called directly using the **Continue at label** setting on the On Error tab.

## **Built-in Error Handling (The On Error Tab)**

Every action in TrueUpdate that could generate an error at run time has an On Error tab on its *Action Properties* screen. You can use the On Error tab to configure how TrueUpdate responds when an error is generated by that action.



The On Error tab of an Action Properties screen

### ***Setting the User Notification Options***

The User Notification section of the On Error tab allows you to control how much information is given to the user when an error occurs—or even whether the user is notified at all.

There are four user notification options you can choose from:

#### **None**

No error message is shown to the user; the user is not notified about the error at all.

### Simple

A simple error message is displayed on a dialog window. The error message lets the user know that something happened, but provides only basic information about the type of error that occurred.

### Verbose

A more specific error message is displayed. The message might include pertinent details about the error, such as the URL that a failed HTTP download action was trying to connect to.

### Custom

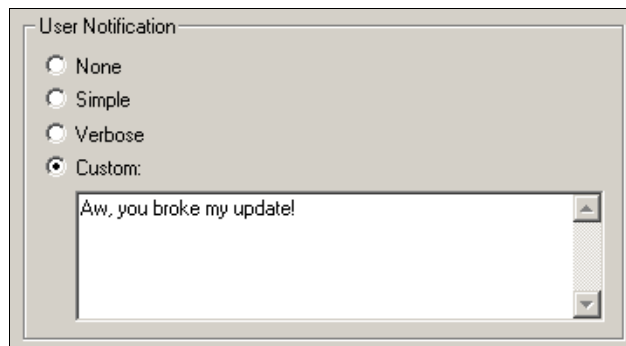
A custom error message of your own is displayed. You can enter your own message text directly onto the On Error tab when this option is selected.

#### TIP



You can use built-in variables like `%LastErrorNum%`, `%LastErrorMsg%` and `%LastErrorDetails%` to incorporate the standard error message text in your custom error messages.

To set the level of user notification you want, simply select the appropriate option in the User Notification section of the On Error tab.



The User Notification section of the On Error tab

### ***Setting the Action Taken After an Error Occurs***

You can choose whether you want the update to abort because of an error, or continue performing actions. If you have any labels defined on the Actions tab, you can even tell TrueUpdate to go to one of the labels and continue the update from there.

There are three on-error action settings you can choose from:

#### **Continue**

After an error, the update continues with the next action in the list—i.e., the next action on the Actions tab.

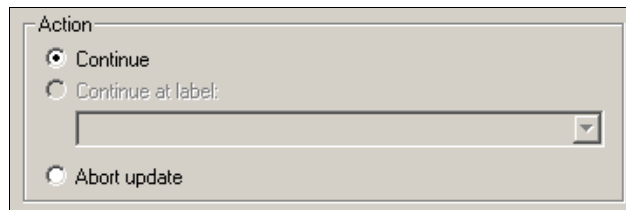
#### **Continue at label**

After an error, the update goes directly to the specified label and continues the update process from there. This option is only available if there is at least one label defined on the Actions tab for the current version.

#### **Abort update**

After an error, the update is aborted (no more actions are performed) and the *Update Failed* screen, if enabled, is shown.

To set the on-error action you want, simply select the appropriate option in the Action section of the On Error tab.



The Action section of the On Error tab

## Custom Error Handling (Using Actions)

In addition to the "built-in" error handling provided by TrueUpdate, you can use server file actions to respond to errors in a custom fashion. For example, you might want to try an alternative download site if an HTTP download fails. Or, you might want to present a Yes/No dialog to the user and let *them* choose whether to abort the update.

There are two ways you can trigger a custom error handling routine:

### Checking %LastErrorNum%

You can use an IF action to check the built-in variable %LastErrorNum% and determine whether the previous action generated an error.

### Using Continue at label

You can use the **Continue at label** option on an action's On Error tab to jump to your custom error-handling routine when that action generates an error.

### Checking %LastErrorNum%

%LastErrorNum% is a built-in variable that gets updated every time an action is performed by the client. If an action is successful, %LastErrorNum% is set to 0. If an action generates an error, %LastErrorNum% is set to a positive integer value that represents the error that occurred.

Each action has its own set of possible error numbers or "return values" that are assigned to %LastErrorNum% when an error occurs. Some actions, like Execute File, only have one error number. Others, like Download FTP, have several.

#### SEE ALSO



You can find a complete list of return values for each action in the Command Reference.

You can use %LastErrorNum% in the **Condition** field of an IF action to test whether the previous action generated an error. Any actions between the IF action and the next corresponding END IF action will only be performed when the preceding action fails.

## Chapter 9

---

Your error handler might look something like this:

```
action that might fail
IF %LastErrorNum%
    bunch of actions
    that are only performed
    when %LastErrorNum% is not 0
END IF
```

If you only wanted to handle specific errors, you could set things up like this instead:

```
action that might fail
IF ((%LastErrorNum% = 3) OR (%LastErrorNum% = 4))
    bunch of actions
    that are only performed
    when %LastErrorNum% is 3 or 4
END IF
```

Three other built-in variables are updated every time an action is performed by the TrueUpdate client: %LastCommand%, %LastErrorMsg% and %LastErrorDetails%.

%LastCommand% is set to the ID of the action.

%LastErrorMsg% is set to the standard "simple" error message for the action.

%LastErrorDetails% is set to the standard "verbose" error message for the action.

(Both error messages are localized—they're taken from the language file.)

**Note:** %LastErrorMsg% and %LastErrorDetails% are only set when the previous action generated an error. Both variables are empty ("") when the previous action was successful.

### *Using Continue at label*

Another way to trigger a custom error-handling routine is to use the **Continue at label** option found on an action's On Error tab. Just group your error-handling actions together, preface them with a label, and select that label for the **Continue at label** option.

You'll probably also want to use GOTO actions and additional labels to skip over your error-handling routines, so they remain isolated from the rest of your actions.

Using the **Continue at label** option, your error handler might look something like this:

```
normal actions
GOTO: Continue
LABEL: Error Handler
    bunch of actions
    that are only performed
    when an action fails
    and its Continue at label option
    is set to "Error Handler"
LABEL: Continue
more actions
```

### SEE ALSO

- ? See `samples\internal database files.uif` in your TrueUpdate program folder for a sample server file with this kind of error handler in it.



---

## Chapter 10

# Integrating TrueUpdate Into Your Application

---

TrueUpdate makes it easy for you to embed dynamic software updating capabilities into your software application. As a self-contained executable, the TrueUpdate client can be integrated into your application with a simple call to the Windows ShellExecute function.

Alternatively, the TrueUpdate client can be called directly from a shortcut on the Windows desktop or Start menu. This makes it perfect for stand-alone use. For instance, you could distribute the client to your sales team, and they could use it to update price lists and other data files from anywhere with Internet access. Or you could deploy TrueUpdate across your corporate network to remotely configure system files and easily update all the software you use.

You can make TrueUpdate as visible to your users as you want it to be, from running it silently at application startup, to letting your users invoke updates manually using a program menu or toolbar button. In fact, you can easily provide a range of visibility, by changing how the client operates using command line options. You could show the wizard interface when TrueUpdate is run from the Start menu, use the dialog interface when it's started from your program menu, and hide all the screens when it checks for updates automatically at program start.

### SEE ALSO




For more information on the client's command line options, see page 179.

## The Integration Two-Step

Integrating TrueUpdate into your application involves two basic steps: adding the client files to your software distribution, and providing one or more ways to initiate an update.

### *Step 1: Adding the Client Files*

The TrueUpdate client usually consists of two files: a client executable, and a single data file. The client executable is named `update.exe` by default, and the data file is named `update.cli` by default. Both files are generated when you press the **Build Now** button (  ) on the Build tab of the Client Configuration Utility.

Depending on the options you specify in the Client Configuration Utility, you may also need to distribute up to two additional files: an icon file for the custom icon feature, and a help file for the *Connection Settings* screen.

#### NOTE



A complete list of the files you need to distribute is displayed on the *Build Status* screen when you generate the client files.

Exactly how you add the client files to your distribution depends on the distribution method you use. It can be as simple as including the two files in a zip archive, or as sophisticated as using a professional installation tool like Indigo Rose's Setup Factory.

You can install the files anywhere on the user's system, but it's usually best to install them in the same place as your application. This allows you to use the built-in variable `%SrcDir%` to refer to that folder when designing your updates.

If you'd prefer to have the client files in a different folder, remember to provide some way for the TrueUpdate client to determine the path to your software. For example, you could write the path to a Registry key or INI file with your installer, and define a custom variable in the Client Configuration Utility to read it in.

### ***Step 2: Triggering TrueUpdate***

Initiating an update is simply a matter of running the TrueUpdate client. In fact, your users could check for an update simply by double-clicking on the client executable.

Of course, there are several more sophisticated ways to initiate an update. Here are some of the different ways TrueUpdate could be started:

#### **Using a Start Menu item**

Install a shortcut to the TrueUpdate client in the user's Start menu.

#### **Using an icon on the Desktop**

Give your users the option to install a shortcut on their Desktop so they can initiate an update at any time. You could use the standard TrueUpdate client icon for this purpose, or perhaps the same icon you use for your application.

#### **Using a menu item in your program**

Provide an item in a program menu like **Help | Check for Updates** that runs the client using a call to the Windows API ShellExecute function.

#### **Using a toolbar button in your program**

Provide a "Check for Updates" button on one of your program's toolbars.

#### **Automatically on starting Windows**

Install a shortcut in the user's Startup folder to automatically run TrueUpdate every time the user's system is started. You could use the `/S:#` command line option to run the client less conspicuously in "silent" mode. Or create a small "launcher" program that checks the number of times the system has been booted since the last update was performed and only launches TrueUpdate periodically. You could even set up an event with the Windows scheduling service if it's active on the user's system.

#### **Automatically when your program starts**

Launch TrueUpdate silently from your program when it's started. You could check the system date and only launch the client if your application was last started more

than  $x$  days ago. Or keep a counter of "application starts" in the Registry, and only check for an update every  $x$  times the user launches your program. You can even let your user specify how often they want the checks to be performed.

As a stand-alone executable, TrueUpdate gives you plenty of freedom to initiate the update process in a way that fits the style of your application. Which method (or methods) you choose to provide is entirely up to you.

## Command Line Options

The TrueUpdate client's command line options allow you to override the default settings that were made in the Client Configuration Utility.

The following options are especially useful when running the client from within your software:

### **/R:#**

This option allows you to run the client program in dialog (`/R:1`) or wizard (`/R:0`) mode. If you are running an update from within your software, you will want to use the interface that best fits the style of your application.

### **/S:#**

This option allows you to choose which screens are allowed to appear during the update. For instance, you can run the client "silently" on program start by using the `/S:250100` option to hide the *Welcome*, *Checking for Update* and *Already Updated* screens. That way, the TrueUpdate interface would only appear whenever an update is available.

### **/CONFIGURE**

This option lets you jump directly to the TrueUpdate client's *Connection Settings* screen. This lets you provide a way for your users to access this screen and configure the TrueUpdate network settings directly from within your program.

### SEE ALSO

? For more information on the client's command line options, see page 179.

## ShellExecute

The TrueUpdate client is simply an executable that can be run normally from Windows. The easiest Windows API function to use for executing programs is ShellExecute.

You can use ShellExecute from any programming language in Windows. Search the Internet for more information about calling ShellExecute from your favorite development language, or refer to the documentation that came with your development tools.

### SEE ALSO

? You can find out more about the ShellExecute function on the Microsoft Web site. At the time of writing, the MSDN documentation for the ShellExecute function could be found at:

<http://msdn.microsoft.com/library/en-us/shellcc/platform/Shell/Functions/ShellExecute.asp>

## Source Code

Here are some source code samples using ShellExecute to call TrueUpdate from Visual Basic and C/C++.

In the first two examples, the `/R:1` command line option is used to run the client in dialog mode.

**Calling ShellExecute from Visual Basic**

```
Private Declare Function ShellExecute Lib "shell32.dll" _
Alias "ShellExecuteA" (ByVal hwnd As Long, _
ByVal lpOperation As String, ByVal lpFile As String, _
ByVal lpParameters As String, ByVal lpDirectory As String, _
ByVal nShowCmd As Long) As Long

Private Const SW_NORMAL = 1

Private Sub RunTrueUpdateClient()

    ' Execute the program
    Dim lReturn As Long

    lReturn = ShellExecute(Me.hwnd, "open", _
        "C:\Program Files\Widget Designer\update.exe", _
        "/R:1", 0, SW_NORMAL)

    If lReturn <= 32 Then
        MsgBox "Error executing client"
    End If

End Sub
```

**Calling ShellExecute from C/C++**

```
BOOL RunTrueUpdateClient()
{
    HWND hWnd;
    int nResult;

    hWnd = GetSafeHwnd();

    nResult = ShellExecute(hWnd, "open",
        "C:\Program Files\Widget Designer\update.exe",
        "/R:1", NULL, SW_NORMAL);

    if(nResult > 32) // values above 32 indicate success
    {
        return TRUE; // update executed
    }
    else
    {
        return FALSE; // update not executed
    }
}
```

## ***Opening the Connection Settings Screen***

Here is the basic MFC/C++ code we use to access the client's *Connection Settings* screen from the Client Configuration Utility:

```
// This function is called when you press the
// Configure Connection button on the Preferences screen
BOOL CAutoUpdateProgram::ShowConnectionConfigDialog()
{
    if((int)ShellExecute(AfxGetMainWnd()->GetSafeHwnd(),
                        "open", m_strUpdateClientPath,
                        "/CONFIGURE", NULL, SW_SHOW) > 32)
    {
        return TRUE; // update executed
    }
    else
    {
        return FALSE; // update not executed
    }
}
```



---

## Chapter 11

# Command Line Options

---

Command line options are special values that can be passed to an executable file when it is run. Also known as "command line switches" or "arguments," command line options are usually used to set program options.

For example, entering "C:\abc.exe /W /F" on a command line would run a program called abc.exe and pass two command line options to it: /W and /F. The abc.exe program would see those options and handle them internally.

You can test command line options by running an executable from the Command Prompt in NT, or the DOS prompt in Windows. You can also use command line options in program shortcuts, or when running an application by using **Start -> Run**.

### NOTE



Not all executables accept command line options, and the list of meaningful command line options is specific to each program. /W might mean "wait for return" in one program, but it could mean "enable wacky walk animation" in another—or it might not even be recognized at all.

The following command line options are supported by the TrueUpdate client executable:

## **/BYPASS**

Forces the client to bypass the *Welcome* screen, regardless of whether it's in Wizard or Dialog mode.

This option is primarily used to support the client executable's ability to update itself. The /BYPASS option allows the updated client to be restarted without showing the *Welcome* screen a second time.

Example: `update.exe /BYPASS`

### ***/C:filename***

Allows you to specify the path and filename of a client data file for the client executable.

#### **NOTE**



At run time, the client executable configures itself according to the settings in an accompanying client data (`.cli`) file. By default, the client assumes that this data file is located in the same directory as the client executable, with the same base name as the client executable and a `.cli` extension. For example, if the client executable is named `update.exe`, it looks for `update.cli` by default.

Replace *filename* with the full path and filename of the client data (`.cli`) file you want TrueUpdate to use. If *filename* contains any spaces, you must enclose this command line option in quotation marks.

Example: `update.exe "/C:C:\Update Files\alternate.cli"`

#### **TIP**



You can use this command line option to share one client executable between multiple products.

### ***/CONFIGURE***

Opens the client directly to the *Connection Settings* screen.

You could use this option to allow users to access the *Connection Settings* screen directly from the start menu, or from a button or menu item in your program.

Example: `update.exe /CONFIGURE`

## ***/DEBUG***

Runs the client in debug mode if debug mode was enabled at design time. (You can enable debug mode on the Build tab of the Client Configuration Utility.) If debug mode wasn't enabled, this option will have no effect on the client executable.

If a password is required, use the `/DEBUG:password` command line option instead.

Example: `update.exe /DEBUG`

## ***/DEBUG:password***

Runs the client in debug mode if debug mode was enabled and password-protected at design time. (You can assign a debug mode password on the Build tab of the Client Configuration Utility.) If debug mode wasn't enabled, this option will have no effect on the client executable. If debug mode was enabled but no password was required, this option will act just like the `/DEBUG` option (i.e. the password will be ignored).

Replace *password* with the password that was specified at design time for debug mode access.

Example: `update.exe /DEBUG:trustno1`

## ***/DF:filename***

Allows you to specify a path and filename for the log file that is generated when the client is run in debug mode. This overrides the default behavior, which is to use the same path and filename as the client, but with a `.log` extension. E.g. the default log file for `c:\client.exe` would be `c:\client.log`.

Replace *filename* with a full path and filename for the log file. If *filename* contains any spaces, you must enclose this command line option in quotation marks.

Example: `update.exe "/DF:D:\My Documents\Logs\tulog.txt"`

### **/L:#**

Forces the client to use the messages associated with a specific language ID, instead of using the messages that correspond with the user's system locale settings.

You can add different sets of messages to the client by using the Messages tab of the Client Configuration Utility. By default, the client detects what language is configured on the user's system, and presents the corresponding set of messages. If there were no messages defined for the user's language, the messages from the language that was specified as the default language are used.

#### **SEE ALSO**



For more information on how the client selects languages, see page 61.

The `/L:#` option allows you to test your updates in multiple languages without having to change your system language and reboot.

Replace `#` with the language ID for the language you want TrueUpdate to "detect." The client will configure itself as though that language was the current system language.

Example: `update.exe /L:17`  
(forces the client to "detect" Japanese as the system language)

#### **TIP**



A complete list of language IDs can be found in the `C:\Program Files\TrueUpdate\languages\langids.ini` file.

## **/P:#**

Allows you to preview a screen.

This option is primarily used by the Client Configuration Utility when previewing screens from the Interface tab.

Replace # with the screen ID for the screen you want to preview.

Example: `update.exe /P:32768`  
*(previews the Update Successful screen using the Dialog interface style)*

### **SEE ALSO**

**?** See page 184 for the complete list of screen IDs.

## **/R:#**

Allows you to specify which interface style (Wizard or Dialog) the client will use.

This overrides the default interface style setting in the `.cli` file, as configured on the Interface tab of the Client Configuration Utility.

Replace # with either 0 or 1:

0 = Wizard interface

1 = Dialog interface

Example: `"C:\Program Files\Widget Designer\update.exe" /R:1`  
*(forces the client to use the Dialog interface style)*

### **/S:#**

Allows you to specify which of the eight Wizard or Dialog screens will be shown. (By default, only the screens with check marks in the Screen List on the Interface tab of the Client Configuration Utility are shown.)

# is a bitmask composed of the following values:

<b>(Interface) Screen:</b>	<b>Bit value / Screen ID:</b>
(Wizard) Welcome	1
(Wizard) Checking for Update	2
(Wizard) Update Available	4
(Wizard) Already Updated	8
(Wizard) Updating Software	16
(Wizard) Update Successful	32
(Wizard) Update Failed	64
(Wizard) Update Cancelled	128
(Dialog) Welcome	1024
(Dialog) Checking for Update	2048
(Dialog) Update Available	4096
(Dialog) Already Updated	8192
(Dialog) Updating Software	16384
(Dialog) Update Successful	32768
(Dialog) Update Failed	65536
(Dialog) Update Cancelled	131072

To build the bitmask, add all the screen IDs together for every screen you want shown.

For example, to show only the *Update Failed* and *Update Cancelled* screens in both Wizard and Dialog mode, you would add 64, 128, 65536 and 131072 to create the bitmask "196800". To hide all screens, you would use the bitmask "0".

```
Examples: update.exe /S:196800  
          update.exe /S:0
```

---

## Appendix A

# Glossary

---

<b>Action</b>	An instruction that the TrueUpdate client can perform once a version is identified.
<b>Application</b>	An executable program capable of performing several tasks or functions.
<b>Associativity</b>	A way of describing the rules that determine which sub-expressions are performed first when operators have equal precedence. If two operators have the same precedence and are left-associative, the operation on the left is performed first. If two operators have the same precedence and are right-associative, the operation on the right is performed first. (Most of the operators in TrueUpdate are left-associative.) [See: <i>expression, operator, precedence, sub-expression</i> ]
<b>Binary Operator</b>	An operator that operates on two operands. In TrueUpdate expressions, all binary operators are infix operators, so one operand precedes the operator, and the other operand follows it. In other words, a binary operator operates on one value to the left of it, and one value to the right. [See: <i>expression, operand, operator</i> ]
<b>Boolean</b>	A term for values that can only have one of two possible states (true or false).
<b>Built-in Variable</b>	A variable automatically defined for you by TrueUpdate. [See: <i>Variable</i> ]
<b>Client</b>	In TrueUpdate, "client" refers to the TrueUpdate client executable ( <code>update.exe</code> ).

## Appendix A

---

<b>Client Side</b>	In TrueUpdate, "client side" refers to the user's system, where the TrueUpdate client is run. [See: <i>Server Side</i> ]
<b>CRC Value</b>	<i>Cyclic Redundancy Check</i> value. A 32-bit checksum number calculated from the contents of a file, that changes whenever the file's contents change.
<b>Current Release</b>	The most up-to-date external (meaning "available to users") version of your software. Also called the "latest release."
<b>Custom Variable</b>	A variable that you define yourself. [See: <i>Variable</i> ]
<b>Design Time</b>	The process of designing your update using the Client Configuration Utility and the Server File Editor.
<b>Dialog</b>	A window that displays options or questions in order to receive input or instruction from a user. Also: a single window in an interface made up of logical steps that must be followed. [See: <i>Wizard</i> ]
<b>Directory</b>	A named location within a file system where other folders or files can be stored. [See: <i>Folder</i> ]
<b>Executable</b>	A program file, usually containing instructions for a computer in the form of machine code. In Windows, executables typically have a .exe extension.
<b>Expression</b>	Any valid combination of operands and operators that resolves to a single result. For example, the expression $(4 + 3) * 2$ resolves to 14. [See: <i>Operand, Operator</i> ]
<b>External Version</b>	A version of your software that is available to users. [See: <i>Release, Version</i> ]

<b>File</b>	<p>A collection of data stored as a single entity in a file system.</p> <p>Some files only serve as receptacles for data (text files, document files, bitmap image files), while others contain instructions for the system to perform (program files, dynamically linked library or ".dll" files).</p>
<b>Filename</b>	<p>The name given to a file when it is created or saved.</p>
<b>Firewall</b>	<p>A firewall is a combination of computer hardware and software used to keep a network secure. The firewall acts as a gatekeeper between an internal network and the Internet, allowing only specific kinds of messages to flow in or out.</p>
<b>Firewall Mode</b>	<p>Firewall mode (also known as "passive mode") is required whenever you need to transfer files to or from an FTP server from behind a firewall. [See: <i>Passive mode</i>]</p>
<b>Folder</b>	<p>The common term for a directory in Windows. A named location within a file system where other folders or files can be stored. [See: <i>Directory</i>]</p>
<b>FTP</b>	<p><i>File Transfer Protocol</i>. An Internet protocol for transferring files between computers.</p>
<b>Full-history Patch</b>	<p>An advanced patch that can update all older versions of a software product using a single self-extracting executable.</p> <p>Indigo Rose's Visual Patch creates full-history patches.</p>
<b>HTTP</b>	<p><i>Hypertext Transfer Protocol</i>. The Internet protocol used to transmit and receive data over the World Wide Web. Often used between a Web browser and a server to request a document and transfer its contents.</p>

## Appendix A

---

<b>Incremental Binary Patch</b>	<p>A patch that only contains the parts of each file that have changed from one version to the next.</p> <p>Incremental binary patches can only update one version at a time and are normally much slower than full-history patches.</p>
<b>Infix Operator</b>	<p>An operator that needs to be placed between the operands it operates on. For example, to multiply 2 by 3 with the Multiply operator (which is a binary infix operator) you would place the Multiply operator between the two values, like so: 2 * 3.</p> <p>[See: <i>binary operator, operand, operator</i>]</p>
<b>INI File</b>	<p>Short for "initialization file." A text file where user, application or system settings can be stored and retrieved by an application or the system as required.</p>
<b>Internal Version</b>	<p>A version of your software that is not available to users. In other words, a version of your software that is never released, but only exists internally. [See: <i>Version</i>]</p>
<b>Intranet</b>	<p>An "internal Internet" designed for use within a single company, university or organization. Essentially, a private network using the same technologies that drive the Internet. The main difference between an intranet and the Internet is that an intranet is not meant to be accessible to the public.</p>
<b>Julian Date</b>	<p>In TrueUpdate, this is a date expressed as an integer using the Chronological Julian Day numbering system. This system counts the number of days since midnight on January 1, 4713 B.C.</p> <p>The Julian Date is primarily useful when comparing dates or performing arithmetic to determine the number of days between two dates.</p>
<b>LAN</b>	<p><i>Local Area Network</i>. A network that connects computers located on the same floor or in the same building or nearby buildings.</p>
<b>Latest Release</b>	<p>The most up-to-date external (meaning "available to users") version of your software. Also called the "current release."</p>

<b>NIC</b>	<p><i>Network Interface Card.</i> A computer component that allows the computer to physically connect to and communicate over a network.</p>
<b>Operand</b>	<p>A value or sub-expression that an operator operates on. For example, in the expression 3 + 4, the values "3" and "4" are both operands of the "+" (or "Add") operator. [See: <i>Expression, Operator</i>]</p>
<b>Operator</b>	<p>A symbol used to represent an operation that can be performed on one or more operands in an expression. For example, in the expression 3 + 4, the "+" is the symbol for the "Add" operator. [See: <i>Expression, Operand</i>]</p>
<b>Passive Mode</b>	<p>Passive mode is required whenever you need to transfer files to or from an FTP server from behind a firewall.</p> <p>Normally, your computer makes a connection to an FTP server, and the FTP server responds by opening a connection back to your computer. This return connection won't work if your computer can't be reached directly from the Internet. In passive mode, both connections are made from your computer, so the FTP server doesn't have to do any connecting—it just waits passively for your computer to make both of the connections to it. [See: <i>Firewall mode</i>]</p>
<b>Password</b>	<p>A string of text used to gain access to something. [See: <i>Serial Number</i>]</p>
<b>Patch</b>	<p>A file that, when run, modifies or replaces specific files on a computer system, usually to bring an already-installed software product up to date.</p>
<b>Path</b>	<p>Text that describes a location within a file structure. Each path describes a route followed by the operating system to find, store or retrieve a file or folder.</p> <p>In Windows and MS-DOS, each folder in a path is separated by a backwards slash (\).</p>

## Appendix A

---

<b>Precedence</b>	A way of describing the rules that determine which operators in a series of sub-expressions are applied first. Operators with higher precedence are applied before operators with lower precedence. [See: <i>associativity, expression, operator, sub-expression</i> ]
<b>Prefix Operator</b>	An operator that needs to precede the operand it operates on. For example, to make the value 2 a negative number with the unary minus operator (which is a prefix operator), you would place the unary minus operator before the value, like so: -2. [See: <i>infix operator, operand, operator, unary operator</i> ]
<b>Protocol</b>	A set of rules that computers use to communicate with each other. Protocols ensure that different kinds of network hardware and software can work together.
<b>Registry</b>	A database used by newer versions of the Windows operating system to store system and software configuration details.  The Windows Registry serves a similar purpose to the <code>win.ini</code> and <code>system.ini</code> files used by earlier versions of Windows.
<b>Release</b>	A set of files that are distributed as a whole unit, i.e. all the files that make up one version of your software. [See: <i>External Version, Version</i> ]
<b>Run Time</b>	When the actual TrueUpdate client executable is run.
<b>Screen</b>	A window in a graphical user interface. [See: <i>Dialog, Window</i> ]
<b>Serial Number</b>	Some text (usually a sequence of numbers and/or letters) used to enable or identify one instance of a software product.
<b>Server</b>	A computer on a network that runs programs and stores data for use by other computers. Servers store information and respond to requests for that information by "serving" the information to other computers.

<b>Server Side</b>	In TrueUpdate, "server side" refers to the location where the TrueUpdate client is downloading the server file from. [See: <i>Client Side</i> ]
<b>Server File</b>	A special data file containing lists of version identifiers and actions for each version of your software. The version identifiers tell the TrueUpdate client how to identify a particular version of your software, and the actions tell the client how to bring a particular version of your software up to date. [See: <i>Action, Version Identifier</i> ]
<b>Server File Location</b>	A place where the server file is made available for download.
<b>Shell</b>	An interface between a user and an operating system. Often used to present an alternative interface which abstracts the details of the operating system and allows a user to perform tasks without accessing the underlying operating system directly. [See: <i>Shell Operation</i> ]
<b>Shell Operation</b>	An operating system task performed by the shell, usually at the request of a user or application. [See: <i>Shell</i> ]
<b>Shortcut</b>	[See: <i>Shortcut File</i> ]
<b>Shortcut File</b>	A very small file that links to another file in the Windows operating system. Items in the Favorites menu and the Start menu are all shortcuts. (Shortcut files have a .lnk extension that is usually hidden by the Windows operating system.)
<b>Shortcut Folder</b>	A folder that contains shortcut files.
<b>Shortcut Icon</b>	A visual representation of a Shortcut File in Windows. [See: <i>Shortcut File</i> ]
<b>SMTP</b>	<i>Simple Mail Transfer Protocol</i> . An Internet protocol for sending email messages.

## Appendix A

---

<b>Sub-directory</b>	A directory that is located within another directory. [See: <i>Sub-folder</i> ]
<b>Sub-expression</b>	An expression used as an operand in a larger expression. For example, $(1 + 2) * (3 - 1)$ is an expression made up of two sub-expressions, $(1 + 2)$ and $(3 - 1)$ . [See: <i>Expression, Operand</i> ]
<b>Sub-folder</b>	A folder that is located within another folder. [See: <i>Sub-directory</i> ]
<b>Unary Operator</b>	An operator that operates on a single operand. The unary operators in TrueUpdate are all prefix operators, which means they precede their operands. In other words, a unary operator operates on the value to the right of it. [See: <i>binary operator, expression, operand, operator</i> ]
<b>Variable</b>	A special named "container" for values that change.  In TrueUpdate, variable names always begin and end with a percentage sign (%).
<b>Version</b>	A single instance or variant of something that has changed or is expected to change over time. When changes are made to a software application, the result is a new "version" of the original application.  Also, the name or number used to identify one version from another. [See: <i>External Version, Internal Version, Release</i> ]
<b>Version Identifier</b>	An instruction that tells the client how to identify which version of your software is installed on the user's system. Each version identifier acts as a criterion that must be met in order for a version to be identified.
<b>WAN</b>	<i>Wide Area Network</i> . A network that connects computers over long distances. A WAN connects computers that are physically or even geographically far apart.

- Window** A rectangular area used as a canvas in the Windows operating system's graphical user interface, upon which objects such as text or buttons can be "drawn" by the operating system or individual applications. These windows can often be manipulated separately (moved, resized, minimized, maximized, etc.).
- Wizard** An interface made up of logical steps that must be followed, presented using successive dialog windows, often including additional explanation or involving simplification to streamline steps or be friendlier to novice users.  
[See: *Dialog*]



---

## Appendix B

# Built-in Variables

---

<b>%CommonFiles%</b>	<p>The user's Common Files directory.</p> <p>Typically, this is something like:</p> <pre>C:\Program Files\Common Files</pre>
<b>%CompanyName%</b>	<p>Your company's name. The value of this variable is set in the <b>Company name</b> field on the Product Info tab of the Client Configuration Utility.</p>
<b>%CompanyPhone%</b>	<p>Your company's phone number. The value of this variable is set in the <b>Company phone</b> field on the Product Info tab of the Client Configuration Utility.</p>
<b>%CompanyURL%</b>	<p>The URL to your company's web site. The value of this variable is set in the <b>Company URL</b> field on the Product Info tab of the Client Configuration Utility.</p>
<b>%Copyright%</b>	<p>The copyright message for your product. The value of this variable is set in the <b>Copyright notice</b> field on the Product Info tab of the Client Configuration Utility.</p>
<b>%CurrentDay%</b>	<p>A number representing the current day of the month, calculated when the update begins.</p>
<b>%CurrentHour%</b>	<p>A number representing the current hour in 24-hour time (e.g. 4:00 PM is 16), calculated when the update begins.</p>
<b>%CurrentMinute%</b>	<p>The current minute, calculated when the update begins. This number is always expressed with two digits, so 4 minutes into the hour will be "04".</p>

## Appendix B

---

<b>%CurrentMonth%</b>	A number representing the current month, calculated when the update begins. January is represented by "1" and December is represented by "12".
<b>%CurrentSeconds%</b>	The current second, calculated when the update begins. This number is always expressed with two digits, so 4 seconds into the minute will be "04".
<b>%CurrentYear%</b>	The four-digit number representing the current year, calculated when the update begins.
<b>%Date%</b>	<p>The current date on the user's system when the patch is run. It's in the format MM/DD/YY.</p> <p>For example, if the user runs the patch on May 23, 2001, %Date% will be:</p> <p>05/23/01</p>
<b>%Desktop%</b>	The path to the user's Desktop directory. On Windows NT, this is the path from the per-user profile.
<b>%DesktopNT%</b>	The path to the user's Desktop directory. On Windows NT, this is the path from the All Users profile.
<b>%Display... &lt;screen name&gt; ...DialogScreen%</b>	<p>Examples: %DisplayUpdatingSoftwareDialogScreen% %DisplayCheckForUpdateWizardScreen%</p> <p>These variables are used to tell the TrueUpdate client which screens to show. Each variable has a value of "TRUE" if a screen should be displayed and "FALSE" if not. They are initially set to match the settings on the Interface tab of the Client Configuration Utility and can be overridden with the /S : # command line option. You can also assign "TRUE" or "FALSE" to these variables at any time to change whether the screens will be displayed.</p>
<b>%Display... &lt;screen name&gt; ...WizardScreen%</b>	
<b>%FontDir%</b>	The path to the user's font directory.

<b>%IsUserNTAdmin%</b>	<p>This variable is set to "True" if the user running the TrueUpdate client is currently logged into Windows NT (or Windows 2000) with Administrator permissions. It's set to "False" otherwise.</p> <p>On systems that aren't running some version of Windows NT, this variable is always set to "False".</p>
<b>%IsWin95%</b>	<p>This variable is set to "True" if the TrueUpdate client is running on Windows 95, and "False" otherwise.</p>
<b>%IsWin98%</b>	<p>This variable is set to "True" if the TrueUpdate client is running on Windows 98, and "False" otherwise.</p>
<b>%IsWinME%</b>	<p>This variable is set to "True" if the TrueUpdate client is running on Windows Millennium, and "False" otherwise.</p>
<b>%IsWinNT4%</b>	<p>This variable is set to "True" if the TrueUpdate client is running on Windows NT 4.0, and "False" otherwise.</p>
<b>%IsWin2000%</b>	<p>This variable is set to "True" if the TrueUpdate client is running on Windows 2000, and "False" otherwise.</p>
<b>%IsWinXP%</b>	<p>This variable is set to "True" if the TrueUpdate client is running on Windows XP, and "False" otherwise.</p>
<b>%JulianDate%</b>	<p>An integer value representing the number of days since midnight on January 1, 4713 B.C. Very useful when comparing dates or performing arithmetic to determine the number of days between two dates.</p>
<b>%LANDomain%</b>	<p>The domain that the user is logged in to. If the user's system is not connected to a LAN, this variable will default to "UNKNOWN".</p>
<b>%LANHost%</b>	<p>The user's local computer name. If the user's system is not connected to a LAN, this variable will default to "UNKNOWN".</p>

## Appendix B

---

<b>%LANIP%</b>	The user's IP address on the local network. If the user's system is not connected to a LAN, this variable will default to "UNKNOWN".
<b>%LANNIC%</b>	The MAC address of the user's NIC ( <i>network interface card</i> ). If the user's system does not contain a network card, this variable will default to "UNKNOWN".
<b>%LANUser%</b>	The user name that the user is currently logged in as. If the user's system is not connected to a LAN, this variable will default to "UNKNOWN".
<b>%LastCommand%</b>	The ID of the last action that was performed. (See the specific actions in the Command Reference for their respective action IDs.)
<b>%LastErrorDetails%</b>	<p>If an error is generated, this variable is set to the "verbose" error message for the last action that occurred (the action that generated the error). When an action doesn't generate an error, this variable is empty ("").</p> <p>For a complete list of the verbose error messages which can be generated by an action, see that action's documentation in the Command Reference.</p>
<b>%LastErrorMsg%</b>	<p>If an error is generated, this variable is set to the "simple" error message for the last action that occurred (the action that generated the error). When an action doesn't generate an error, this variable is empty ("").</p> <p>For a complete list of the simple error messages which can be generated by an action, see that action's documentation in the Command Reference.</p>

<b>%LastErrorNum%</b>	<p>If an error is generated, this variable is set to an action-specific error number that identifies the last error that occurred. When an action does not generate an error, this variable is set to 0.</p> <p>For a complete list of the error numbers associated with an action, see that action's documentation in the Command Reference.</p>
<b>%OS%</b>	<p>A number indicating the user's operating system:</p> <ul style="list-style-type: none"><li>1 = Windows 95</li><li>2 = Windows 98</li><li>3 = Windows NT 4.0</li><li>4 = Windows 2000</li><li>5 = Windows ME</li><li>6 = Windows XP</li></ul>
<b>%ProductName%</b>	<p>The name of the product that you are updating. The value of this variable is set in the <b>Product name</b> field on the Product Info tab of the Client Configuration Utility.</p>
<b>%ProgramFiles%</b>	<p>The path to the user's Program Files directory.</p>
<b>%RegOwner%</b>	<p>The name of the registered user of the system.</p>
<b>%RegOrganization%</b>	<p>The organization of the registered user of the system.</p>
<b>%ScreenHeight%</b>	<p>The user's screen height in pixels.</p>
<b>%ScreenWidth%</b>	<p>The user's screen width in pixels.</p>
<b>%SrcDir%</b>	<p>The full path of the folder the TrueUpdate client was run from, for example:</p> <pre>C:\Program Files\Widget Designer\Updates</pre>

## Appendix B

---

<b>%SrcDrv%</b>	The drive that the TrueUpdate client was run from, for example:  C:
<b>%StartMenu%</b>	The path to the user's Start menu directory. On Windows NT, this is the path from the per-user profile.
<b>%StartMenuNT%</b>	The path to the user's Start menu directory. On Windows NT, this is the path from the All Users profile.
<b>%StartMenuPrograms%</b>	The path to the Programs folder in the user's Start menu. On Windows NT, this is the path from the per-user profile.
<b>%StartMenuProgramsNT%</b>	The path to the Programs folder in the user's Start menu. On Windows NT, this is the path from the All Users profile.
<b>%Startup%</b>	The path to the user's Startup folder. On Windows NT, this is the path from the per-user profile.
<b>%StartupNT%</b>	The path to the user's Startup folder. On Windows NT, this is the path from the All Users profile.
<b>%SysDir%</b>	The path to the user's Windows System directory, for example:  C:\Windows\System
<b>%SysDrv%</b>	The drive that the user's Windows System directory is located on. For example:  C:
<b>%SysLanguage%</b>	The numeric primary language ID for the user's system language.
<b>%TempDir%</b>	The path to the user's Temp directory.

<b>%UserEmailAddress%</b>	The user's email address as set by the user on the Email tab of the <i>Connection Settings</i> screen. If the user has not configured this option or has not changed it from the default value, this variable is set to "Not Configured".
<b>%UserSMTPPort%</b>	The user's SMTP port as set by the user on the Email tab of the <i>Connection Settings</i> screen. Set to 25 by default.
<b>%UserSMTPServer%</b>	The user's SMTP server address as set by the user on the Email tab of the <i>Connection Settings</i> screen. If the user has not configured this option or has not changed it from the default value, this variable is set to "Not Configured".
<b>%WinDir%</b>	The path to the user's Windows directory. For example:  C:\Windows



---

## Appendix C

# Contact Info

---

Indigo Rose Corporation is a world leader in software installation and deployment tools. Programmers, electronic publishers, multimedia developers and software professionals from all over the world turn to Indigo Rose Corporation for state-of-the-art solutions.

## Corporate Headquarters

Web: <http://www.indigorose.com>  
Email: [info@indigorose.com](mailto:info@indigorose.com)  
Office Hours: Monday to Friday  
9:00 AM to 5:00 PM Central Standard Time

## Sales

The Indigo Rose sales team is ready to answer your questions Monday to Friday from 9:00 AM to 5:00 PM Central Standard Time.

Contact a sales representative for information on the latest Indigo Rose products or to purchase technical support subscriptions, upgrades or additional product licenses.

If your question is technical in nature, please contact Technical Support.

Phone: (204) 946-0263  
Fax: (204) 942-3421  
Web: <http://www.indigorose.com>  
Email: [sales@indigorose.com](mailto:sales@indigorose.com)

# Technical Support

Indigo Rose Corporation offers a variety of technical support programs designed to match your specific needs. Both complimentary and fee-based support options are available. Please refer to the *Developer Support Programs* brochure included with your package for complete details on the available support programs.

Your connection to all of our support resources can be found at:  
<http://www.indigrose.com/support/>

## ***Before You Contact Our Support Department***

Save yourself both time and money by referring to our supplementary resources *before* you contact our technical support department. Answers to just about any question can be found in these self-help resources. You'll be able to get the answers you need 24 hours a day, 7 days a week.

### **Product Documentation**

The product documentation includes this User's Guide, the Command Reference, and any last-minute notes in a `readme.txt` file. Answers to most common support issues can be found in the product documentation.

### **Knowledge Base**

A collection of informational articles and how-to tutorials. These articles cover both common and uncommon situations, including advanced technical issues. Many articles are written in response to consulting-type situations. You'll find a wealth of good information in the knowledge base.

### **Discussion Groups**

Indigo Rose maintains a number of community message boards. These discussion groups are frequented by many developers, including our own technical support representatives. The discussion groups are a great source for ideas, solutions and peer support.

## ***Limitations of Technical Support***

Technical Support is limited to general product usage questions. Questions relating to external functionality such as operating systems, development environments, third-party products or various Windows technologies are not included. All services are provided subject to the current *Support Terms and Conditions* as listed at our web site.

Limitations include, but are not limited to:

- Problems with your development tool such as Visual Basic, Delphi, Director, etc. Please consult your development tool's documentation for these issues.
- Third-party technologies you are distributing (such as BDE or QuickTime).
- Issues relating to distribution media such as CD-R burning.
- A specific design issue that is unique to your distribution.

### **TIP**



If your situation requires a more personalized solution, you might want to consider using our Consulting Services. Our consulting department can help you with matters that go beyond standard technical product issues.

For more information on Consulting Services, please visit our web site.



---

## Appendix D

# Minimum System Requirements

---

TrueUpdate requires the following minimum system configuration in order to operate properly:

### TrueUpdate Design Environment

- Windows 95, 98, ME, NT 4.0, 2000
- Pentium processor
- 32 MB RAM
- 800x600 SVGA display
- Video card set to 15 bit 32K color or greater (recommended)
- Mouse
- 20+ MB free hard drive space

### TrueUpdate Client Executable

- Windows 95, 98, ME, NT 4.0, 2000
- 486 processor
- 8 MB RAM
- 640x480 SVGA display
- Video card set to 8 bit 256 color or greater (recommended)
- Mouse
- 10+ MB free hard drive space



---

# Index

- actions.....27, 117
  - adding ..... 118
  - changing the order of ..... 120
  - cutting, copying, pasting ..... 121
  - editing ..... 120
  - exporting ..... 122
  - importing ..... 122
  - indenting ..... 120
  - removing ..... 122
  - unindenting ..... 121
- actions list ..... 162
- adding a local publish site ..... 93
- adding a version ..... 98
- adding actions ..... 118
- adding an FTP publish site ..... 94
- adding comments ..... 114
- adding FTP server file locations ..... 46
- adding HTTP server file locations ..... 45
- adding indentation to actions ..... 120
- adding indentation to version identifiers ..... 116
- adding languages ..... 63
- adding local server file locations ..... 44
- adding the client files to your software ..... 172
- arithmetic operators ..... 153
- Assign Value screens ..... 145
- assigning a value to a variable ..... 68
- before you begin ..... 29
- Boolean ..... 185, 192
- Boolean operators ..... 151
- build expression button ..... 146
- building the client ..... 30, 73, 76
- building the server file ..... 33
- built-in error handling ..... 163
  - setting the action taken after an error occurs ..... 166
  - setting the user notification options ..... 164
- built-in variables ..... 22, 133, 195–201
  - %CommonFiles% ..... 195
  - %CompanyName% ..... 195
  - %CompanyPhone% ..... 195
  - %CompanyURL% ..... 195
  - %Copyright% ..... 195
  - %CurrentDay% ..... 195
  - %CurrentHour% ..... 195
  - %CurrentMinute% ..... 195
  - %CurrentMonth% ..... 196
  - %CurrentSeconds% ..... 196
  - %CurrentYear% ..... 196
  - %Date% ..... 196
  - %Desktop% ..... 196
  - %DesktopNT% ..... 196
  - %Display[...]DialogScreen% ..... 196
  - %Display[...]WizardScreen% ..... 196
  - %FontDir% ..... 196
  - %IsUserNTAdmin% ..... 197
  - %IsWin2000% ..... 197
  - %IsWin95% ..... 197
  - %IsWin98% ..... 197
  - %IsWinME% ..... 197
  - %IsWinNT4% ..... 197
  - %IsWinXP% ..... 197
  - %JulianDate% ..... 197
  - %LANDomain% ..... 197
  - %LANHost% ..... 197
  - %LANIP% ..... 198
  - %LANNIC% ..... 198
  - %LANUser% ..... 198
  - %LastCommand% ..... 168, 198
  - %LastErrorDetails% ..... 168, 198
  - %LastErrorMsg% ..... 168, 198
  - %LastErrorNum% ..... 167, 199
  - %OS% ..... 199
  - %ProductName% ..... 199
  - %ProgramFiles% ..... 199
  - %RegOrganization% ..... 199
  - %RegOwner% ..... 199
  - %ScreenHeight% ..... 199
  - %ScreenWidth% ..... 199
  - %SrcDir% ..... 199
  - %SrcDrv% ..... 200
  - %StartMenu% ..... 200
  - %StartMenuNT% ..... 200
  - %StartMenuPrograms% ..... 200
  - %StartMenuProgramsNT% ..... 200
  - %Startup% ..... 200
  - %StartupNT% ..... 200
  - %SysDir% ..... 200
  - %SysDrv% ..... 200
  - %SysLanguage% ..... 200
  - %TempDir% ..... 200
  - %UserEmailAddress% ..... 201
  - %UserSMTPPort% ..... 201
  - %UserSMTPServer% ..... 201

## Index

---

- %WinDir% ..... 201
- calling ShellExecute from C/C+... 176
- calling ShellExecute from Visual Basic
  - ..... 176
- changing the actions list colors ..... 128
- changing the client executable filename
  - ..... 74
- changing the double-click on publish site options ..... 129
- changing the image used on a wizard screen ..... 52
- changing the indent size ..... 129
- changing the location of the publish sites data file..... 125
- changing the order actions are performed in ..... 120
- changing the order that variables are assigned in..... 72
- changing the output folder ..... 74
- changing the search order for server file locations ..... 48
- changing the version identifiers list colors ..... 128
- changing the version search order... 100
- changing user configuration settings.. 56
- choosing startup options ..... 78, 125
- client ..... 25, 185
- Client Configuration Utility..... 14, 39–83
  - building the client ..... 73
  - interface settings ..... 49
  - messages..... 61
  - product info ..... 41
  - projects ..... 40
  - server file locations ..... 42
  - variables..... 66
- client side ..... 25
- colors ..... 128
- command line options..... 184
  - /BYPASS ..... 179
  - /C:filename ..... 180
  - /CONFIGURE ..... 174, 180
  - /DEBUG ..... 181
  - /DEBUG:password..... 181
  - /DF:filename ..... 181
  - /L: # ..... 182
  - /P: # ..... 183
  - /R: # ..... 174, 183
  - /S: # ..... 174, 184
- Command Reference ..... 19
- comparing CRC values ..... 111
- comparing file versions ..... 109
- condition tabs ..... 143
- Configure button ..... 57
- configuring user tools ..... 82, 132
- connection settings..... 60
- continue at label..... 168
- copying an existing version..... 99
- CRC values ..... 23–24
- CRC-32 ..... 24
- creating a new server file ..... 86
- custom error handling ..... 163, 167
  - checking %LastErrorNum% ..... 167
  - using continue at label ..... 168
- custom error messages ..... 165
- custom icons..... 54
- custom variables ..... 22, 134
- cutting, copying and pasting actions 121
- cutting, copying and pasting version identifiers..... 115
- debug mode
  - enabling..... 74
  - password-protecting ..... 76
- default language files..... 79
- default output folder ..... 77
- defining variables with server file actions ..... 139
- design time ..... 21
- dialog mode..... 49
- different file patching methods..... 36
- disabling support for user configuration ..... 55
- document conventions ..... 17
- double-click options ..... 129
- editing a publish site..... 96
- editing actions ..... 120
- editing messages ..... 62
- emptying the log file ..... 127
- enabling debug mode in the client executable ..... 74
- enabling the More Information button ..... 102
- environment preferences..... 128–29
- error handling
  - built-in ..... 163
  - custom ..... 163
- evaluate value as expression ..... 145
- exporting actions..... 122
- exporting messages to a language file 64
- expressions ..... 143
  - arithmetic operators ..... 153
  - Boolean operators ..... 151
  - integers ..... 147

- logical operators ..... 151
- notes..... 154
- operator associativity ..... 150
- operator precedence ..... 150
- operators ..... 149
- parentheses ..... 151
- relational operators ..... 152
- string operators ..... 153
- strings ..... 147
- syntax rules ..... 155
- values ..... 147
- variables ..... 148
- version operators ..... 154
- versions ..... 147
- where can you use expressions? . 143
- firewall mode..... 47, 96, 187
- forums..... 20
- FTP publish sites ..... 92
- FTP server file locations ..... 43
- generating a report ..... 123
- getting started ..... 37
- glossary ..... 185–93
- how TrueUpdate works ..... 13
- HTTP server file locations ..... 43
- importing actions ..... 122
- indent size ..... 129
- indenting actions..... 120
- indenting version identifiers..... 116
- Indigo Rose
  - sales ..... 203
  - tech support..... 204
- inserting variables..... 136
- integrating the client into your software
  - ..... 35, 172
  - source code..... 175
- interface settings ..... 49
  - dialog mode ..... 49
  - wizard mode ..... 49
- Julian date ..... 188
- labeling a version ..... 101
- list colors ..... 128
- local publish sites ..... 92
- local server file locations ..... 43
- log file preferences..... 126–27
- logging server file editor activity ..... 127
- logical operators ..... 151
- messages..... 61
- minimum system requirements..... 207
- More Information button ..... 102
- naming variables..... 138
- notes ..... 105
- On Error tab ..... 163
  - abort update..... 166
  - continue..... 166
  - continue at label ..... 166, 167
- online help ..... 19
- opening an existing project..... 40
- opening an existing server file..... 86
- opening the connection settings screen
  - ..... 177
- operators ..... 149
  - associativity ..... 150
  - precedence..... 150
  - usage notes..... 154
- other resources ..... 19
- parentheses..... 151
- passive mode..... 47, 96, 189
- password protecting the server file .... 89
- previewing screens ..... 54
- product information ..... 41
- projects..... 40
- providing an update size ..... 104
- publish sites ..... 92
- publish sites data file ..... 125
- publishing the current server file ..... 87
- publishing the server file ..... 34
- reading a value from an INI file.. 71, 108
- reading a value from the Registry..... 70, 106
- rearranging version identifiers..... 115
- refreshing the log file tab ..... 127
- relational operators ..... 152
- removing a publish site ..... 96
- removing a version ..... 99
- removing actions..... 122
- removing indentation from actions... 121
- removing indentation from version
  - identifiers..... 117
- removing languages ..... 63
- removing server file locations ..... 47
- removing variables ..... 72
- removing version identifiers..... 116
- reopening a recent project..... 41
- reopening a recent server file..... 87
- reports..... 123
- requiring a debug mode password .... 76
- run time ..... 21
- running TrueUpdate ..... 173
- saving connection settings..... 60
- saving the current project ..... 41
- saving the current server file ..... 86
- selecting the default interface style.... 51

## Index

---

- selecting which events to log ..... 127
- Server File Editor .....14, 85–132
  - actions ..... 117
  - activity log ..... 127
  - publish sites ..... 92
  - reports ..... 123
  - version identifiers ..... 105
  - versions bar ..... 97
- server file locations .....26, 42
  - FTP ..... 43
  - HTTP ..... 43
  - local ..... 43
- server files .....25, 86
  - building the server file ..... 33
  - publishing the server file ..... 34
- server side ..... 25
- setting conditions for server file locations ..... 48
- setting default language files ..... 79
- setting product information ..... 42
- setting the default language ..... 64
- setting the default output folder ..... 77
- setting the temporary build folder .... 77, 124
- sharing actions with other users..... 122
- ShellExecute..... 175
- showing and hiding screens ..... 51
- simple error messages ..... 165
- source code ..... 175
- starting a new project ..... 40
- startup options ..... 78, 125
- string operators ..... 153
- syntax rules ..... 155
- tech support ..... 204
- temporary build folder ..... 77, 124
- the update process ..... 28
- translating messages ..... 65
- triggering TrueUpdate ..... 173
- TrueUpdate ..... 13
  - design environment ..... 14
  - integrating ..... 172
  - update preferences ... 81–82, 130–32
  - user tools ..... 82, 132
  - web site ..... 19
- TrueUpdate client ..... 25
- TrueUpdate technology ..... 15
- unindenting actions ..... 121
- unindenting version identifiers ..... 117
- user tools ..... 82, 132
- using a custom icon in the title bar .... 54
- using continue at label ..... 168
- using expressions in IF and WHILE actions ..... 144
- using expressions on Assign Value screens ..... 145
- using expressions on condition tabs ..... 143
- using OR operators ..... 112
- using variables in expressions ..... 140
- validating the current server file ..... 91
- value-delimiting characters ..... 141
- values ..... 147
- variables ..... 22, 66, 133, 148
  - built-in ..... 22, 133
  - built-in (list) ..... 195–201
  - custom ..... 22, 134
  - defining variables with server file actions ..... 139
  - inserting ..... 136
  - naming ..... 138
  - using variables in expressions ..... 140
  - what are variables? ..... 133
  - what can you do with variables?.. 135
- verbose error messages ..... 165
- version identifiers ..... 26, 105
  - comments ..... 114
  - compare CRC values ..... 111
  - compare file versions ..... 109
  - cutting, copying, pasting ..... 115
  - indenting ..... 116
  - OR operators ..... 112
  - read from INI file ..... 108
  - read from Registry ..... 106
  - rearranging ..... 115
  - removing ..... 116
  - unindenting ..... 117
- version operators ..... 154
- versions
  - adding ..... 98
  - copying ..... 99
  - labeling ..... 101
  - notes ..... 105
  - removing ..... 99
  - search order ..... 100
  - update size ..... 104
- versions bar ..... 97
- viewing the log file contents ..... 127
- wizard mode ..... 49