Build Automator

BUILD AUTOMATOR

Build Automator

Users Guide

by Arnór Baldvinsson

Welcome to the Build Automator Users Guide.

To familiarize yourself with the Build Automator we suggest that you read the Introduction and Getting Started chapters.

If you run into problems, the Support chapter has information about how to contact us and report problems, feature requests or general comments.

Published: Thursday, November 17, 2016

Build Automator

Published: November 2016

Publisher

Icetips Alta LLC

Managing Editor

Arnor Baldvinsson

Table of Contents

Build Automator

Part I

Chapter 1 - Build Automator

1 Build Automator

Thank you for using Build Automator™.

Current version: 6.11.1366, November 17, 2016

If you have not used the **Build Automator™** before we suggest that you read the Introduction 2 and Getting Started 4 sections first to get aquainted with it.

The **Build Automator™** is a very powerful tool that you can use to minimize the time it takes for you to build new software install. It can also help you reduce the time it takes to do many repetitive tasks and we are certain that you will find many uses for the **Build Automator™** that we did not envison when we designed it!

If you have questions, check out **Frequenty Asked Questions** 150 section in this document and also our FAQ pages on our website at http://www.buildautomator.com/faq.php The **Build Automator™** has a built-in support module that will let you post a question directly to our on-line support. If you request support, please provide your full name when prompted so we know who we are communicating with:)

We also provide peer support on our support forum at http://www.buildautomator.com/forum as well as a blog site at http://www.buildautomator.com/blog. We are also working on a trouble ticket system online.

The Build Automator™ includes a web update module that can check for availability of updates to the software. If new updates are available the module can download them and install them completely automatically. Updates can only be installed if you have a valid maintenance plan.

Please check the Vesrion history 154 for information about each release. This section details what is included and fixed in each build that we release.

This document was last updated on 11/17/2016 at 12:51 PM.

1.1 Introduction

With the Build Automator™ you can make difficult and repetitive tasks simple and easy to do, saving time when in a crunch to get a new release out. While it's designed with Software developers in mind to make it easy to create new builds of software, the Build Automator™ is very versatile and can be used for all kinds of automation.

How to get started?

- Read the Getting Started 4 chapter. It explains how to create a simple project.
- Read the Glossary 5 to familiarize yourself with the terminology used in this documentation.
- Read the Project Window 30 chapter. It covers the project window in detail.
- Check out tutorials on our website

1.2 Getting Started

Once you have installed the Build Automator™ we would suggest that you create a simple project to get to know the software. We have kept the interface as clean and simple as we possibly could to make it easy to use and easy to get around in. We intend to stay true to that design even though the interface may change in the years to come.

To create a new project please follow these steps:

1. Select "File | New" from the main menu 7.
2. Navigate to the folder that you want to save the project file in and click the "Save" button
3. The Project Window 30 will now show up, with the Project Item list on the left and the Action items in the middle and both lists will be empty.
4. You can view the Project Properties 33 by clicking on the button farthest to the left in the Project Window 30 toolbar.
5. Create a new project item by right clicking in the "Project Items" list on the left and select "Add Item" from the popup menu.
6. Press F2 or click twice on the item to change the name. You may need to click on the item first to select it. There is no update window for the Project Items as the only information that we need is the name (this may change in future releases).
7. Look at the available actions in the "Actions" list on the right and double click on the action 67 you want to add.
8. You will get an action update window where you can specify the settings for the action.
9. To save the project use "File | Save" from the main menu 7 or press Ctrl-S on the keyboard.
10. Right click on the Action item and select "Execute this Action" to execute this one action. Check the Project Properties 33 topic to read about options that affect the execution of the project.
11. Close the project by clicking the big red X up in the right corner. If you have made changes to the project and it hasn't been saved you will be prompted to save the project.

Congratulations, you now have created your first Build Automator™ project!

For tutorials on the Build Automator™, please visit the tutorial page on our website

Webinar presentation from June 26, 2009 - 284MB

1.3 Glossary

Action Files Files with .actn extension that contain information about one of more Vendor Actions.

Project Action Files Each action in the project has it's own Project Action file. It's name consists of the GUID of the Project Action and a .patn extension. These files are created by copying corresponding Vendor Action Files to the project folder which is named with the GUID of the project. NOTE: As of version 1.6 this is no longer the case. The information is now stored in the Project file.

Plugins A win32 DLL that contains functions to update and execute project actions.

Call Function A single exported function in the Plugin DLL that is called by the Build Automator™.

IDE Integrated Development Environment. The Build Automator™ is an IDE with plugins that integrate into the IDE.

Project Items The Project Items group together related Action Items. Each project item can have one or more action items. You need to have at least one project item, but you can group and organize the project items any way you want. There is no limit to the number of action items each project item can contain.

Action Items Each action item performs a specific action during execution of the script. For example copy files, create folders, upload a file via FTP. A project item is a group of action items.

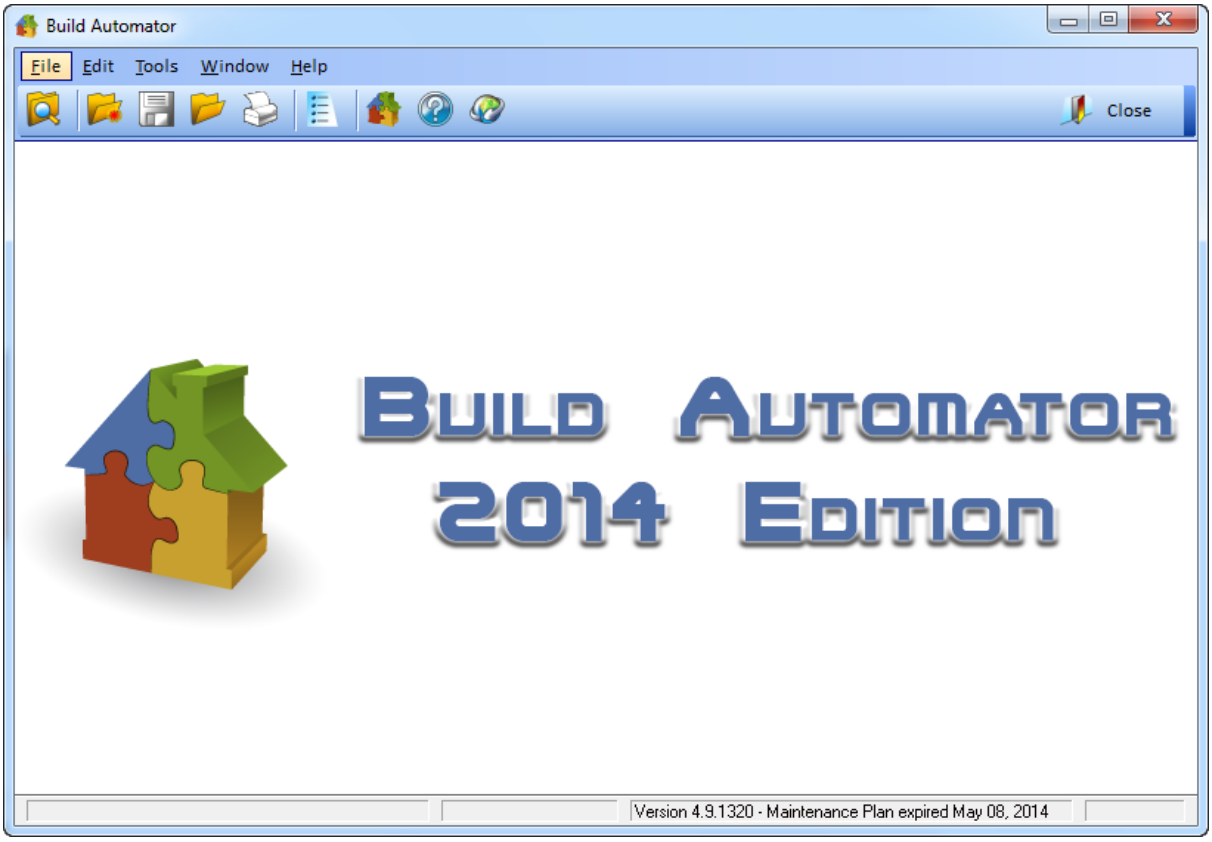
Build Automator

Part II

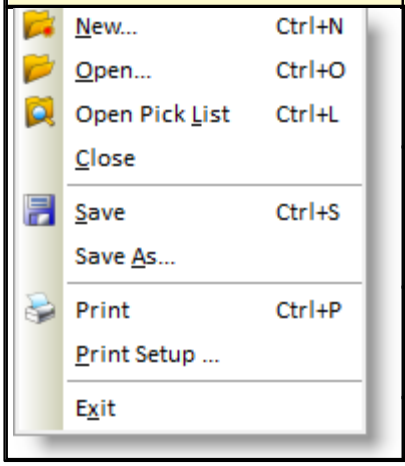
Chapter 2 - Integrated Environment

2 Integrated Environment

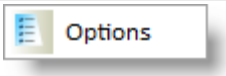
The Build Automator™ main window, or what we call an Integrated Development Environment - IDE for short, is uncluttered and easy to work with. All functionality of the software can be accessed from the main menu and the toolbar at the top.

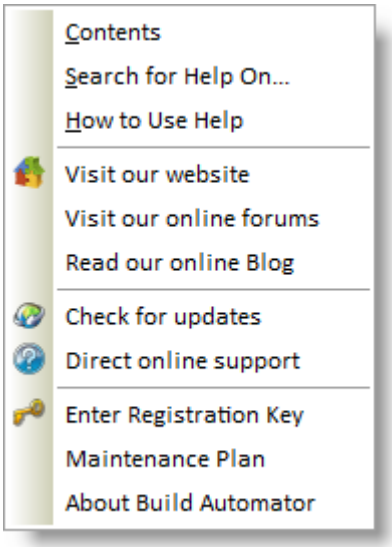


The main components of the menu are the File menu, the Tools menu and the Help menu.

File Menu	Item	Details
	New... 14	Creates a new project. You are prompted to create a new .aprj file which is then opened in the Project Window 30 for you to work with. You can use the Ctrl-N hotkey to access this option.
	Open... 15	Opens an existing project. You are prompted to open an existing .aprj file which is then opened in the Project Window 30 for you to work with. You can use the Ctrl-O hotkey to access this option.
	Open Picklist	This item opens the Picklist which makes it quick and easy to select recently used projects. You can use the Ctrl-L hotkey to access this option.
	Close	Closes the active project. The active project is which

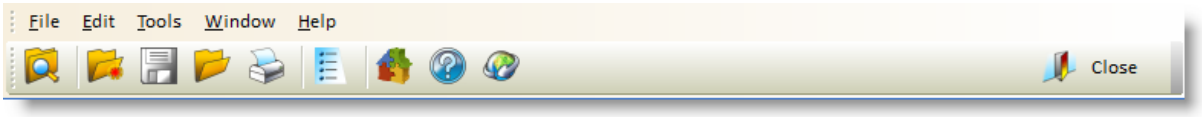
		ever Project Window 30 that has focus.
	Save	Saves any changes that have been made to the active project. The active project is which ever Project Window 30 that has focus. You can use the Ctrl-S hotkey to access this option.
	Save As...	Saves the project to a new file/location.
	Print	Prints the active project. The active project is which ever Project Window 30 that has focus. You can use the Ctrl-P hotkey to access this option.
	Print Setup	Calls the Printer setup dialog for you to select a default printer for the program. Rarely needed as you are always prompted to select a printer when printing.
	Exit	Closes the program. If any unsaved projects are open you will be prompted to save them before closing the program.

Tools Menu	Item	Details
	Options	Opens the Build Automator™ Options 17 window.

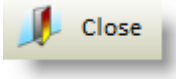
Help Menu	Item	Details
	Contents	Opens the context sensitive help.
	Search for Help...	Opens the context sensitive help.
	How to use help	Opens the context sensitive help.
	Visit our website	Opens a browser and navigates to our website at http://www.buildautomator.com
	Visit our Forum	Opens a browser and navigates to our forum website at http://www.buildautomator.com/forum
	Read our Blog	Opens a browser and navigates to our blog website at http://www.buildautomator.com/blog
	Check for updates	Checks for updates. If updates are available you will be prompted if you want to update the software. If you do not have a valid maintenance plan 148 the update will not be applied.
	Direct support	Opens a window where you can click on a button to access our direct online chat support. We may not

		always be available to take your call but you can leave a message with your email address and we will get back to you as quickly as possible.
	Enter Registration Key	Opens a window where you can enter your registration information and unlock the program if it is in demo mode.
	Maintenance Plan	Opens the Maintenance Plan 148 window where you enter your Maintenance Plan 148 code.
	About Build Auto...	Brings up the Splash Screen 23 with information about the Build Automator including version information for the main components.

The Build Automator Toolbar is equally easy to navigate through.

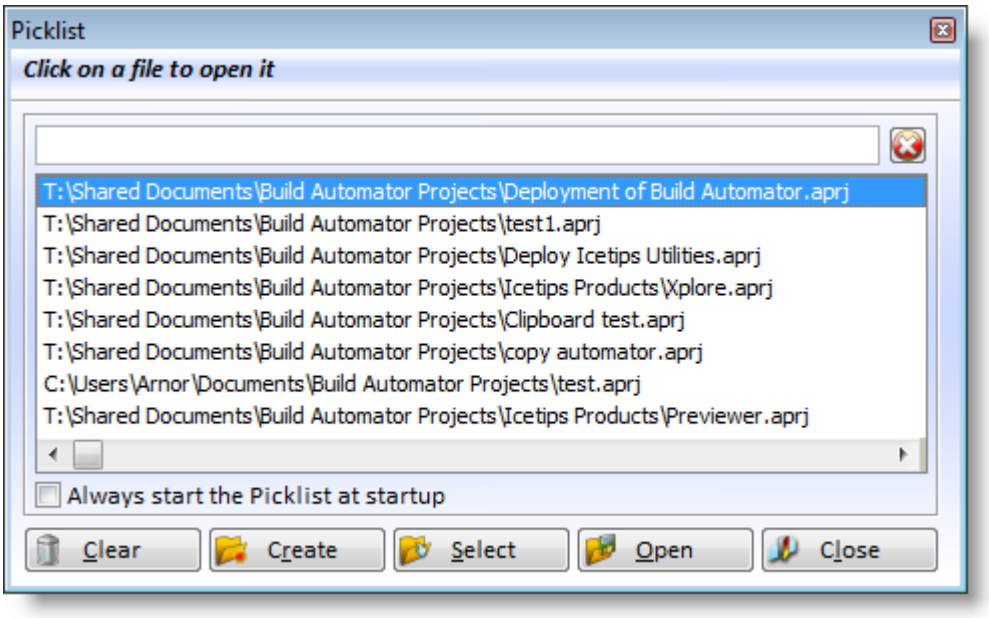


Toolbar Button	Details
	This item opens the Picklist which makes it quick and easy to select recently used projects. You can use the Ctrl-L hotkey to access this option.
	Creates a new project. You are prompted to create a new .aprj file which is then opened in the Project Window 30 for you to work with. You can use the Ctrl-N hotkey to access this option.
	Saves any changes that have been made to the active project. The active project is which ever Project Window 30 that has focus. You can use the Ctrl-S hotkey to access this option.
	Opens an existing project. You are prompted to open an existing .aprj file which is then opened in the Project Window 30 for you to work with. You can use the Ctrl-O hotkey to access this option.
	Prints the active project. The active project is which ever Project Window 30 that has focus. You can use the Ctrl-P hotkey to access this option.
	Opens the Build Automator™ Options 17 window.
	Opens a browser and navigates to our website at http://www.buildautomator.com
	Opens a window where you can click on a button to access our direct online chat support. We may not always be available to take your call but you can leave a message with your email address and we will get back to you as quickly as possible.

	Checks for updates. If updates are available you will be prompted if you want to update the software. If you do not have a valid maintenance plan 148 the update will not be applied.
	Closes the program. If any unsaved projects are open you will be prompted to save them before closing the program.

2.1 Picklist

The Picklist stores the last opened files and displays them in a list, so it is easy for you to pick what file to open. To open a file, just click with the left mouse button on the filename in the list. The file will open in the Project Window and the Picklist window will automatically close. You can open the Picklist from anywhere in the program by selecting "File | Picklist" from the main menu or by pressing Ctrl-L hotkey on the keyboard. You can resize the window and it will remember it's position and size next time you open it.



Clear — Click this button to remove all the files from the list. You will be prompted to confirm if you want to remove all the files. Please note that this only removes the files from the list and does NOT remove the actual files from your hard drive.

Create — Click on this button to create a new project file. The file is automatically added to the list. You will be prompted with a file selection window where you can select the location and name for the new project file. This is exactly the same as selecting "File | New" from the main program menu or clicking on the New button in the toolbar.

Select — Click on this button after selecting a project to open in the list.

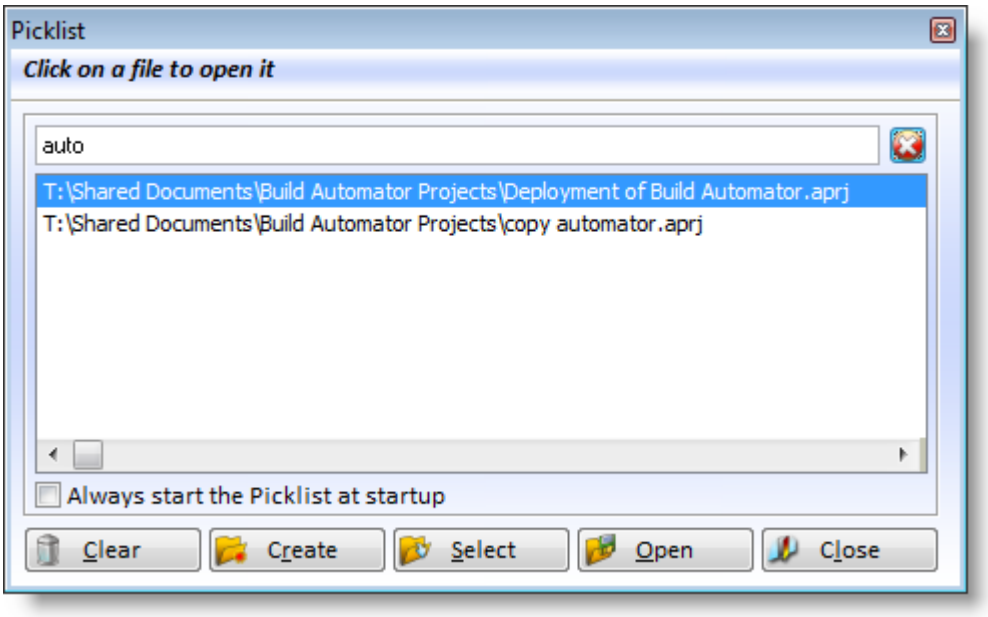
Open — Click on this button to open an existing project. The file is automatically added to the list. You will be prompted with a file selection window where you can select the file to open. This is exactly the same as selecting "File | Open" from the main program menu or clicking on the Open button in the toolbar.

Close — Click on this button to close the Picklist window.

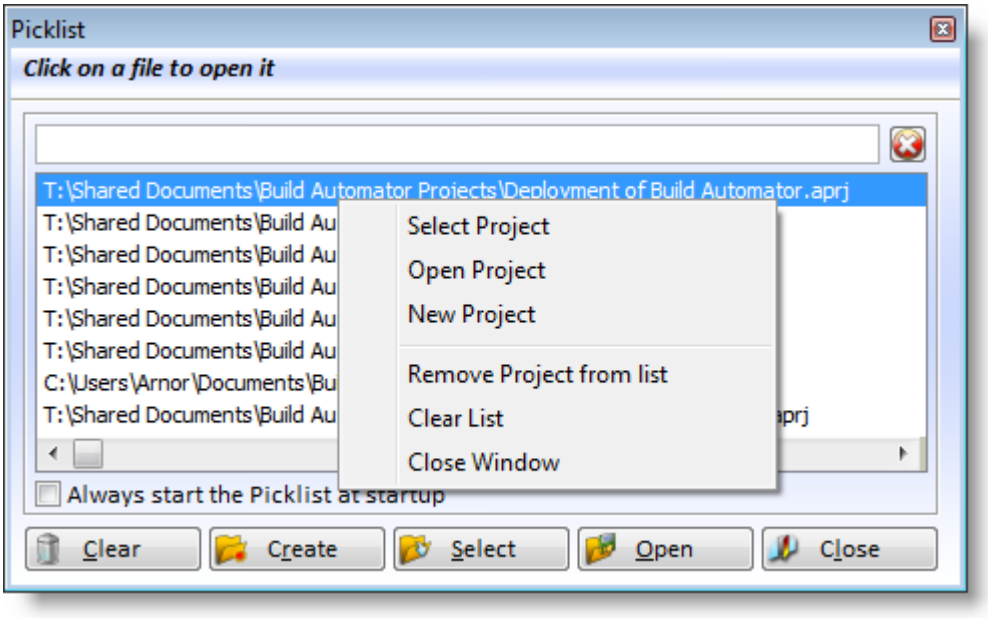
Always start... — Check this if you want the picklist to start up every time you run the Build Automator. Uncheck it if you don't want the picklist to show up. You can

change this setting on the Options 17 window.

The search entry above the file list can be used to filter the list. Note that the search only filters on the filenames. In the screenshot below it has filtered everything out of the list except filenames that include "auto" The filter is not case sensitive. To clear the filter, click on the red X button on the right of the locator or use the Ctrl-R key. To jump to the search entry use the Ctrl-F key.



Right click on the item that you want to remove and you will get a popup menu that you can select from.



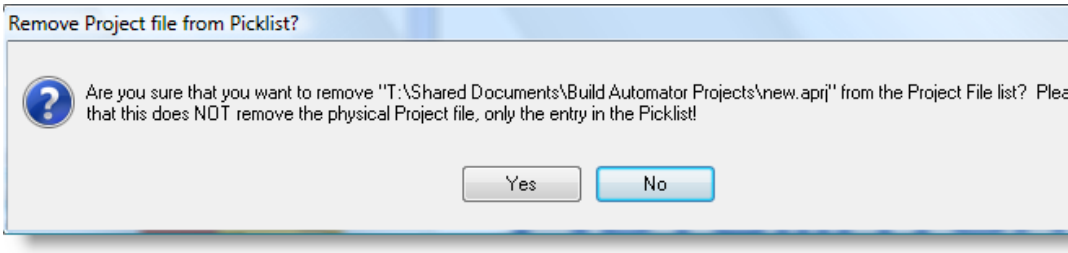
The popup menu has similar functions as the buttons at the bottom of the window, but also some additional ones.

Select Project Click on this menu item to open the file that is selected. The selected file is shown in the list with an underline like a hyperlink.

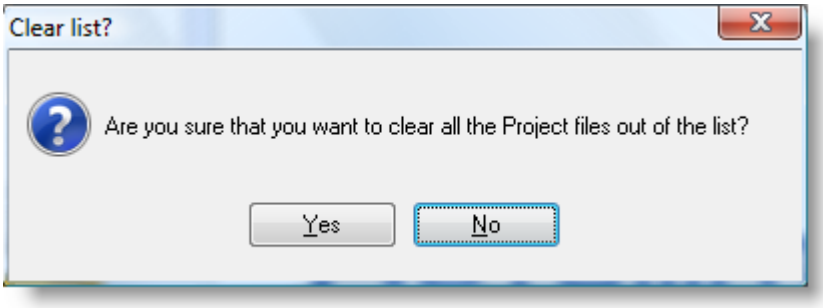
Open Project Click this menu item to open an existing project that is not in the list.

New Project Click this menu item to create a new project file.

Remove Project Click this menu item to remove the selected file from the list. The files below will be shuffled up. You will be prompted for confirmation.



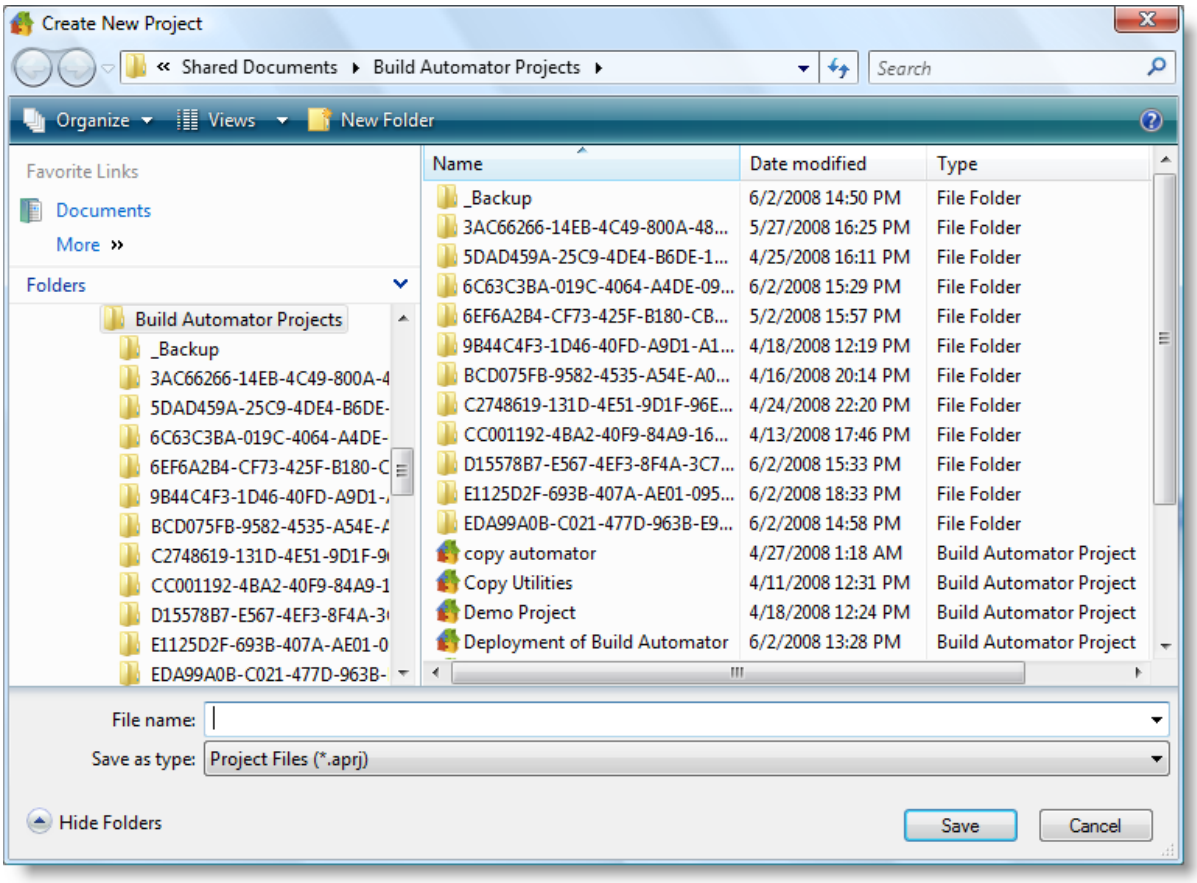
Clear List Click this menu item to clear the list completely. You will be prompted for confirmation.



Close Window Click this menu item to close the picklist window.

2.2 Create Project

To create a project, use the "File | New... 7" menu from the main menu, or click on the "New" button in the toolbar. You can also use the "Create" button in the picklist 11. You will be prompted for a filename and location for the new project.



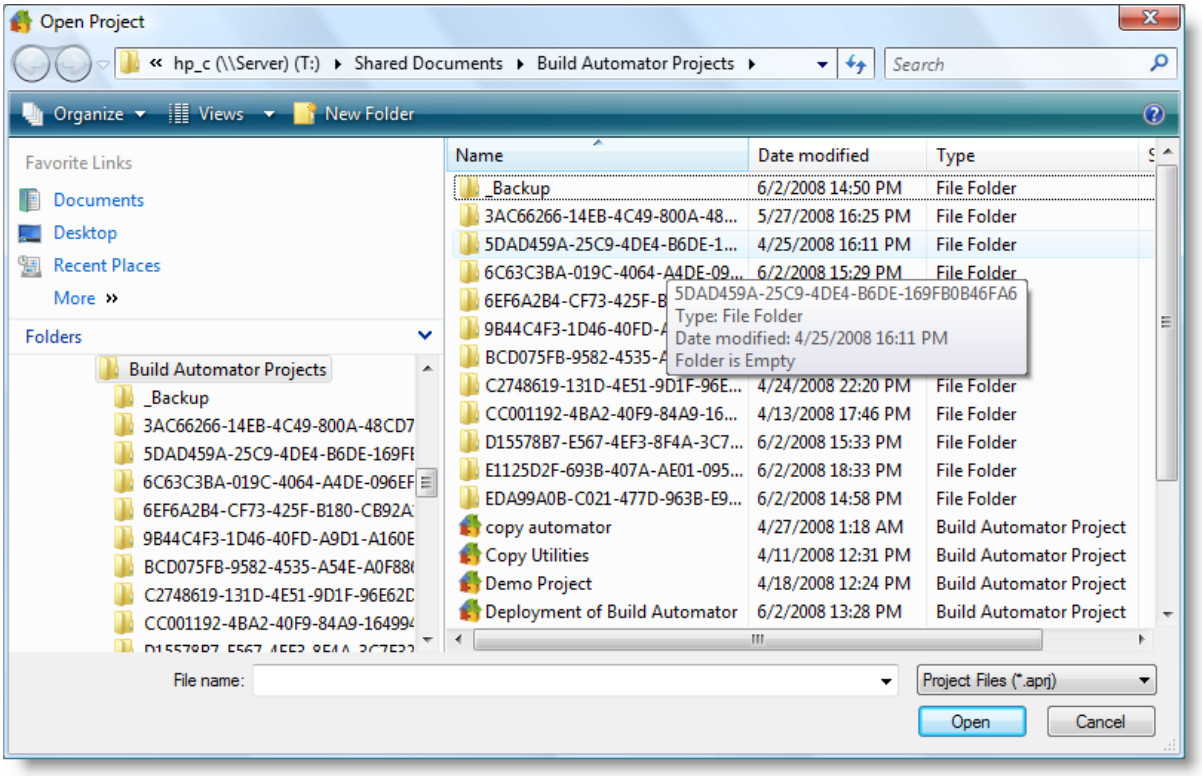
You can use any valid filename. Please note that each project is composed of 3 different parts:

1. The **Project file** that has an .aprj extension
2. The **Variables file** that has an .avar extension, but uses the same filename as the Project file.
3. The **Data folder** which uses the GUID from the Project as a folder name. You can see the GUID on the Project Properties 33 window at the top under "Project GUID" This folder contains all the Action Items datafiles.

When you create a new project it is automatically added to the Picklist 11.

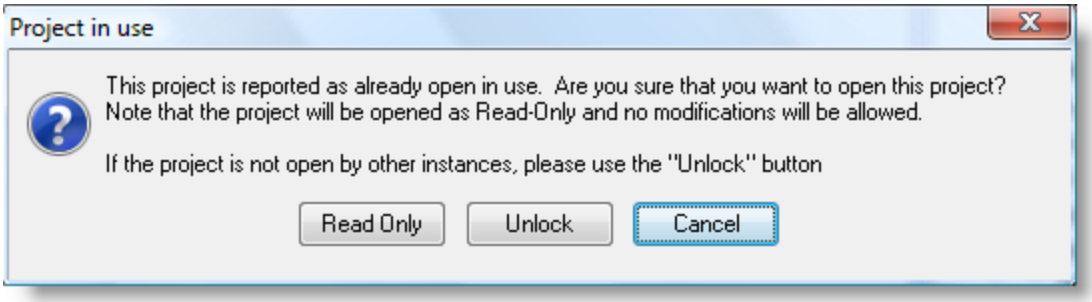
2.3 Open Project

When you open a Project, you get a standard file selection window to select a project file. The Build Automator Project files have a .aprj extension. When you install the Build Automator you can optionally associate the .aprj extension with the Build Automator. That allows you to create shortcuts for your projects and easily identify Build Automator files when you are browsing files in Windows Explorer because they will show up with the familiar Build Automator image.



When you open a project file, it is automatically added to the top of the Picklist 11. If the file already exists in the Picklist, it is not added again.

If the Project file that you are opening happens to be in use by another user or open in another instance of the Build Automator or if you have already opened it, you will be prompted how you want to handle it.



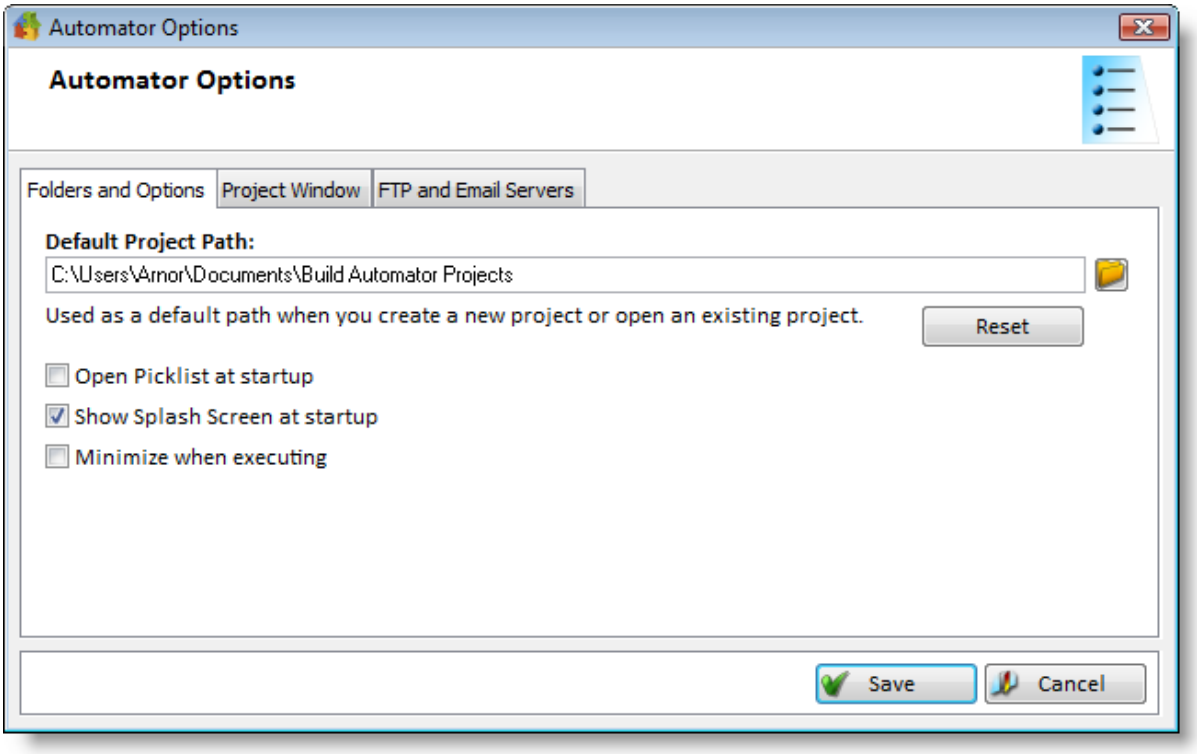
1. **Read Only** Using this option you can open the file as read-only, which means that you can still browse through it, but you cannot add anything and you cannot make any changes, but you can

copy Action Items to the clipboard to paste them into a different project.

2. **Unlock** If the Project was not closed properly, it will be left locked and marked as if it was in use. When that happens you need to unlock it when you open it. This should only be used if the Build Automator could not close the project normally last time it was opened. This could happen if for example the electricity went out before you closed the Build Automator, something forced you to turn the computer off or something happened to cause the Build Automator to crash. In those cases the file would still be identified as open and in this case you need to use the Unlock option to open it again. If the file is in use, you should not use the Unlock option - it can cause problems with your Project file.
3. **Cancel** This simply cancels the operation and does not attempt to open the Project file.

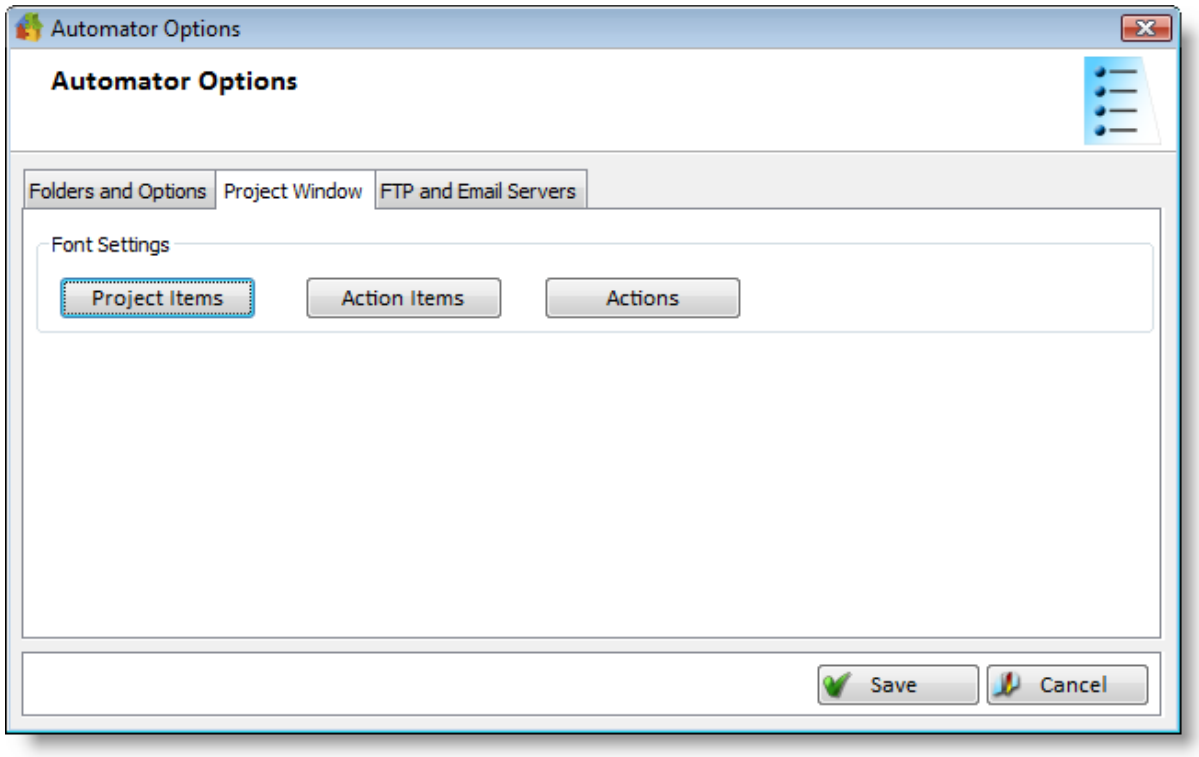
2.4 Automator Options

The Build Automator™ can be customized to serve your needs. We will be adding more new options, so the window you open may look slightly different from the screenshots, although the documentation will keep as much up to date with the software as possible.

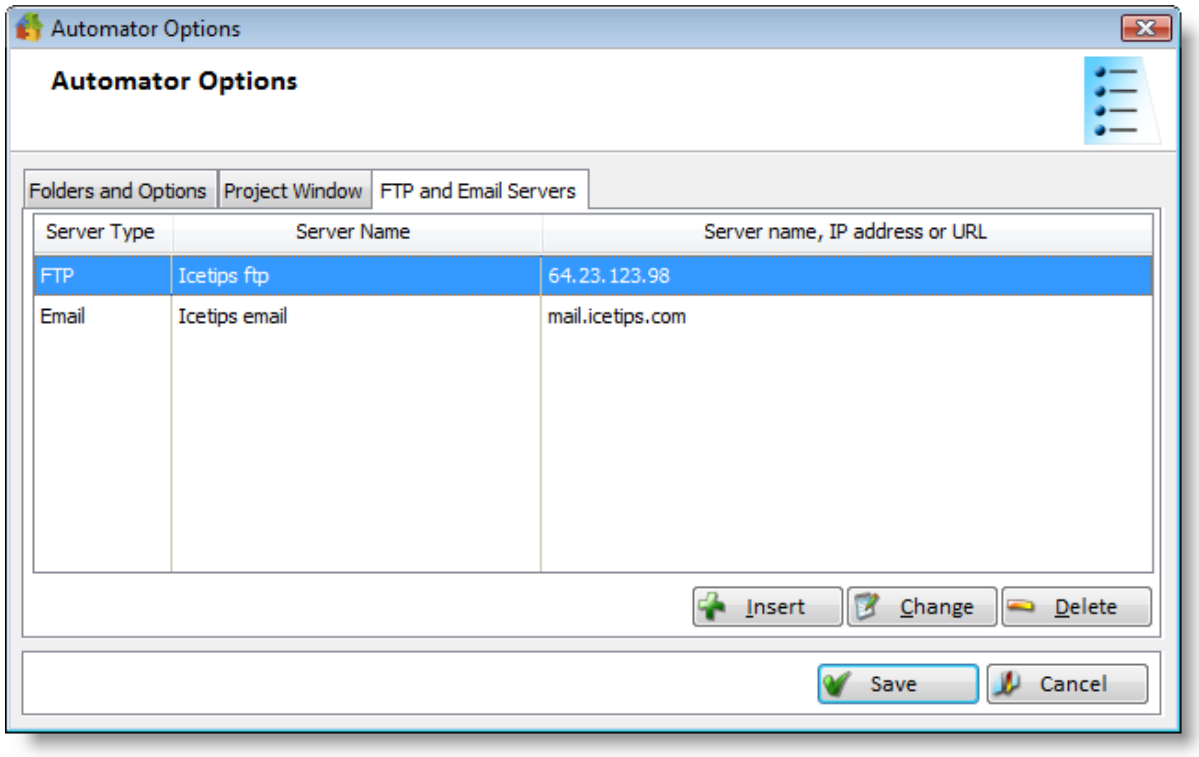


On the first tab you can specify the default project path. This path is used when you create or open a file as the default path. It defaults to your "My Documents\Build Automator Projects" folder, but you can change it to any folder. To set your default path back to the default "My Documents" folder, click on the Reset button. You will be prompted for a confirmation to reset it.

You can check or uncheck if you want to open the picklist at startup or not. You can also check or uncheck if you want to see the splash screen at startup.



On the "Project Window" tab you can set the fonts for each of the lists on the window. Note that the font settings are applied when the project window is opened.

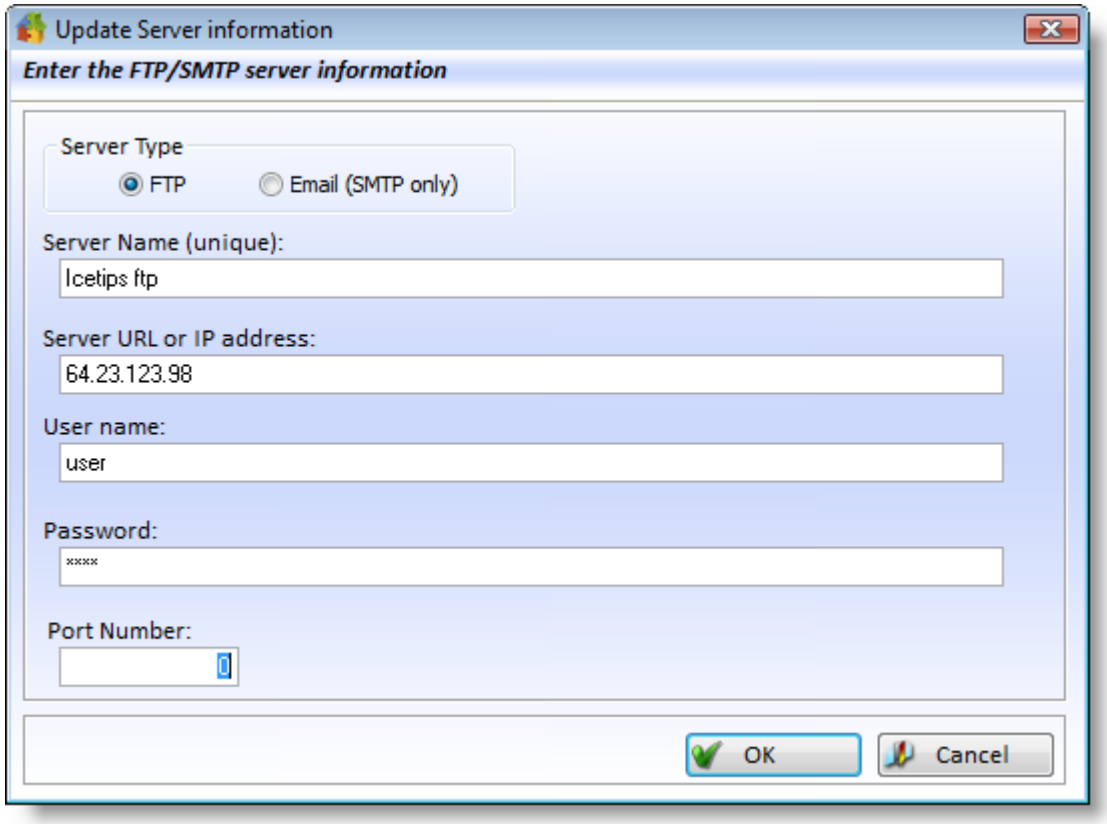


On the FTP and Email Servers tab you can set up FTP and email servers for Build Automator. Use

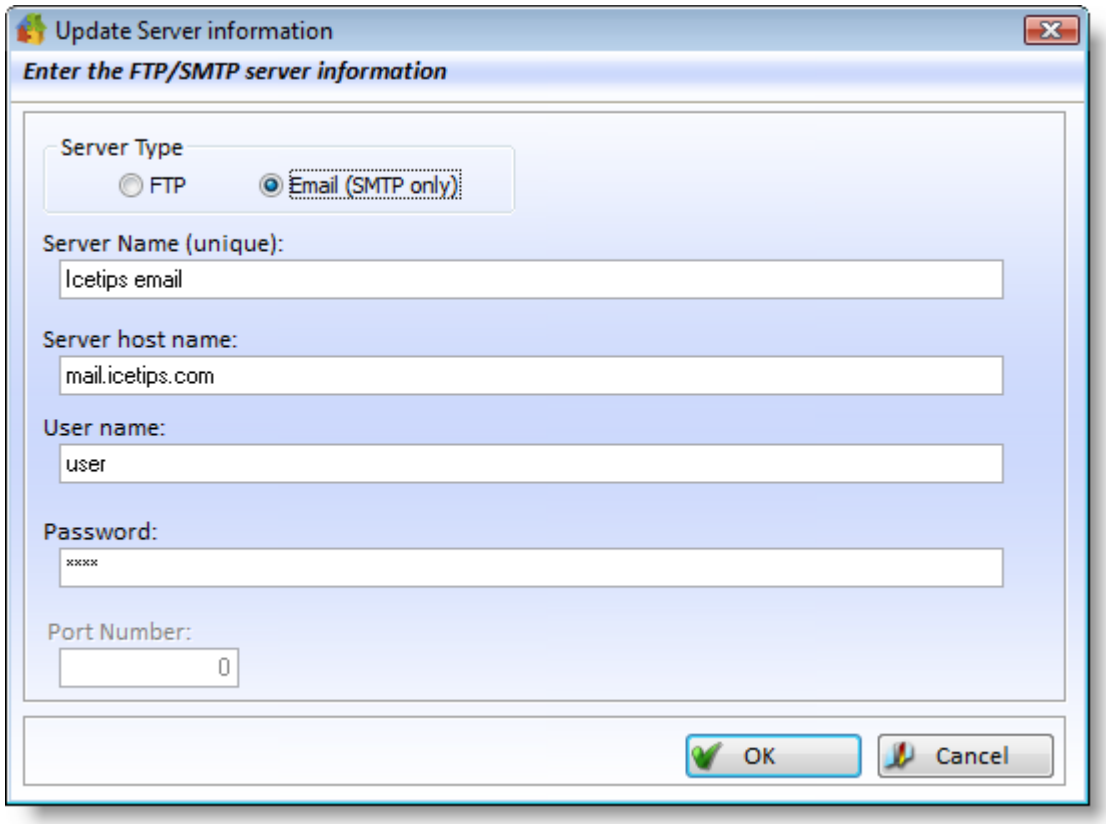
the buttons under the list to insert a new server, change an existing server or delete.

2.5 Update Servers

Build Automator allows you to set up two types of servers, FTP and SMTP.



The two servers are identical except that the Port Number is only available on the FTP server. Note that the information that you see on those screenshots is not valid.



As of the first build of version 2.0 the email servers are not in use in any actions, but we are working on options that will allow you to send emails upon completion of build process or if there are errors during automated build execution.

Properties	Explanation
Server Type	Select either FTP or SMTP.
Server Name	Descriptive name of the FTP server. This name is used in the server selection dropdown.
Server IP or URL	Here you need to enter either the IP address or the URL of the ftp server.
Username	Enter your username for the server.
Password	Enter your password for the server.
Port number	Currently not used but reserved for future use. Specify the port number to use for the server connection.

2.6 Command Line Parameters

The Build Automator can be called with the name of a Build Automator project file (.aprj file) to open it. You can also add the /E flag after the filename to execute it. Note that this changed in version 1.30.100 compared to what it was in version 1.20

/E	The Build Automator™ can be called with a parameter that tells it to run a specific project. Note that currently this option runs the whole project and does not respect the "Allow Full Execution" option on the project. For example if "Allow Full Execution" is turned OFF in the project you are executing on the command line it will still execute the whole project. We will provide an optional command line parameter that will tell the Build Automator™ what project item to execute. This makes it easy for you to create shortcuts to specific projects that you want to execute. **Example:** BuildAutomator.exe "C:\projects\MyProject.aprj" /E
/NL	This flag tells the Build Automator **N**ot to display the **L**ogfile at the end of an execution process. This is only effective if the /E flag is also used. *- added in version 1.30.150- June 24, 2008*

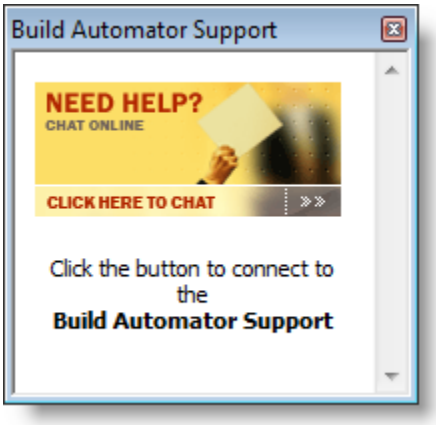
2.7 Splash Screen

The splash screen is used both as a splash screen for the program and also used as the About screen. When it is used as a splash screen it will close automatically after 3 seconds. When used as the About screen it will stay open until you click the Close button or press the Esc key on the keyboard.

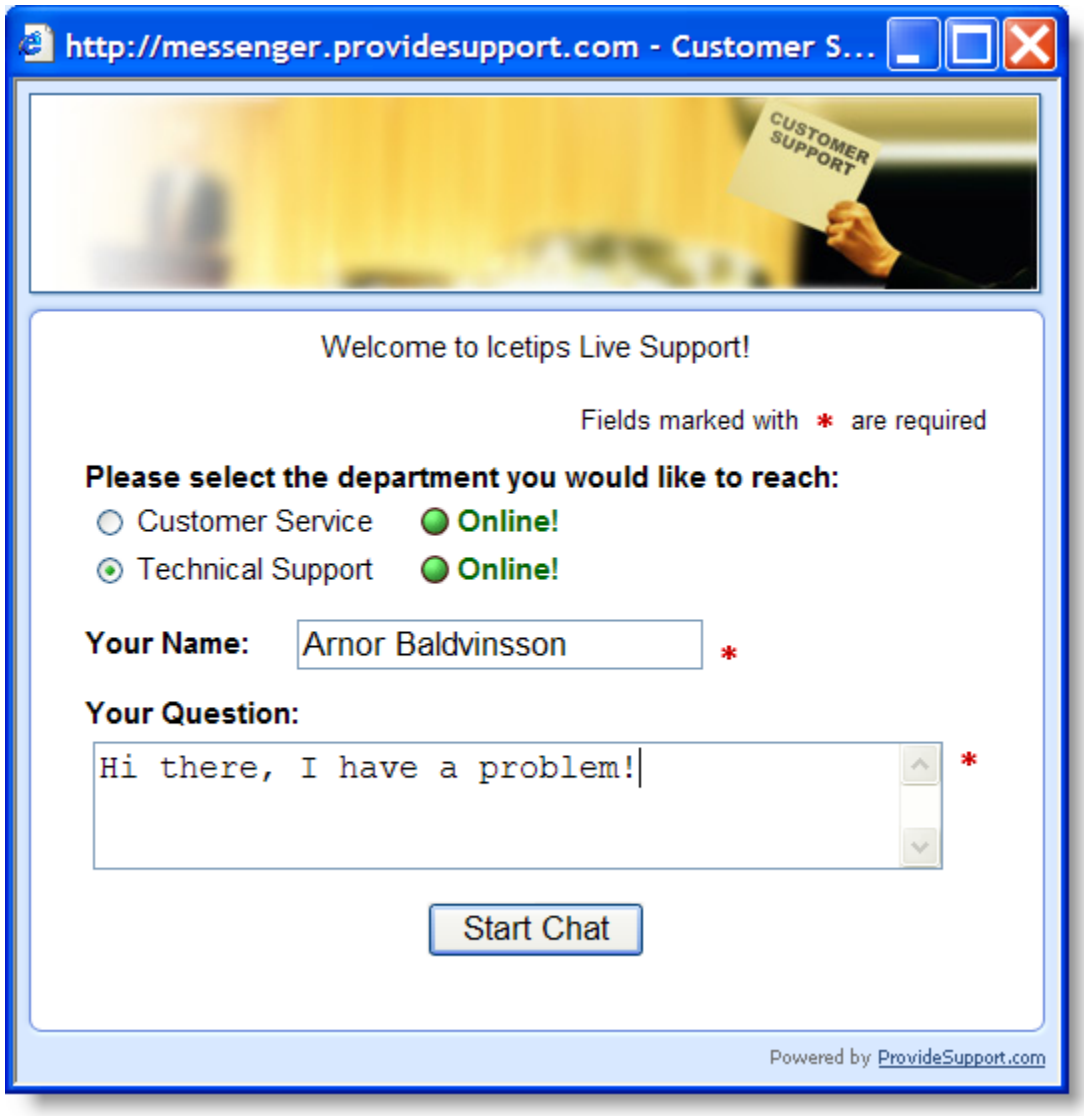
The splash screen shows some version information and information about where the program is located. You can check or uncheck the "Show Splash Screen at startup" to show it or not show it.

2.8 Direct Online Support

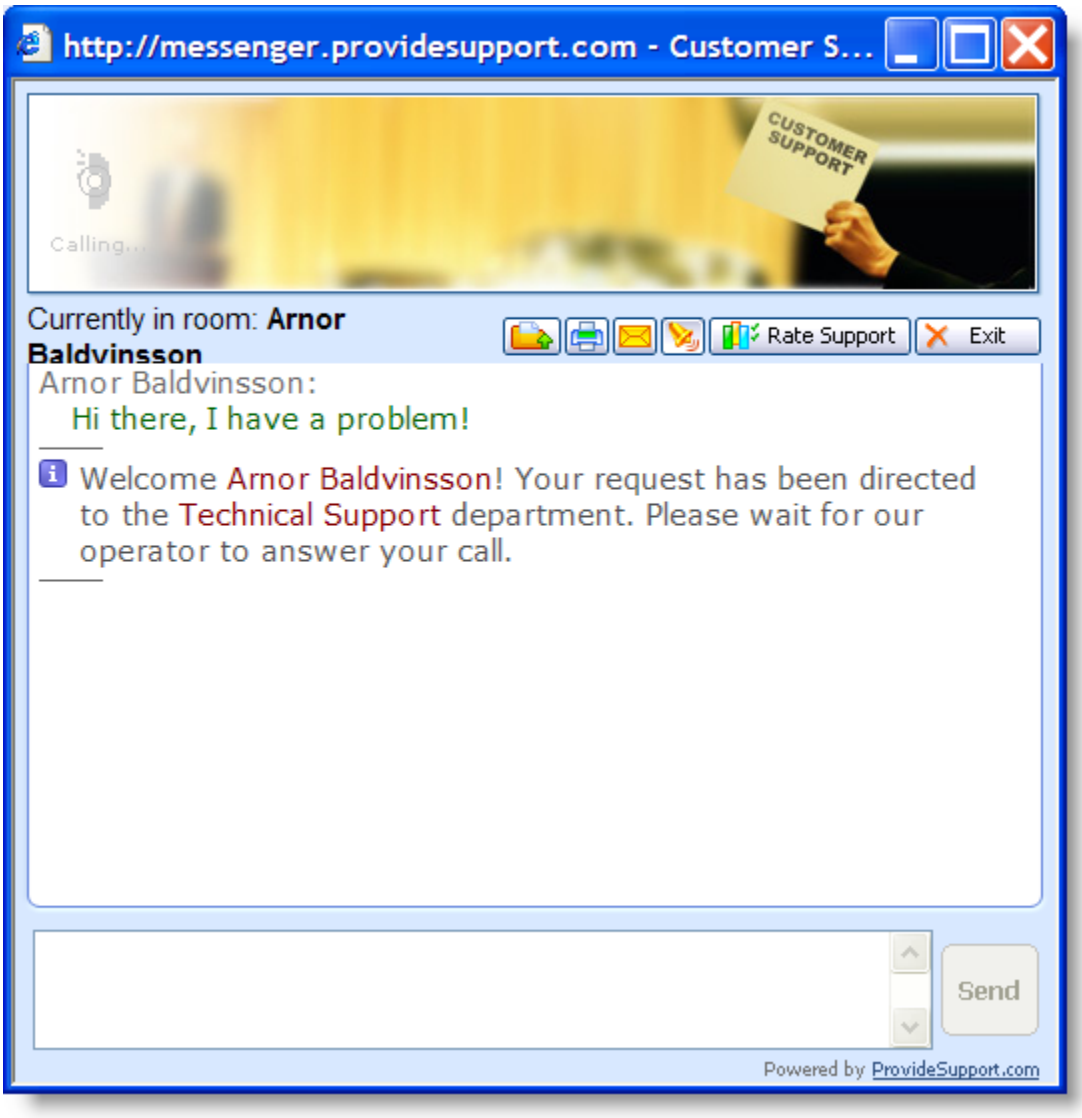
This option is available from the "Help | Direct online support" in the main menu. This gives you a window with a "Need Help?" button.



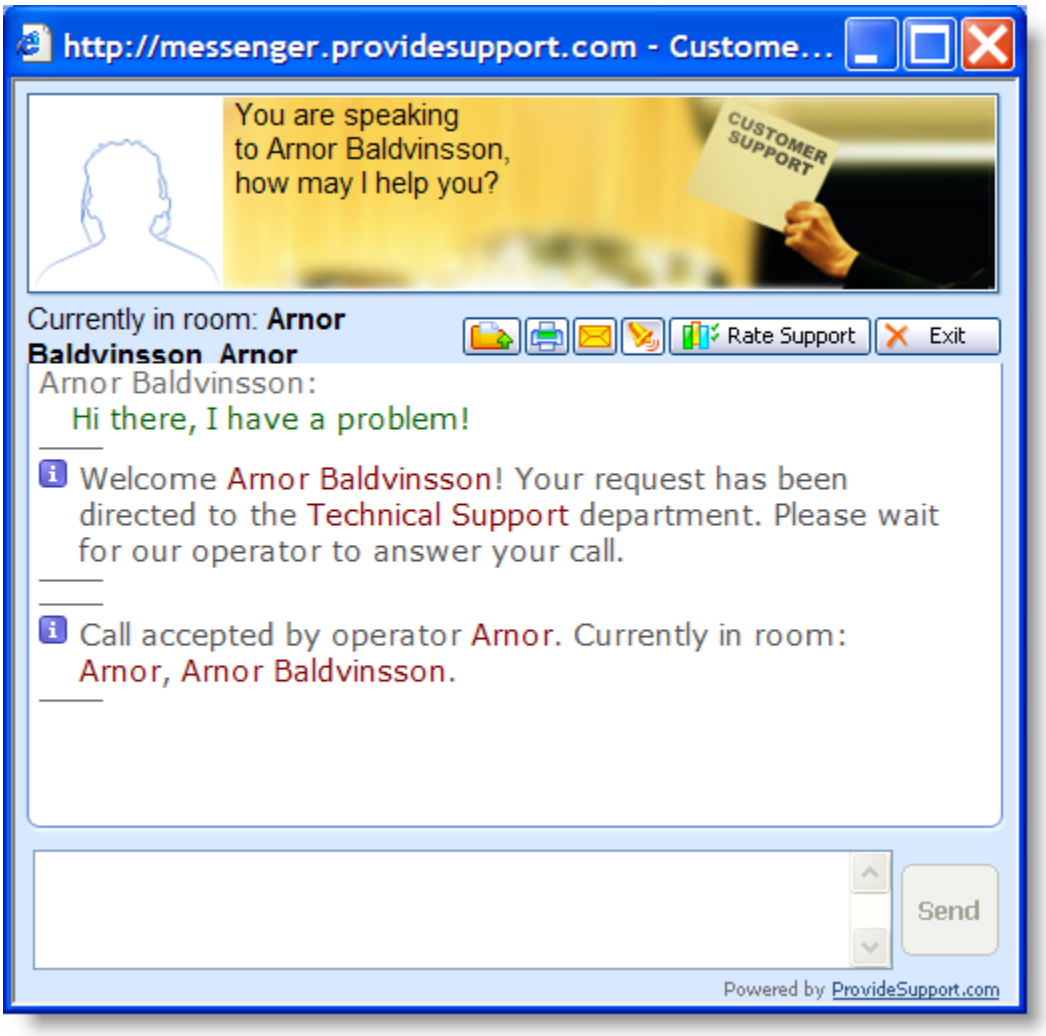
Click on the button and you will get a browser window that looks like this:



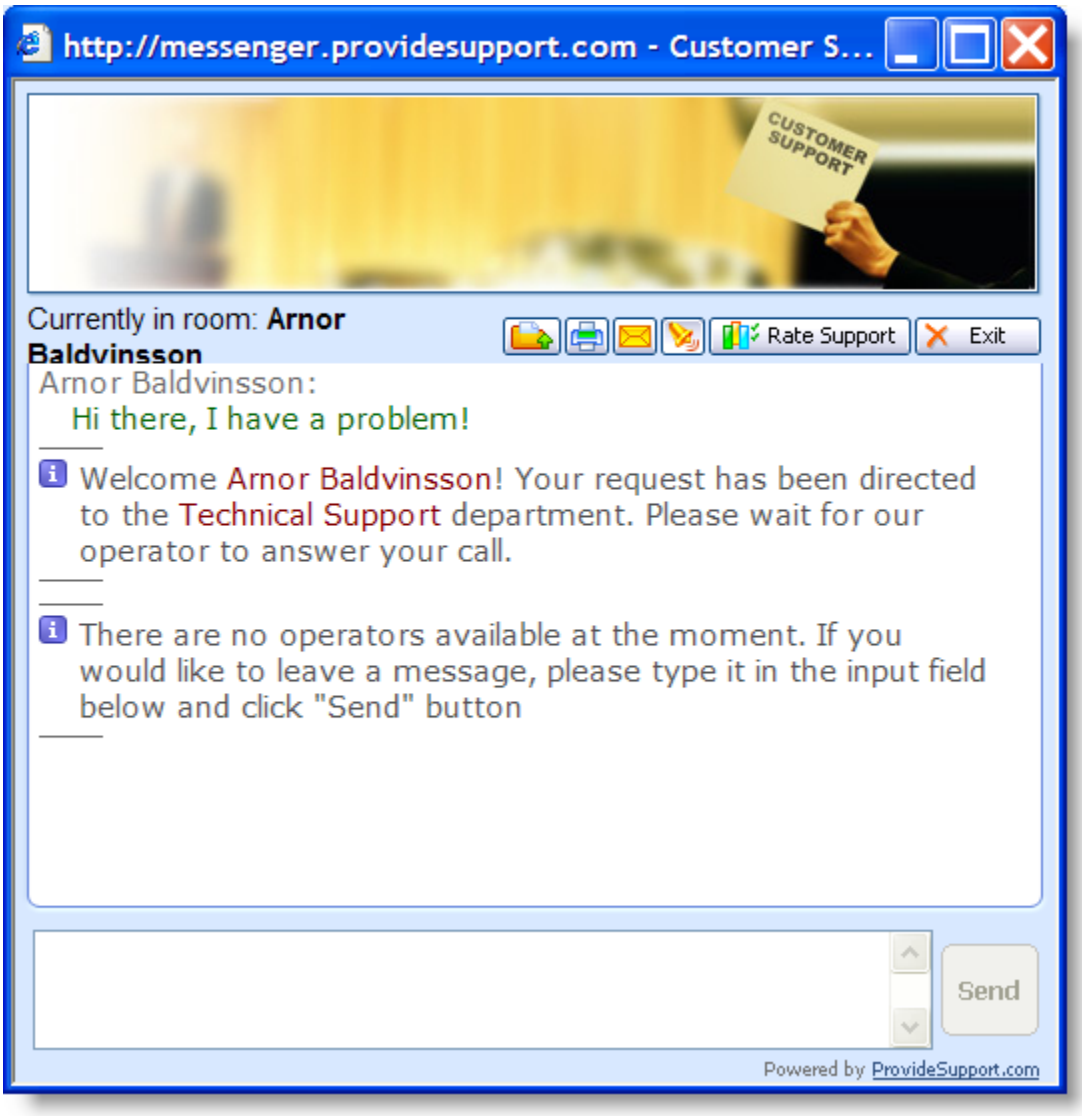
When you hit the "Start Chat" button, your message is immediately displayed on our support computers, ready for an operator to take your "call" Once you press the "Start Chat" button you will get a new browser screen:



When someone picks up on our end, the screen will display more information:



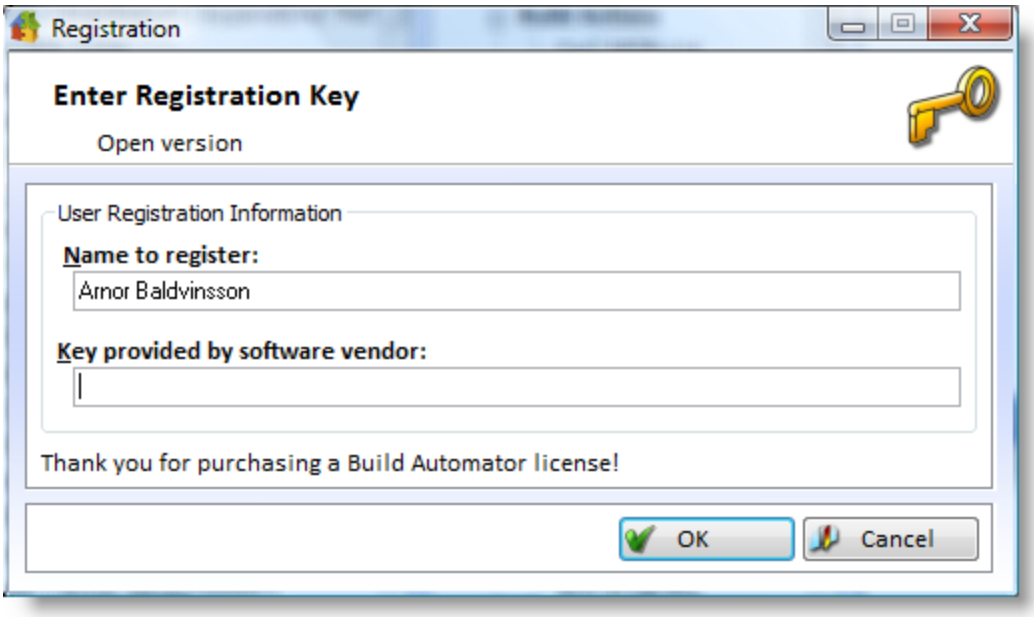
If, on the other hand, nobody is available to take your call, the screen will also display it and give you an option to leave us a message:



In this case, please leave us a message with your email address and we will get back to you as soon as possible. Please note that the "Send" button is not enabled unless you type something into the text box at the bottom to prevent empty messages being sent to us.

2.9 Enter Registration Key

This window is used to unlock the software and enter your registration key that you get from Icetips Alta LLC.



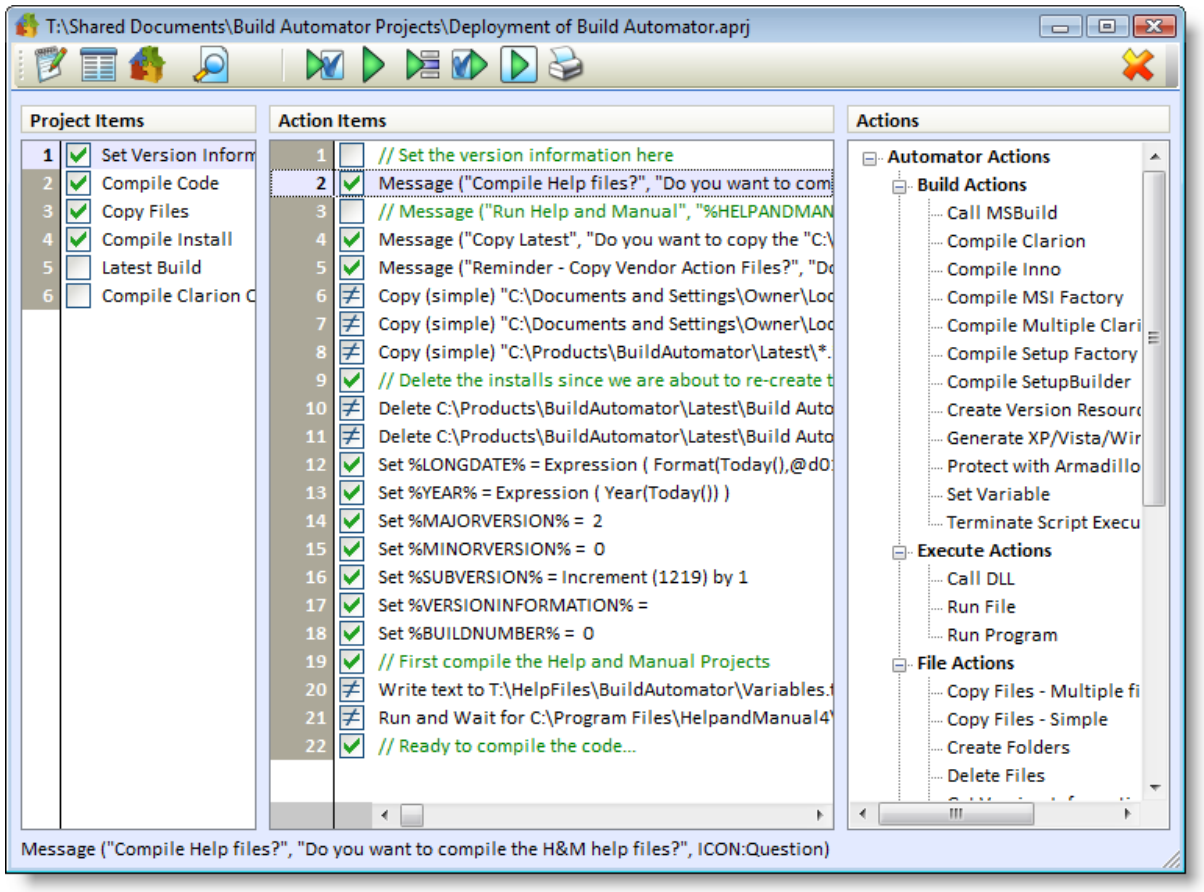
Name to register: This is the name that your registration is linked to. In our emails this will be listed as "Registered Name" It must match exactly with the registered name as it is shown in our registration email.

Key: This is the key as we provide it in our emails listed as "Key code". It must match exactly with the key code as it is shown in our registration email.

Once you have entered the keycode and it is verified the software will be unlocked and you can continue to use it. To receive free upgrades and updates to Build Automator you must have a valid maintenance plan 148 subscription.

2.10 Project Window

The Project Window is the main work window of the Build Automator™'s IDE. This is where you create your automation project.



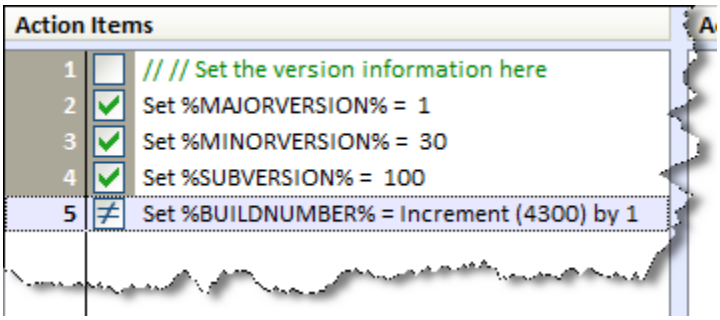
On the left are Project Items [39]. This is a way for you to group together related actions, such as copying of files, or compilation of code. On the right are the Actions [42] that are available for use. Double click on the action to insert it into the Action Items [40] list for the selected Project Item. Each action has an update window that will be displayed when the Action Item is created. The checkboxes on the Project Items and Action items list control is that particular item will be executed or not. Double click on the checkbox to toggle between checked and unchecked. This way you can quickly exclude or include sections of the project. Right click on the Actions Items list, or use the Application Key on the keyboard, to to bring up a popup menu.

You can change the size of the Project Items, Actions Items and Actions lists by placing the mouse between those elements and click-and-drag to change the sizes. A splitter bar wills how up between the controls and allow you to change the sizes.

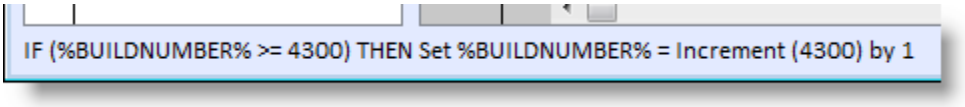
At the top of the window is a toolbar with several buttons that affect the project and the item list.

	Opens the Project Properties 33 window.
	Opens the Project Variables 34 window
	Opens the Create Shortcut 44 window
	Opens the Log File Viewer 48 window with the last logfile
	Executes all Action items from the start of the script up to and including the selected item. If the Execute All is selected, it will start with the first Project Item. If Execute Project item is selected, it will start at the first Action item in the selected Project item and end on the selected item.
	Executes all Action items from the start of the scrip to the finish. If the Execute All is selected, it will start with the first Project Item. If Execute Project item is selected, it will start at the first Action item in the selected Project item and end on the last Action item.
	Executes all Action items in the selected Project Action item.
	Executes from the selected item to the end. If the Execute All is selected it will execute from the current Action item to the final Action item in the last Project Item. If the "Execute Project item" is selected it will execute from the current Action item to the last Action item in the selected Project Item.
	Executes the selected item. If multiple items have been selected it will execute all selected item. If only one item is selected it will execute only that one item.
	Print the Project 46.
	Move the selected item up. Alt-Up-arrow key.
	Move the selected item down. Alt-Down-arrow key.
	Closes the Project. If the project has changed you will be prompted to save if it has changed since you saved last time.

Each Project Item and Action Item can be checked or unchecked. When unchecked the item will be excluded from execution. Each Action Item can also be conditional. In that case a checkbox with ≠ is shown instead of the checked or unchecked boxes. See screenshot below. The top Action Item is unchecked and would not be included in execution.

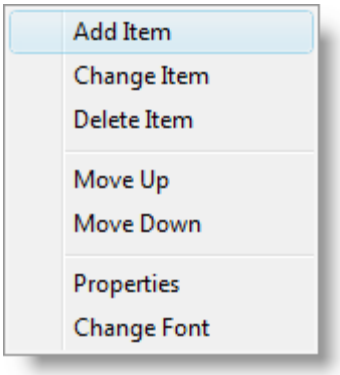


The bottom item is a conditional item, indicated with the ≠ in the checkbox. When a condition is present, the Action Item is displayed at the bottom with an IF (condition) THEN statement - see screenshot.



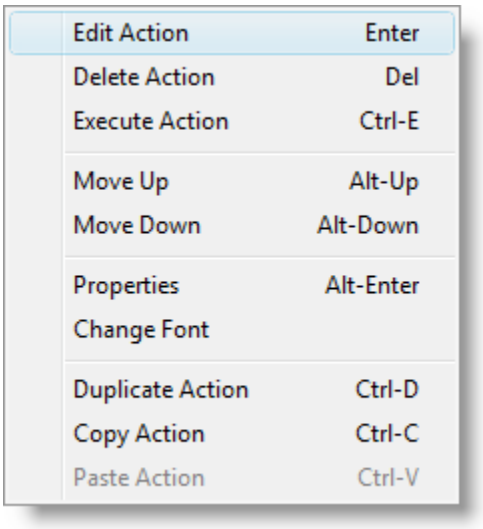
You can use variables in the conditions, but the condition must result in a logical true or false statement. The condition is evaluated and if the result is either an empty string or numerical zero, the Action Item is not executed. If any other values are returned by the evaluation of the condition, the Action Item will execute.

If you click with the right mouse button, or press the Application Key, on the Project Items 39 list the program will show you a popup menu.



You can use this popup menu to easily Insert, Change or Delete the selected Project Item 39. The Change Font will bring up a font selection window where you can select whatever font you are comfortable with for the Project Items list.

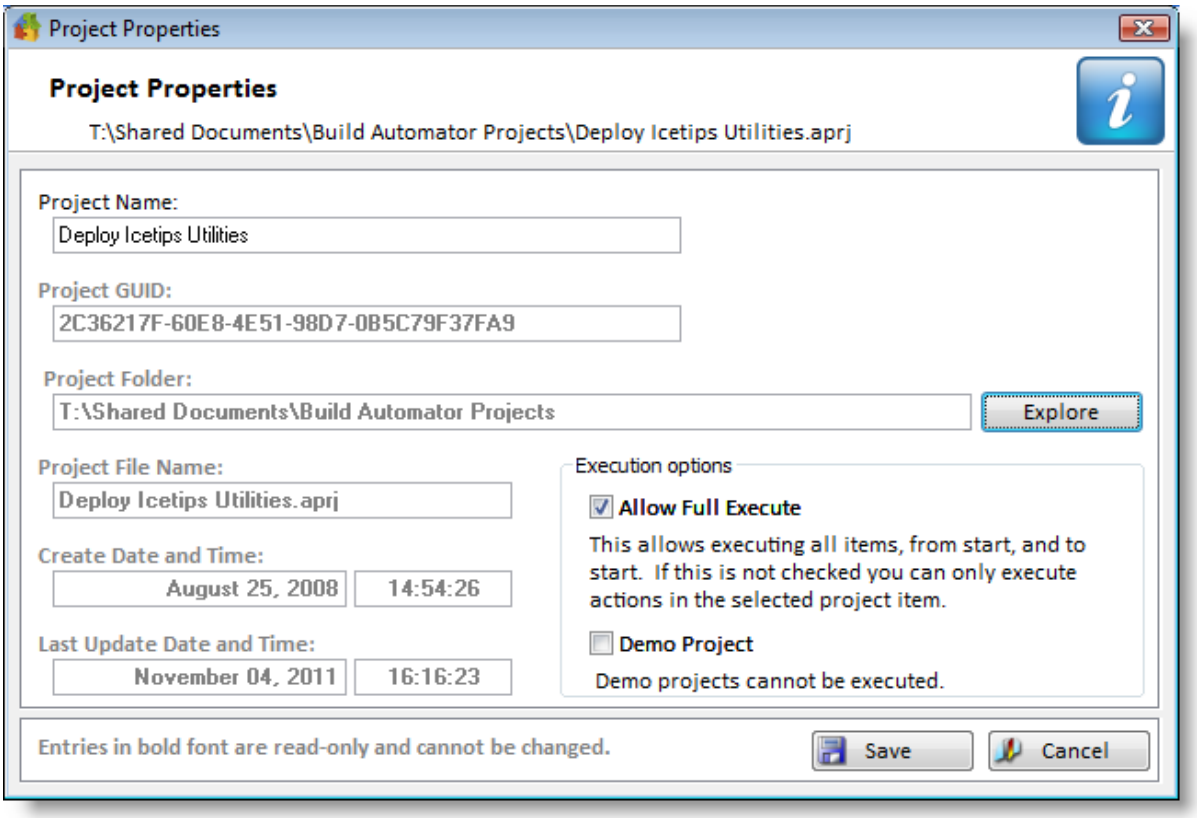
If you click with the right mouse button, or press the Application Key, on the Action Items 40 list the program will show you a popup menu.



You can use this popup menu to easily Insert, Change or Delete the select Action Item 40. The Properties option will bring up the Action Items Properties 43 window. The Change Font will bring up a font selection window where you can select whatever font you are comfortable with for the Actions Items list.

2.10.1 Project Properties — Project Window

The Project Properties are mostly set by the system when the project is created and most of those properties cannot be changed. Currently you can set the **Project Name** which will be used on the project reports. You can also set the "**Allow Full Execute**" which select if you want to allow the whole project to be execute or just the currently selected Project item. By not allowing the whole project to be executed, you can easily have project items that are not really related, but may seem logical to have in one place. This could for example be a project for backing up all files related to a certain software project. It might not be a bad idea to have a single project for that and have multiple project items, one for each software project that you are working on. That way you can execute the copy process for each software project without risking accidentally to set off the whole copy process which could take a long time and might not be appropriate.



Note that if you check the "**Demo Project**" you no longer can execute the project. This should only be used if you want to create a project to send to someone for demonstration purposes. The reason it cannot execute is that unless the recipient has all the tools in the project and has all the files in the appropriate places and everything is basically identical, the execution of the project could have completely unpredictable results. Of course if you know that everything is ok, for example if you are copying files based on standard CSIDL folders or executing programs that you know exist on the recipient machine, you can share projects. Note that for that you will need to copy both the project file (in this case new.aprj) and the project variables (new.avar)

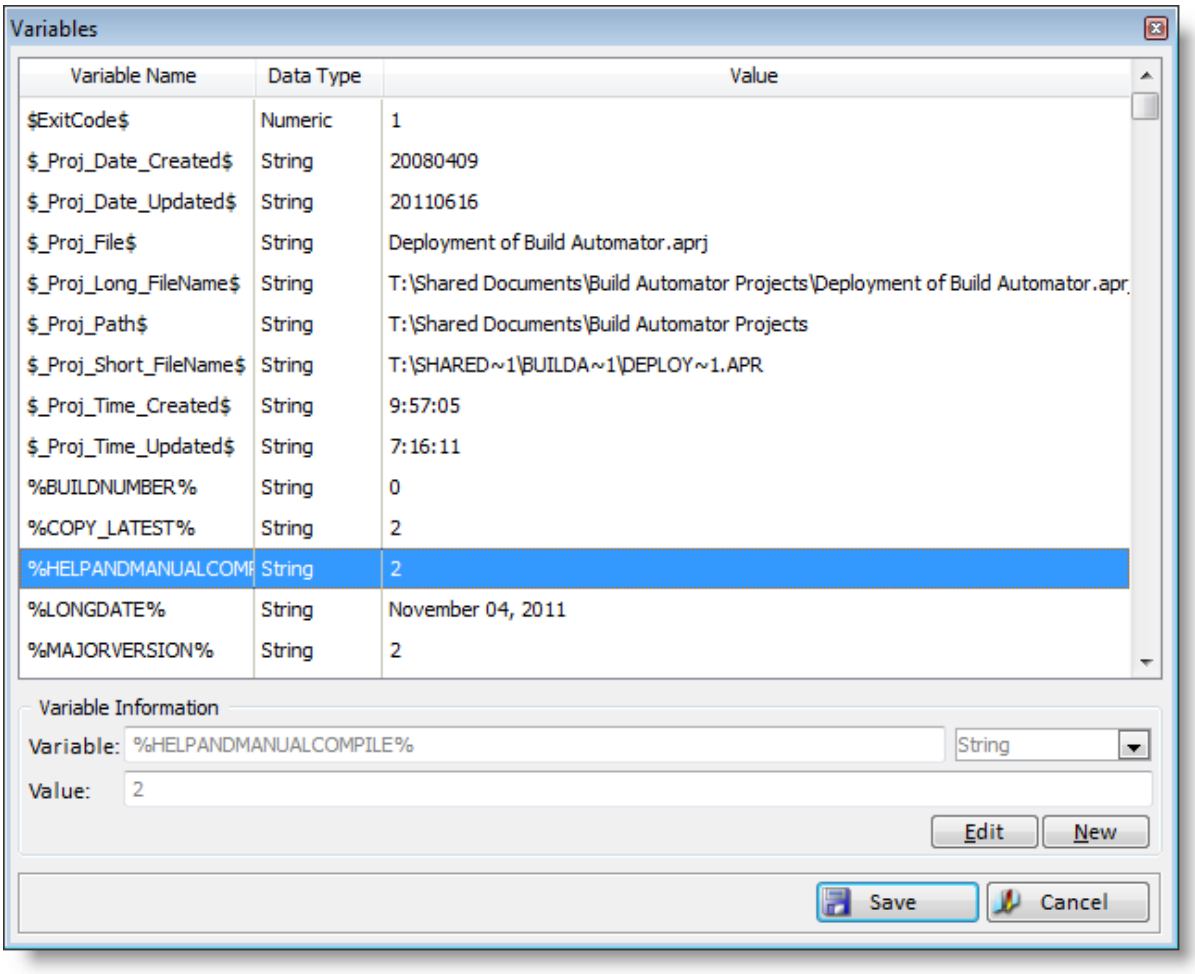
The Explore button opens a Windows Explorer window on the folder where the project is.

For security reasons we do not allow you to change the demo project back to a normal project. If you make a mistake on a project that you didn't want to make a demo project please let us know and we will fix it for you.

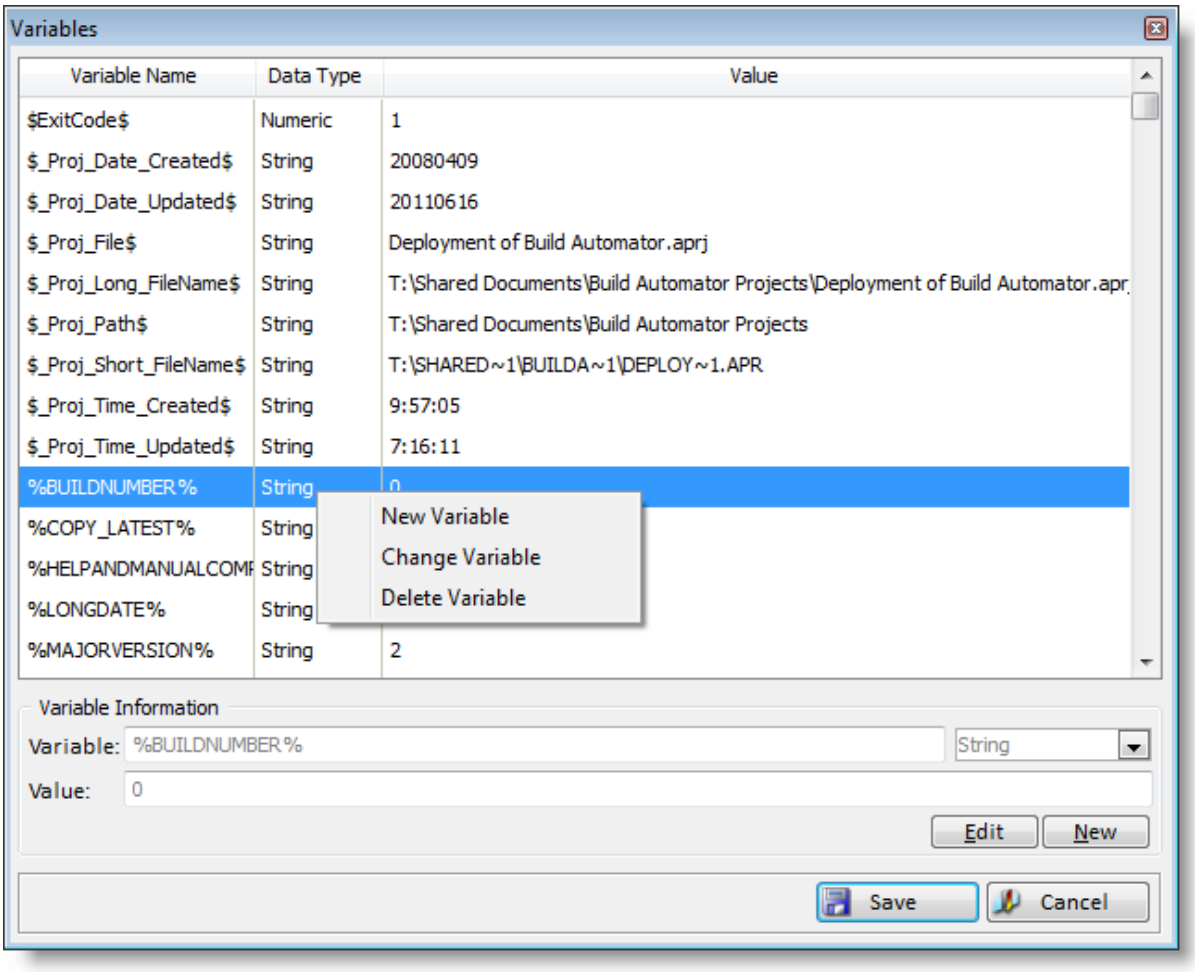
2.10.2 Project Variables — Project Window

Each project can have it's own variables. These variables can be used to store data and add information into the various update forms. When you select a variable to use in one of the update windows, you will also have access to system variables which contain certain system related information such as date, time and the various special folder locations, such as Program Files, My Documents etc. etc.

You can click on the list headers to sort the list. By default Build Automator adds several project variables to the list and those variables are kept updated internally. Those variables all start with "$_Proj" and they include the name of the project, when it was created, last updated, name of the project file, etc. By default the variables are listed in alphabetical order, so the project variables end up at the top.

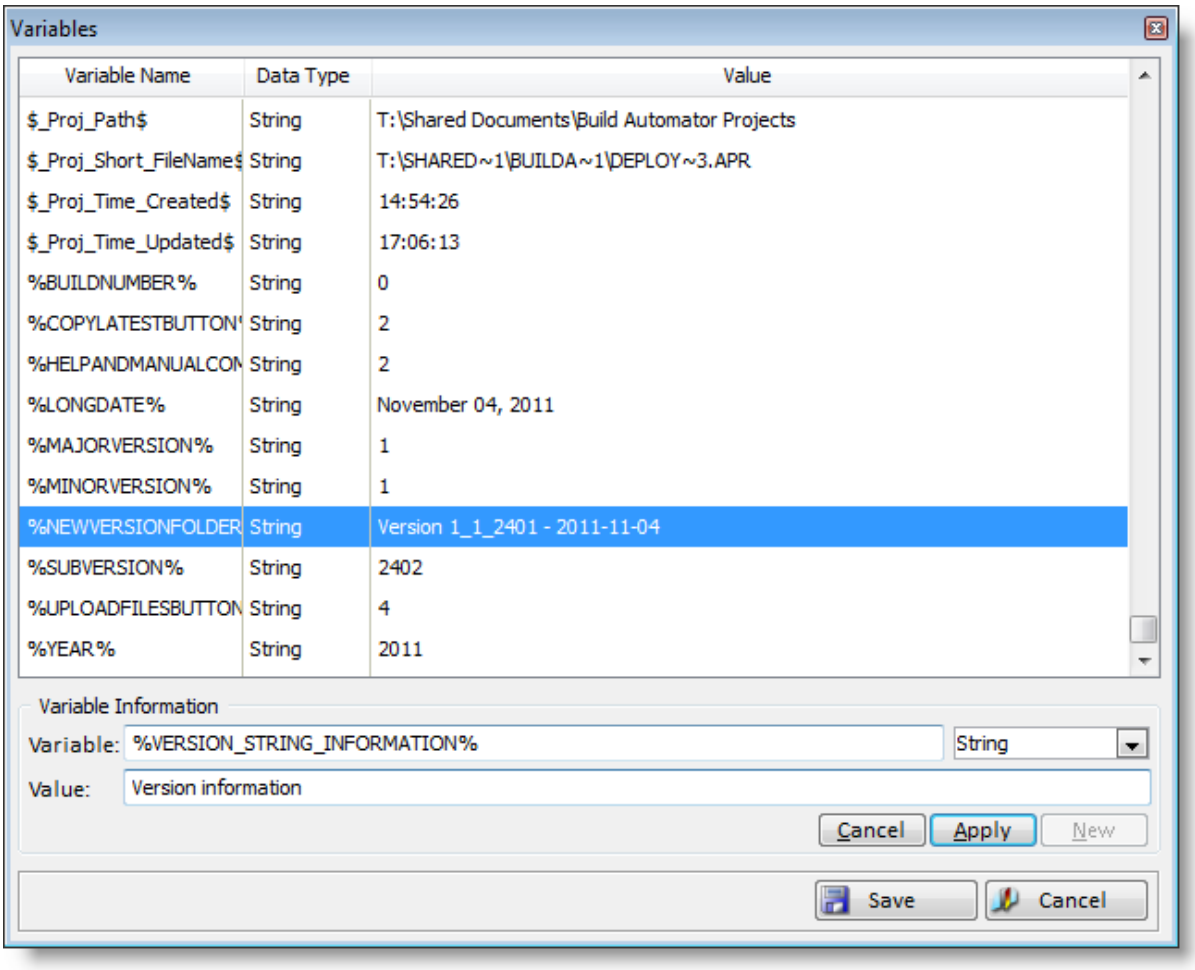


To create a new variable, you can click on the New button, hit the Insert key on the keyboard or right click on the list and select "New Variable".
To edit the selected variable, you can click the Edit button, hit the Enter key on the keyboard or right click on the list and select "Change Variable".
To delete the selected variable hit the Delete key on the keyboard or right click on the list and select "Delete Variable". You will be prompted to confirm deletion of the variable.
To cancel editing or inserting a new variable, press the Esc key or click the Cancel button next to the Edit button. The Alt-C key on the keyboard also cancels the editing/insert.

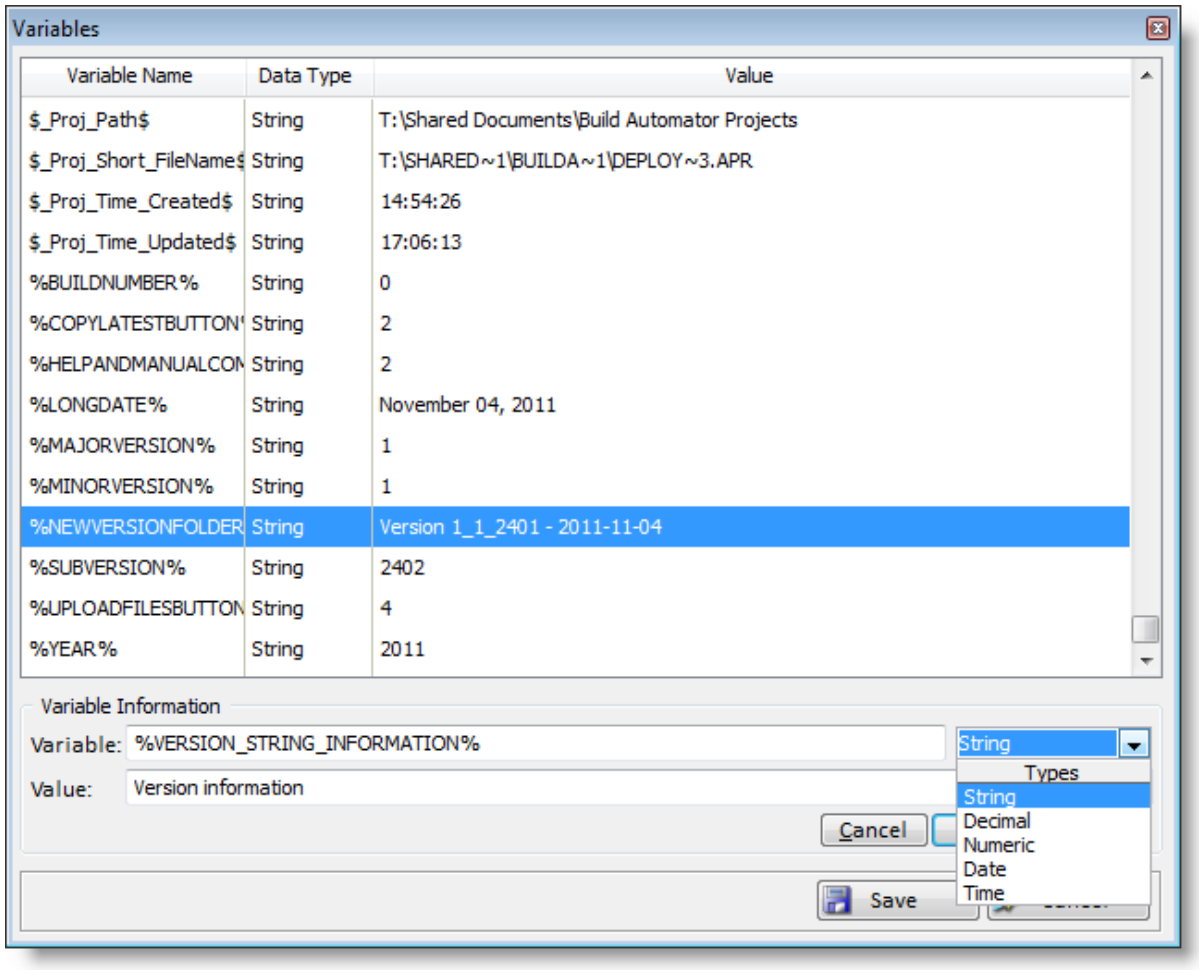


All variables that you create must start and end with a percent character - %. This uniquely identifies them in entries so the variable can be expanded to it's correct value. System Variables start and end with a dollar character - $. If you do not place the % before and after the variable name, the Build Automator™ will do it for you. If you notice that the variable name is not being formed correctly, please report it to our bug tracking system at http://www.buildautomator.com/bugtracker.php

The Variable name, including the two % at the beginning and end, is limited to 40 characters. The static variable value is currently limited to 255 characters. Variable names are forced upper case. Note that SOME system variables are NOT upper case, such as $ExitCode$, $ToDay$ and Now as well as the $_Proj_ variables. You do not have to type the enclosing %, Build Automator will add them as you accept the variable name.



Variables can have one of 5 data types, String, Decimal, Numeric, Date or Time. When you are working with Actions, you can also enter variables to use in certain actions. These variables always default to the String type and cannot be changed when you enter them. You can, however, change them in the project variables window to the proper data type if String is not suitable.



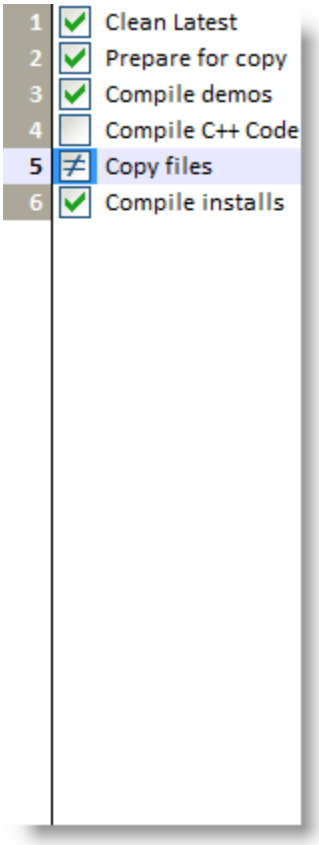
The formats are hard coded to provide 100% consistency in data types.

String @s255
Decimal @n-20.4
Integer @n-14
Date @d10-
Time @t4

Data Type	Formatting	Explanation
String	@s255	255 character string without any special formatting
Decimal	@n-20.4	Decimal format with 11 digits, a period and 4 decimal digits. Grouping by comma is by default. Example: 12,345,678,901.1234
Integer	@n-14	Signed Integer (31bit) format with 14 digits. Grouping by comma is by default. Example: 2,147,483,647
Date	@d10-	Date formatted as YYYY-MM-DD. This format is used as default as it allows dates to be used in filenames or pathnames.
Time	@t4	Time formatted as HH:MM:SS. This format is **not** suitable for use in filenames and pathnames.

2.10.3 Project Items List — Project Window

The Project Items list is on the far left of the Project Window 30. The Project Items can be used to break up your project into easily manageable pieces or into logical construction where each Project Item performs well defined part of the whole project.



The Project Items list is constructed of 3 components:

- Line number
- Checkbox
- Name

The line number serves as a placeholder so it is easy to identify each line.

The checkbox in the Project Items list has two possible states, checked or unchecked. unchecked, the Project Item will not be included in the execution of the project. It will b Action Items in it will be skipped. To change the checkbox double click on it.

The Name can be maximum 30 characters long.

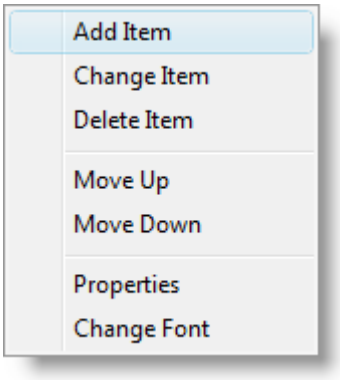
To create a new Project Item, press the Insert key on the keyboard or right click with th popup menu and select "Add Item" from the menu. This will insert the new item below item.

To change the name, select it and press the Enter key on the keyboard or press the F2 You can also double click the name with the mouse to change it. Right click and select popup menu is yet another way to edit the Project Item.

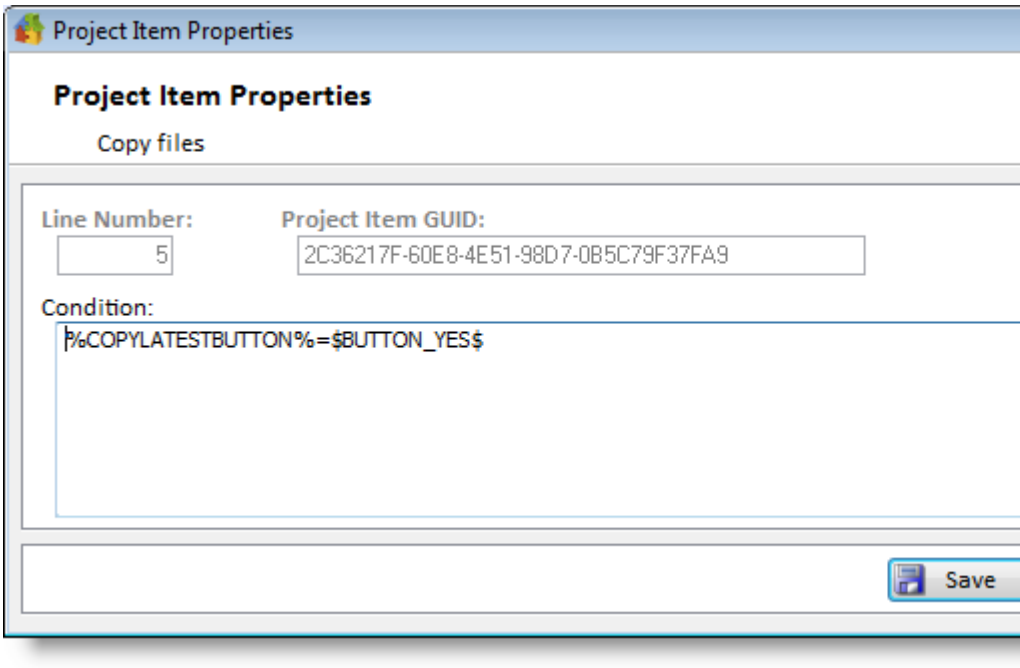
Deleting a Project Item will also delete ALL of it's Action Items. To limit the risk of accid Project Item, the Build Automator prompts you for confirmation before deleting. To dele can use the Delete key on the keyboard or right click and select "Delete Item" from the

There are no limits to how many Project Items can be in a single project.

Items can be moved up and down in the list by using Ctrl-Up arrow and Ctrl-Down arrow right clicking on the items list and using "Move Up" or "Move Down"



Properties for the Project Item can be access by right clicking and selecting "Properties



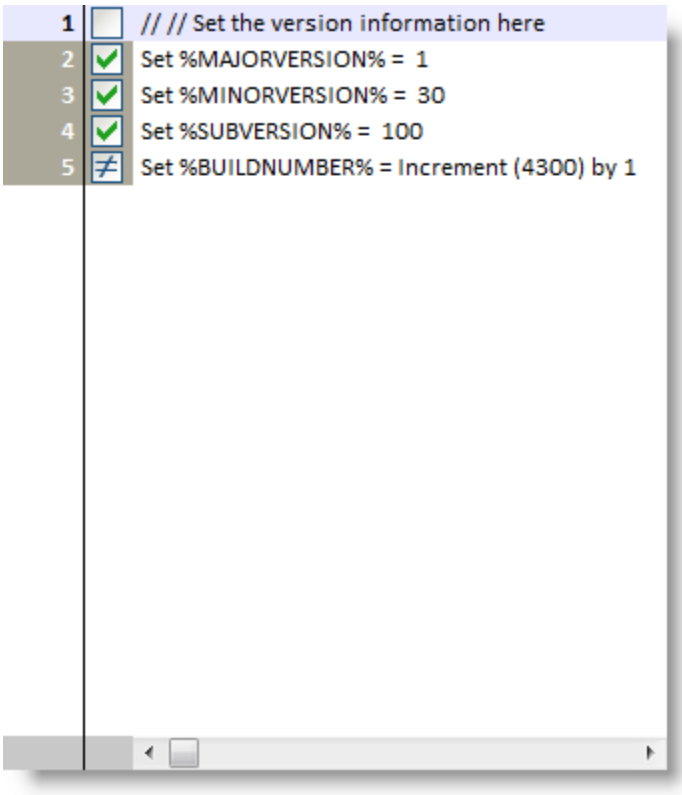
A condition can be enforced on the Project Item and if the condition is not met, the Proj
Action items will be skipped during execution. Example of a condition could be:

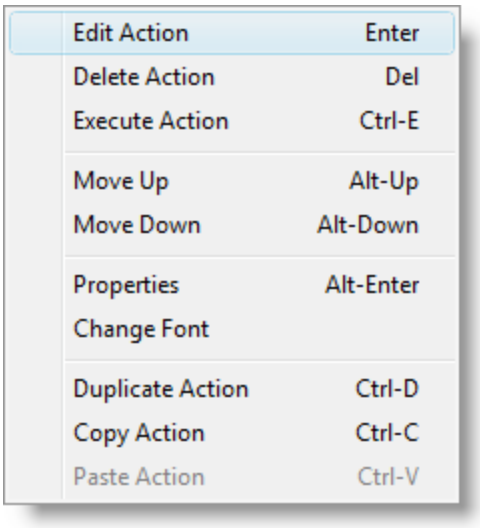
```
%COPYLATESTBUTTON%=$BUTTON_YES$
```

When a condition is applied, the checkbox in front of the Project Item changes, see line
the list above.

2.10.4 Action Items List — Project Window

The Actions Items list is in the middle of the Project Window 30. The Action Items list shows information about each action that has been added to the project. How the information is displayed is determined by how the Action was set up in the *Action Editor*.





The Action Items list is constructed of 3 components:

- Line number
- Checkbox
- Description

The line number serves as a placeholder so it is easy to i

The checkbox in the Action Items list has three possible s


Unchecked


Checked


Condition

If the checkbox is unchecked, the Action Item will not be i execution of the project - it will be skipped. To toggle the checked and unchecked double click on it. In order to ch right click on the Action Item - or use the AppsKey on the select "Properties" This opens the Action Items Propertie will show you information about the Action Item as well as a condition.

The Description is constructed by the software and canno Action Items list.

To create a new Action Item, you need to locate the Actio use and double click on the Action Tree to insert it into th You can also use the keyboard to select the Action and th or Insert key on the keyboard.

To edit the Action Item, select it and press the Enter key double click on it with the mouse. You can also right click and select "Change Item" from the popup menu.

To limit the risk of accidentally deleting an Action Item, th prompts you for confirmation before deleting. To delete a can use the Delete key on the keyboard or right click and Action" from the popup menu.

You can use the popup menu to Execute the selected Ac clicking on the item - or use the Apps Key on the keyboar "Execute this Action" from the popup menu.

There are no limits to how many Actions Items can be in Item or in a single Project for that matter.

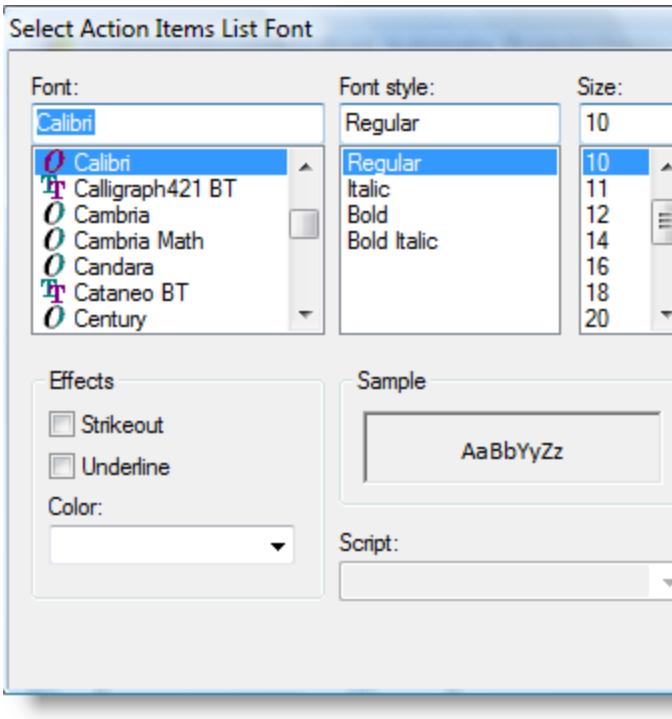
If multiple items have been selected, the popup menu ind showing "Delete Actions" and "Execute Actions".

Items can be moved up and down in the list by using Ctrl- Down arrow on the keyboard or by right clicking on the ite "Move Up" or "Move Down"

To view the Action Item Properties 43, select "Properties"

menu.

The font of the action item list can be changed by using th
option.



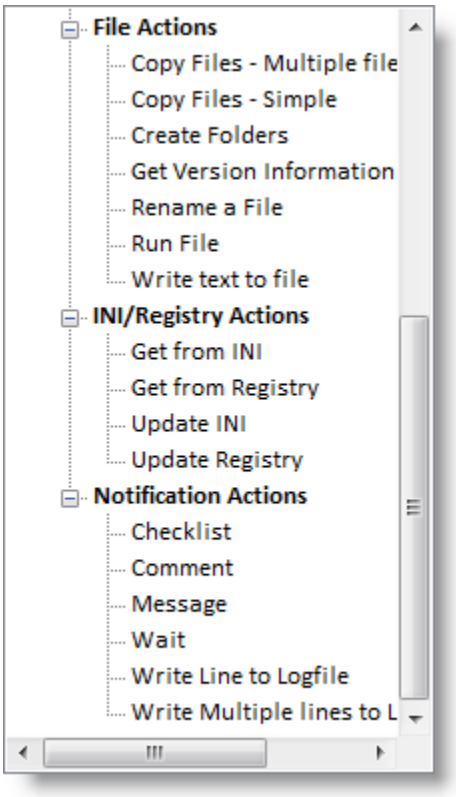
It brings up a standard Fonts dialog.

Clipboard actions are supported for single or multiple item
and paste actions within the same Project Item, into a ne
into a different project. You can also duplicate the selecte
the popup menu or by using Ctrl-D on the keyboard.

Standard clipboard keyboard shortcuts are available such
and Ctrl-V to paste

2.10.5 Actions Tree — Project Window

The Actions tree is on the far right of the Project Window 30. The Actions tree contains all the actions that are available for you to use.

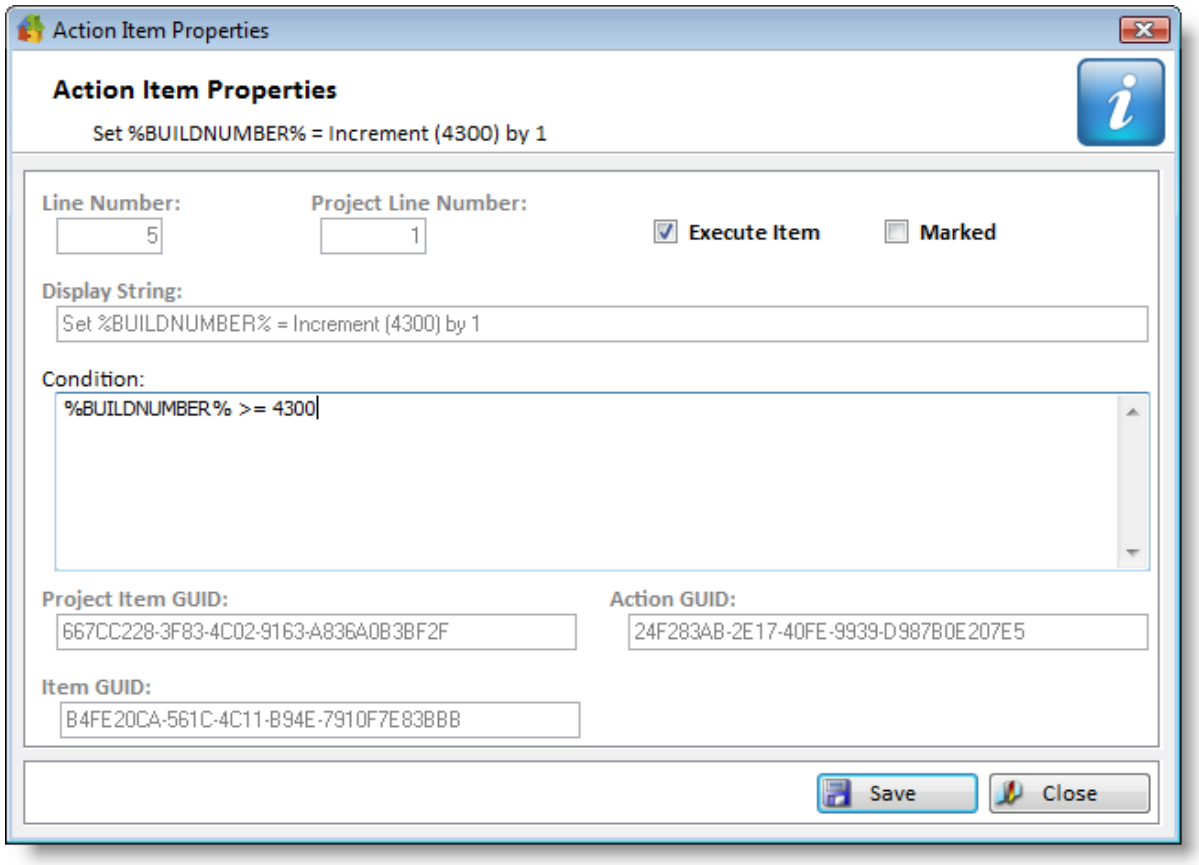


The Actions are grouped together in groups of similar functionality. You cannot add or remove Actions from the tree and the only thing that is allowed is expanding and collapsing the nodes of the tree as well as using the Enter or Insert keys on the keyboard to create a new Action Item. Double clicking with the mouse will also create a new Action Item. The new Action Item will be created above the currently selected item, i. e. it will push the selected item down and be created on the currently selected line.

You can change the font setting of the Action Tree from the Build Automator Options window. There is currently no popup menu for the Action tree.

2.10.6 Action Items Properties — Project Window

The Action Items Properties window can be accessed from the Project Window 30 by right clicking with the mouse and select "Properties" For the most part the information there is fixed and cannot be changed, with the exception of the Condition. The Condition gives you the power to set a condition for the execution of this Action Item.

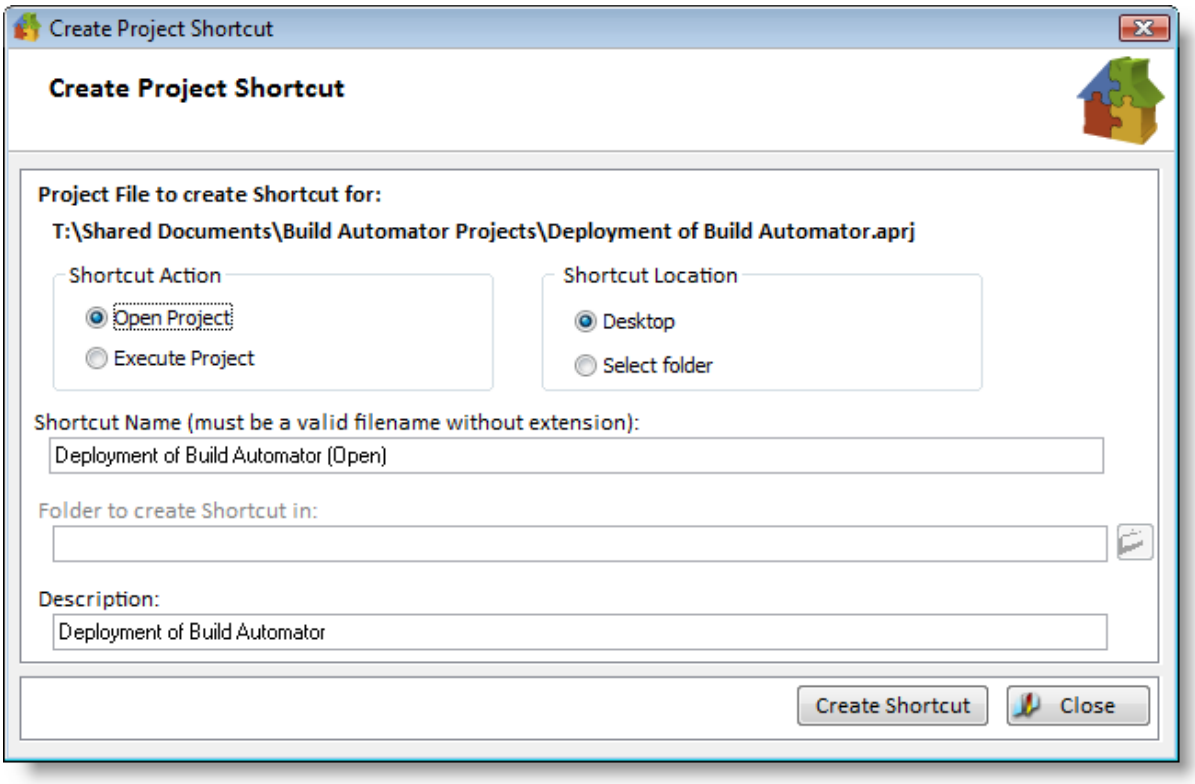


This window shows information about the Action Item including all the GUIDs that are associated with the Action Item. It also shows the Line number, Project Line Number, Execute Item and if the item is Marked as part of multiple selection. All these entries and check boxes are read-only and only for information purposes.

Only the Condition entry is not read-only and you can construct whatever condition you need for this Action Item to execute.

2.10.7 Create Project Shortcut — Project Window

You can create shortcuts to your projects directly from the Build Automator.



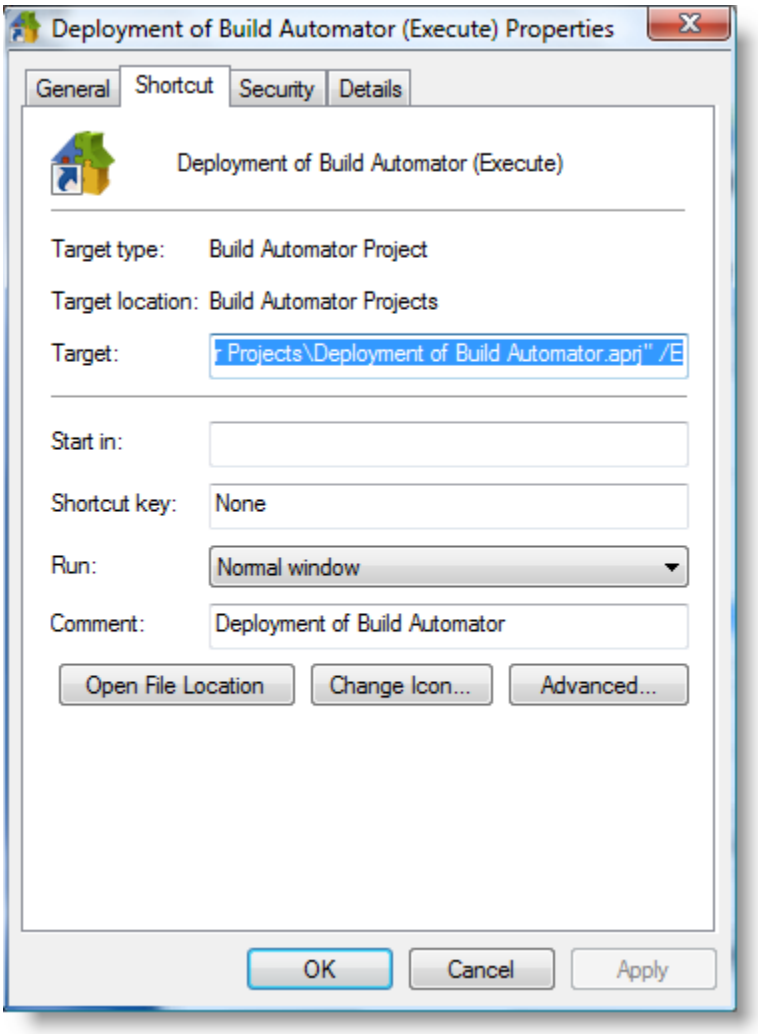
A shortcut can be created for either the Opening of a project or Execution of a project. The only difference is that the command line 22 for the Execute will have a trailing /E after the name of the project to indication execution. The default option is a shortcut to Open a project.

You can either create the shortcut on the Desktop or in a selected folder. If you use the "Select folder" option, the "Folder to create Shortcut in" and the selection button on the right will be enabled and you can select a folder.

The Shortcut Name is defaults to the name of the project file or the Project Name that you can specify in the Project properties 33 window. You can change the name of the shortcut filename, but you must provide a valid file name, without extension.

The "Folder to create Shortcut in" is only used if you use the "Select folder" option for the Shortcut location. Click on the button on the right to select a folder to create the shortcut in.

You can enter description of the shortcut which shows up in the "Comment" field on the shortcut properties - see screenshot below.



2.10.8 Print Checklist — Project Window

This option is not yet implemented.

2.10.9 Print Project — Project Window

This report prints the project in a similar format as it shows in the Project Window 30. Conditional items are shown in IF/END structures in bold, blue color. Comments are shown in green color. Currently the report only prints as a 8.5x11" US Letter size report in portrait mode.

Below is an example page from the Print Project option.

Project: Deploy Icetips Utilities

Project File: T:\Shared Documents\Build Automator Projects\Deploy Icetips Utilities.aprj
Created: 8/25/2008 at 14:54:26
Last modified: 11/04/2011 at 16:20:19
Printed: 11/04/2011 at 17:32:08

1. Clean Latest

```
1. // This part cleans out the Latest folder and starts over
2. Message ("Copy Existing", "Do you want to copy the existing Latest Folder?", ICON:Question)
3. Message ("Compile Help files?", "Do you want to compile the H&M help files?", ICON:Question)
4. Message ("Upload Files?", "Upload the install?", ICON:Question)
5. //
   IF %COPYLATESTBUTTON%=$BUTTON_YES$
6.   Set %NEWVERSIONFOLDER% =  Version %MAJORVERSION%_%MINORVERSION%_%SUBVERSION% - $ToDay$
7.   Create multiple subfolders for C:\Products\ITUtilities
8.   Copy (simple) "C:\Products\ITUtilities\Latest\*.*" to "C:\Products\ITUtilities\%NEWVERSIONFOLDER%\",
9.   Delete C:\Products\ITUtilities\Latest\3rdParty\*.*
10.  Delete C:\Products\ITUtilities\Latest\Data\*.*
11.  Delete C:\Products\ITUtilities\Latest\ITUtilities\*.*
12.  Delete C:\Products\ITUtilities\Latest\LibSrc\*.*
   END
```

2. Prepare for copy

```
1. Set %MAJORVERSION% =  1
2. Set %MINORVERSION% =  1
3. Set %SUBVERSION% = Increment (2306) by 1
4. Set %BUILDNUMBER% =  0
5. // First compile the Help and Manual Projects
6. Message ("Compile Help files?", "Do you want to compile the H&M help files?", ICON:Question)
7. Set %LONGDATE% = Expression ( Format(Today(),@d18) )
8. Set %YEAR% = Expression ( Year(Today()) )
   IF %HELPANDMANUALCOMPILE%=$BUTTON_YES$
9.   Write text to T:\HelpFiles\UtilityClasses\Variables.txt
10.  Run and Wait for C:\Program Files\HelpandManual4\HelpMan.exe T:\HelpFiles\UtilityClasses\ITUtility.hmx /PDF /
   END
11. // Set Variables
12. Copy (simple) "C:\Products\ITUtilities\Welcome.rtf" to "C:\Products\ITUtilities\Latest",
13. Search for "##VERSION##" and replace with "%MAJORVERSION%.%MINORVERSION%.%SUBVERSION%"
14. Search for "##RELEASEDATE##" and replace with "%LONGDATE%"
```

3. Compile demos

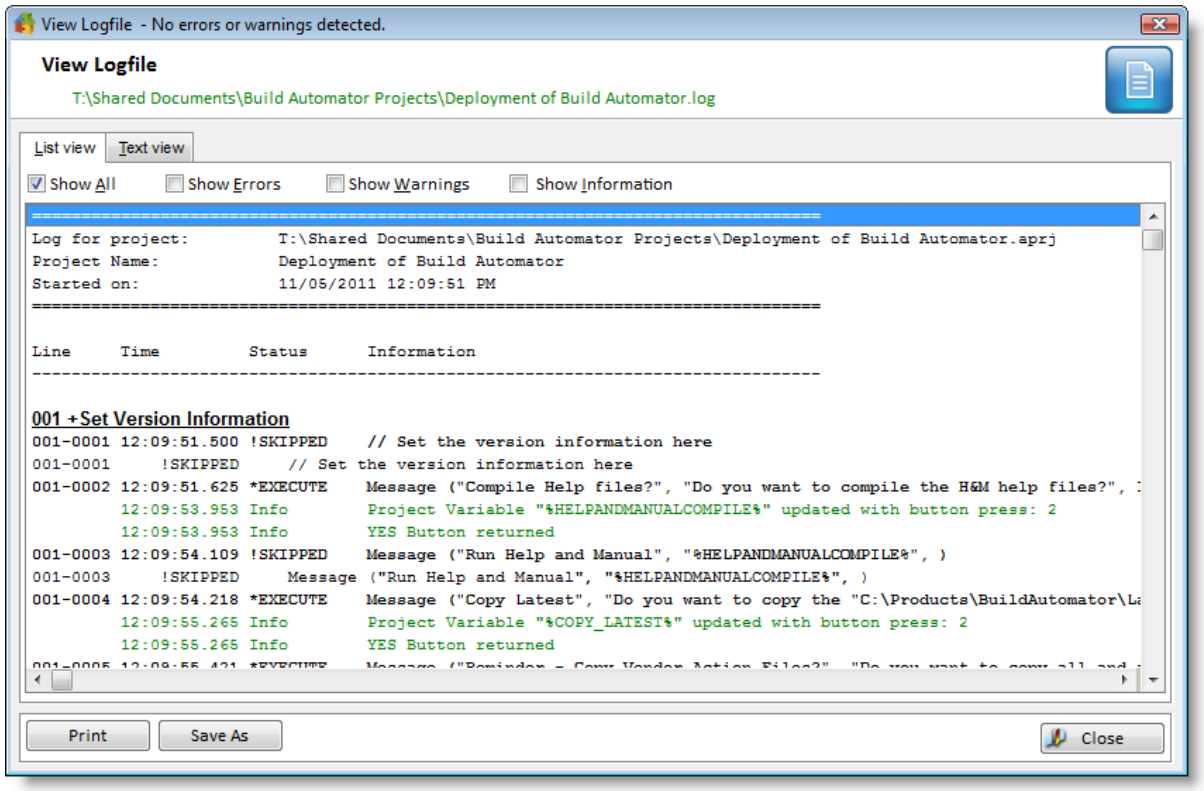
```
1. // Set version resources and compile Clarion demo apps
2. Write text to C:\Dev\Prod\Utilities\Demos\CoreClassDemo.Version
3. Write text to C:\Dev\Prod\Utilities\Demos\UtilDemo.Version
4. Write text to C:\Dev\Prod\Utilities\Demos\WindowsClassDemo.Version
5. Compile CoreClassDemo.app, WindowsClassDemo.app, UtilDemo.app apps
6. Compile in Clarion 6.0: C:\Dev\Prod\Utilities\Demos\WindowsClassDemo.app -> C:\Dev\Prod\Utilities\Demos\Windows
```

4. Compile C++ Code

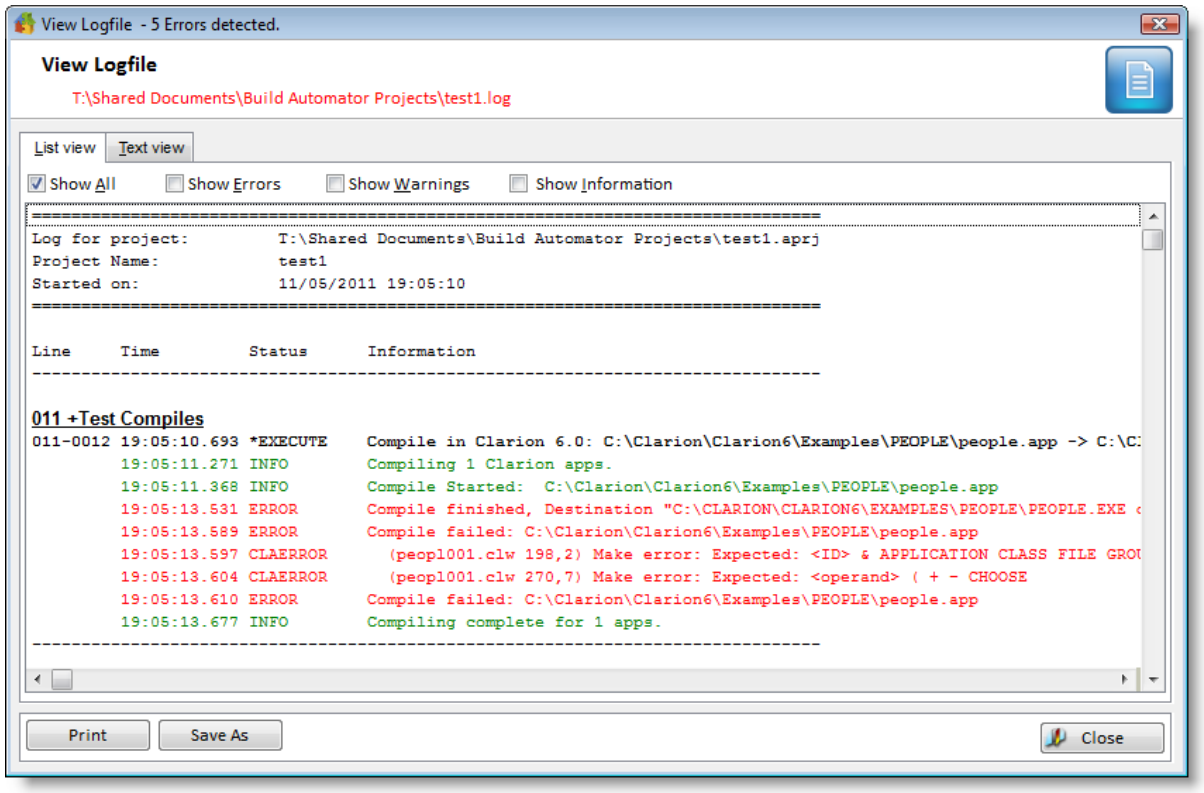
```
1. // Compile the C6 version first, make sure that we have #define __RUNDLL__ 1 in the .cpp file.
2. Search for "#define __RUNDLL__ 0" and replace with "#define __RUNDLL__ 1"
3. Run and Wait for C:\Clarion\CBuilderX\Apps\ITRun32\compileITRun32.bat
4. Copy (simple) "C:\Clarion\CBuilderX\Apps\ITRun32\windows\Release_Build\ITRun32.dll" to "C:\Products\_Binaries",
5. // Compile the C7 version.
6. Search for "#define __RUNDLL__ 1" and replace with "#define __RUNDLL__ 0"
7. Run and Wait for C:\Clarion\CBuilderX\Apps\ITRun32\compileITRun32.bat
```

2.10.10 Log File Viewer — Project Window

When a project is executed and the execution completes the Log File Viewer comes up and shows a log of the process.

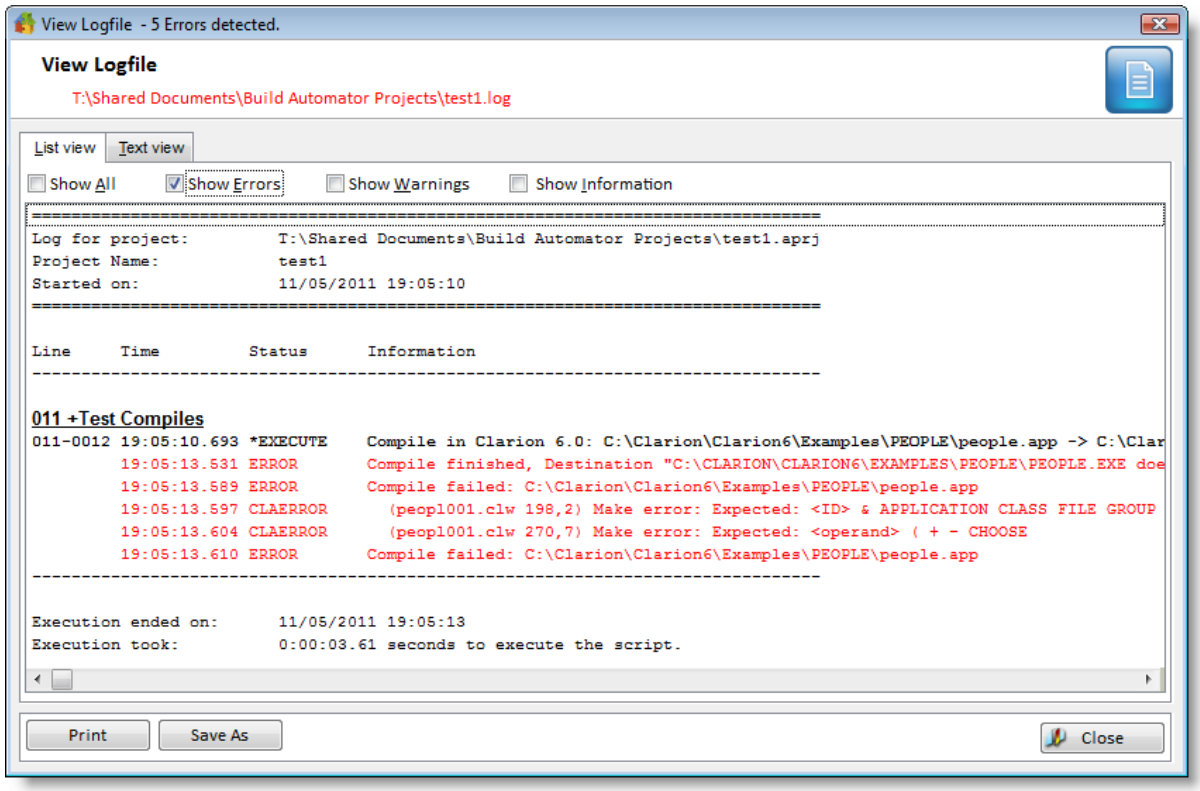


The list can be filtered to show everything, errors, warnings or information. Errors, Warnings and Information can all be checked at the same time.



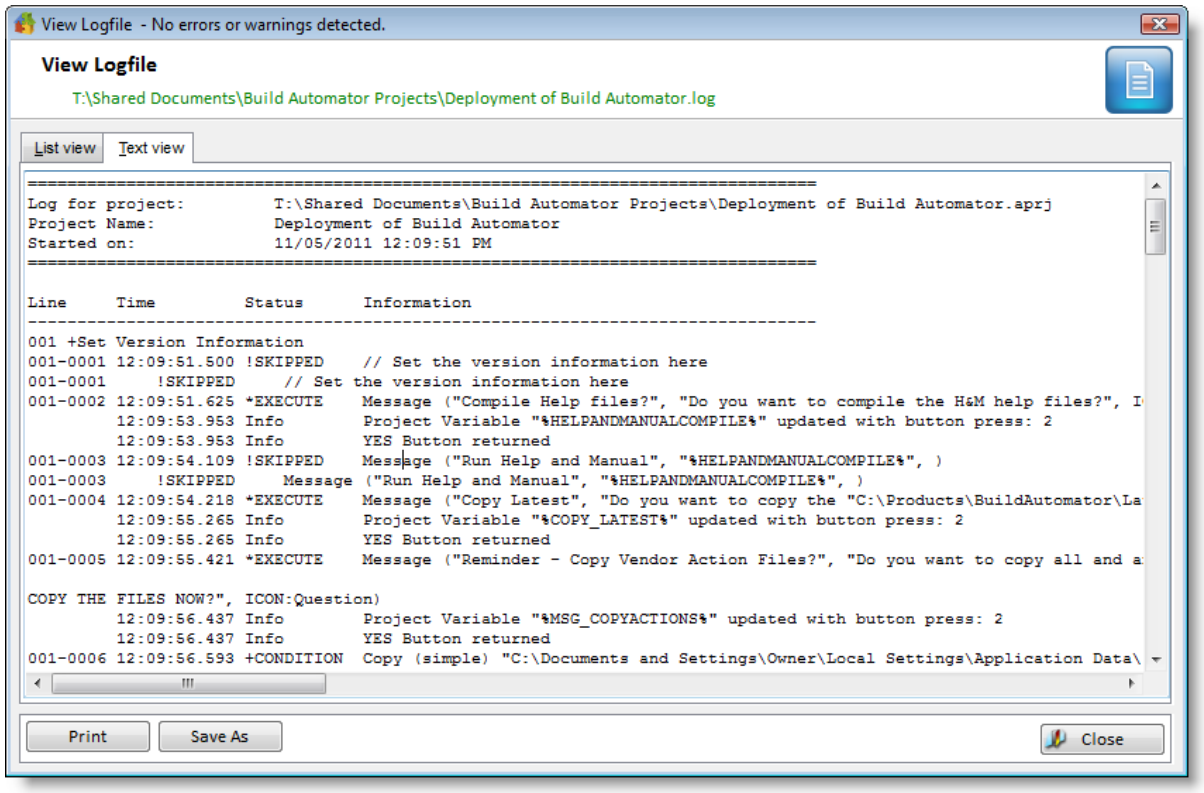
Errors show up in red color, warnings in Blue, information in Green and everything else in black. The log file points to the line numbers for the Project item and the Action item and indicates the time when the action completed. It also indicates what operation happened, execute, skipped i.e. commented out, condition passed or failed.

Below is a screenshot of the same screen as above but with only Errors checked.



As you can see the "INFO" lines are not shows, but the log still shows you the relevant Project Item and Action Item information.

The Text tab shows the same data but without any of the color and font formatting, i.e. it shows the raw log file.



The log file can be printed or saved to a new filename. The printed output is the same as the raw output without formatting.

The log file is overwritten each time you execute the project.

2.11 Keyboard shortcuts

We use keyboard shortcuts in various places in the program to make it easy to navigate and do certain things. This topic is still under construction as we go through the beta testing and get a feel for what is needed and what keyboard shortcuts are best.

Main menu 52

Project Window 52

Actions Windows 52

General

Ctrl-Alt-R	On action update windows with lists, this hotkey resets the list format to original.

2.11.1 Actions Windows — Keyboard shortcuts

The various Actions Windows use some standard keyboard shortcutrs.

Ctrl+Down-Arrow	On entry fields that support variables, this pops up the Select Variable 135 window so you can select a variable.
Ctrl+Right-Click	On entry fields that support variables, this pops up the Select Variable 135 window so you can select a variable.
Ctrl+Shift +F12	Prompts you to copy the GUID of the project action to the clipboard.

2.11.2 Main Menu — Keyboard shortcuts

The main Build Automator™ menu uses several keyboard shortcuts.

Ctrl+N	Create a new project.
Ctrl+O	Open an existing project.
Ctrl+L	Open the Pick list 11
Ctrl+S	Saves the current project.
Ctrl-P	Prints the current project (not implemented yet)

2.11.3 Project Window — Keyboard shortcuts

Project Item list:

Enter	Change the name of the selected Project Item

F2	Change the name of the selected Project Item
Double click	Change the name of the selected Project Item
Right Mouse Button	Pops up a menu to insert, change or delete an item
Ctrl-I	Toggles the checkbox which controls if the item is executed or not
Insert	Inserts a new item in the list
Delete	Deletes the currently selected item
Alt-Up arrow	Moves the selected item up
Alt-Down arrow	Moves the selected item down
Apps-Key	Pops up a menu to insert, change or delete an item, same as the Right Mouse Button.

Action Item list:

Enter	Open the update form for the selected Action Item
Alt-Enter	Opens the property window for the Action Item
F2	Change the name of the selected **Project Item! Note:** this key works on the Project Items list, **not** the Action Items.
Double click	Open the update form for the selected Action Item
Right Mouse Button	Pops up a menu to edit, delete or execute the currently selected item.
Ctrl-I	Toggles the checkbox which controls if the item is executed or not
Ctrl-E	Executes the selected Action Item(s)
Ctrl-R	Executes the selected Action Item(s) without opening the Log File Viewer 48 window after it's finished.
Ctrl-T	Copies selected Action Item(s) as text.
Ctrl-A	Selects all the Action Items for the selected Project Item.
Ctrl-C	Copies the selected item(s) to the clipboard
Ctrl-V	Pastes items from the clipboard to the Action item list, above the currently selected item.
~~Ctrl-X~~	~~Cuts the selected item(s) to the clipboard~~
Ctrl-D	Duplicates the selected Action Item(s) ***Note:*** *This also places the item(s) on the clipboard.*
Delete	Deletes the currently selected item
Alt-Up arrow	Moves the selected item up
Alt-Down arrow	Moves the selected item down
Apps-Key	Pops up a menu to edit, delete or execute the currently selected item, same as the Right Mouse Button.
C	Comment selected Action Item(s)
U	Uncomment selected Action Item(s)

Actions Tree:

Enter	Add the currently selected Action to the Action Items
Insert	Add the currently selected Action to the Action Items
Double click	Add the Action being double clicked to the Action Items

Build Automator

Part III

Chapter 3 - Function Reference

3 Function Reference

This is a reference guide to the functions that are available to use in the Evaluate Expression operation in the Set Variable 94 action.

This includes most of the standard runtime library functions that are not related to files and data. Below is a list of functions that you can use in the expression in the Set Variable action. Below is simple documentation for the available functions. We may also add some functions of our own that are not part of the Clarion Runtime Library but could be useful in the Build Automator. Parameters in square brackets (i.e. inside []) indicate optional parameters. We may add more detailed help for functions that need it.

Note that you MUST enclose strings being passed to those functions with double quotes (")

Functions marked with ** are functions that are only available in Build Automator and are not Clarion RTL functions.

Function	Description
Abs (*expression*)	Returns the absolute value of expression.
Acos (*expression*)	Returns the arccosine value of the expression.
Age (*birthday [, basedate]*)	Returns a string containing age calculation for the *birthday*. If the *basedate* is omitted, the system date is used. The returned string is in the following format: 1-60 days: "nn DAYS" 61 days - 24 months: "nn MOS" 2 years - 999 years: "nnn YRS" See also: Date, Day, Month, Year function.
All (*string [,length]*)	Returns a string that is *length* characters long, repeatedly filled with *string*. **Example:** `All("-",80)` would fill the variable with 80 dashes (-)
And	Logical AND
Asin (*expression*)	Returns the Arcsine of the value of the expression.
Atan (*expression*)	Returns the Arctangent of the value of the expression.
Band (*value, mask*)	Returns the results of bitwise AND of each bit.
Bor (*value, mask*)	Returns the results of bitwise OR of each bit.
Bshift (*value, count*)	Returns the value of *value* after shifting the bits *count* left or right. Positive *count* shifts the bits to the left, negative *count* shifts the bits to the right.
Bxor (*value, mask*)	Returns the results of bitwise exclusive OR operation on value.

Center (*string ,length*)	Returns string that is centered based on the length parameter. For example: Center("123",7) would return " 123 " - i.e. 123 with two spaces on each side.
Choose (*expression, value, value[,value...]*)	Returns a value based on the expression. **Example:** `Choose(%X%,2,4,6)` If %X% is 1, then it would return 2 If %X% is 2, then it would return 4 If %X% is 3, then it would return 6
Choose (condition[, *value, value]*)	Returns true or false (1 or 0) depending on the condition. **Example:** `Choose(%X% = 1,'True','False')` This would return 'True' if the value of %X% was 1, otherwise it would return 'False'
Chr (*charactercode*)	Returns the character for the ASCII *charactercode*. **Example:** Chr(32) would return a space Chr(65) would return 'A' (uppercase a)
Clip (*string*)	Returns the *string* with no trailing spaces.
Clock ()	Returns the system clock value. Note that this is in centiseconds, 1/100 of a second, not milliseconds. To get ms, use: Clock() * 10
****CompareVersionStrings (*ver1, ver2, level*)**	Compares version strings in *ver1* and *ver2*. The level specifies the *level* to compare, 1 compares only the first number, 2 compares first and second number, etc. The level must range from 1 to 4. This function can only compare strings that are formatted as aaa.bbb.ccc.ddd. Each level is individually cast to an integer and compared, for example 1.20 is greater than 1.2. The function returns one of 3 possible values: 0 = ver1 and ver2 are equal to the level specified 1 = ver1 is greater than ver2 2 = ver2 is greater than ver1 **Example:** `CompareVersionStrings("%VERSION1%","%VERSION2%",3)` This will compare the first 3 levels of the %VERSION1% and %VERSION2% string variables. For example if %VERSION1% is "1.2.3.4" and %VERSION2% is "1.2.3.5" the result would be 0 as the first 3 levels ("1.2.3") are equal. If the level was set to 4, the results would be 2 because %VERSION2% is greater than %VERSION1%

	Demo: **Compare Version Strings.aprj** demo project.
Cos (*radians*)	Returns the cosine of the *radians* value.
Date (*month, day, year*)	Return a standard Clarion date based on the value for *month*, *day* and *year*. A Clarion date is set to 0 on December 29, 1800. **Example:** `Date(Month()+1,1,Year())` This would return the first day of next month. Note that you can use out of bounds values, such as 13 for months, or 40 for days to advance the date calculation by specific period.
Day (*date*)	Returns the day of month for the *date* value. **Example:** `Day(Today())` would return 15 on June 15, 2014.
Deformat (*string, format*)	Returns the *string* with formatting removed based on the *format* parameter. The format can be any of the format tokens used in standard Clarion data formatting 62.
****ExpandEnv(*string*)**	Expands the environment variable passed in *string* and returns the expanded variable contents. **Example:** `ExpandEnv("PATH")` will return the contents of the %PATH% environment variable. Note that starting in version 2.0 you have access to all available environment variables directly in your actions.
****FileExists(string)**	Returns 1 (true) or 0 (false) depending on if the file or folder specified in *string* exists. **Example:** `FileExists("%MYFOLDER%")`
Format (*string, format*)	Returns the *string* formatted based on the *format* parameter. The format can be any of the format tokens used in standard Clarion data formatting 62.
****GetFileDate(filename)**	Returns the Clarion Date value of the file. Filename must be enclosed in double quotes. Use Format() to format the date value. See Date formats 62.
****GetFileSize(filename)**	Returns the size of the file as an unformatted 32bit signed integer. Filename must be enclosed in double quotes. Use format to format the value. See Number formats 63.
****GetFileTime(filename)**	Returns the Clarion Time value of the file. Filename must be enclosed in double quotes. Use Format() to format the time value. See Time formats 65.

Getini (*section, entry [,default] [,file]*)	Retrieves a value from an INI file.
	Example:
	`GetIni("Source","Root","C:\","C:\test.ini")`
	
	If the test.ini file contained:
	`[Source]`
	`Root=F:\files`
	
	Then the GetIni would return "F:\files"
	If the *file* is not specified it defaults to win.ini. If path is not specified in the *file* parameter, it defaults to the windows folder. Note that GetReg is not available, but you can always use the Get From INI 117 and Get From Registry 117 actions.

Inlist (*searchstring, liststring, liststring[,liststring...]*)	Inlist compares the value of *searchstring* to the *liststring* values and returns the index number of the first *liststring* that matches the *searchstring*. If no match is found it returns zero.
Inrange (*expression, low, high*)	Compares a numeric *expression* to the range of *low* and *high* value and returns True (1) if the *expression* is within the *low* to *high* range. If the expression is lower than the *low* parameter or higher than the *high* parameter, the function returns False (0)
Instring (*searchstring, string, step,start*)	The Instring function searches for the *searchstring* in the *string* starting at the *start* position. The *step* is used to specify the step length of the search. Normally the *step* is set to 1. *Step* can be negative and then it searches backward, but then *searchstring* is limited to a single character. The function returns the position of the first occurance of the *searchstring* in the *string* or zero (0) if the *searchstring* was not found. **Example:** `Instring("X","THIS IS X",1,1)` would return 9.

Int (*expression*)	Returns the integer portion of expression. No rounding is performed.
	Example:
	`Int(12.44)`
	returns 12
	`Int(-6.3)`
	returns -6

Left (*string [,length]*)	Left justifies the *string*. All leading spaces are removed. The length is used to determine the length of the resulting string: **Example:** `Left("  ABC")` returns "ABC "
Len (*string*)	Returns the length of the *string*.
Log10 (*expression*)	Returns base 10 logarithm for the *expression*. If the expression is 0 or less, the returned value is 0.

Loge (*expression*)	Returns the natural logarithm for the *expression*. The value of ***e*** is determined to be 2.71828182846
Longpath (*shortfilename*)	Returns the long path or filename for the *shortfilename*.
Lower (*string*)	Returns the *string* converted to lower case.
Match (*first, second, mode*)	Returns true or false (1 or 0) based on comparison between the *first* and the *second* parameters.
Month (*date*)	Returns the month number (1..12) for the date. The date must be a standard Clarion date with base date on December 29, 1800. See Date.
Not	Logical NOT
Numeric (*string*)	Returns true (1) if the *string* only contains a valid numeric value. Valid numeric characters are 0123456789-.
Or	Logical OR
Path ()	Returns the current path.
Random (*low, high*)	Returns a random integer between *low* and *high*, both included.
Right (*string [,length]*)	Returns a right justified string. The length is used to determine the length of the resulting string. **Example:** `Right("ABC  ")` returns " ABC"

Round (*expression, order*)	Rounds the value of *expression* rounded to the power of 10.
	Example:
	`Round(1234,100)`
	returns 1200
	`Round(675.6,1)`
	returns 676
	`Round(25.12595,0.01)`
	returns 25.13

****SearchReplace(*search, replace, text*)**	Searches for "search" in "text" and replaces it with "replace" and returns the results. **Example:** `SearchReplace(".","_","This.is.a.string.with.periods.in.it")` returns: "This_is_a_string_with_periods_in_it"
Shortpath (*longfilename*)	Returns the shortened path for the *longfilename*.
Sin (*radians*)	Returns the sine of an angle measured in *radians*.
Sqrt (*expression*)	Returns the square root of *expression*.
Strpos (*first, second[, mode]*)	Returns the starting position where the *first* and *second* strings match.

Sub (*string, position, length*)	Returns a *length* long substring from *string* starting at *position*.
Tan (radians)	Returns the tangent of an angle measured in *radians*.
Today ()	Returns the standard Clarion date value for today.
Upper (*string*)	Converts the *string* to upper case.
Val (*character*)	Returns the ASCII value of the *character*.
Xor	Boolean exclusive OR
Year (date)	Returns the year of a standard Clarion *date*.

3.1 Format tokens

The Build Automator uses format tokens to format numbers, dates, times etc. These are standard Clarion format tokens so if you are familiar with Clarion, you can use most of the format tokens. The @K token is not supported and doesn't really make sense in the context of the Build Automator since it is used for data entry.

3.1.1 Date formats — Format tokens

Dates
Syntax: @Dn [s] [B]

All date formats start with @D or @d
The **n** determines the date format and ranges from 1 to 18.
The **s** is an optional separation character between year, month and days. It is forward slash (/) by default. Valid characters for **s** are:
. (period) produces periods.
` (grave accent - ASCII 96) produces commas.
- (hyphen) produces hyphens.
_ (underscore) produces spaces.
The trailing **B** will produce a **B**lank string if the date is empty instead of something like " / / " for @d1 for example.

Token	Format	Results
@D1	mm/dd/yy	10/31/59
@D01	mm/dd/yy	01/01/59
@D2	mm/dd/yyyy	10/31/1959
@D3	mmm dd,yyyy	OCT 31,1959
@D4	mmmmmmmmm dd, yyyy	October 31, 1959
@D5	dd/mm/yy	31/10/59
@D6	dd/mm/yyyy	31/10/1959
@D7	dd mmm yy	31-Oct-59
@D8	dd mmm yyyy	31 OCT 1959
@D9	yy/mm/dd	59/10/31
@D10	yyyy/mm/dd	1959/10/31
@D11	yymmdd	591031
@D12	yyyymmdd	19591031

@D13	mm/yy	10/59
@D14	mm/yyyy	10/1959
@D15	yy/mm	59/10
@D16	yyyy/mm	1959/10
@D17		Standard Short Date
@D18		Standard Long Date
	Alternate separators	
@D1.	mm.dd.yy	Period separator
@D2-	mm-dd-yyyy	Dash separator
@D5_	dd mm yy	Underscore produces space separator
@D6`	dd,mm,yyyy	Grave accent produces comma separator

3.1.2 Number formats

Numbers
Syntax: @N [currency] [sign] [fill] size [grouping] [places] [sign] [currency] [B]
All numeric and currency formats start with @N or @n
The **currency** can be a dollar sign ($) or any character(s) enclosed in tildes (~) It can be either at the beginning or end of the format.
The **sign** can be a dash (-) or parenthesis (()) If dash it can be either before the fill or after the places. If parentheses they must be in both places.
The **fill** can be a zero (0), underscore (_) or asterisk (*) and specifies leading fill character.
The **size** is required to specify the total number of characters that the number will display including any grouping and formatting characters.
The **grouping** is used to separate the numbers in groups of 3. This should be used along with the hyphen to indicate sign. The grouping can be one of 3 characters: Period (.), hyphen (-) or underscore (_) which produces spaces.
The **places** is used to indicate decimal separator and decimal places. Decimal separator can be either period (.) or grave accent (`) which produces a comma (,) as decimal separator.
The trailing **B** will produce a **B**lank string if the variable is empty or zero.

Numeric	Result	Comment
@N9	4,550,000	Nine digits, group with commas (default)
@N_9B	4550000	Nine digits, no grouping, leading blanks if zero
@N09	004550000	Nine digits, leading zero
@N*9	***45,000	Nine digits, asterisk fill, group with commas
@N9_	4 550 000	Nine digits, group with spaces
@N9.	4.550.000	Nine digits, group with periods

Decimal	Result	Comment
@N9.2	4,550.75	Two decimal places, period decimal separator
@N_9.2B	4550.75	Two decimal places, period decimal separator, no grouping, blank if zero
@N_9`2	4550,75	Two decimal places, comma decimal separator
@N9.`2	4.550,75	Comma decimal separator, group with periods
@N9_`2	4 550,75	Comma decimal separator, group with spaces

Signed	Result	Comment
@N-9.2B	-2,347.25	Leading minus sign, blank if zero
@N9.2-	2,347.25-	Trailing minus sign
@N(10.2)	(2,347.25)	Enclosed in parenthesis when negative

Dollar Currentcy	Result	Comment
@N$9.2B	$2,347.25	Leading dollar sign, blank if zero
@N$10.2-	$2,347.25-	Leading dollar sign, trailing minus when negative
@N $(11.2)	$(2,347.25)	Leading dollar sign, in parenthesis when negative

Other Currentcy	Result	Comment
@N12_`2~ F~	1 5430,50 F	France
@N~L.~12`	L. 1.430.050	Italy
@N~£~12.2	£1,240.50	Great Britain
@N~kr~12`2	kr1.430,50	Norway
@N~DM~12`2	DM1.430,50	Germany
@N12_`2~ mk~	1 430,50 mk	Finland
@N12`2~kr~	1.430,50 kr	Sweden

3.1.3 Time formats Format tokens

Time
Syntax: @Tn[s][B]
All time formats start with @T or @t
The **n** determines the time format and ranges from 1 to 8.
The **s** is an optional separation character between hour, minutes and seconds. It is colon (:) by default. Valid characters for **s** are:
. (period) produces periods.
` (grave accent - ASCII 96) produces commas.
- (hyphen) produces hyphens.
_ (underscore) produces spaces.
The trailing **B** will produce a **B**lank string if the time is empty instead of something like " : : " for @t4 for example.

Token	Format	Results
@T1	hh:mm	17:30
@T2	hhmm	1730
@T3	hh:mmXM	5:30PM
@T03	hh:mmXM	05:30PM
@T4	hh:mm:ss	17:30:00
@T5	hhmmss	173000
@T6	hh:mm:ssXM	5:30:00PM
@T7		Standard Short Time
@T8		Standard Long Time
	Alternate separators	
@T1.	hh.mm	Period separator
@T1-	hh-mm	Dash separator
@T3_	hh mmXM	Underscore produces space separator
@T4`	hh,mm,ss	Grave accent produces comma separator

Build Automator

Part IV

Chapter 4 - Build Automator Actions

4 Build Automator Actions

Build Automator™ comes with a growing library of actions that are ready for you to use. For more information about what is planned, see our Future plans 151 chapter. If you have suggestions for new actions or options, please do not hesitate to let us know by filling out the Feature Request form on our website.

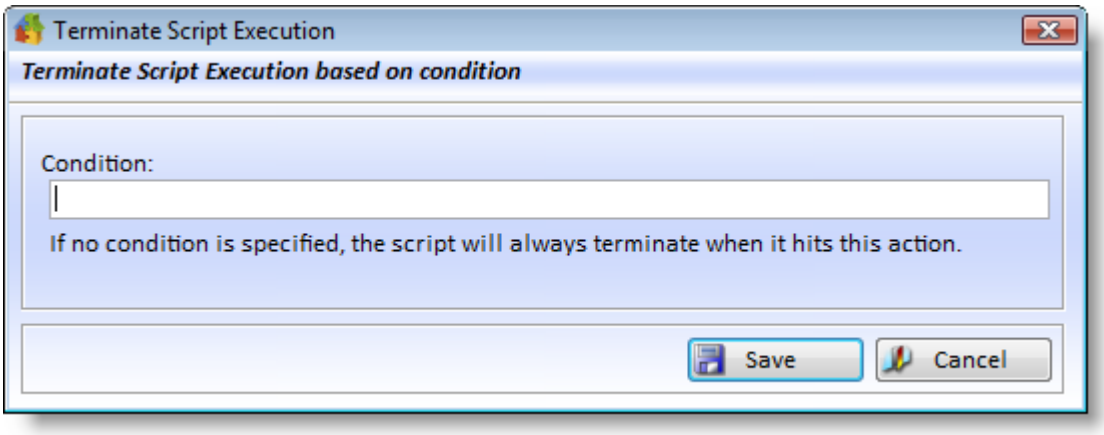
Note that all entry fields where you can select variables is indicated in the documentation with an image that looks like this: 

4.1 Script Actions

4.1.1 TerminateScript Script Actions

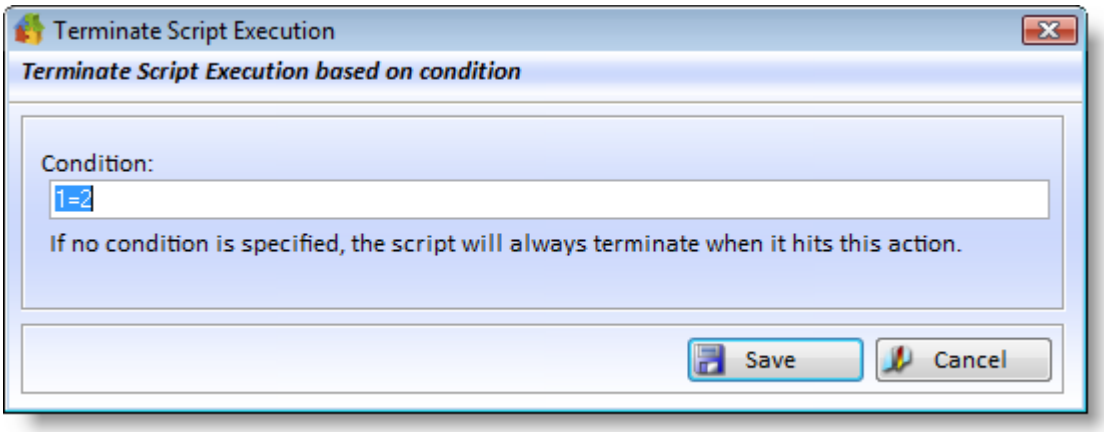
This action terminates the script execution immediately. No Action Items after this action will be executed and the script is aborted. This action has a condition that is evaluated at runtime to determine if the script will be terminated or not. If no condition is specified the script is terminated unconditionally. You can use constants or variables in the condition.

By using the action property you can also set a condition for this action, like any other action. The action condition has priority over the termination condition, i.e. the termination condition is never evaluated if the action condition is false and the action is not executed. For example if you have a condition on the action item that looks like 1=2 and a condition on the termination that looks like 1=1, then the action will not execute because of the 1=2 condition which will always be false.

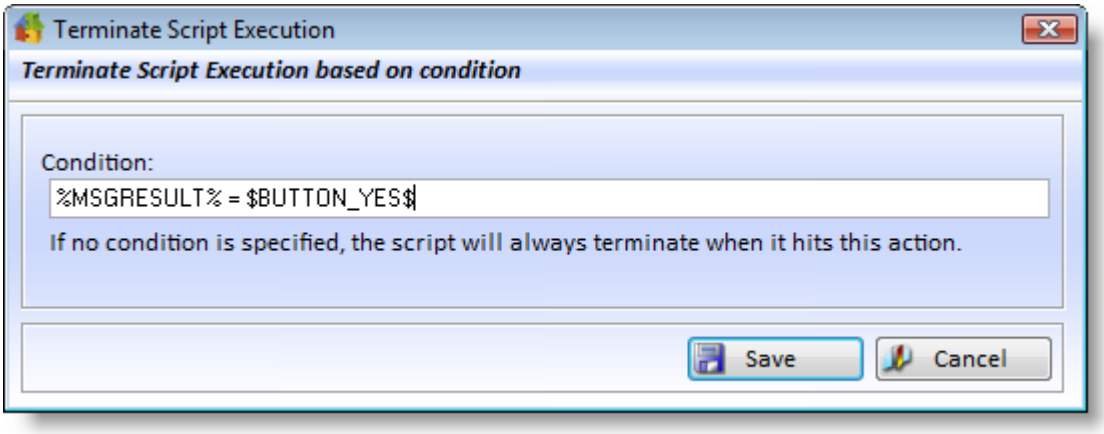


Properties	Explanation
Condition	Condition for termination of the script. This entry can contain constant values such as 1=2 if you want to temporarily disable the termination action or 1=1 if you want it to run always. You can also use variables and the contents will be evaluated at runtime. This could be used for example with the Message 127 action to terminate the script if the user presses a certain button.

The screenshot above shows the window without any condition in which case the terminate is unconditional, i.e. it will always trigger a termination of the script.



This screenshot shows a literal condition, such as 1=2 which would always be false so the termination would never occur. This can be useful when you are debugging or testing a script and you want it to terminate at a certain point. Then you could use condition like 1=2 to disable the termination temporarily.



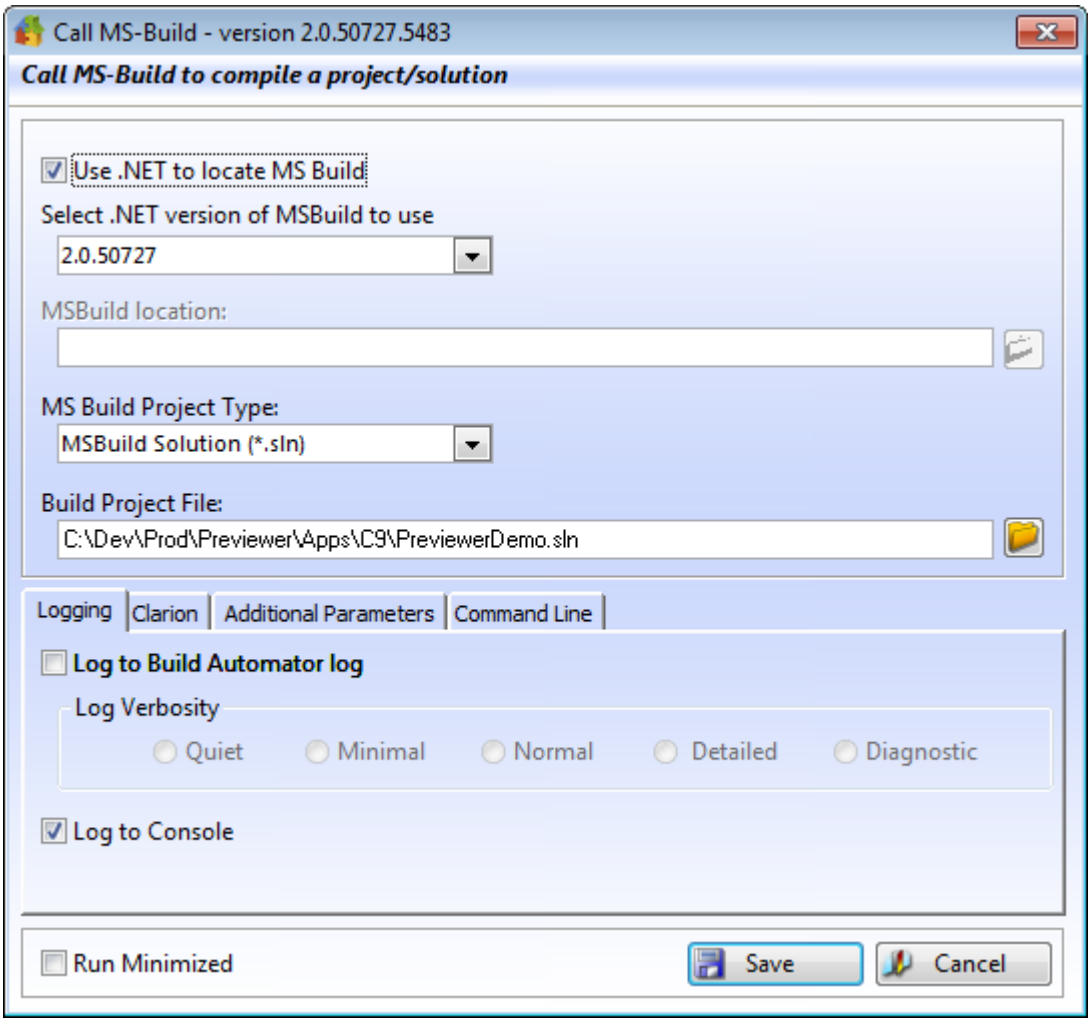
This screenshot shows a variable condition that is evaluated based on the values of those variables. In this case the %MSGRESULT% variable contains what button was pressed on a previously executed Message action 127 and the condition compares it to the $BUTTON_YES$ constant. If the user pressed the Yes button in the previous Message action 127 then it would trigger the Terminate Script action to execute and terminate the script.

4.2 Build Actions

4.2.1 Call MS Build Build Actions

MS-Build is a command line compiler that is used by many programming languages to compile projects and solutions in various programming languages. The Build Automator™ includes an action that can call the selected MS-Build version to compile your project or solution. Note that the Build Automator™ does NOT build your solution or project files for you, it simply takes your existing solution or project and MS-Build then compiles it using the related compiler. The Build Automator™ can work with any MS-Build Solution files (.sln files) and several project files, such as C#, Visual Basic, Visual C ++, Delphi and Clarion.

> ***NOTE:*** *Clarion developers using Clarion 7 and up: As of August 1st, 2014, Build Automator does support the use of* ***/ConfigDir*** *to set the folder for the Clarion configuration files. This allows setting up multiple environments with different settings. We have added an option so this can be used with both ClarionCL.exe as well as with MS-Build!*

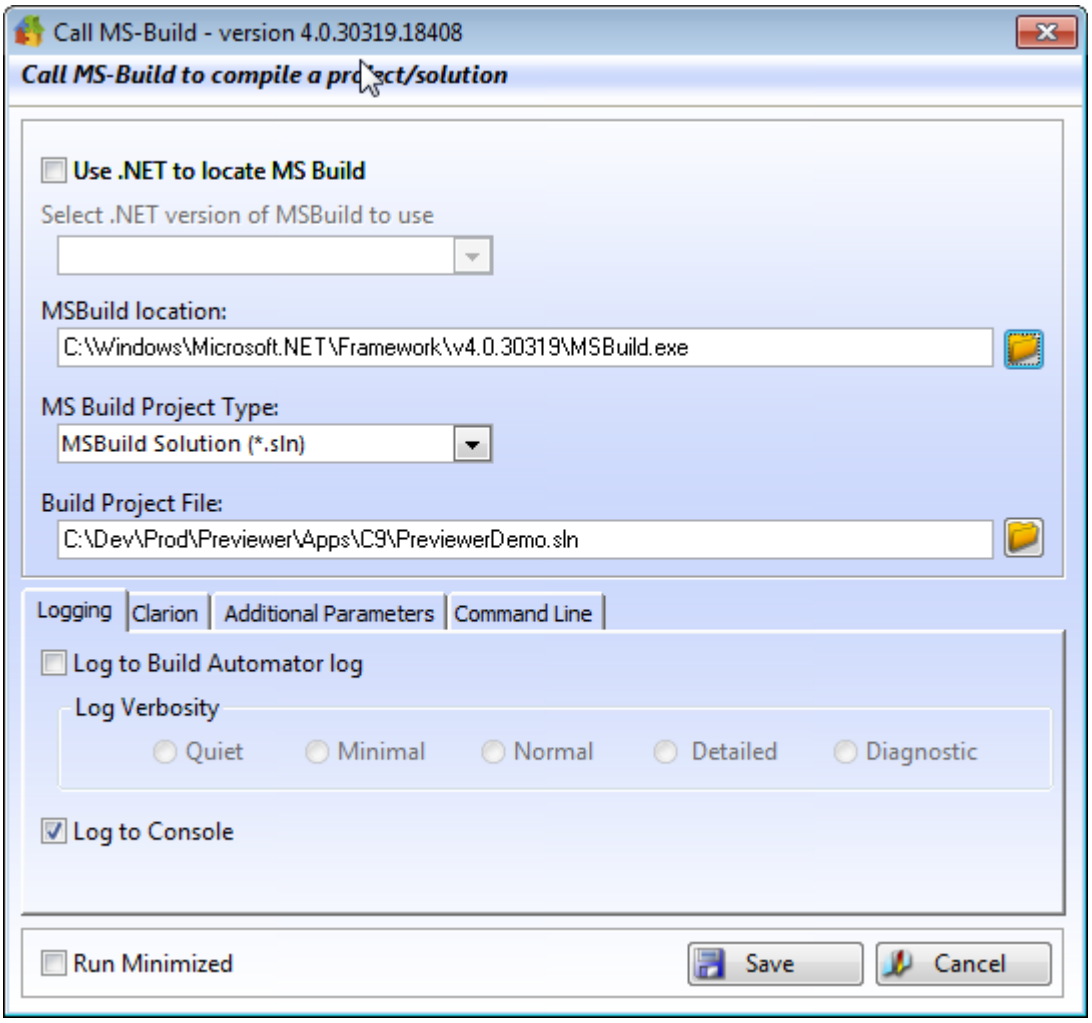


Properties	Explanation
Use .NET to locate...	If checked, you can select what version of the .NET framework is installed and the Build Automator™ will use that selection to locate MS-Build
Select .NET version...	Select the appropriate version from the dropdown
MS-Build location 	Option for you to manually select the MS-Build that you want to use. Use the button on the right of the entry to select the MS-Build.
Project Type	Select the MS-Build project type that you want to use. This is used when you select the solution or project to specify the correct file extension.
Build Project File 	This is the path and name of the solution or project file that you want to compile. Use the button on the right to select the MS-Build
Logging	On this tab you can specify if you want the MS-Build log to be added into the Build Automator log and the verbosity of the MS-Build log.
Clarion 74	On this tab you can specify what version of the IDE to use and optionally what version of the compiler. You can also specify if you want to generate code before you compile and if you wan t to specify a different configuration folder. This tab is only visible if the project is a Clarion project or the solution is made with Clarion.
Additional Parameters 	On this tab you can specify any additional parameters that you may want to use. Please refer to the MS-Build command line parameters information on MSDN for more information.
Command Line	On this tab you can see the generated command line based on the settings you have chosen. Note that this is a read-only version. If you need to modify the command line change the settings or add additional parameters.

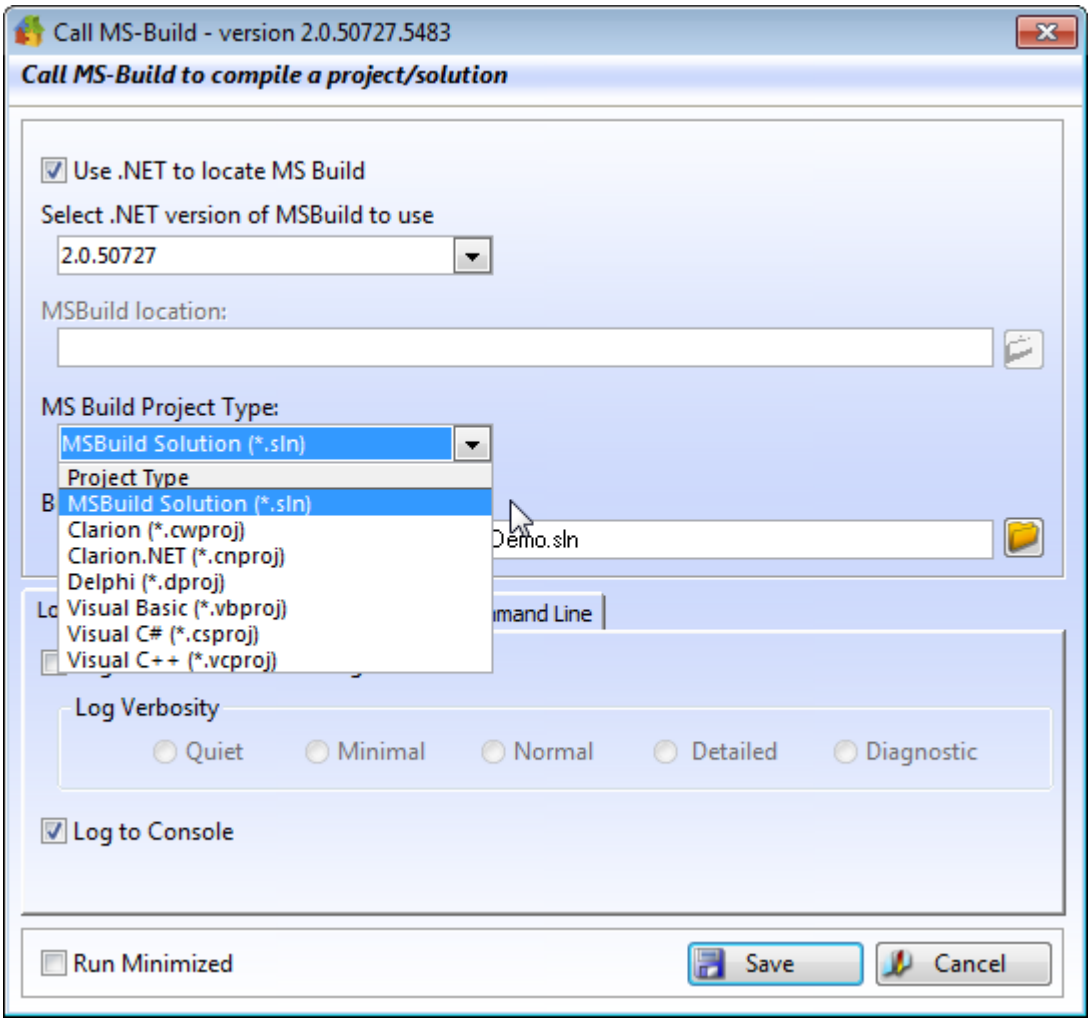
MS-Build comes in several different versions and the parameters vary based on what version you are using. You can locate the MS-Build executable on your computer in two slightly different ways.

First is by either selecting a version of .NET and the Build Automator™ will locate the MS-Build based on the .NET version. To use this option check the "Use .NET to locate MS Build" and then select the appropriate .NET version.

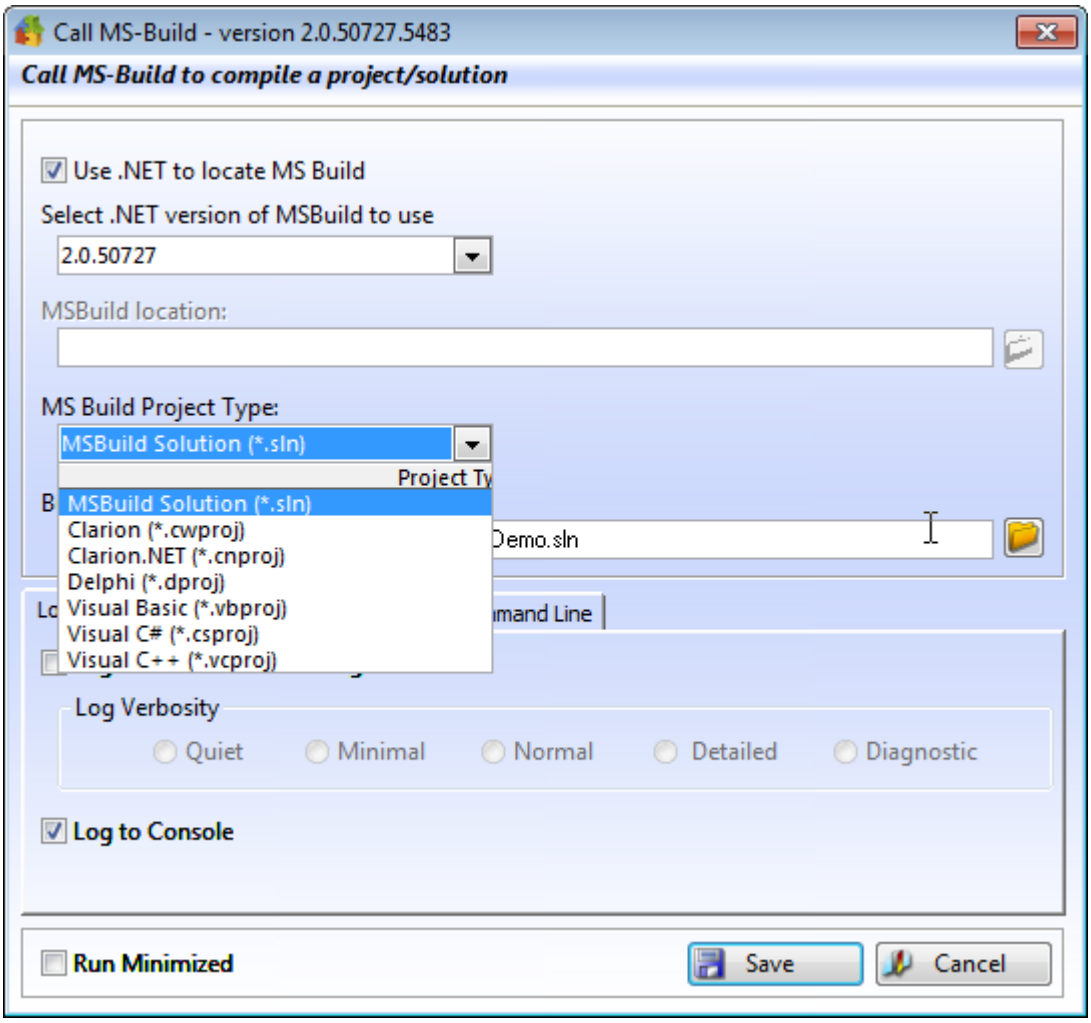
The second option is for you to select the MSBuild.exe manually. To use this option uncheck the "Use .NET to locate MS Build" and the MSBuild location entry and button will now be enabled. Click on the button to select the appropriate version of MS Build. By default it will go to the latest version of . NET that you have installed as a start folder for you to look for the MSBuild.exe.



Once you have selected the appropriate way to locate the MS-Build executable, you can now select the MS Build project type that you want to compile. The options you have are MSBuild Solution (*.sln), Clarion 7 (*.cwproj), Delphi (*.dproj), Visual Basic (*.vbproj), Visual C# (*.csproj) and Visual C++ (*.vcproj). Select the appropriate project type from the drop down.



Once you have selected the type, click on the button on the right of the entry for the "Build Project File" This will open a "Select File" dialog for you to pick the correct file to compile.



Finally there is a big box for you to enter any additional parameters that you may want. For a simple rebuild to a Release you would need something like:

/t:rebuild /p:Configuration=Release

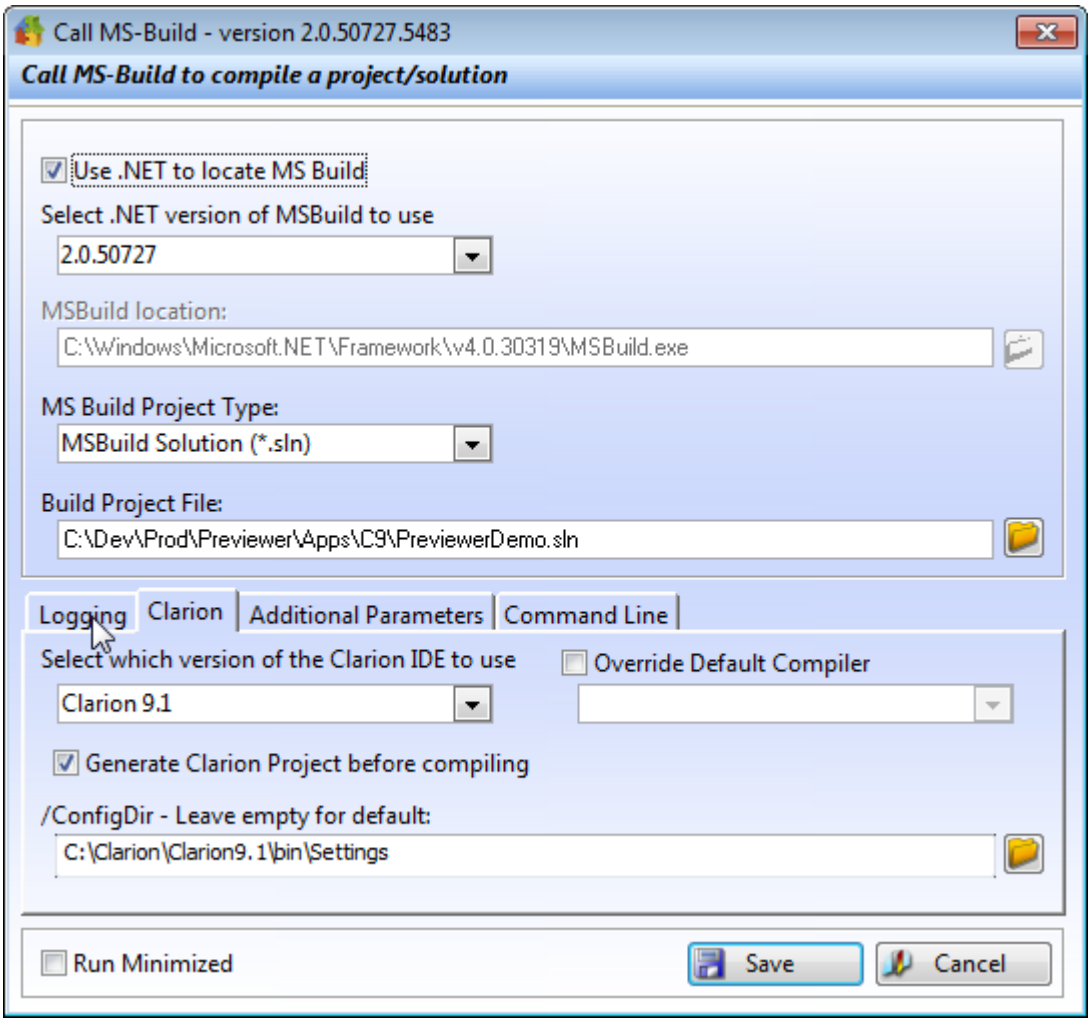
This will rebuild the target in Release mode. For more information on MS-Build parameters, please check out the online MS Build command line reference. To get more information about the parameters for your particular version, use:

MSBuild.exe /help

That will give you a list of the available parameters for your particular version.

Clarion tab

The Clarion tab is only used when compiling Clarion projects or solutions. If you have multiple installs of Clarion 7 and higher, you can select which IDE to use. You can also select what compiler to use if you need to use a version that is installed for the selected IDE.



Generate Clarion Project before compiling will call ClarionCL.exe to generate the code. There are limitations: This ONLY works with Clarion Enterprise Edition. The Professional Edition does **NOT** have the option to generate code using ClarionCL.exe.

If you use /ConfigDir when you run Clarion you can now specify the folder in Build Automator™. Please note that IF you use /ConfigDir without specifying the folder, you still need to use the folder here in Build Automator because it knows nothing about your Clarion shortcut settings, so you need to help by specifying the folder.

Note that if you have multiple builds of the same Clarion version installed, lets say Clarion 10, into multiple folders, then that can cause a problems for generation if the REDirection files are different for those different builds. The reason is that the Clarion installer saves the root folder into the registry. That root folder will always point to the root of the last installed Clarion build. Lets say you install 3 builds of Clarion 10:

C:\Clarion\Clarion10.11975
C:\Clarion\Clarion10.12028
C:\Clarion\Clarion10.12278

and in this order. Then the HKEY_LOCAL_MACHINE\SOFTWARE\SoftVelocity\Clarion10\root value will be C:\Clarion\Clarion10.12278. This only seems to cause an issue if the RED files are different,

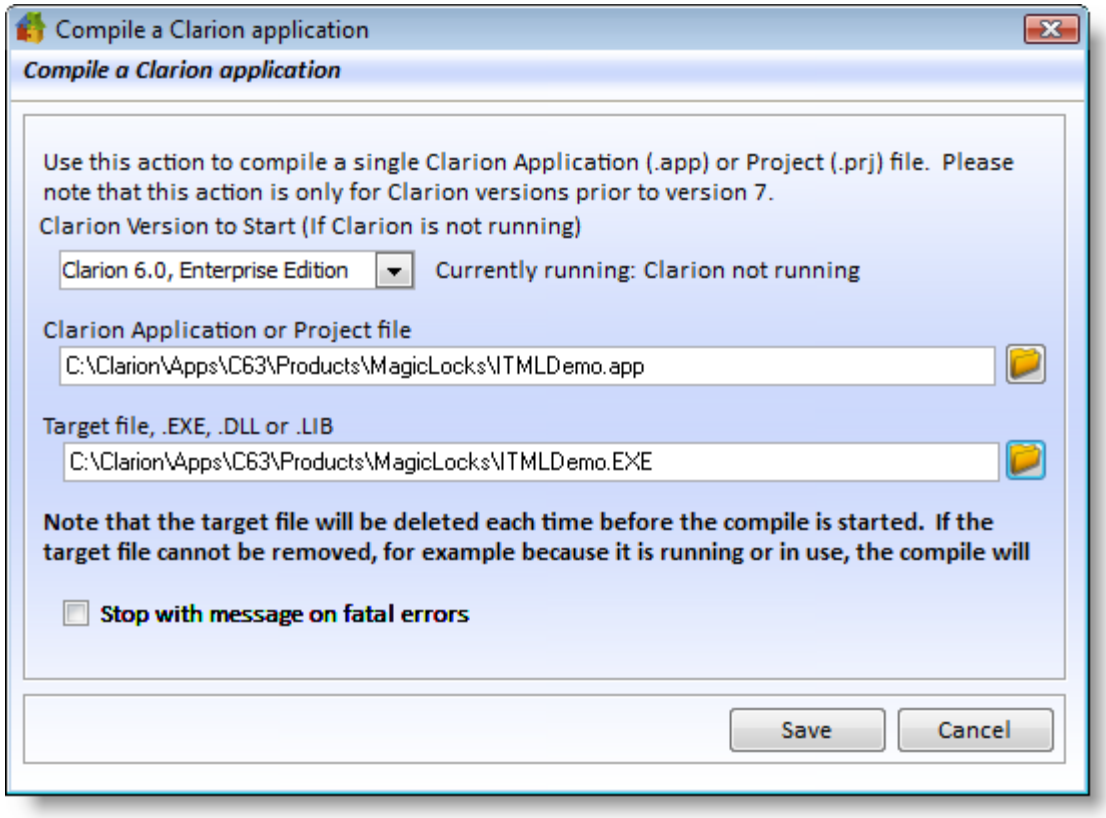
but it is possible that this causes problems with different versions of code being compiled. If you have this situation, please make sure that the HKEY_LOCAL_MACHINE\SOFTWARE\SoftVelocity \Clarion10\root points to the correct Clarion version that you are using.

Additional web resources for MSBuild (last revised in May 2016):
MSBuild Reference
MSBuild Command Line reference
MSBuild Reserved and Well-Known Properties
MSBUild Common Project Properties

Older MSBuild links:
http://www.codeproject.com/KB/books/msbuild.aspx
http://blogs.msdn.com/msbuild/
MSDN TV video on MS-Build
[****]

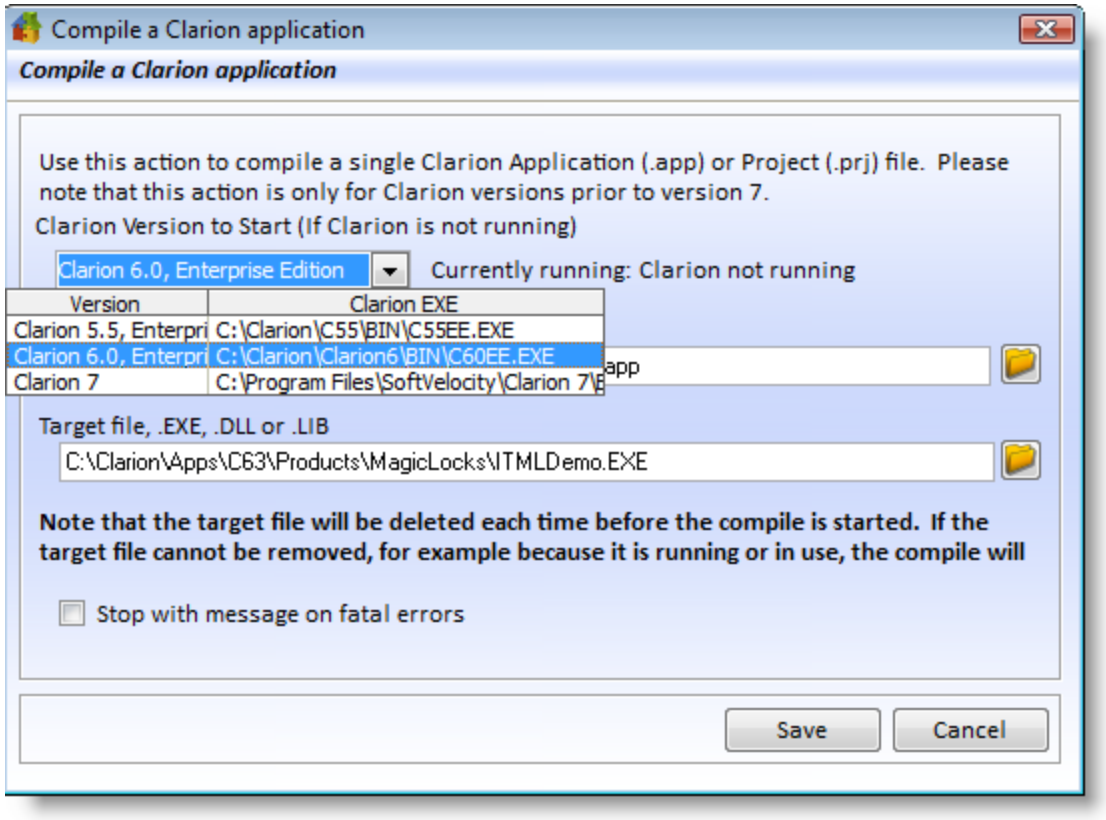
4.2.2 Compile Clarion — Build Actions

This action compiles a selected Clarion application file (*.app) or a selected Clarion project file (*.prj). **Note that this action is only for Clarion 6 and older.** For Clarion 7, Clarion 8 and Clarion.NET use the Call MS Build 70 action. There are some limitations in this action. Even though you can specify which Clarion version to run the action cannot guarantee that the app will compile in the correct version IF you have multiple versions of Clarion running. This can cause complete confusion because the DDE server in each version has the same name and that DDE server has to be called to compile the application. So the safest way is to make sure that either Clarion is **not** running or **only** the correct version of Clarion is running before you execute this action.



Properties	Explanation
Clarion version...	Select the Clarion version to start if it isn't already started.
Application or Project	Select the .app or .prj file to compile.
Target file	You must select the target file that the compilation will create. This must match what you have set in your Clarion app/project as the "Target File"
Stop with message...	Stop the script on fatal errors allowing you to determine if you want to continue or not.

First you need to select the correct Clarion version to start. This is **only** used if no Clarion version is running.

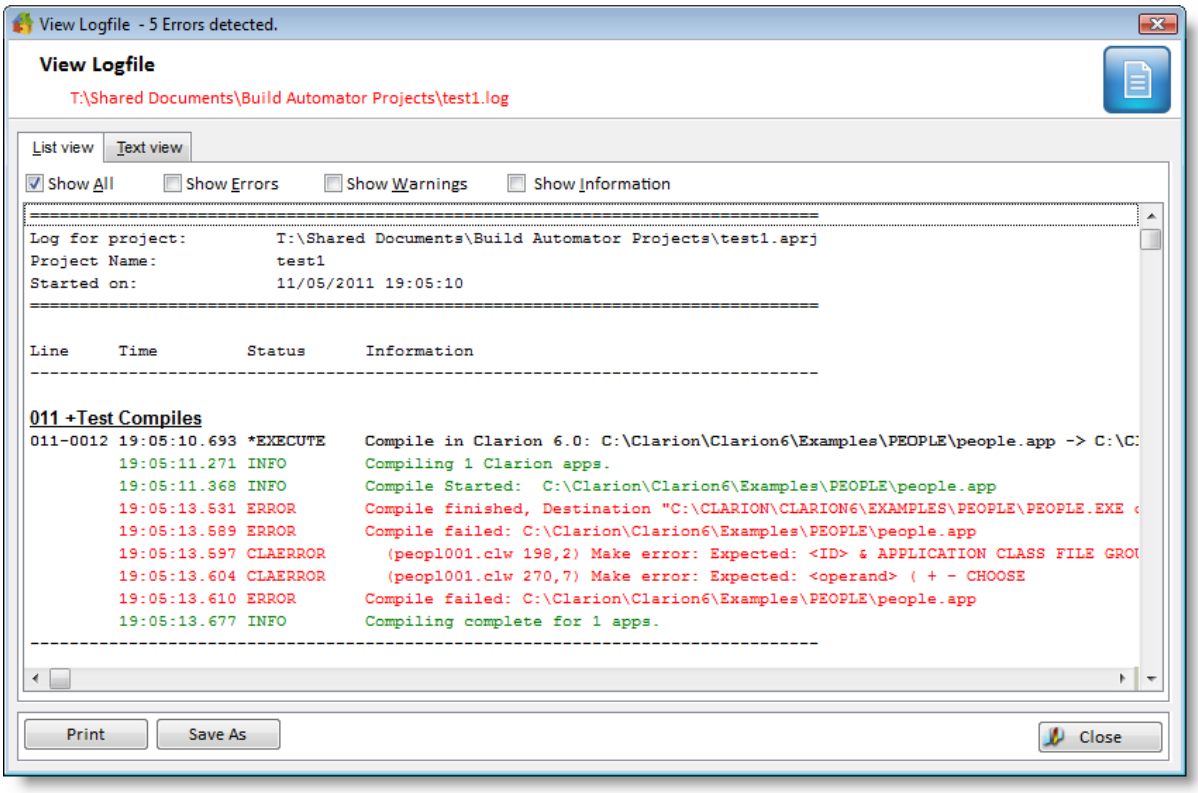


Select the appropriate version from the dropdown. It will show you both the version information and the location of the Clarion executable.

You must select both the application or project file to compile and also the destination file. If the destination is not specified, the action will not run. Before the compile starts, the destination file is removed. The creation of the destination file is the indication that the compile process was successful.

If you want the compile process to stop on a fatal error during the Clarion compile process, then check the "Stop with a message on fatal errors" This will allow you to terminate the execution of the script.

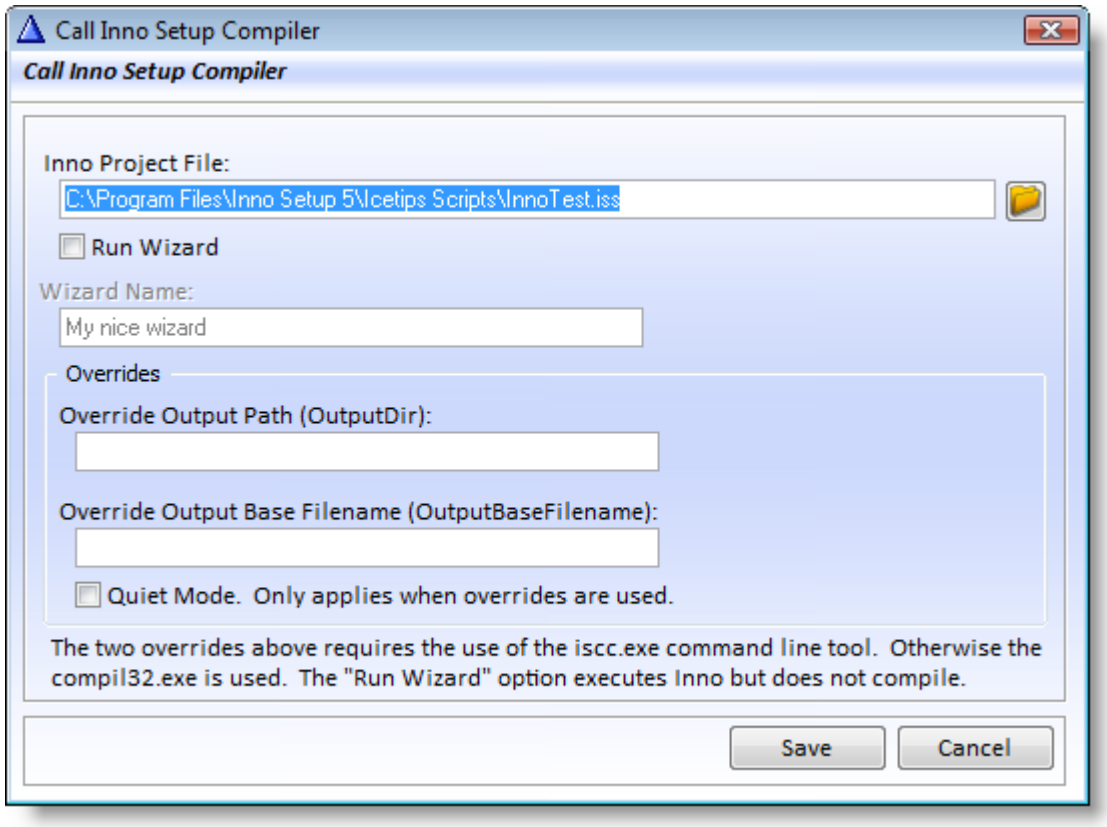
Starting in version 1.5 you now get the same details in the Build Automator log as you would get in the Clarion Compiler window. This makes it possible for you to look at the Build Automator log and see exactly where errors may have occurred during the compiling process. See example below of a compile that caused 4 errors during compiling.



See also:
Compile Multiple Clarion apps 81

4.2.3 Compile Inno — Build Actions

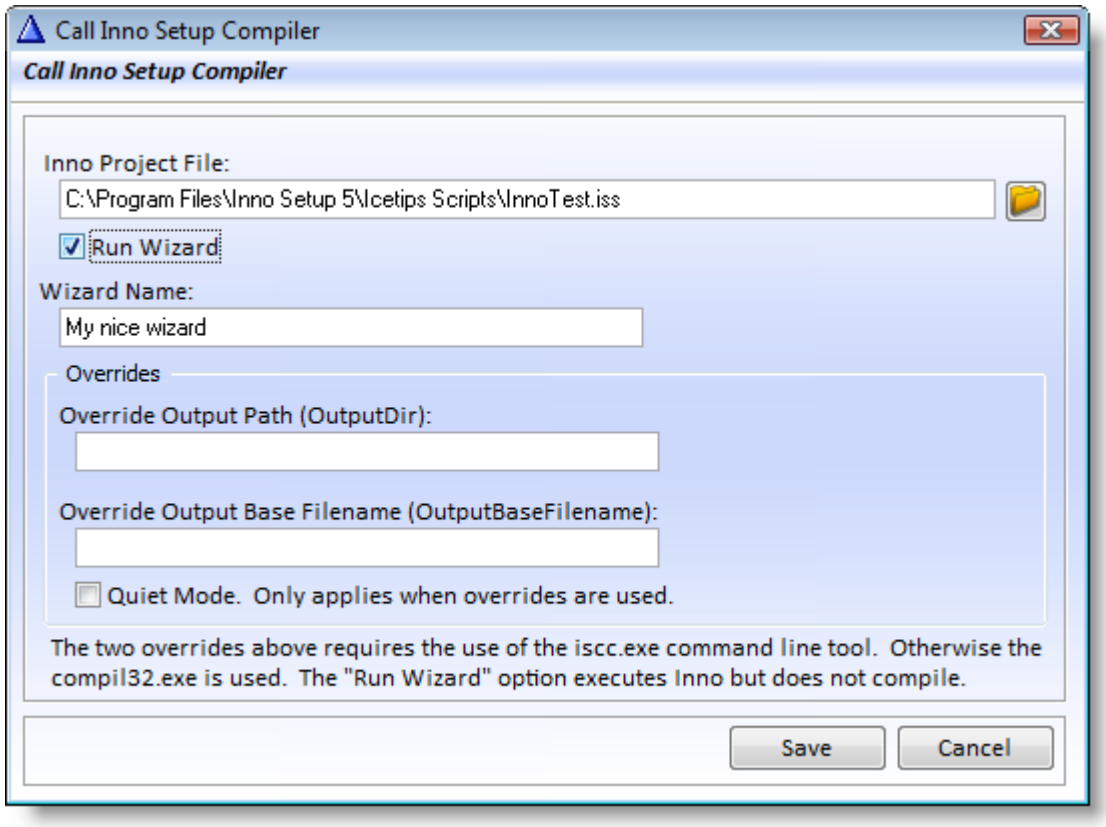
This action compiles installation projects created with Inno Setup. Inno offers only a few command line parameters that can be used to modify it's output.



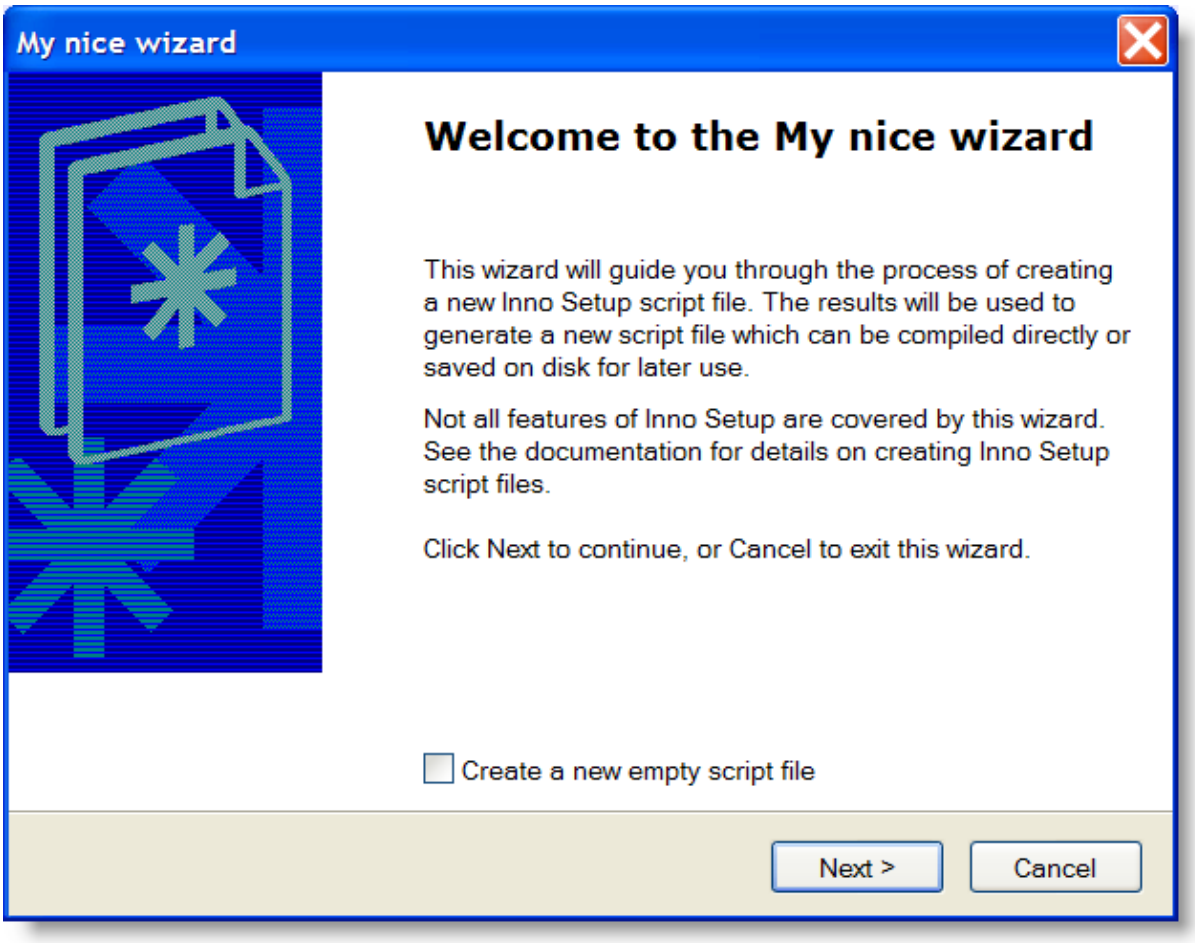
Properties	Explanation
Inno Project File	The Inno Project file (*.iss) to compile. If not overrides are specified, this will execute the Compil32.exe compiler.
Run Wizard	Runs the Inno Wizard instead of running the script. The Wizard creates a new install script.
Wizard Name	Name used in the wizard (see screenshot below). This will execute the Compil32.exe compiler.
Override Output...	Overrides the output path for the compiled install executable. This is a single folder name that is used instead of Output. This will execute the ISCC.exe compiler.
Override Base file...	This is the filename without extension to use for the compiled install executable. This will execute the ISCC.exe compiler.
Quiet Mode	This only applies with either of the overrides are used as then the project is compiled with the ISCC.exe command line compiler.

First you need to select the Inno project file (.iss) that you want to compile.

If you want to use Inno to create a new install script, check the "Run Wizard" The name of the Wizard, entered in Wizard Name, will become the name of the wizard being run.



This calls up the Wizard dialog in Inno using the "My nice wizard" as the caption and also the name in the install. See screenshot below.

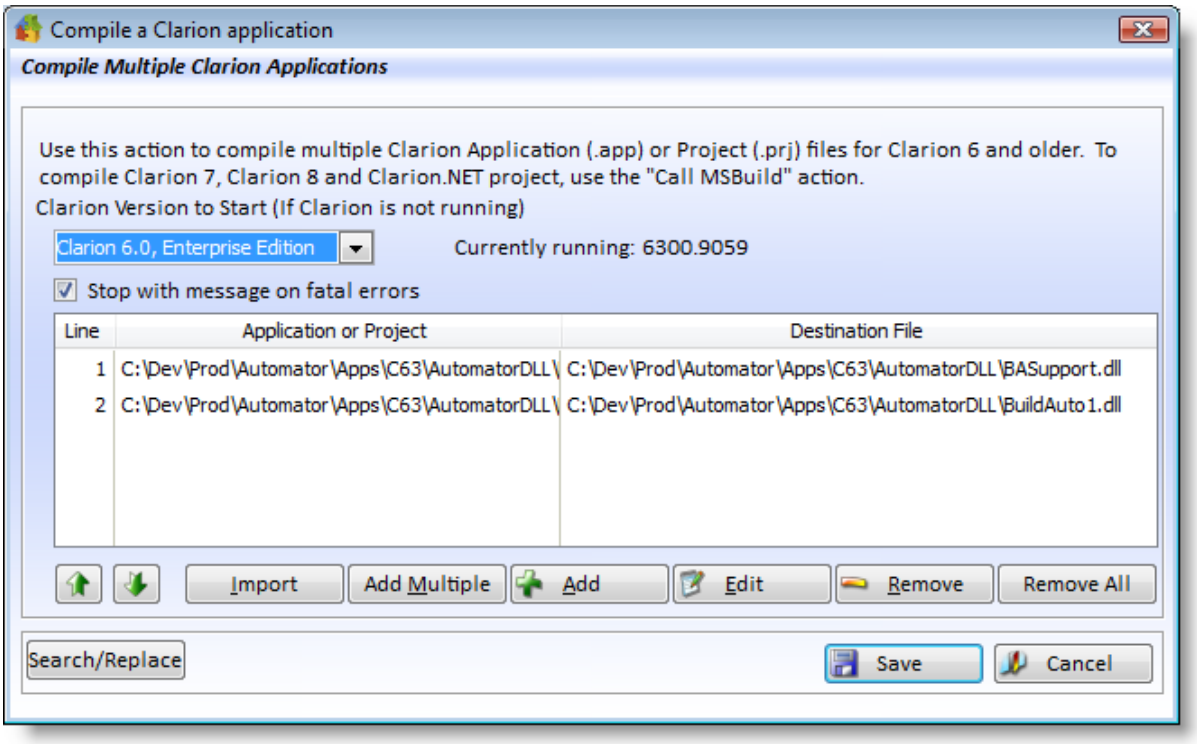


You can override the output path and the output base filename, corresponding to the /O and /F flags respectively.

For more information about the command line options in Inno, please see the "Command Line Compiler Execution" topic in the Inno help.

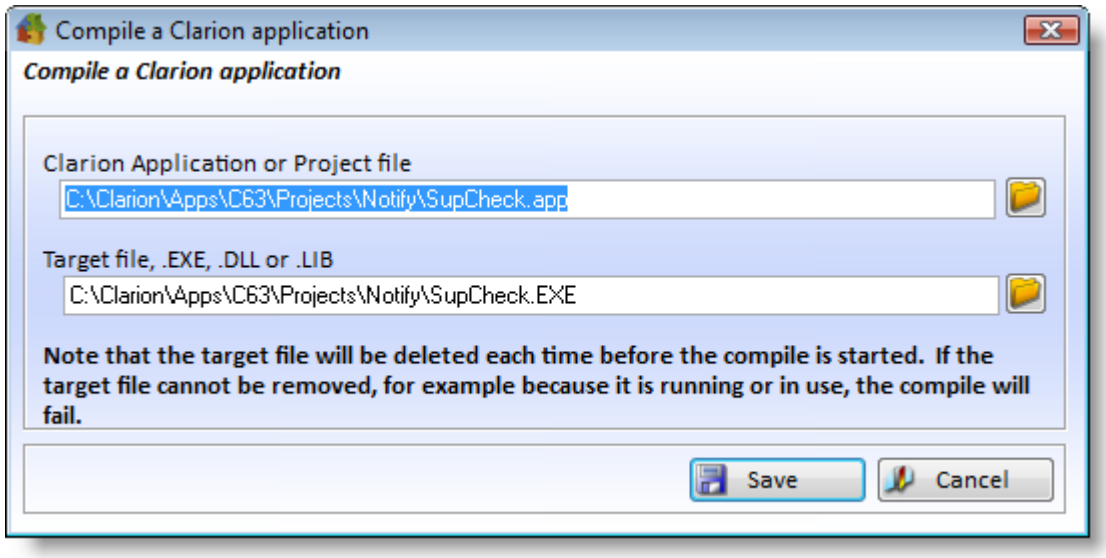
4.2.4 Compile Multiple Clarion apps — Build Actions

This action makes it possible to select multiple applications or projects to compile in Clarion, rather than pick one application at the time. **Note that this action is only for Clarion 6 and older.** For Clarion 7, Clarion 8 and Clarion.NET use the Call MS Build 70 action. There are some limitations in this action. Even though you can specify which Clarion version to run the action cannot guarantee that the app will compile in the correct version IF you have multiple versions of Clarion running. This can cause complete confusion because the DDE server in each version has the same name and that DDE server has to be called to compile the application. So the safest way is to make sure that either Clarion is **not** running or **only** the correct version of Clarion is running before you execute this action.



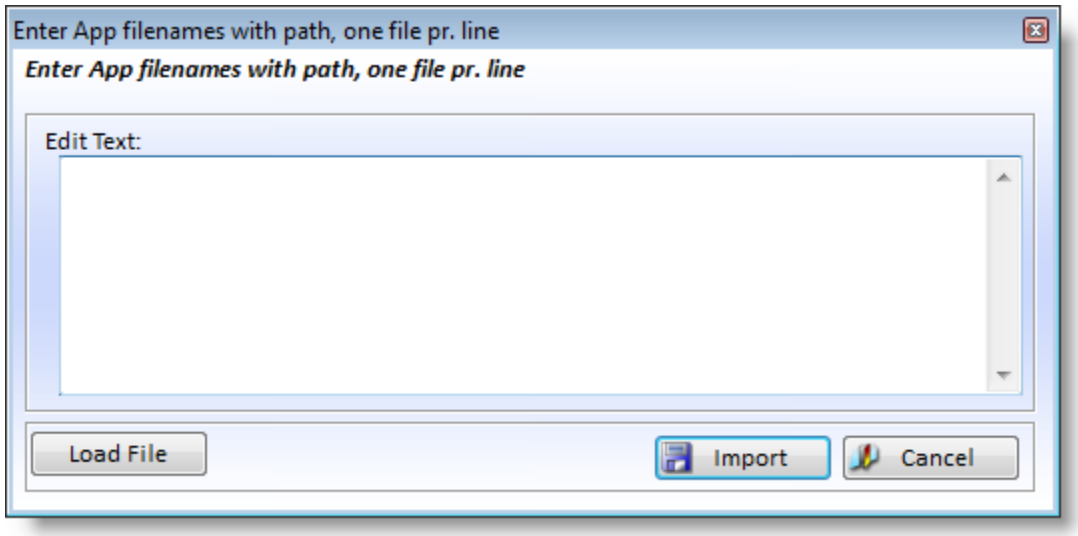
Properties	Explanation
Clarion version...	Select the Clarion version to start if it isn't already started.
Stop with message...	Stop the script on fatal errors allowing you to determine if you want to continue or not.
Applications	List of applications or project files to compile along with their destination files

To insert a single application or project click on the "Add" button. To edit the currently selected item in the list, click on the "Edit" button. To delete the selected item, click on the "Delete" button.

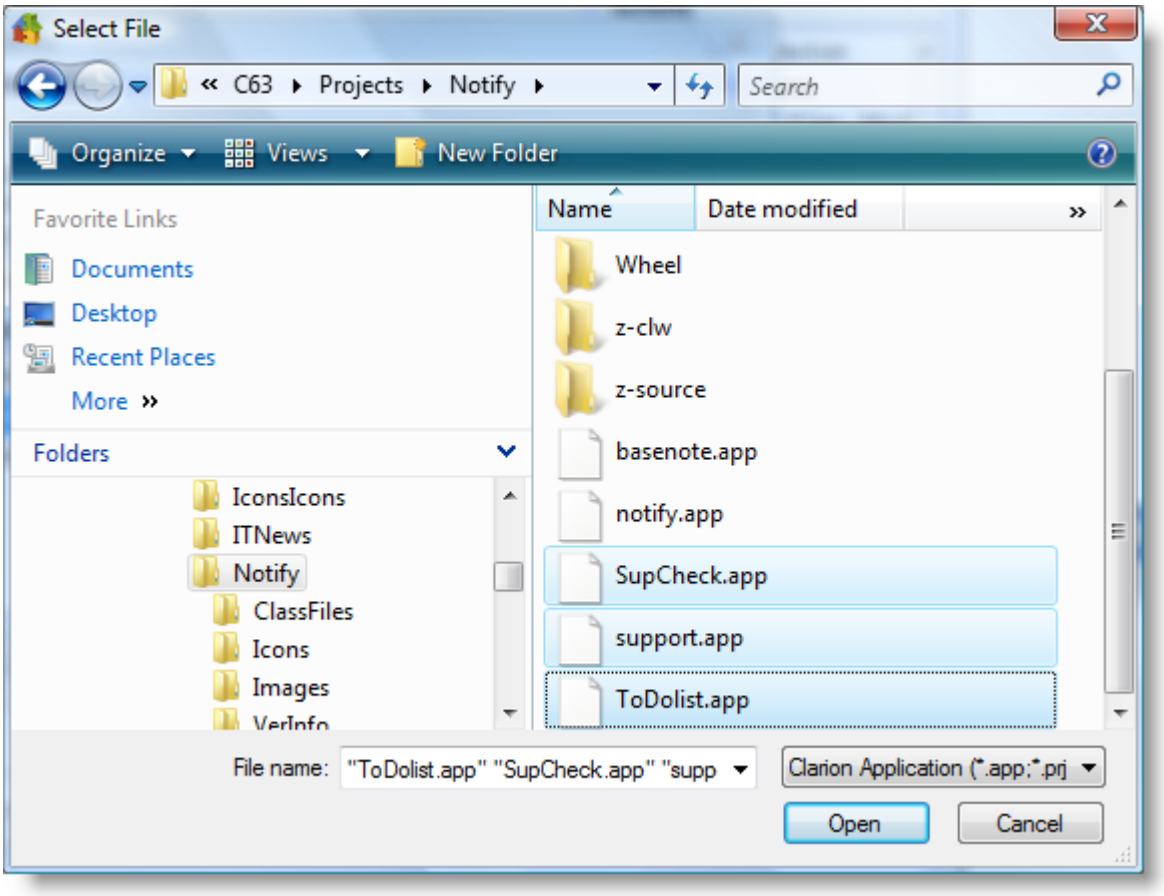


When you select the Application or Project file, Build Automator reads the selected .app file and determines the name of the target file. If it cannot extract the target name from the .app file, it will fill the "Target File" entry with the path and filename of the application or project file but without the extension and prompt you to add the extension. **The target file must match what you have set in your Clarion application or project as the "Target File" or the compile process may not work properly.**

The "Import" button can be used to add files from a list or from a file. It opens a window where you can enter a file list or copy it from some other source, or load it from an external file.

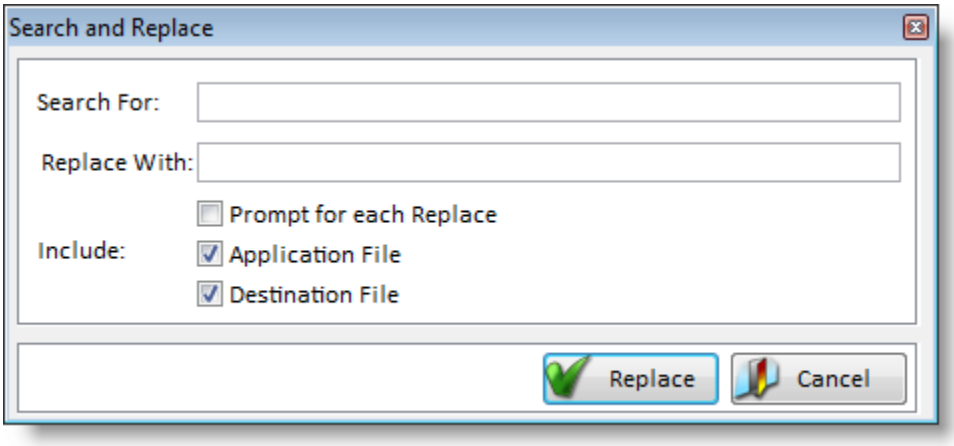


The "Add Multiple" button can be used to add multiple applications to the list with a single selection. The apps added with the "Add Multiple" must be in the same folder, but the files generally added to the list can be anywhere. When you press the "Add Multiple" buttons you see a file selection dialog pop up.



Hold down the Control key on the keyboard and left click on the files that you want to pick, or use Shift and left click to select a range of file. When you click on the "Open" button the files will be brought into the file list list. The Build Automator will attempt to locate the target files in the same folder as the .app or .prj files. If it can, it will add them automatically as target files, otherwise it will leave the target blank. The target file entry on the update window must be filled out or the compile will fail.

You can do a quick search and replace operation on the Application and Destination columns by clicking the "Search and Replace" button. This will bring up the following window:

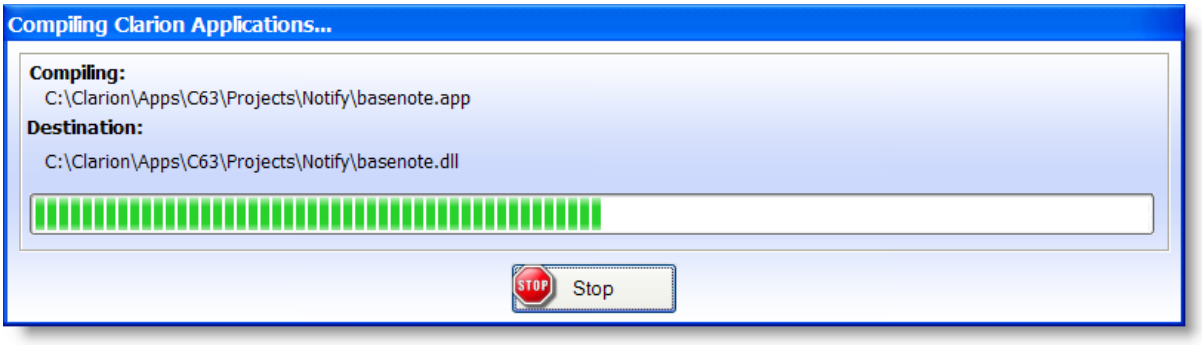


This is an excellent option to set variables to use for either or both the Application file and the

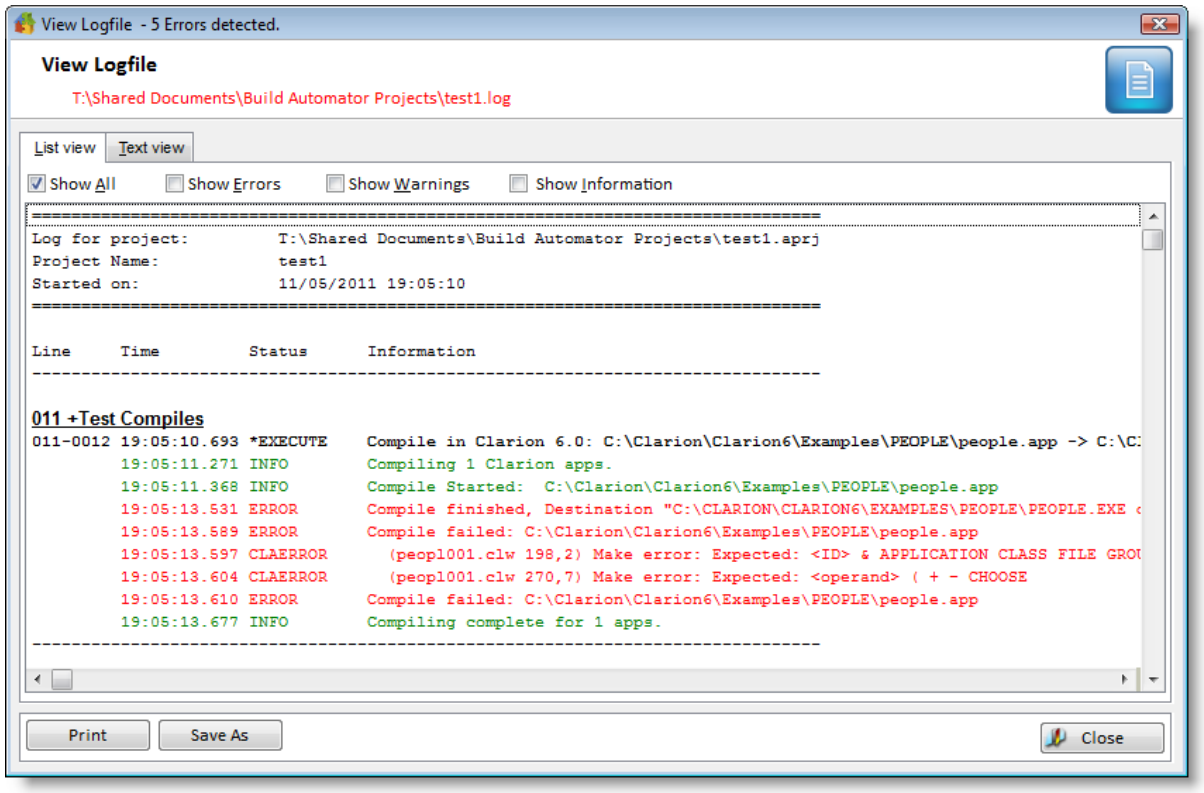
Destination file. That allows you to set the source and destination with variables and thereby make it very easy to change locations if needed.

The "Remove All" button will remove all entries in the list. It will prompt you for confirmation before it deletes.

During the execution of the compile process, you will see a progress window that shows what application is being compiled. The Clarion IDE will get focus as soon as it is connected and communication with it starts. Once the compiling is done, the Build Automator IDE will gain focus again.



Starting in version 1.5 you now get the same details in the Build Automator log as you would get in the Clarion Compiler window. This makes it possible for you to look at the Build Automator log and see exactly where errors may have occurred during the compiling process. See example below of a compile that caused 2 errors during compiling. Note that this is not 100% fool proof! If your compile is extremely fast and hits an error immediately and closes the compiler window, Build Automator may not have a chance to get the information from it. This happens rarely and mostly with very small hand coded projects. Build Automator extracts the information from the compiler window but it takes it a few milliseconds to find it and set it up so that it can retrieve the information from it.



See also:
Compile Multiple Clarion apps 81

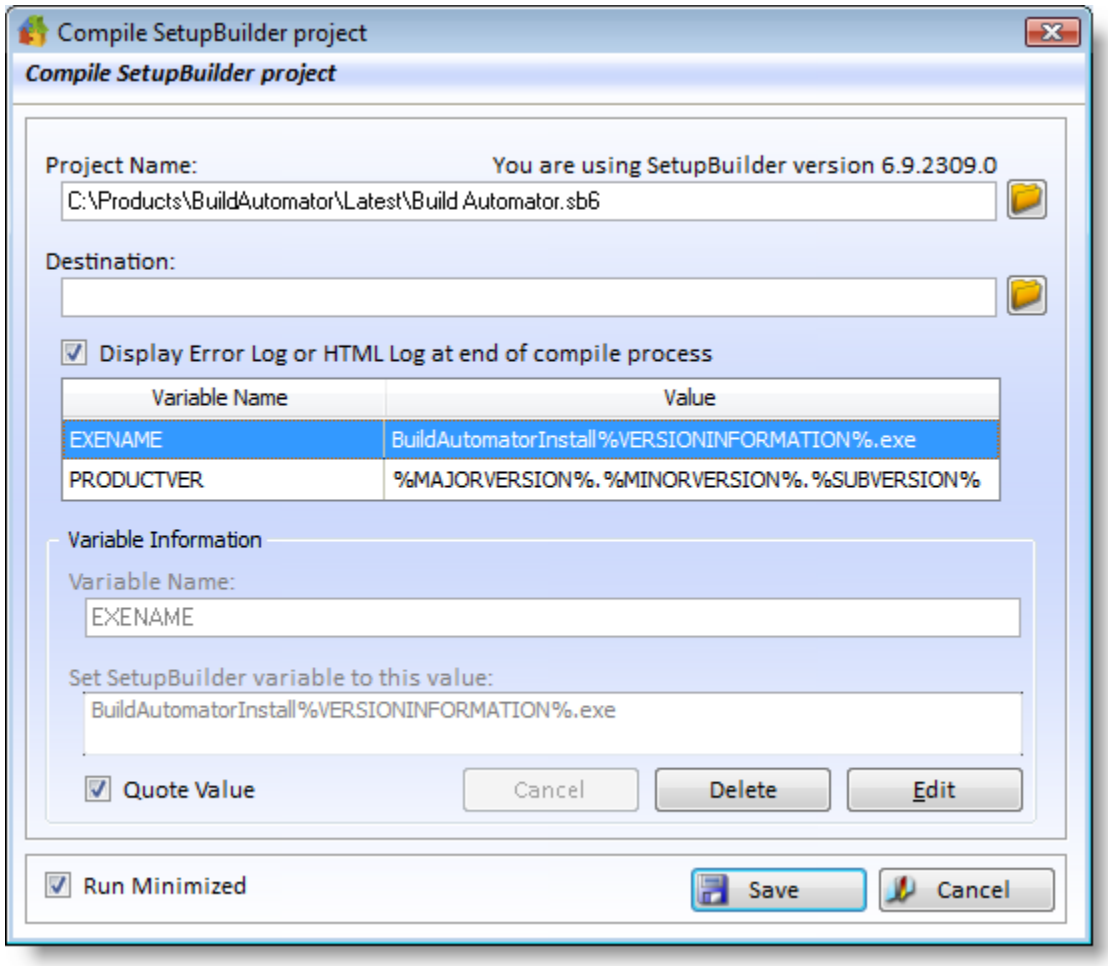
4.2.5 Compile SetupBuilder — Build Actions

This action takes care of compiling Setup Builder projects.

There are some limitations to this action. If you happen to have Setup Builder version 5 and version 6 installed, there is a problem picking which compiler to use. Since Setup Builder 6 associates both .sb5 and .sb6 files with Setup Builder 6 there is no way to find the appropriate program to compile with. Setup Builder 7 is fully supported.

Please note that you can **only update compiler variables** in your Setup Builder project from the Build Automator. There is currently no way to pass information to set normal variables in Setup Builder. What you can do is create compiler variable and in your Setup Builder script set your normal variable to the value of the compiler variable, then pass that value to Setup Builder from the Build Automator.

There are also differences in the various builds in regards of changing Setup Builder variables, making it impossible to reliably set the output file (EXENAME) before Setup Builder 6.5 build 2000.

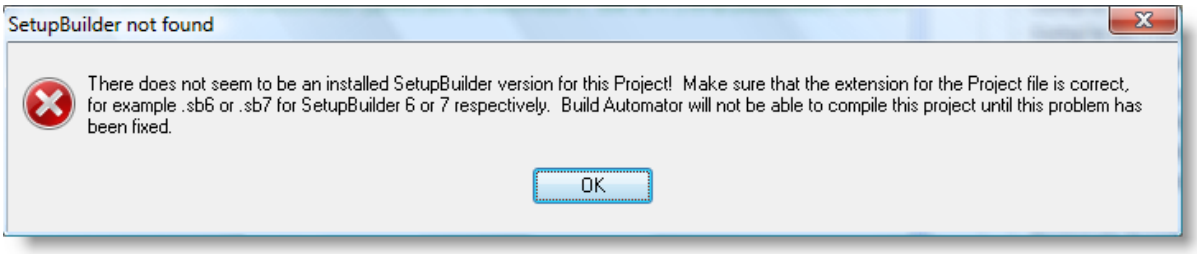


Properties	Explanation
Project Name	The project file to compile. Use the button on the right to select a Setup Builder project.
Destination	Optional setting that specifies an alternate destination of the compiled install executable. Note that this does not work before Setup Builder 6.5 build 2000.
Display Error Log...	Check this to open the error log or the html log at the end of the Setup Builder compile process. Note that the Build Automator™ will continue execution of the script.
Variables	You can specify any number of variables to update in the Setup Builder project. Note that these variables are **not** saved in the Setup Builder project, only temporarily updated with the passed in values. For example if you have PRODUCTVER set in your Setup Builder project as 1.2.0.0000 and you let the Build Automator™ update PRODUCTVER to 1.2.4.1234, your Setup Builder project will still contain 1.2.0.0000 after you compile. Also note that these variables are currently being sent on the command line and that there are limitations to how much data can be sent.
Variable Name	Name of the variable in SetupBuilder to update. The variable name is forced uppercase to perfectly match Setup Builder variables.
Set	Specify the value that you want to set the SetupBuilder variable to.

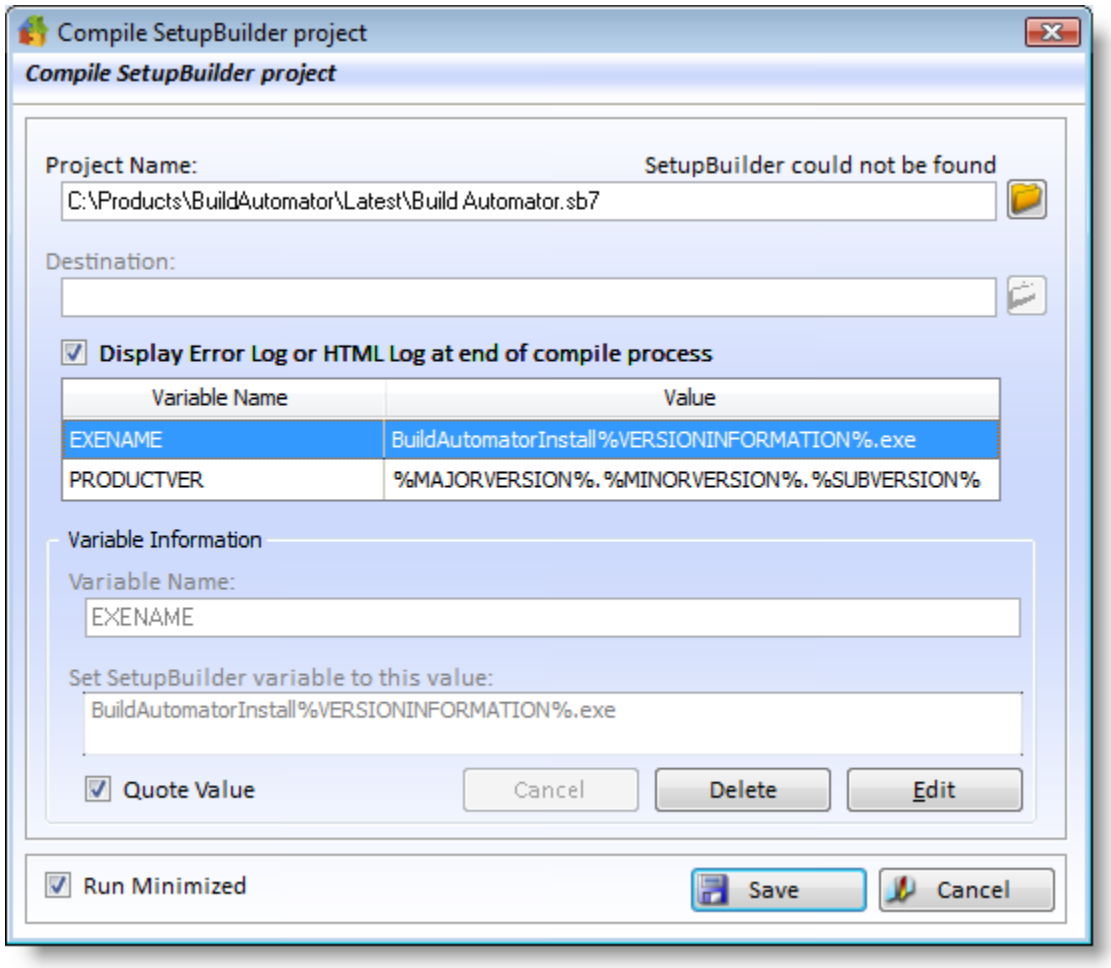
Variable...	
Quote Value	When this is checked the value passed on the command line is placed in double quotes. This is normally what is required so it is checked by default.

To create a variable, right click on the variable list and select "New Variable". To edit a variable click on the Edit button. Specify the variable name and the value you want to set it to.

If the Setup Builder version required to compile the project is not found on the machine where the script is running, you will get a warning when you try to open the action.

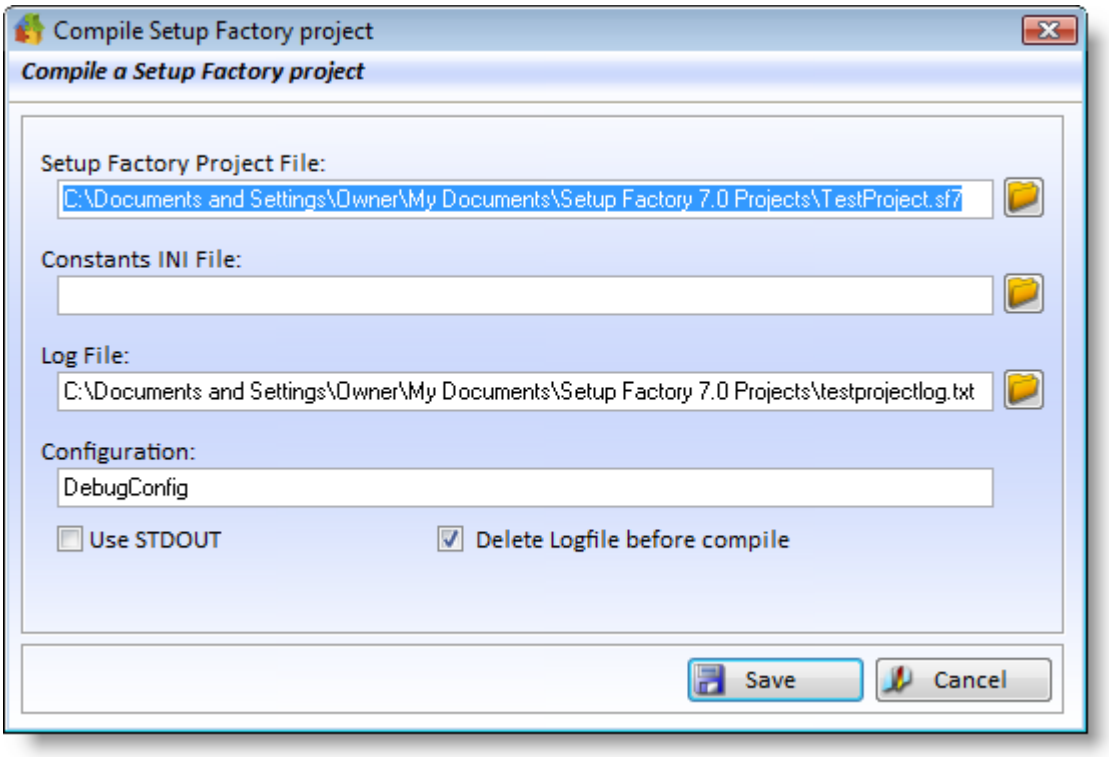


The action window also indicates that it could not find the required version of Setup Builder.



4.2.6 Compile Setup Factory Build Actions

This action compiles scripts created with Setup Factory from Indigo Rose. Please note that we completed this action for Setup Factory version 7. Setup Factory version 8 (released August 2008) should not have any problems since it uses exactly the same command line parameters as version 7.



Properties	Explanation
Project File	The project file to compile. Use the button on the right to select a Setup Factory project.

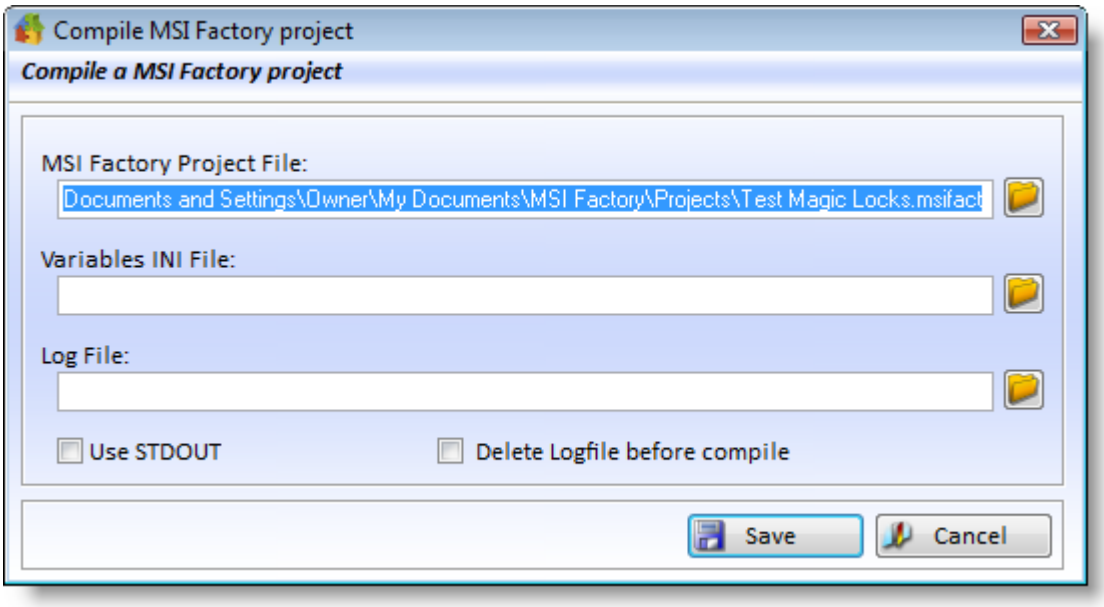
Constant INI	Lets you specify an INI file that contains design-time constants to override the ones in the project. You can define as many design-time constants as you want in the INI file, with each constant on a separate line beneath the [Constants] section. Each constant that is defined in the INI file must already be defined in the project file. For example:
	`[Constants]`
	`#OUTPUTDIR#=C:\Output\Foobar 2002\Release`
	`#SETUPNAME#=foobar2002setup.exe`
	`#BUILD#=release`

Log File	Select the name of the logfile if you want to redirect logging to a different file. If the file does not exist Setup Factory will create it. If it does exist Setup Factory will append the new log information to the existing file unless you check the "Delete Logfile before compile"
Configuration	Specifies the build configuration to use in the build.

Use STDOUT	Redirects the build status output to the Standard Output device (console)
Delete Logfile...	Use this to create a new logfile every time to compile the project. Setup Factory will append the log to an existing logfile but this option allows you to create a new logfile each time you build.

4.2.7 Compile MSI Factory — Build Actions

This action compiles scripts created with MSI Factory 2.0 from Indigo Rose. MSI Factory creates Microsoft Windows Installer (.msi) format software installers.



Properties	Explanation
Project File	The project file to compile. Use the button on the right to select a MSI Factory project.

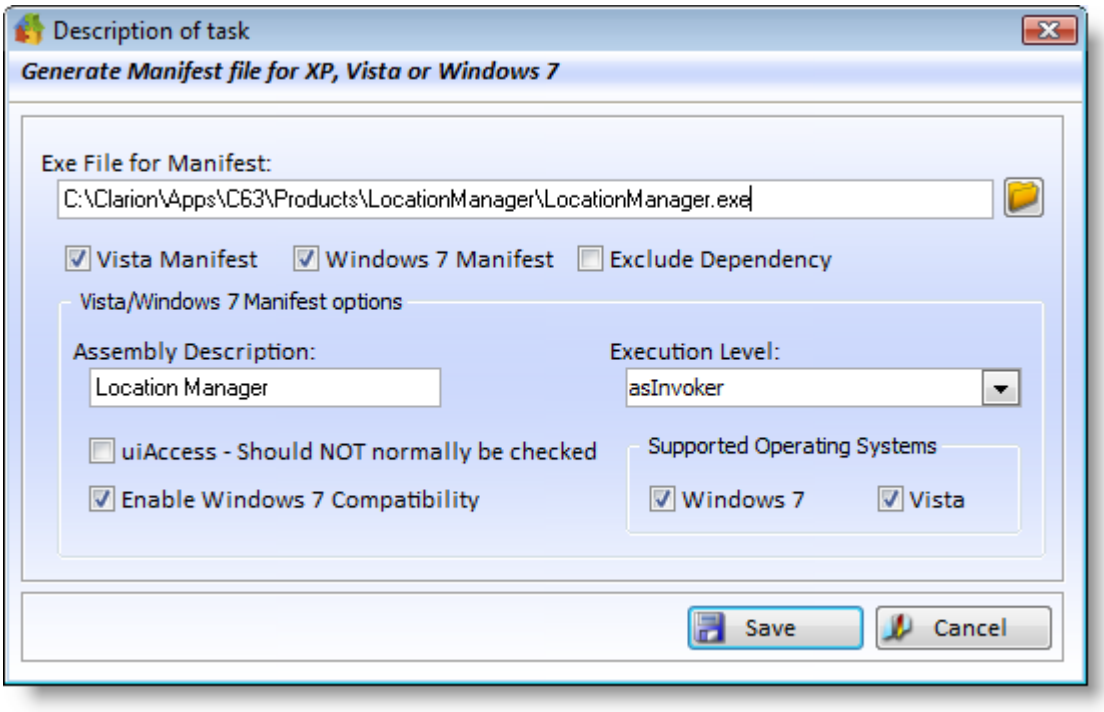
Variables INI	Lets you specify an INI file that contains build variables to override the ones in the project. You can define as many build variables as you want in the INI file, with each variable on a separate line beneath the [Variables] section. Each variable that is defined in the INI file must already be defined in the project file. For example:
	`[Variables]`
	`OUTPUTDIR=C:\Output\Foobar 2002\Release`
	`SETUPNAME=foobar2002setup`
	`BUILD=release`

Log File	Select the name of the logfile if you want to redirect logging to a different file. If the file does not exist MSI Factory will create it. If it does exist MSI Factory will append the new log information to the existing file unless you check the "Delete Logfile before compile"
Use STDOUT	Redirects the build status output to the Standard Output device (console)
Delete Logfile...	Use this to create a new logfile every time to compile the project. MSI Factory will append the log to an existing logfile but this option allows you to create a new logfile

	each time you build.

4.2.8 Generate XP/Vista/Win7 Manifest Build Actions

Part of completing software is to create and link manifest files for Windows XP™, Windows Vista™ or Windows 7™. While those files are simple XML files it can be very handy to be able to automatically generate them. This action does exactly that. It can create either XP or Vista manifests. Normally you would just create a Vista manifest to be 100% compatible with both operating systems, since the Vista manifest is backward compatible with XP. For more information about Vista manifests, please visit this page on MSDN: http://msdn.microsoft.com/en-us/library/bb756929.aspx. For more information about **User Account Control** (UAC), please visit this page on MSDN (http://msdn2.microsoft.com/en-us/library/bb530410.aspx)



Properties	Explanation
Exe File for Manifest	Select the executable file to create the manifest for.

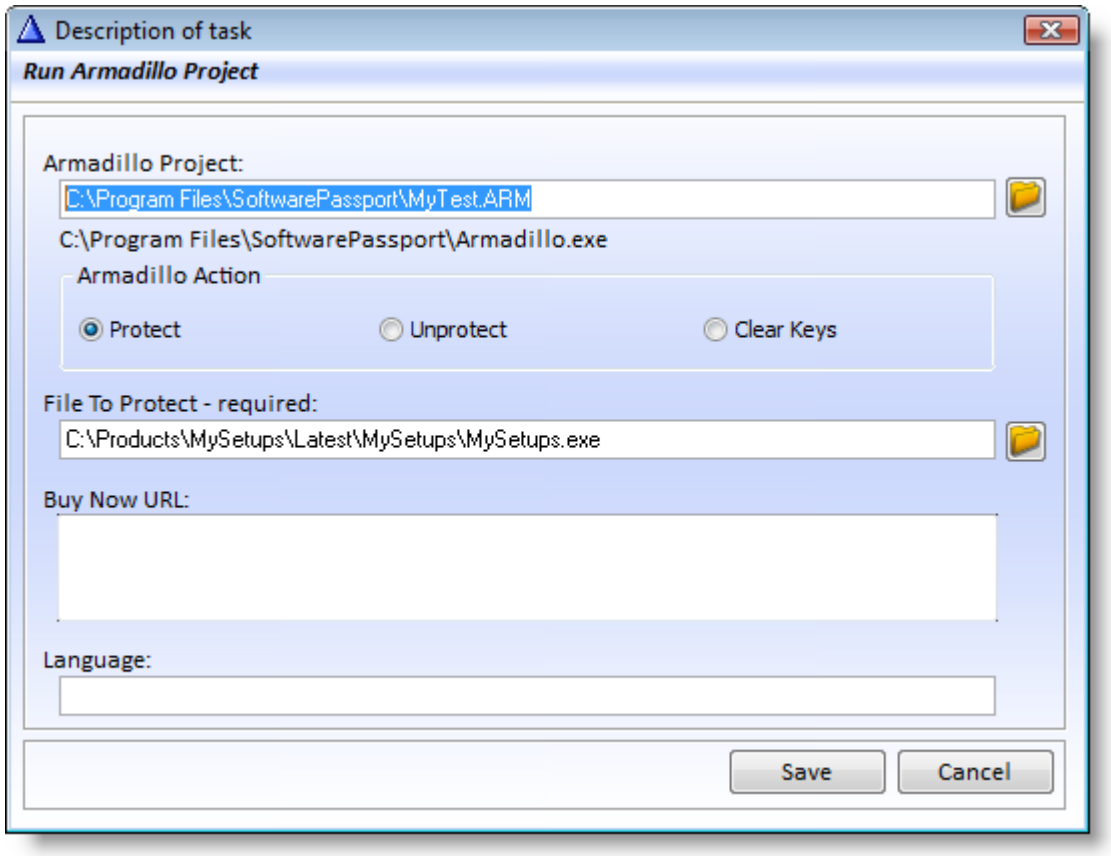
Vista Manifest	If you are creating a manifest for Vista, then check this. This option is checked by default and you should normally just leave it checked. The Vista manifest is backward compatible with Windows XP and does not cause any problems (that we know of) This will result in this section of the manifest being added:
	
	`<trustInfo xmlns="urn:schemas-microsoft-com:asm.v3">`
	`  <security>`
	`  <requestedPrivileges>`
	`  <requestedExecutionLevel`

	level="asInvoker" uiAccess="true"/> </requestedPrivileges> </security> </trustInfo>
Windows 7 Manifest	This enables the "Enable Windows 7 Compatibility" option below. This does not directly affect the generation of the manifest, just enables the Windows 7 related options on the window.
Exclude Dependency	This lets you exclude the dependency on Microsoft Common-Controls and in essence it allows you to create a manifest for UAC without changing the look of your application. No controls will be themed, but the execution level etc. will all be enforced when the program is running under UAC on Vista or Windows 7. Checking "Exclude Dependency" will result in this section of the manifest being excluded: <dependency> <dependentAssembly> <assemblyIdentity type="win32" name="Microsoft.Windows.Common-Controls" version="6.0.0.0" processorArchitecture="X86" publicKeyToken="6595b64144ccf1df" language="*"/> </dependentAssembly> </dependency>
Assembly descr. 	The name of the assembly. This is placed inside the <description></description> tags in the manifest XML. This will result in this section of the manifest being added: <description>T Scan</description>
Execution Level	Select one of the execution levels. For normal programs it should be set to "asInvoker". If you compile your program using "requireAdministrator" your program will always require administration elevation under Windows Vista, Windows Server 2008, Windows 7 and Windows 8 if UAC is turned on. This will result in this section of the manifest being added: <requestedExecutionLevel **level="asInvoker"** uiAccess="true"/> </requestedPrivileges>
uiAccess	This should normally be unchecked. This controls if the program can drive input to the user interface of another program, for examply by posting messages to another program or directly sending keyboard input to it. For more information on this property see http://msdn2.microsoft.com/en-us/library/bb756929.aspx - about half way down the page. This will result in this section of the manifest being added:

	<requestedExecutionLevel level="asInvoker" **uiAccess="true"/>** </requestedPrivileges>
Enable Windows 7...	This enabled Windows 7 compatibility in the manifest, i.e. "urn:schemas-microsoft-com:compatibility.v1" This will result in this section of the manifest being added: <compatibility xmlns="urn:schemas-microsoft-com:compatibility.v1"> <application> </application> </compatibility>
Supported OS	This is only enabled if you are creating a Windows 7 Manifest and have checked "Enable Windows 7 Compatibility" You can check either or both Windows 7 and Vista. For more information about the changes to manifests in Windows 7, please see http://msdn.microsoft.com/en-us/library/dd371711(v=vs.85).aspx The Enable Windows 7 Compatibility and checking both Windows 7 and Vista will result in the red section of manifest being added: <compatibility xmlns="urn:schemas-microsoft-com:compatibility.v1"> <application> **<!--The ID below indicates application support for Windows Vista -->** **<supportedOS Id="{e2011457-1546-43c5-a5fe-008deee3d3f0}"/>** **<!--The ID below indicates application support for Windows 7 -->** **<supportedOS Id="{35138b9a-5d96-4fbd-8e2d-a2440225f93a}"/>** </application> </compatibility>

4.2.9 Protect with Armadillo Build Actions

This action calls Armadillo / Software Passport project.

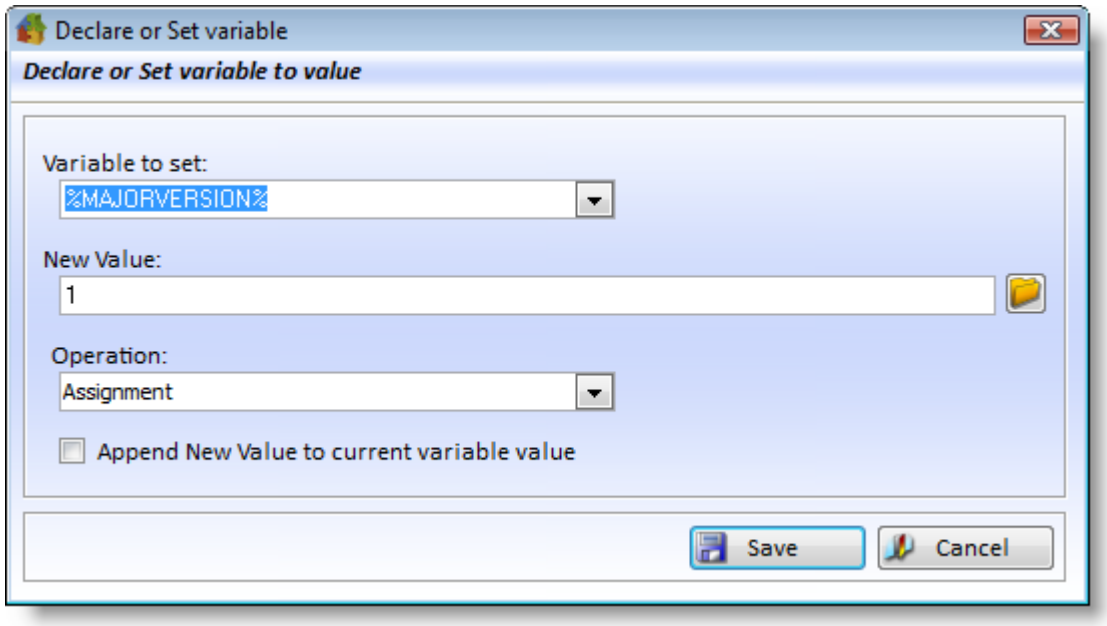


Properties	Explanation
Armadillo Project	Select the Armadillo Project file (*.arm) that contains the protection that you want to apply.
Armadillo Action	You can do 3 things with this action, Protect, which is the default action to take, Unprotect and Clear keys. This is equivalent to the /P /U and /C command line flags respectively.
File to Protect	This is the name of the file to protect. We have found that at least in some versions of Armadillo you **must** provide the file to protect, even though it is specified in the project. This is equivalent to the /file flag.
Buy Now URL	The URL used in the Buy Now button. This is equivalent to the /web flag.
Language	The language used in the protection. This is equivalent to the /language flag.

Please refer to the "Automating Protection" topic in the Armadillo help for more information.

4.2.10 Set Variable — Build Actions

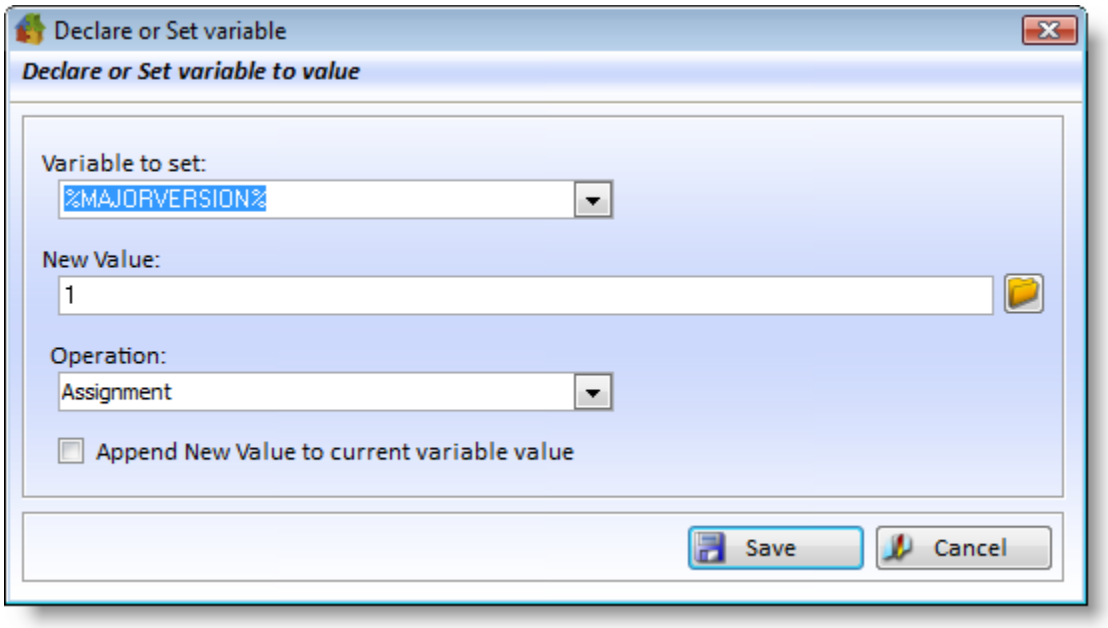
The Set Variable action is designed to be used to assign values to variables during the execution process of the action or project.



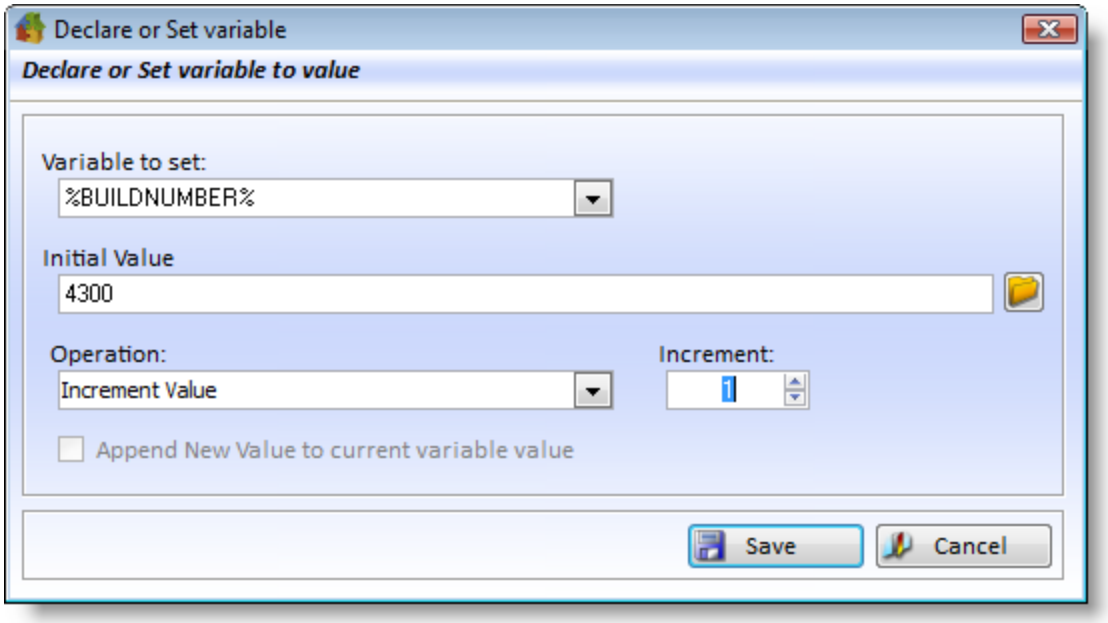
Properties	Explanation
Variable to set	Select or enter the variable name that you want to set. This can be used to declare a variable, or set it.
New Value	Enter the new value for the variable. Click the button on the right to show a window to select various data elements, such as files, folders, etc. This entry can include variables.
Operation	Here you can select the operation to perform. The available operations are: Assignment, Increment Value, Decrement Value, Convert to Uppercase, Convert to Lowercase and Evaluate Expression. See below for more detailed information.
Append New...	This can be used to append the results of the New Value assignment to the current value of the variable.

Below is a list of the various operations that apply to the Set Variable action. We will be documenting the functions that are available for the Evaluate Expression as we add more to the list. Please check our Function Reference 56 for more information.

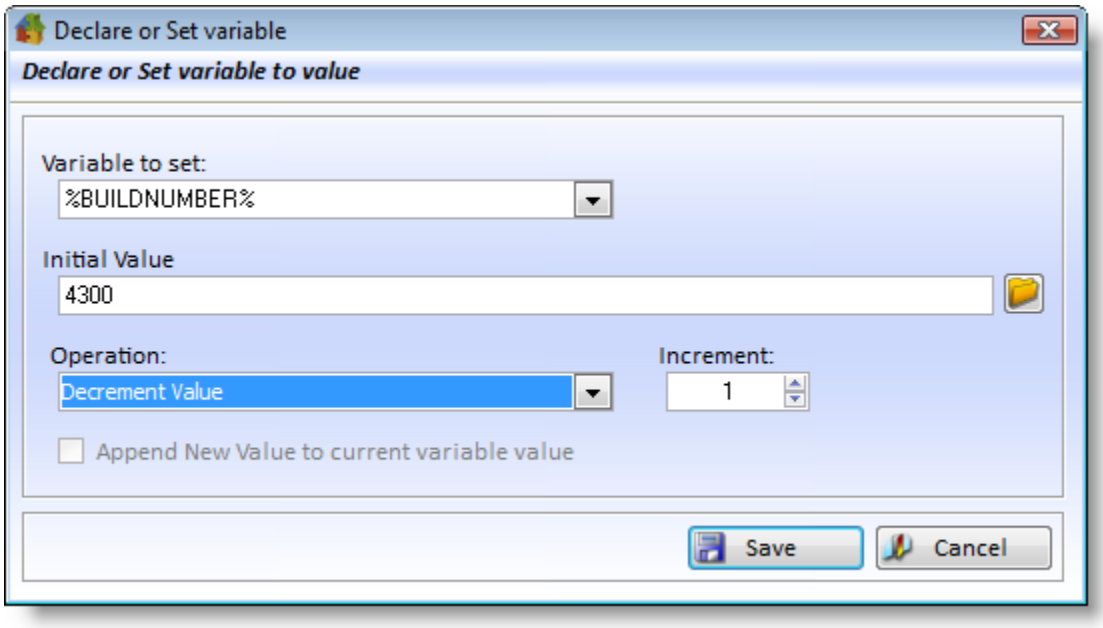
Assignment This is a direct assignment to the variable. If "Append New Value..." is unchecked then whatever the variable contained before will be overwritten with the contents of the New Value. If the "Append New Value..." is checked this operation will append the New Value to the existing value.



Increment Value This increments a numeric value by the Increment value. If a value is specified in the Initial Value, then the variable will be assign the Initial Value plus the increment value the first time it executes. For example if the Initial Value is set to 4000 and the Increment Value is set to 1, then the first time the action is executed the variable will contain 4001. The next time it will contain 4002, then 4003 and so on.

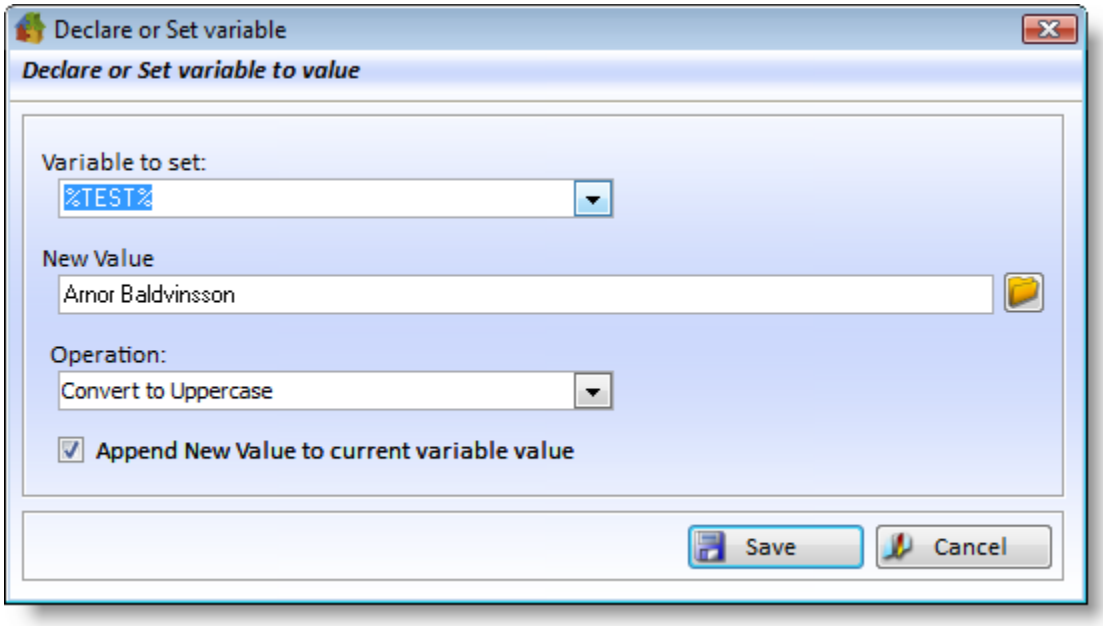


Decrement Value This decrements a numeric value by the Increment value. If a value is specified in the Initial Value, then the variable will be assign the Initial Value minus the increment value the first time it executes. For example if the Initial Value is set to 4000 and the Increment Value is set to 1, then the first time the action is executed the variable will contain 3999. The next time it will contain 3998, then 3997 and so on.



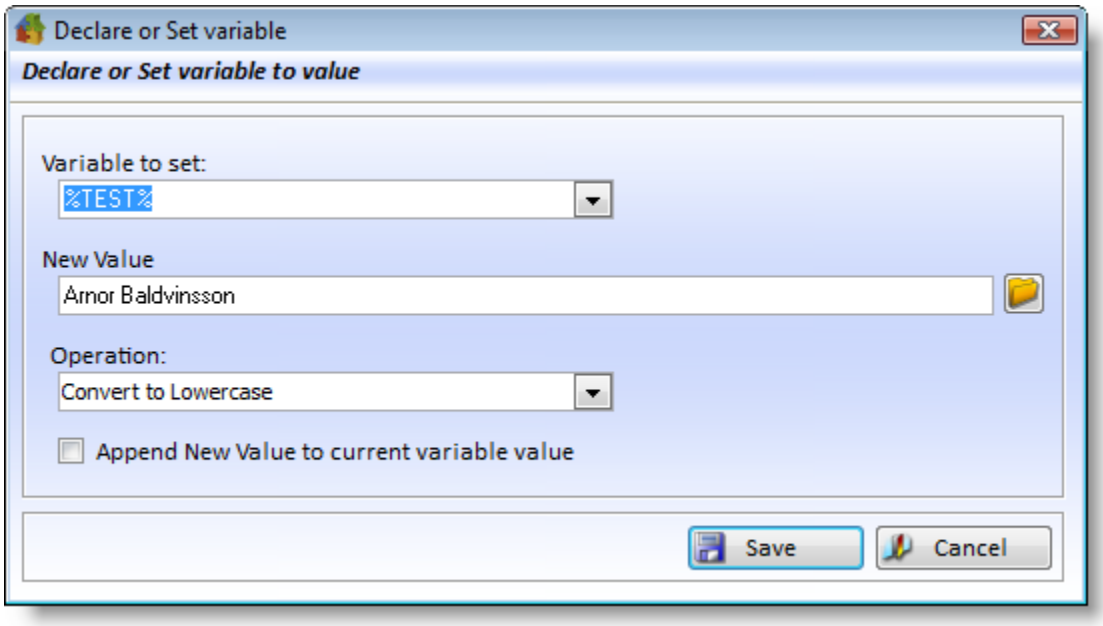
Convert to Uppercase

This operation takes the contents of the Variable and converts it to uppercase. If the New Value is set, then that value is converted to uppercase. If the "Append new Value..." is checked, then the New Value is appended to it, otherwise the New Value is assigned to the variable.



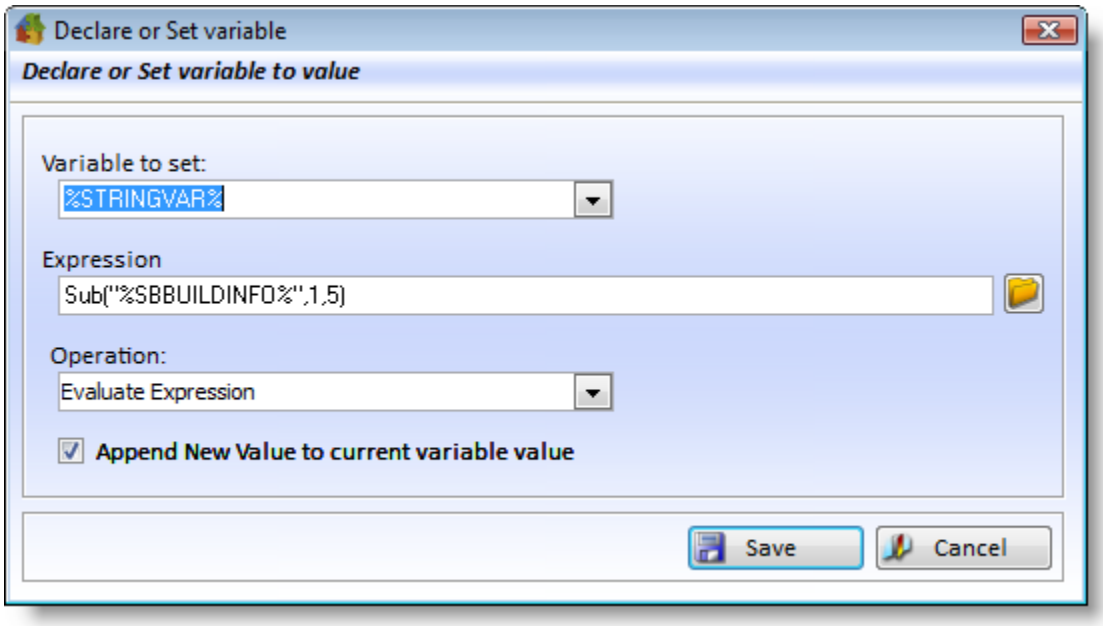
Convert to Lowercase

This operation takes the contents of the Variable and converts it to lowercase. If the New Value is set, then that value is converted to lowercase. If the "Append new Value..." is checked, then the New Value is appended to it, otherwise the New Value is assigned to the variable.

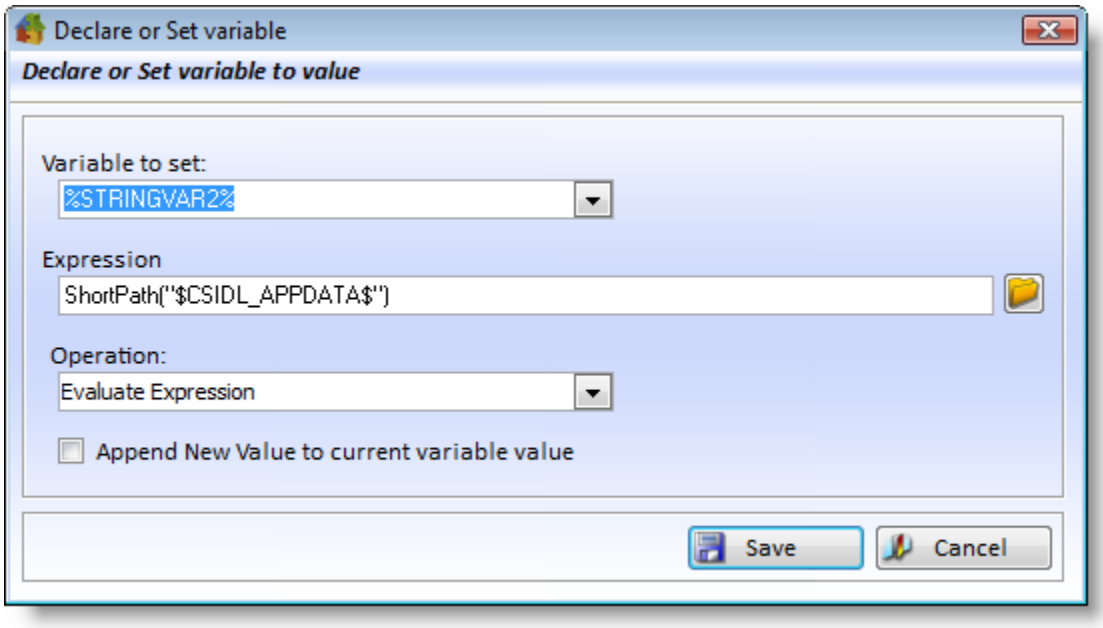


Evaluate Expression

The Evaluate Expression operation is extremely powerful. It can evaluate simple arithmetic, such as 1+2/5*10, but it can also work with a variety of internal functions. In the example below the %STRINGVAR% will be assigned the first 5 characters of the %SBBUILDINFO% variable. Please note that all string type variables MUST have double quotes around them where they are used in functions. In the example below this will be expanded to something like Sub('Something or other',1,5) before it is evaluated, and then the %STRINGVAR% will be assigned the first 5 characters, i.e. 'Somet'



The example below returns the ShortPath of the $CSIDL_APPDATA$ system variable.



We will be documenting the functions that are available for the Evaluate Expression as we add more to the list. Please check our Function Reference 56 for more information.

You can create some very powerful things with the Set Variable action. Such as format data to your liking, for example by using Format(Today(),@d17) to format a date to the standard short windows format. Or retrieve part of a string by using Sub() or Instring().

You can also use Build Automator variables in your expressions. The only requirement is that **if the variables contain alphanumeric data the variables MUST be enclosed in Double Quotes**, such as "$CSIDL_APPDATA$" or "%THISVALUE%" Numeric variables, or string variables that contain numeric values only do not need to be enclosed in double quotes if they are used as numeric parameters.

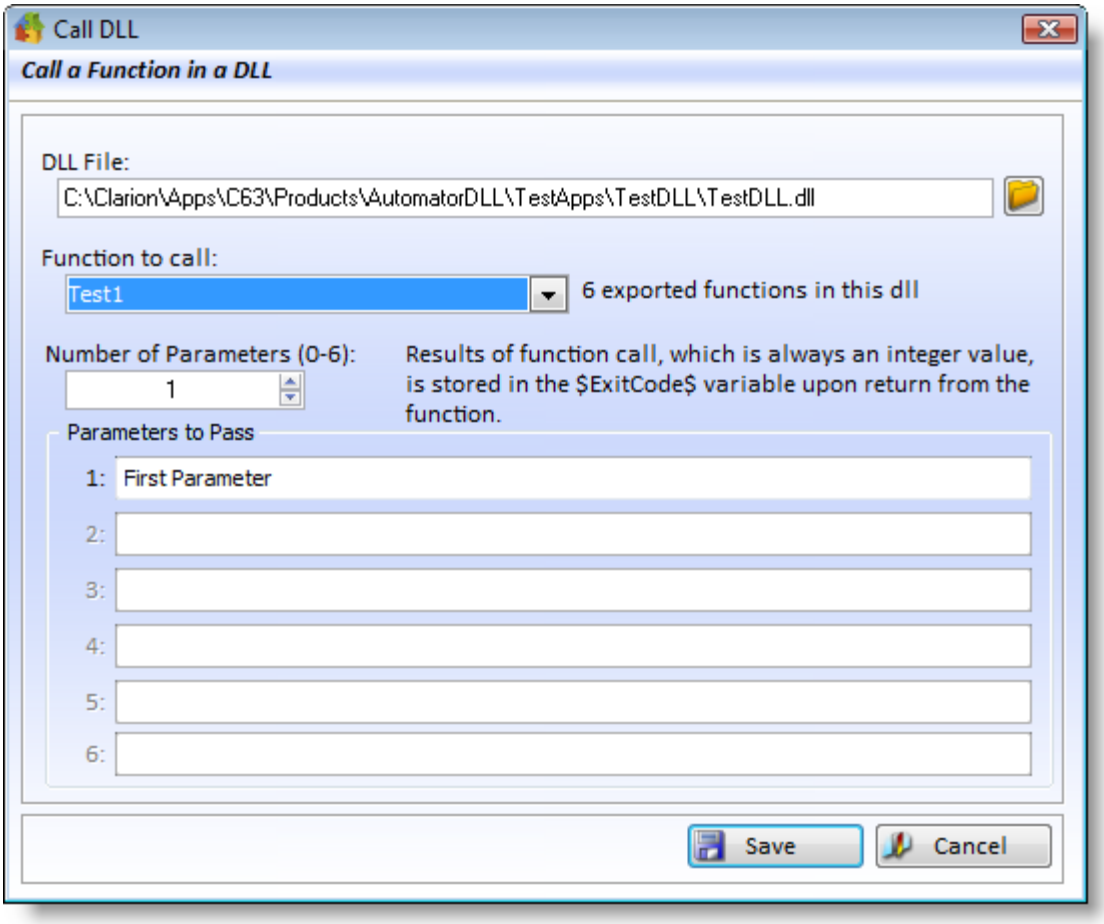
4.3 Execute Actions

4.3.1 Call DLL Execute Actions

The Call DLL action is a very powerful extension to the Build Automator. You can call any function in any dll as long as it follows certain rules about parameters and return values.

The functions being called with this function must be prototyped with CONST *CSTRING parameters in Clarion or char* parameters in C. The function can take no parameter or it can take anywhere from 1 to 6 parameters. The function must return an integer value. In Clarion it must be declared with the PASCAL attribute and in C it must be declared with the __stdcall atribute.

Please note that if the number of parameters as entered into the window, see below, is not correct the action will GPF immediately when it is called to execute. **We suggest that you make sure that your project file is saved before you execute this function!**



Properties	Explanation
DLL File	Select the DLL file to call. The DLL does not to be in the search path and the Build Automator will load the DLL from the path that is specified here without attempting to

	locate it on the path first.
Function to call	Select the function that you want to call from the dropdown. The entries in the dropdown are reset every time you select a new DLL to call.
Number of Parameters	The number of *CSTRING/char* parameters that the function accepts. Make sure that this matches the prototype of the function you selected in the "Function to call" drop down list.
Parameters 	Enter the appropriate parameters to pass to the function. Normal strings can be typed in without single or double quotes, for example in the screenshot above the first parameter will be passed as 'First Paramter' You can use string literals or you can use variables or a mix of both, for example: Today is $ToDay$ and the time is Now

Below is a very simple DLL example compiled in Clarion 6.3.

TestDLL.CLW

```
 Program
 Map
Test1                  FUNCTION(CONST *CString pP1),Long,PASCAL,Name('Test1')
Test2                  FUNCTION(CONST *CString pP1, |
                                CONST *CString pP2),Long,PASCAL,Name('Test2')
Test3                  FUNCTION(CONST *CString pP1, |
                                CONST *CString pP2, |
                                CONST *CString pP3),Long,PASCAL,Name('Test3')
Test4                  FUNCTION(CONST *CString pP1, |
                                CONST *CString pP2, |
                                CONST *CString pP3, |
                                CONST *CString pP4),Long,PASCAL,Name('Test4')
Test5                  FUNCTION(CONST *CString pP1, |
                                CONST *CString pP2, |
                                CONST *CString pP3, |
                                CONST *CString pP4, |
                                CONST *CString pP5),Long,PASCAL,Name('Test5')
Test6                  FUNCTION(CONST *CString pP1, |
                                CONST *CString pP2, |
                                CONST *CString pP3, |
                                CONST *CString pP4, |
                                CONST *CString pP5, |
                                CONST *CString pP6),Long,PASCAL,Name('Test6')
 End

 Code


Test1                PROCEDURE  (CONST *CString pP1)!!,Long,PASCAL,Name('Test1')
 Code
 Message('1:  ' & pP1,'Test with 1 parameter',ICON:Exclamation)
 Return(1)

Test2                  FUNCTION(CONST *CString pP1, CONST *CString pP2)!!,Long,PASCAL,Name('Tes
 Code
 Message('1:  ' & pP1 & '|2:  ' & pP2,'Test with 2 parameters',ICON:Exclamation)
 Return(2)

Test3                  FUNCTION(CONST *CString pP1, CONST *CString pP2, CONST *CString pP3)!!,L
 Code
 Message('1:  ' & pP1 & '|2:  ' & pP2 & '|3:  ' & pP3,'Test with 3 parameters',ICON:Exclamation
 Return(3)
```

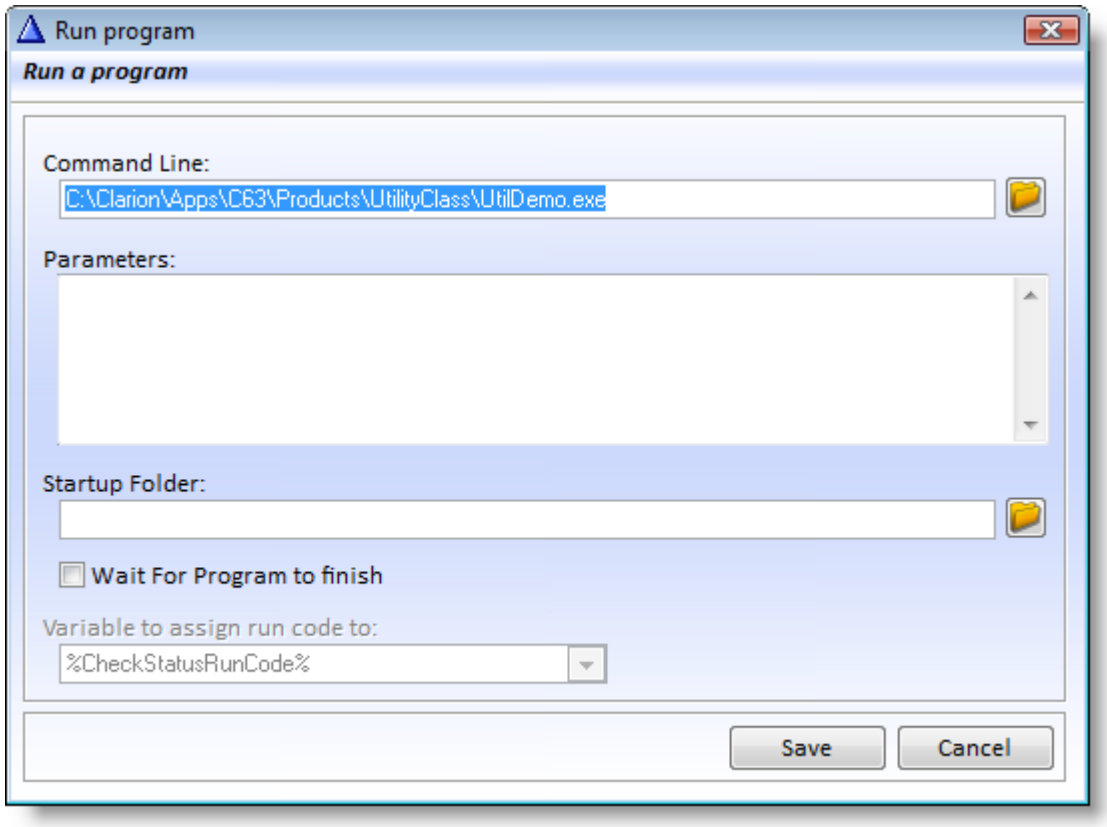
```
Test4                FUNCTION(CONST *CString pP1, CONST *CString pP2, CONST *CString pP3, CON
 Code
 Message('1:  ' & pP1 & '|2:  ' & pP2 & '|3:  ' & pP3 & '|4:  ' & pP4,'Test with 4 parameters',
 Return(4)

Test5                FUNCTION(CONST *CString pP1, CONST *CString pP2, CONST *CString pP3, CON
 Code
 Message('1:  ' & pP1 & '|2:  ' & pP2 & '|3:  ' & pP3 & '|4:  ' & pP4 & '|5:  ' & pP5,'Test wit
 Return(5)

Test6                FUNCTION(CONST *CString pP1, CONST *CString pP2, CONST *CString pP3, CON
 Code
 Message('1:  ' & pP1 & '|2:  ' & pP2 & '|3:  ' & pP3 & '|4:  ' & pP4 & '|5:  ' & pP5 & '|6:  '
 Return(6)
```

TestDLL.PRJ

```
-- Test DLL
#noedit
#system win32
#model clarion dll
#pragma define(maincode=>off)
#set RELEASE = on
#pragma debug(vid=>off)
#pragma optimize(cpu=>386)
#compile "TestDLL.clw"
#link "TestDLL.dll"
```

4.3.2 Run Program — Execute Actions

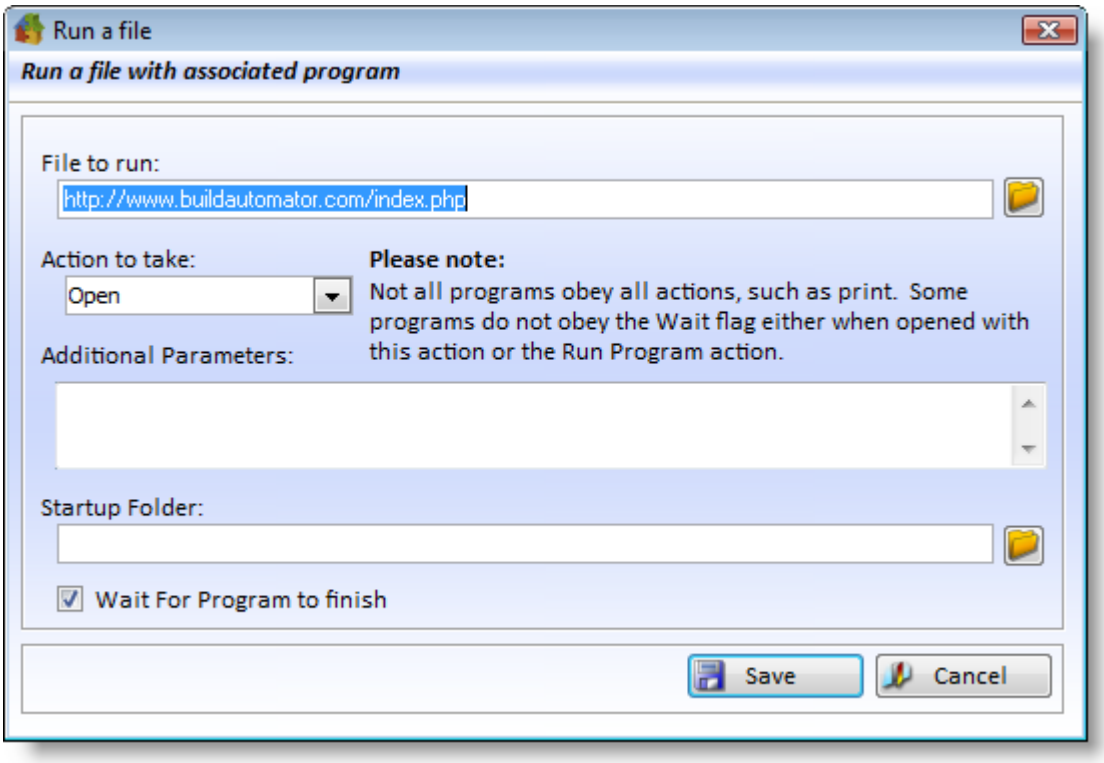
This action can be used to run any kind of program on your machine. This action is very versatile.



Properties	Explanation
Command Line	The program to execute. This can be an EXE, COM or BAT file or any other executable file. **Please note that this should not include any command line parameters or flags**.
Parameters	Any parameter(s) that you need to pass on to the program.
Startup folder	You can optionally specify a startup folder if necessary. If not startup folder is specified it will default to the folder where the program is, i.e. the Command Line folder.
Wait for Program...	If this is checked the Build Automator™ will be suspended until the program that you run terminates. You will not be able to access or do anything in the Build Automator™ until the program that you run closes or you close it manually.
(Variable for run code)	*Allows you to select a variable to put the run code from the executed program into.* ***This is temporary only*** *as we are already putting this into a system variable called $ExitCode$* ***We recommend that you do not use this*** *drop down for that reason.*

4.3.3 Run File

This action is very similar to the Run Program 102 action, except you select a file to run, not a program. This is the same as double clicking on a filename in Windows Explorer.



Properties	Explanation
File to run	Select the file to run.
Action to take	Normally you would use Open, but you can also use other verbs. RunAs can be used to run programs under Windows Vista and force elevation when using UAC. Explore can be used on a folder path and will open the folder in Windows Explorer. Properties will open the "File Properties" window for the selected file.
Additional Param.	Add any additional parameters. Normally this should be empty.
Startup Folder	Normally not needed, but can be used to specify the startup folder.
Wait for Program...	Check this to force the Build Automator to wait for the called program to finish. Note that some programs do not obey this for security reasons, including Windows Explorer and Internet Explorer.

4.4 File Actions

4.4.1 Copy Files - Multiple File Actions

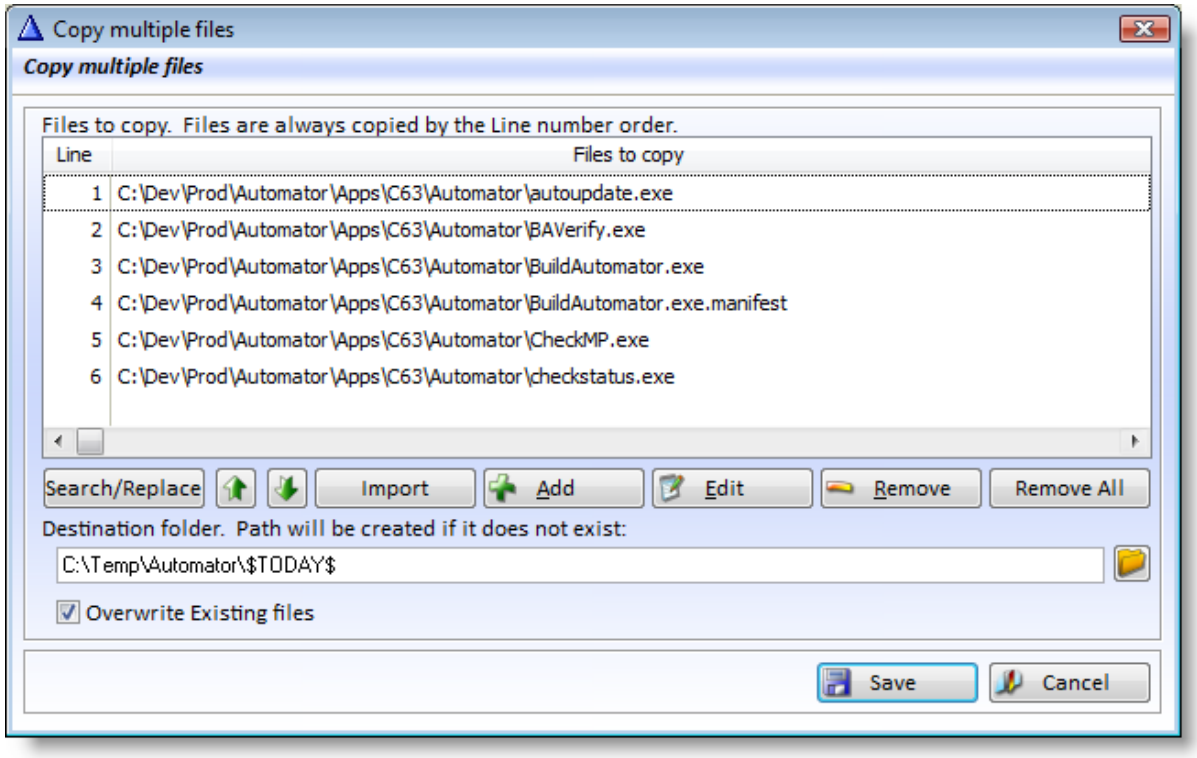
This action copies a list of files to a selected destination. You can add files to the list or remove files from it and also move individual entries up or down. Please note that there is only one destination folder so if you select files from multiple source folders you may end up with duplicates.

For example:

Source files:
C:\Temp\MyFile.txt
C:\Programs\MyFile.txt

Destination:
C:\Backup

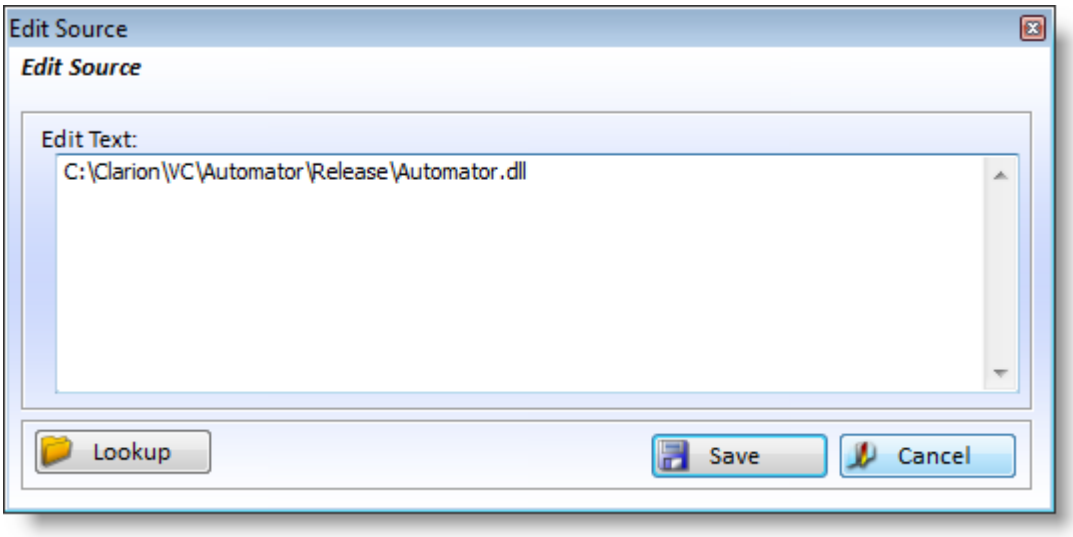
Both of the MyFile.txt will end up as C:\Backup\MyFile.txt, the second copy overwriting the first one unless you uncheck "Overwrite Existing files".



Properties	Explanation
Files to copy	List of source files to copy to the destination. You can use the Add and Remove buttons to add or remove entries in the list and use the up and down arrow buttons to move the files. You can use the + key or the Alt-Down arrow keys to move entries down. You can use the - key or the Alt-Up arrow keys to move entries up.

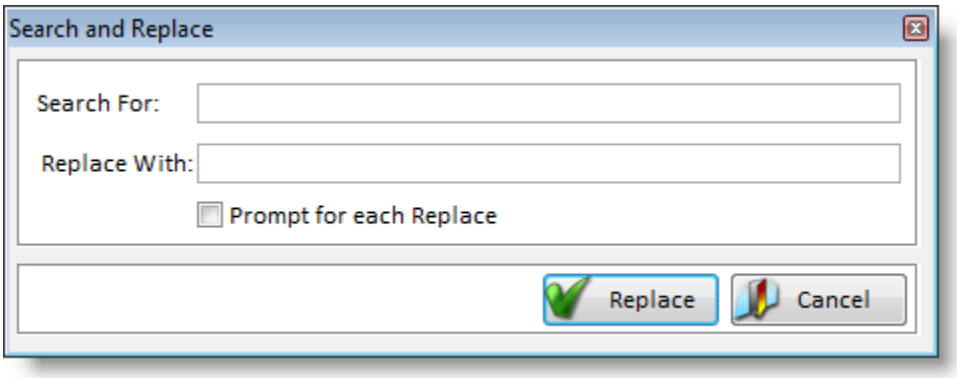
Destination	Single folder destination. See note above about duplicate files.
Overwrite Existing...	If this is checked existing files will be overwritten if files with the same name are being copied into the destination folder.

Use the "Add" button to add one or more file to the list or use the Insert key on the keyboard. To remove a file from the list, use the "Remove" button or use the Delete key on the keyboard. To move the files up and down, use the up/down arrow buttons or use Alt-Up arrow or Alt-Down arrow buttons on the keyboard. To edit the currently selected entry, click on the Edit button or use the Enter key on the keyboard or double click with the mouse. That will bring up an edit window:



To select a new file, click on the "Lookup" button to bring up a file selection dialog.

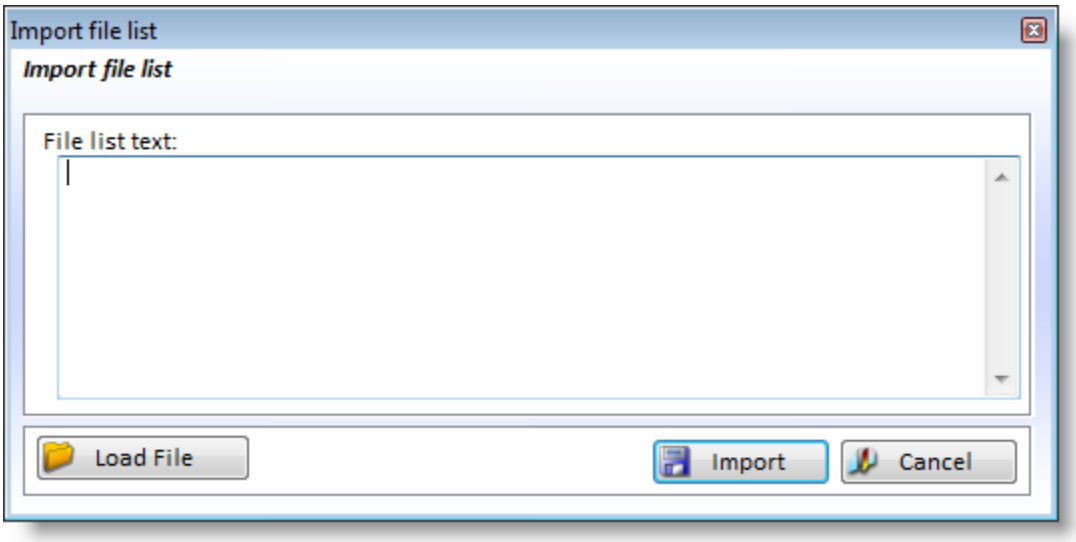
You can do a quick search and replace operation on the filenames in the list by using the "Search/ Replace" button. It will bring up a simple window where you can enter the text that you want to search for and replace with.



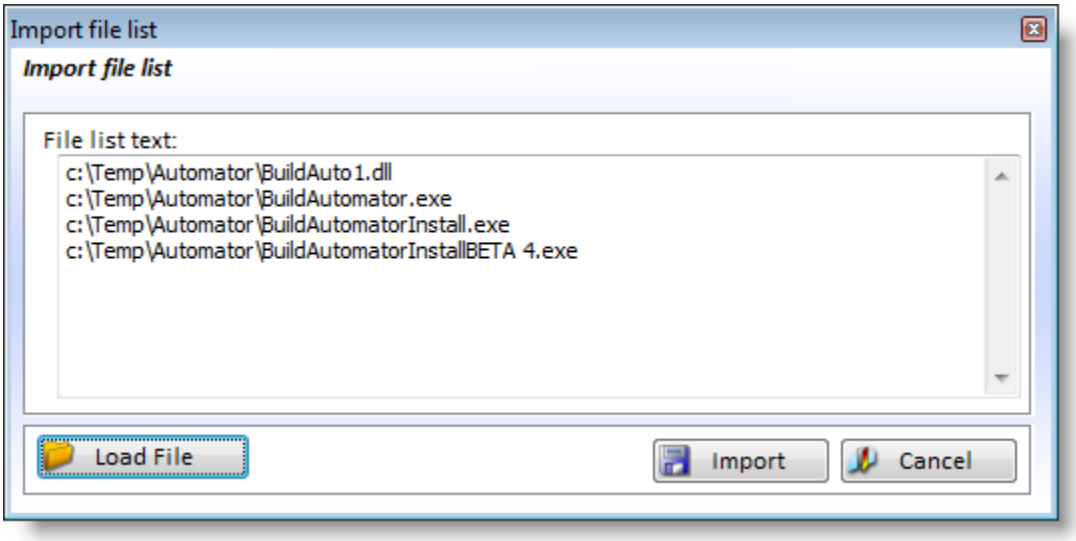
Note that you can of course use variables in both entries. This is an excellent way to set the source of all files to be the same based on a preset variable.

With the Import button you can import a list of files. It opens a window where you can create the list, or past it from the clipboard, or you can select a text file with

a list of filenames.



Click on the "Load File" button to select a text file with a list of filenames and the file with the list will be importe dinto the "File list text:" text box.



This is how it looks like after selecting a text file called "files.txt" that contained this text:

c:\Temp\Automator\BuildAuto1.dll
c:\Temp\Automator\BuildAutomator.exe
c:\Temp\Automator\BuildAutomatorInstall.exe
c:\Temp\Automator\BuildAutomatorInstallBETA 4.exe

This makes it easy to construct this list from external sources if needed.

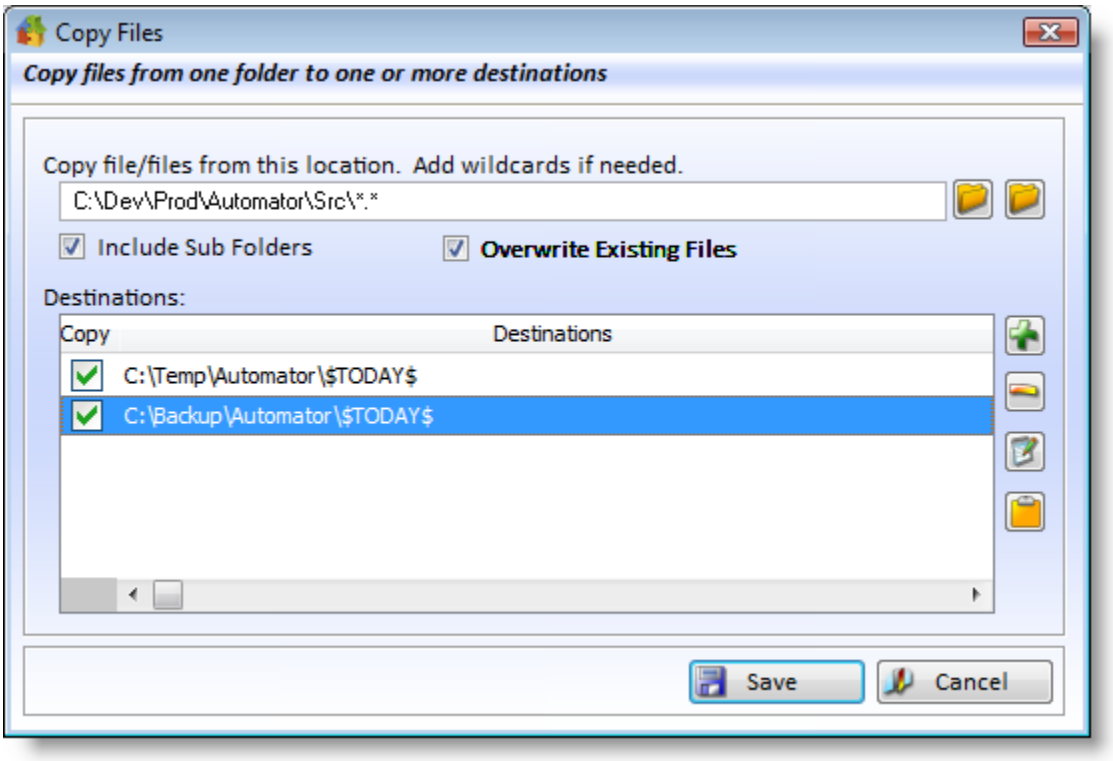
The "Files to copy" list can never contain duplicates. When a file is added Build Automator checks first if it already exists in the list. The file is only added if it doesn't exist already. This also applies when using the Import button.

The "Remove All" button will remove all files from the "File to Copy" list, but Build Automator will

prompt you to confirm before it removes any files from the list.

4.4.2 Copy Files - Simple File Actions

This action can copy a single source using wildcards to one or more destinations. You can choose to include sub folders. Only files that match the wildcard will be copied from the parent folder as well as the sub folders. The sub folder structure will be created in all the specified destinations.



Properties	Explanation
Copy file/files	Select a file to copy. You can enter a path name or file name into the entry field, including wildcards but you can also use the buttons on the right of the field to select a folder or a file.
	Button on the left opens a folder dialog and the one on the right opens a file dialog. Using the folder select button on the left will result in the path being placed into the "Copy file/files" entry with a *.* extension added. Using the file select button on the right will result in a filename being placed in the entry.
Include sub folders	Check this to include all sub folders from the folder where the selected file is located.
Overwrite Existing...	If this is checked existing files will be overwritten if files with the same name are being copied into the destination folder.
Destinations	List of one or more destinations for the file(s) to be copied to. You can add or remove destinations by using the [+] and [-] buttons to the right of the list.
	With this button you can paste a list of paths into the destination. Note that this does not check the contents of the clipboard for valid paths. If the listbox is selected, you

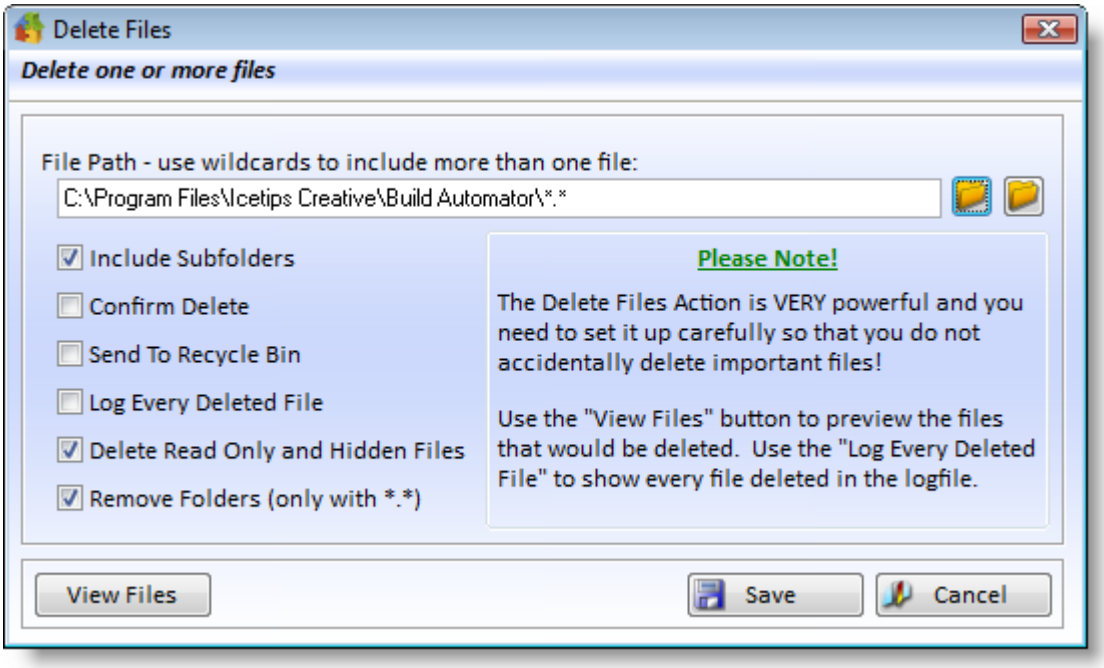
	can also use Ctrl-V to paste the list.

4.4.3 Delete Files File Actions

The Delete Files action is a very powerful action.

Warning: You need to make sure that you understand it's power before you execute it and make sure that you do not accidentally delete files that you don't want to delete! We do not accept responsibility for any files that may be deleted - accidentally or otherwise.

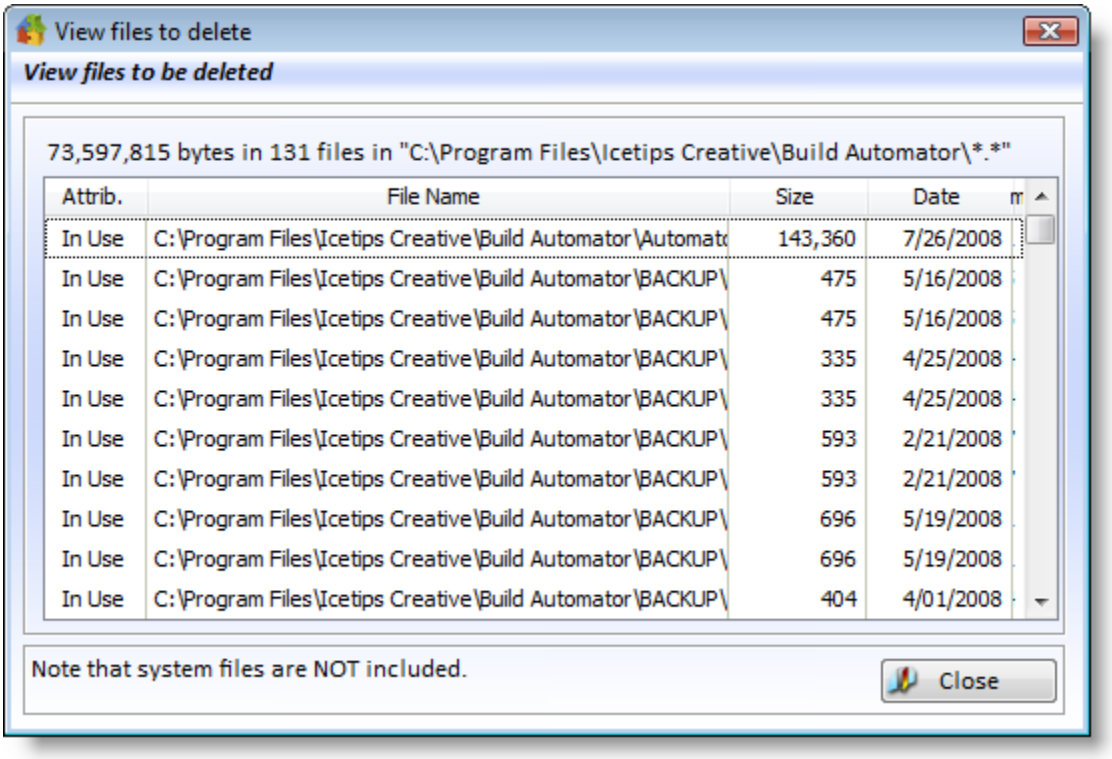
That said, let's take a look at what this can do!



Properties	Explanation
File path	The file or path that you want to delete. The left button on the right will select a folder to use and then automatically append the *.* wildcard to the file path. The right button selects a single file. You can edit the File path once you have selected the correct path to anything like like as long as it follows the standard wildcard rules for the operating system. For example you could use C:\Clarion\Apps\C63\Junk*.clw to clean out all .clw files from your C:\Clarion\Apps\C63\Junk\ path. If you check the Include subfolders, the delete action would recursively delete from all subfolders.
Include Subfolders	When this is checked the delete process will include ALL subfolders that may be under the main folder.
Confirm Delete	This will show a message asking for confirmation. The message will show how many files and folders are involved.
Send to Recycle Bin	You can optionally not send the files to the Recycle Bin. It is considerably faster than using the Recycle Bin, but then you won't be able to recover the files in case something went wrong! We suggest that you leave this checked while you are testing

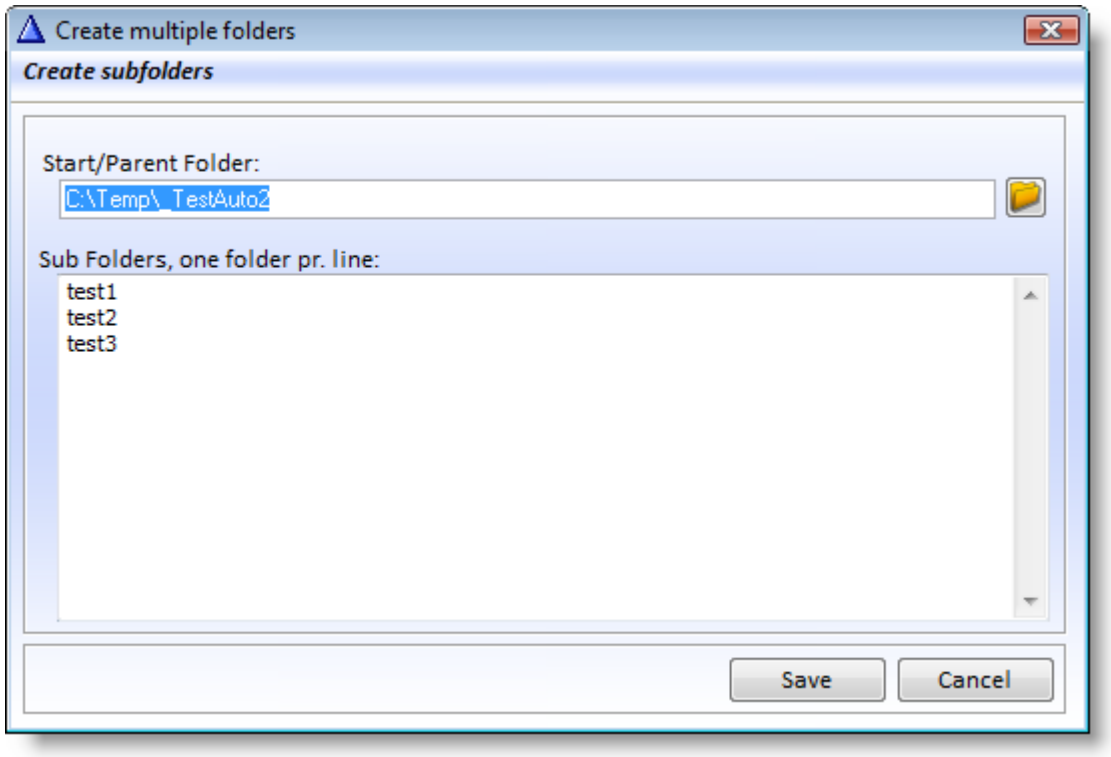
	the action to make sure it is deleting the correct files.
Log Every Deleted File	When this is checked the Build Automator will add an entry for every file and folder that was deleted. If it is unchecked then the Build Automator will only log files that could not be deleted.
Delete Read Only...	This allows you to delete Read-Only and Hidden files. To avoid error messages during unattended builds, the Build Automator first changes the file attributes to Archive. If any file can't be changed the Build Automator will not attempt to delete it. The Build Automator never includes System files in the delete process.
Remove Folders	This option is only available if you use the *.* wildcard. It will go through all the subfolders and remove them if they are empty. Note that this action will **not** remove the parent folder, i.e. the folder selected in the "File path"

The "View Files" button shows you a window with all the files that fit the selection criteria on the main window. Use it to view and check that you are only including the files that you expected to be included. Note that the Attrib. column will show "r" or "h" for Read-Only and "Hidden" files or "In Use" if the file is in use by another program.



4.4.4 Create Folders — File Actions

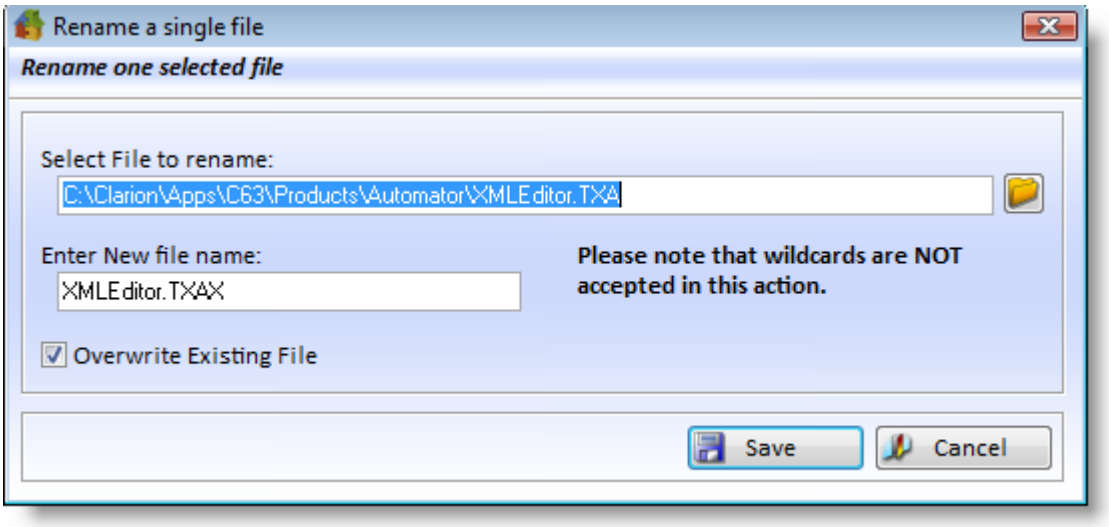
This action creates one or more folders in a given start or parent folder.



Properties	Explanation
Start/Parent	Select a start of parent folder. All the folders will be created as sub folder in this folder.
Sub folders	Enter the folder names you want to create, one folder per line. You can enter whole sub paths, such as test1\testing\testing.

4.4.5 Rename a File File Actions

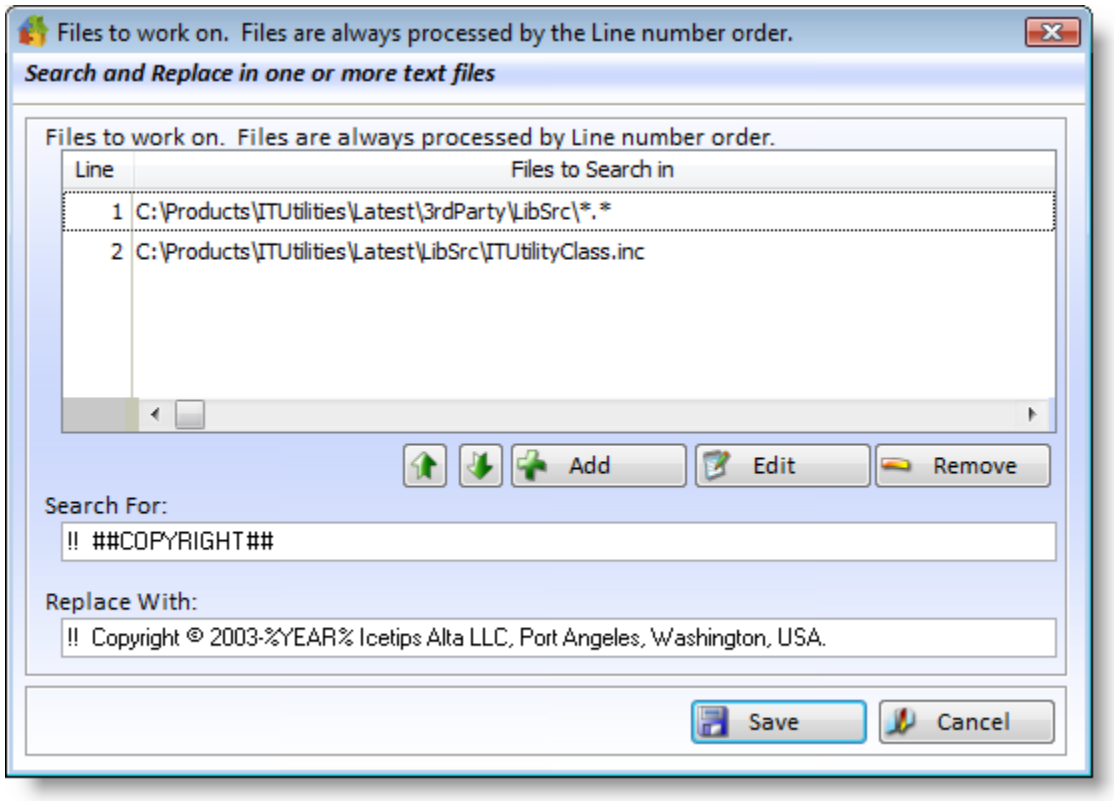
This action renames a single file to a new name in that same folder. This action does not support wildcards as of release 1.10.100.



Properties	Explanation
Select File...	Select the file that you want to rename.
Enter New file name	Enter the new filename. The file is renamed within the same folder as the source file.
Overwrite Existing File	Check this if you want the destination file to be overwritten. If this is not checked and the file exists the rename fails.

4.4.6 Search and Replace File Actions

This action searches and replaces text in selected files.



Properties	Explanation
Files to work on...	Select the file(s) that you want to do the search and replace in. Note that you can add multiple files and you can use wildcards. You can also use variables in the filenames.
Search For	Enter the text you want to search for in the files specified in "Files to work on". You can use also variables here.
Replace With	Enter the text you want to replace the searched text with. You can use also variables here.

You can use this to replace tokens in source files, even in text documentation such as plain text files, RTF files, html files etc.

Example of header in a source file with tokens ready to be replaced:

```
!!  ------------------------------------------------------------------------------
!!  Icetips Utilities Source file
!!  ##COPYRIGHT##
!!  This file can not be distributed in any way by anyone except Icetips Alta LLC
!!  ##VERSIONINFO##
!!  ------------------------------------------------------------------------------
```

With the settings in the screenshot, this would be replaced with something like this:

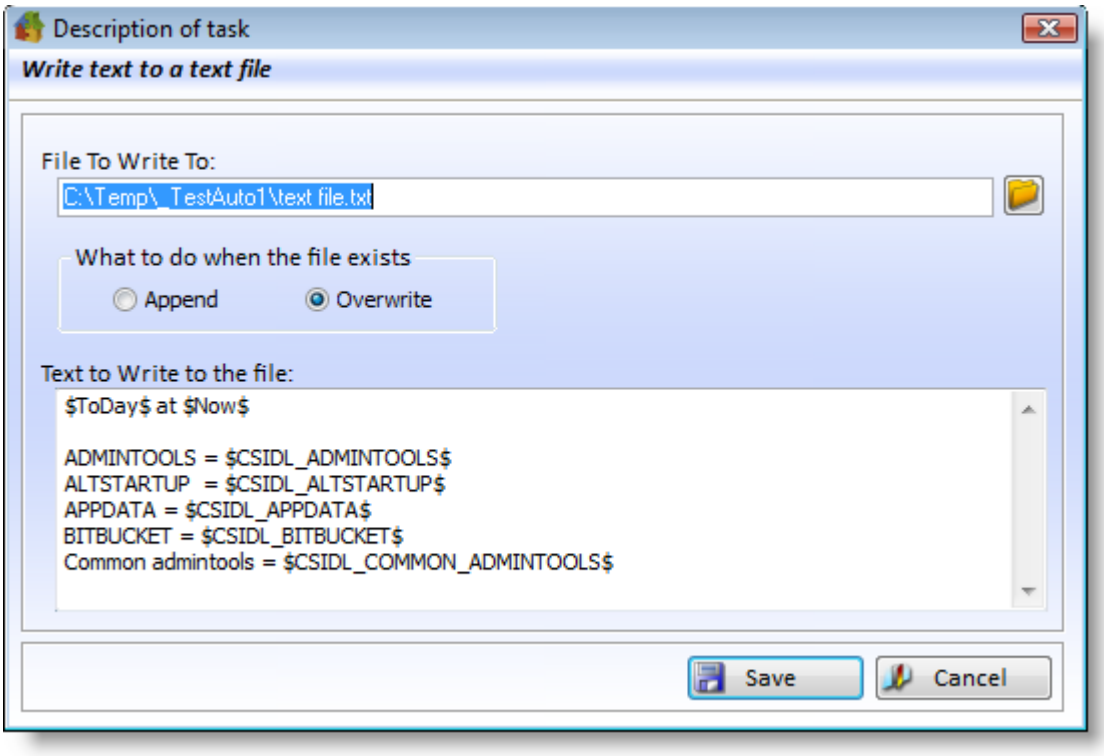
```
!!  ------------------------------------------------------------------------------
!!  Icetips Utilities Source file
!!  Copyright © 2003-2015 Icetips Alta LLC, Port Angles, Washington, USA.
```

!! This file can not be distributed in any way by anyone except Icetips Alta LLC
!! Version: 1.1.2351, April 15, 2015
!! ----------

4.4.7 Write text to file — File Actions

This action writes the text entered to a specified text file. This can be very powerful when combined with compilers to change project settings before a compile, then change them again before another compile, for example to compile different variations of the same program.

Version 2.0: December 7, 2013. You can now use Include to include files inside the text.



Properties	Explanation
File to write to	The name of the file to write the text to. If the path does not exist when the action is run, the folder structure is created automatically.
What to do...	This action can either overwrite the file if it exists, or it can append to it.
Text to Write	Enter the text to write. The text is currently limited to 10K, but if you use variables in the text, please allow for room to expand them.

The above settings produce a text file like this (this will not format properly in the PDF manual because the lines are too long):

```
2015-09-15 at 15:19:32
```

```
ADMINTOOLS = C:\Users\User Name\AppData\Roaming\Microsoft\Windows\Start Menu
\Programs\Administrative Tools
ALTSTARTUP  = C:\Users\User Name\AppData\Roaming\Microsoft\Windows\Start Menu
\Programs\Startup
APPDATA = C:\Users\User Name\AppData\Roaming
BITBUCKET =
Common admintools = C:\ProgramData\Microsoft\Windows\Start Menu\Programs
\Administrative Tools
```

INCLUDE

By using Include() you can now include other files in the text. This allows you to build files, such as code files on the fly before compiling. **Note that the {#INCLUDE( )} statement MUST be in upper case**. The filename must be enclosed in double quotes. Single quotes will not work. The whole statement is enclosed in curly brackets. Note that no spaces are allowed in the {#INCLUDE()} statement except in the filename. Here is an example of a text that I use to construct Clarion template that exports classes and methods:

```
#!!
------------------------------------------------------------------------------
---
#!!  Icetips Outlookbar Template file
#!!  ##COPYRIGHT##
#!!  This file can not be distributed in any way by anyone except Icetips Alta LLC
#!!  ##VERSIONINFO##
#!!
------------------------------------------------------------------------------
---
#!!
#!----------------------------------------------------------------------------
------
#PREPARE
  #CALL(%ReadABCFiles(ABC))
  #CALL(%SetClassDefaults(ABC), 'ITAuto', 'ITAuto', %ClassName)
  #SET(%ITATemplateName,'Icetips Automator Classes Global')
#ENDPREPARE
{#INCLUDE("%EXPORT_FILE%")}
```

%EXPORT_FILE% is set earlier in the script to point to an automatically generated .exp file that is then included in this .TPW file before it is included in the distributed product.

Example use in Clarion:

Let's say that you need to compile 3 different models of the same application. You can do that easily by using a project #define that you then use in COMPILE and OMIT statements.

In your code you have these 3 equates:

```
MODEL:Standard
MODEL:Professional
MODEL:TeamProject
```

You decide to use _Model_ pragma define in your project and it could be set to 0, 1 or 2 for the 3 different MODEL: values in that order. In your code you can now have something like this, for example in the "After Program Code" global code embed:

```
COMPILE('***Standard***',_Model_=0)
Glo:Model = MODEL:Standard
 ***Standard***
COMPILE('***Professional***',_Model_=1)
Glo:Model = MODEL:Professional
```

```
 ***Professional***
COMPILE('***TeampProject***',_Model_=2)
Glo:Model = MODEL:Standard
 ***TeampProject***
```

You create a file called ModelProject.prj You don't have to put anything into it, or you could put:

```
#pragma define(_Model_=>0)
```

into it to test with. Save the file and go back to your Clarion application and add the ModelProject.prj to your application project under "Projects to include". Now you have this all set up to make the Build Automator do the rest for you.

You can now combine this in a single script in the Build Automator:

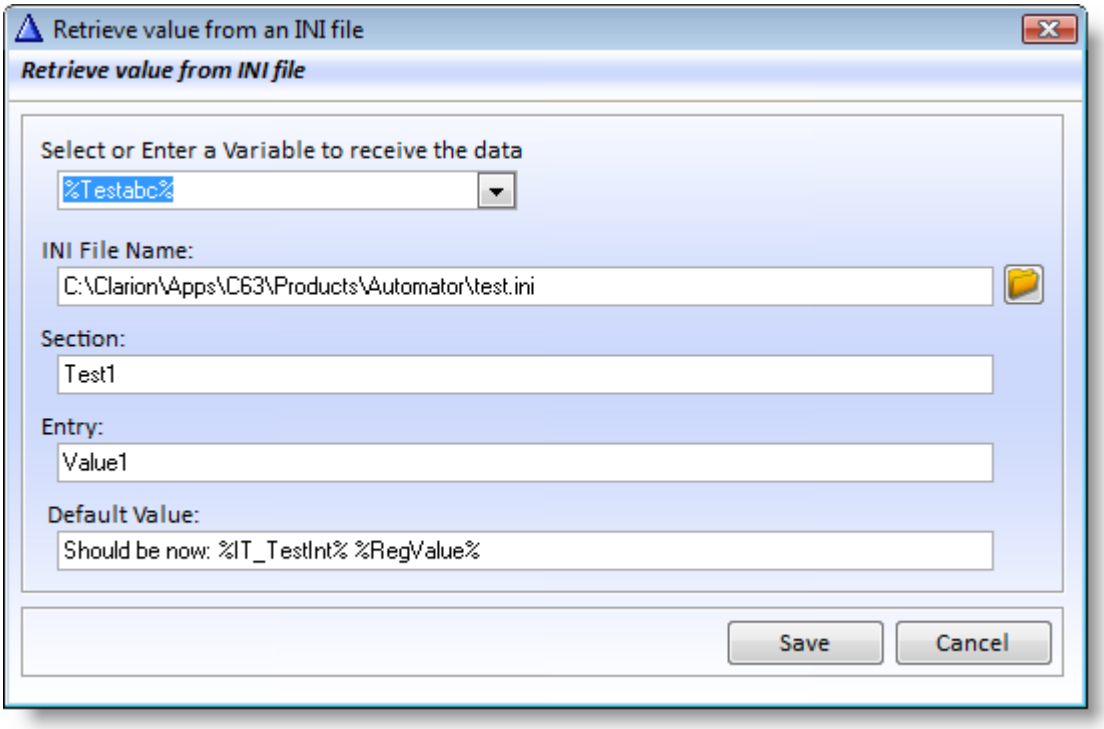
```
// Create the Standard executable
WriteTextToFile:  #pragma define(_Model_=>0)
Compile Clarion:  MyApp.app
Rename File:      MyApp.exe -> MyAppStandard.exe

// Create the Professional executable
WriteTextToFile:  #pragma define(_Model_=>1)
Compile Clarion:  MyApp.app
Rename File:      MyApp.exe -> MyAppProfessional.exe

// Create the TeamProject executable
WriteTextToFile:  #pragma define(_Model_=>2)
Compile Clarion:  MyApp.app
Rename File:      MyApp.exe -> MyAppTeam.exe
```

The WriteTextToFile changes the #pragma in the project that is included and forces the compiler to set the Glo:Model to the appropriate value. Now you can check Glo:Model in your application and it will tell you what program you are running. Instead of renaming the file you may want to copy it to a different install folder or copy it to an install folder and run SetupBuilder on it and specify a different install name based on what option you are using. I.e. the .exe name would be the same from Clarion, but the .exe name created by SetupBuilder would be different. This gives you a LOT of power with one very simple action!

4.5 INI/Registry Actions

4.5.1 Get from INI INI/Registry Actions

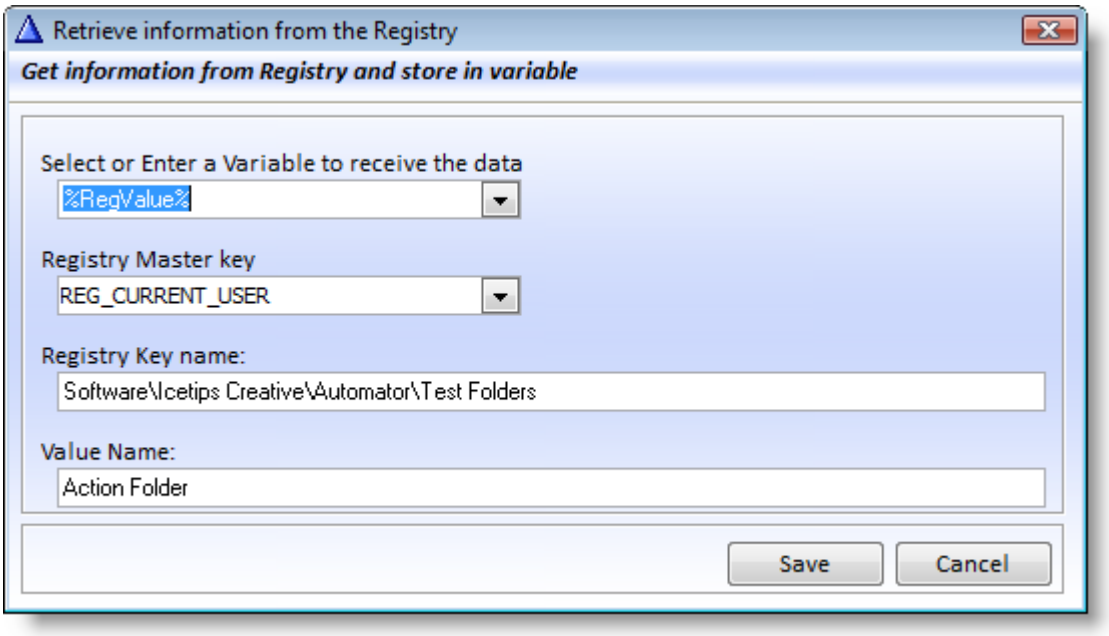
This action retrieves information from an INI file and places the resulting value into a variable.



Properties	Explanation
Select/enter variable	Select a variable to place the resulting value in or enter a variable name to create.
INI File name	Enter the location of the INI file. Click the button on the right of the entry to select an INI file.
Section	The INI section that you want to access.
Entry	The INI Entry that you want to access.
Default Value	Optional default value.

4.5.2 Get from Registry INI/Registry Actions

This action retrieves information from the registry and places the resulting value into a variable.



Properties	Explanation
Select/enter variable	Select a variable to place the resulting value in or enter a variable name to create.
Registry Master key	Select the master key.
Registry Key name	Enter the registry key name that you want to access.
Value Name	Enter the registry value name that you want to access.

The following master keys are available to select from the Registry Master key drop down:

REG_CLASSES_ROOT

REG_CURRENT_USER

REG_LOCAL_MACHINE

REG_USERS

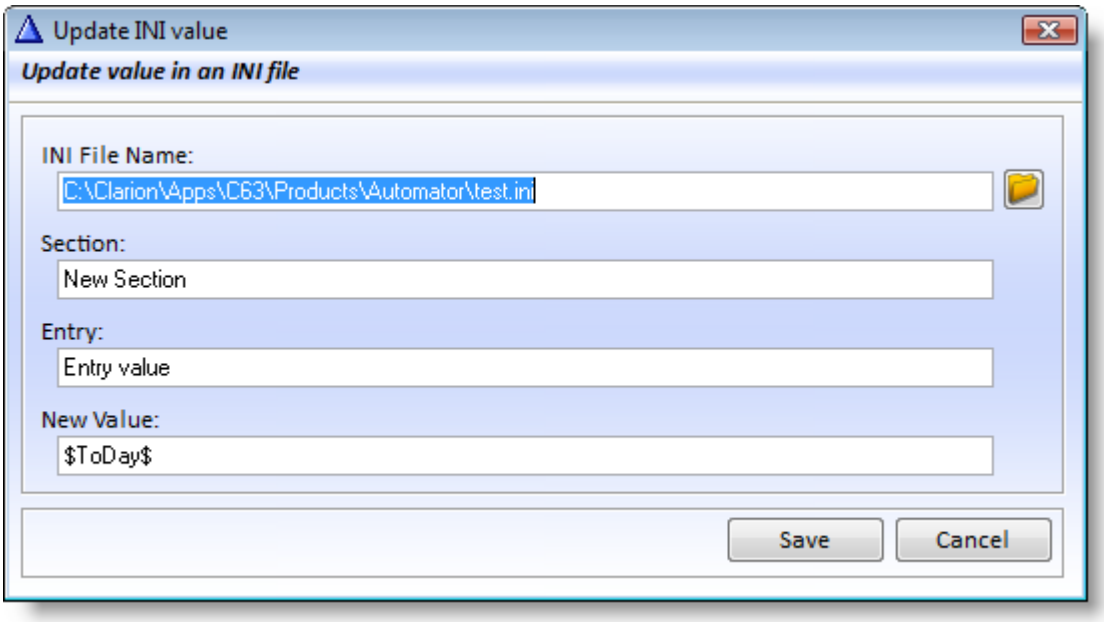
REG_PERFORMANCE_DATA

REG_CURRENT_CONFIG

REG_DYN_DATA

4.5.3 Update INI — INI/Registry Actions

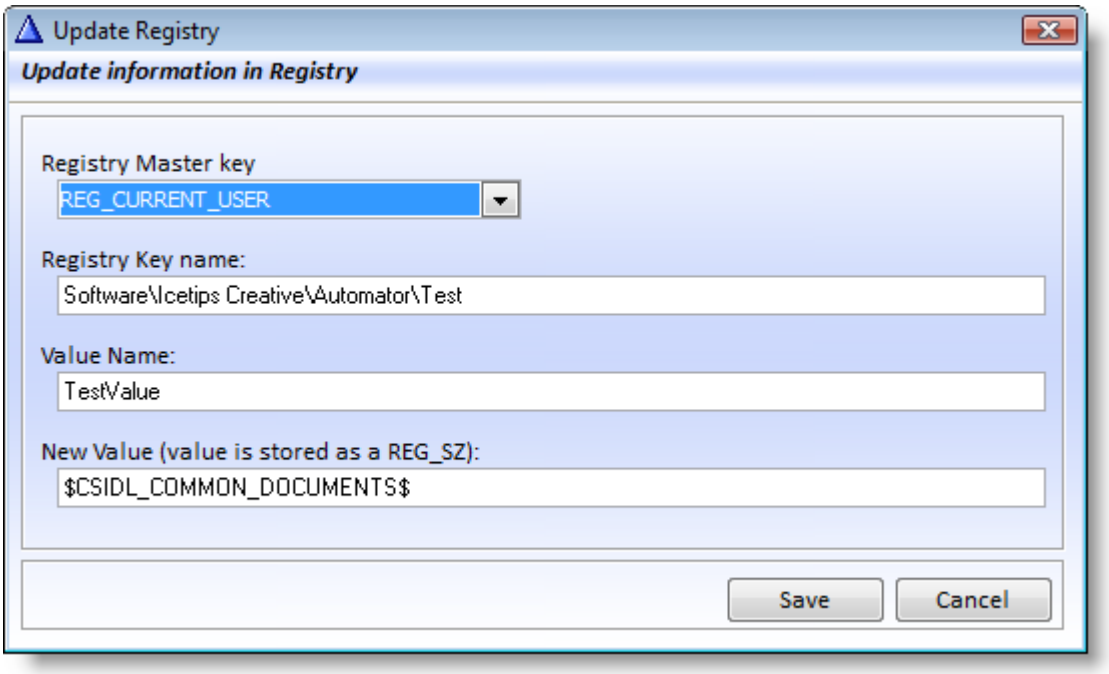
This action updates an entry in an INI file with a given value.



Properties	Explanation
INI File name	Enter the location of the INI file. Click the button on the right of the entry to select an INI file.
Section	The INI Section that you want to update.
Entry	The INI Entry that you want to update.
New Value	The new value for that entry.

4.5.4 Update Registry INI/Registry Actions

This action updates the specified key in the registry with a new value.



Properties	Explanation
Registry Master key	Select the master key.
Registry Key name	Enter the registry key name that you want to update.
Value Name	Enter the registry value name that you want to update.
New value	The new value for the registry value. *Note: Currently this action only support REG_SZ data types for the value.*

The following master keys are available to select from the Registry Master key drop down:

REG_CLASSES_ROOT

REG_CURRENT_USER

REG_LOCAL_MACHINE

REG_USERS

REG_PERFORMANCE_DATA

REG_CURRENT_CONFIG

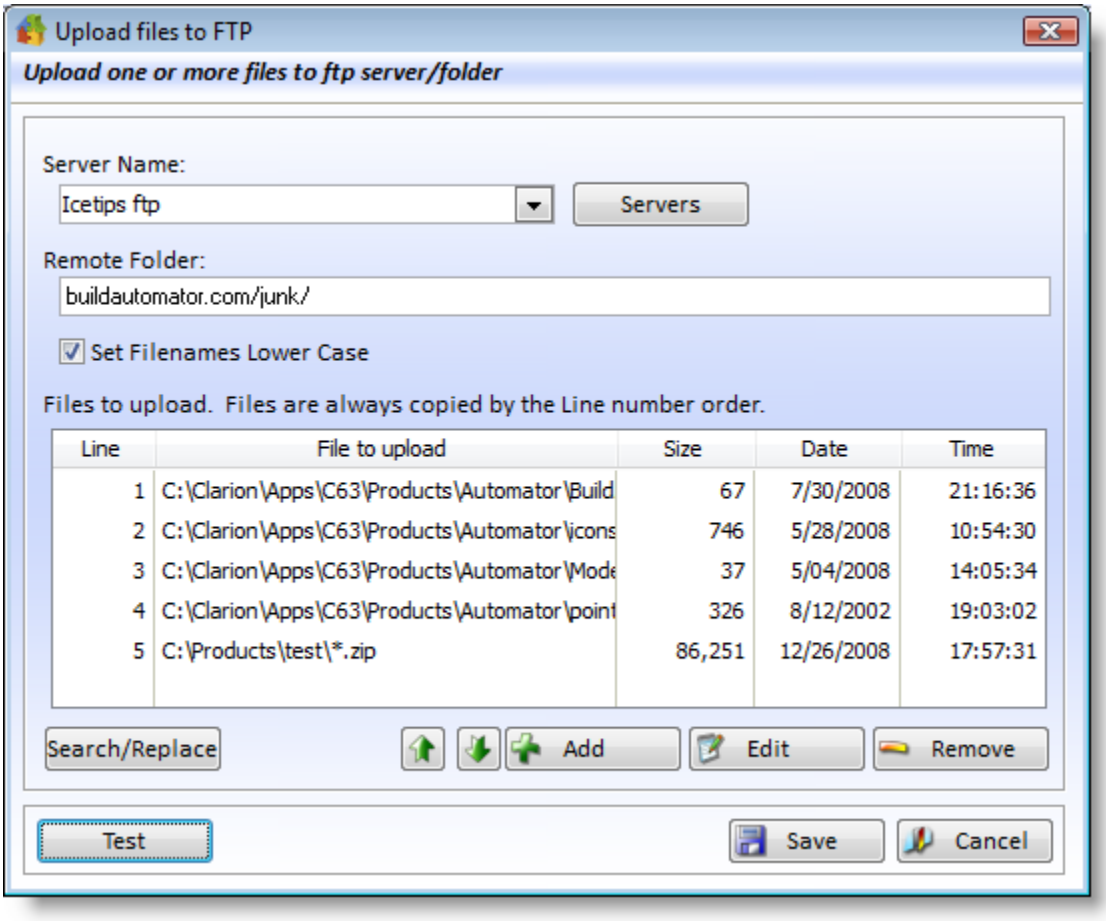
REG_DYN_DATA

4.6 Internet Actions

Enter topic text here.

4.6.1 FTP Upload Internet Actions

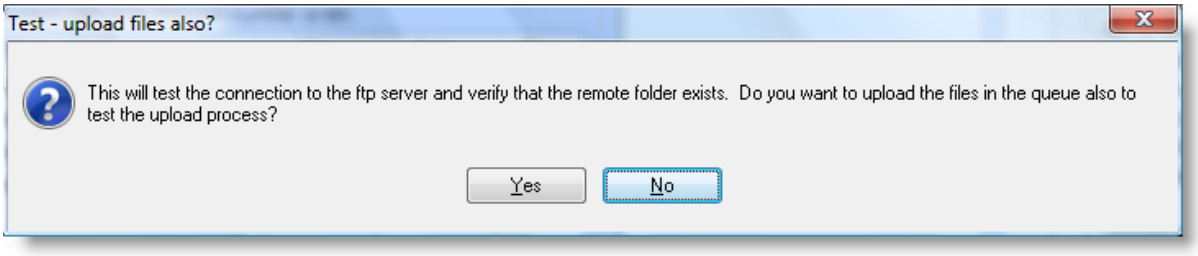
The FTP Upload action makes it possible to upload one or more functions. You can select a single file or multiple files to add to the file list. You can move the files up or down to change the order how they will be uploaded.



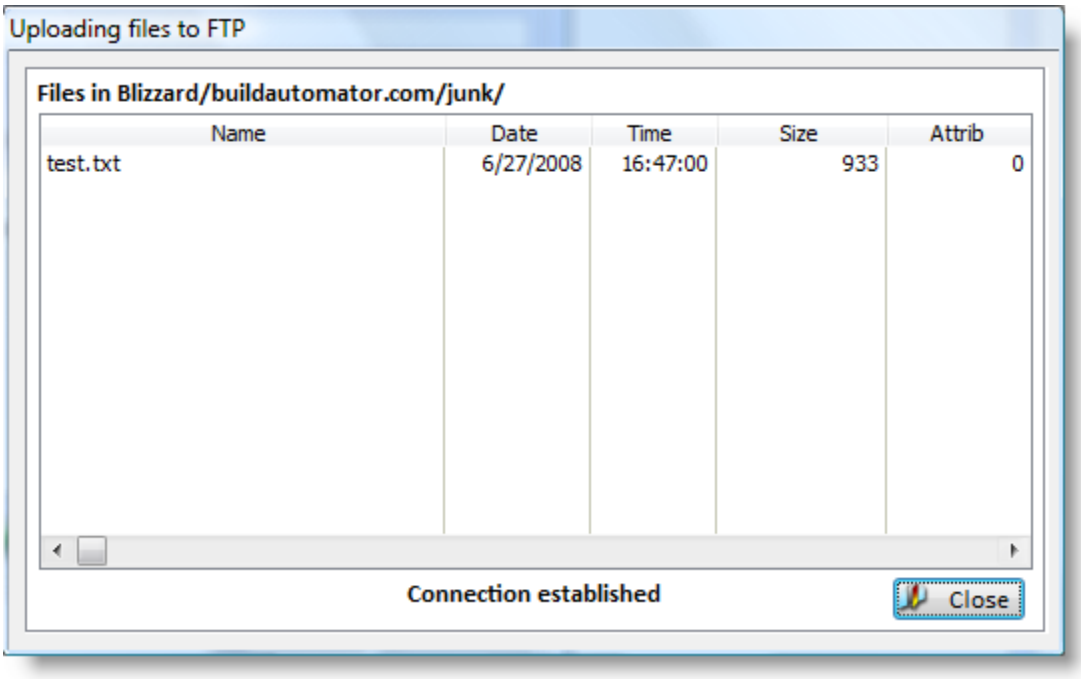
Properties	Explanation
Server Name	The name of the server as it was entered into the Server list 123.
Remote Folder	Enter the path from the root on the server to where you want it to upload the files.
Set Filenames lower...	This changes all the filenames to lowercase as they are uploaded. This does not affect the local filenames.
Files to upload	Files to upload to the server. This list can contain wildcards.

The Servers button takes you to the FTP Servers list where you can add, edit and delete FTP servers.

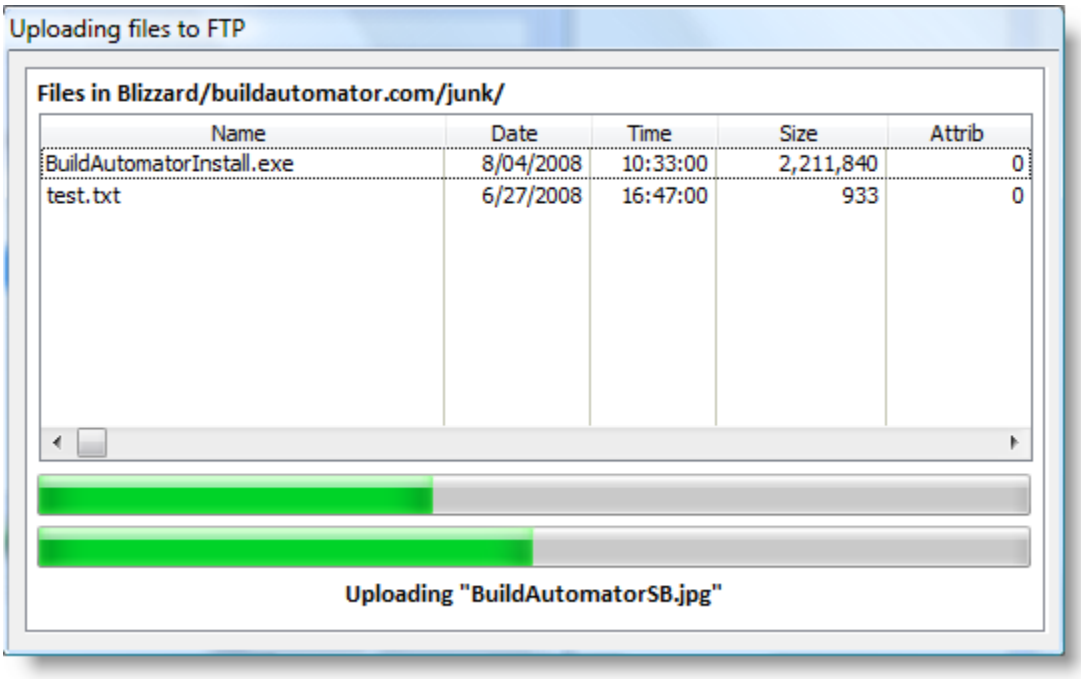
The Test button allows you to test the connection to make sure that the server setup is correct and also to make sure that the remote folder is correct. When you press the button you will get a message asking if you want to upload the files.



If you click on the No button you will get a window that attempts to connect to the server and then navigate to the "Remote Folder"



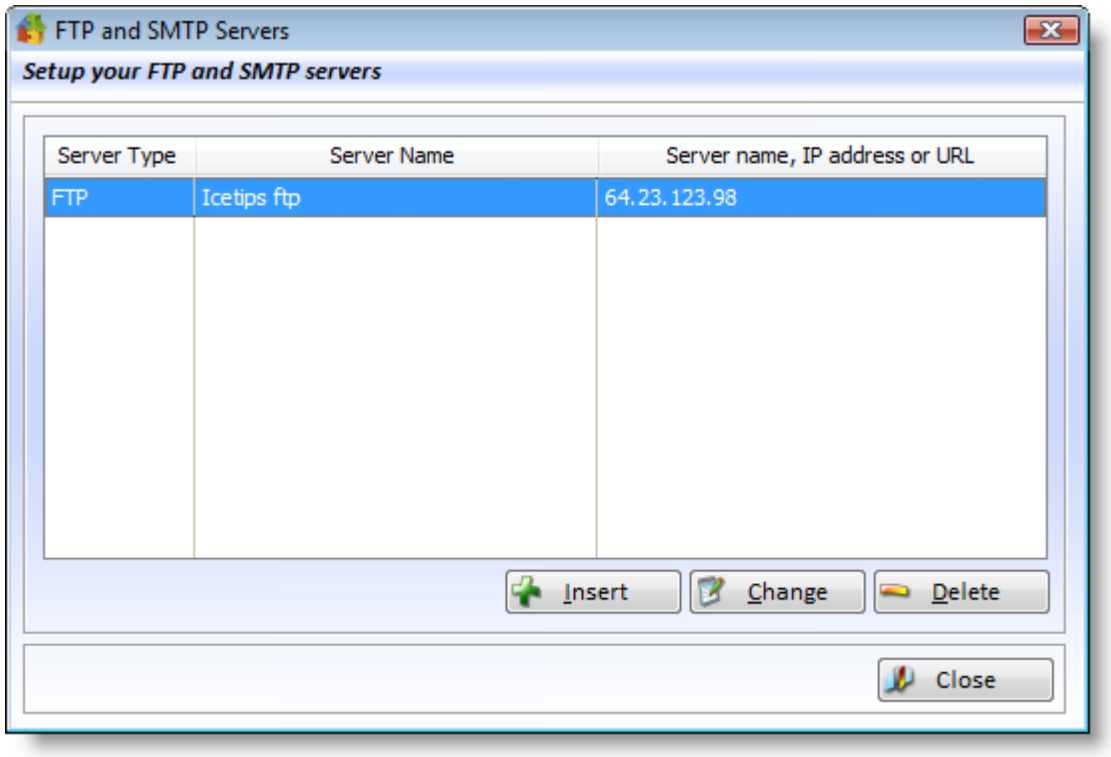
If you click on the Yes button on the message the window will open and it will proceed with uploading the files in the list to the server.



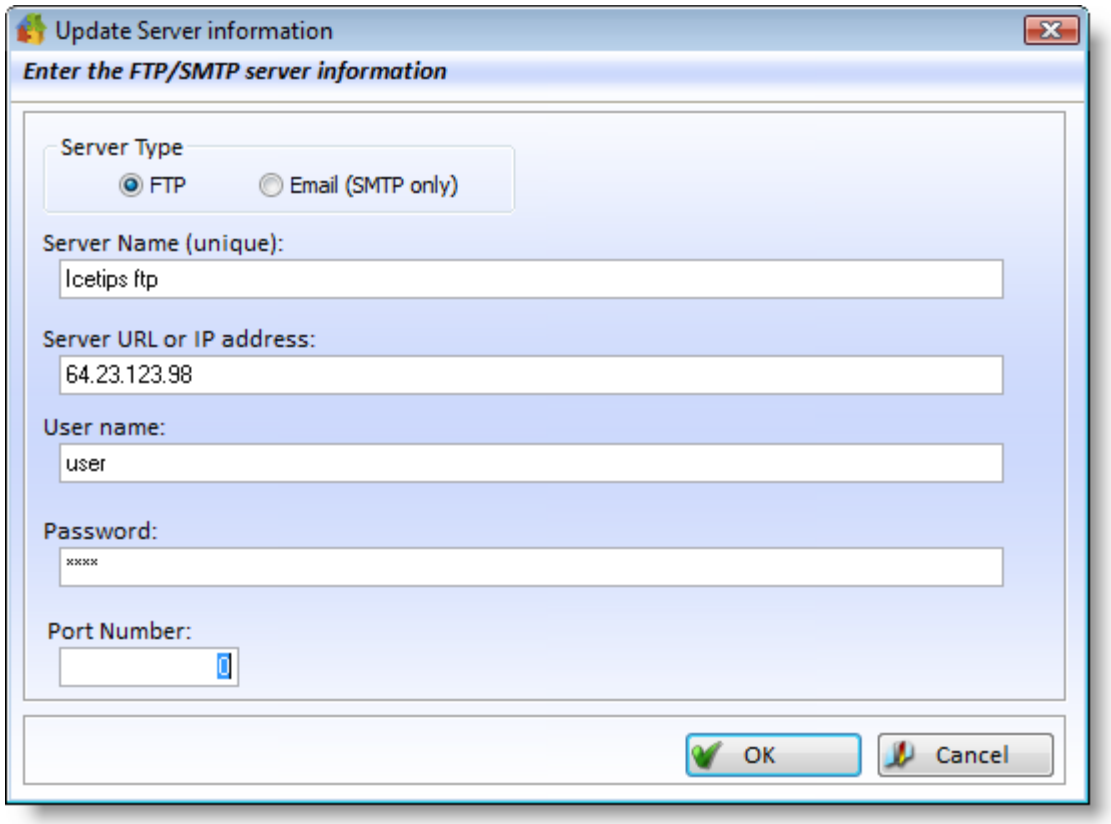
The top progressbar shows the overall progress through the files to upload while the bottom progress bar shows the progress through the file currently being uploaded.

4.6.1.1 Servers

Servers are stored in a separate file and can be selected from a dropdown in the various internet related actions. Servers can be either FTP or SMTP servers for FTP and email actions.



Use the Insert button to create a new entry.



Properties	Explanation
Server Type	Select either FTP or SMTP.
Server Name	Descriptive name of the FTP server. This name is used in the server selection dropdown.
Server IP or URL	Here you need to enter either the IP address or the URL of the ftp server.
Username	Enter your user name for the server.
Password	Enter your password for the server.
Port number	Currently not used but reserved for future use. Specify the port number to use for the server connection.

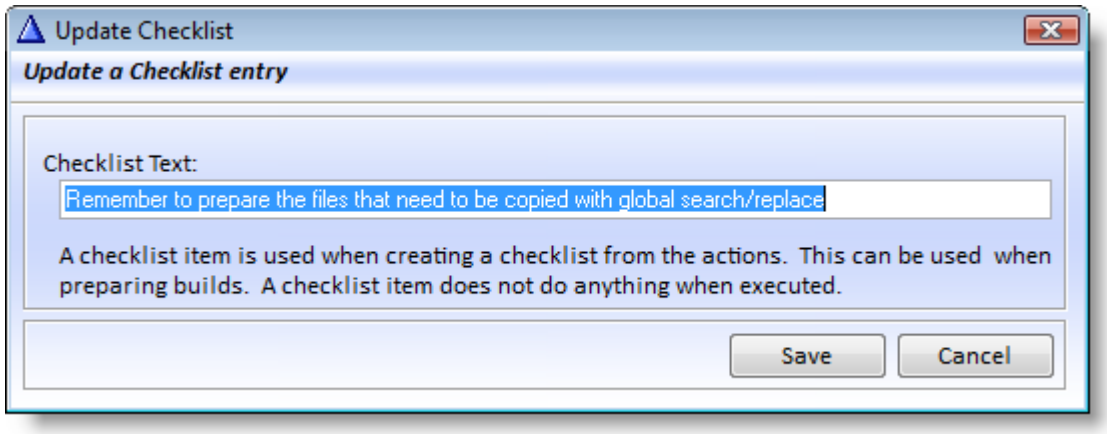
See also:

FTP Upload 121
Automator Options 17

4.7 Notification Action

4.7.1 Checklist Notification Action

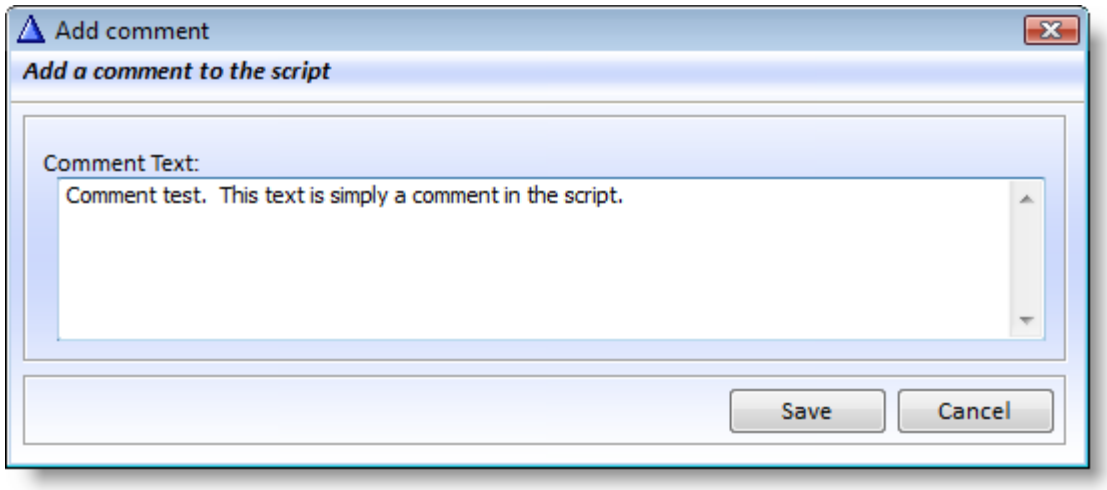
The Checklist action adds an item that is intended for the checklist report 46. This action does not execute.



Properties	Explanation
Checklist Text	One line text for the checklist report.

4.7.2 Comments Notification Action

The Comments action adds a simple, single line comment to your script. This action does not execute.

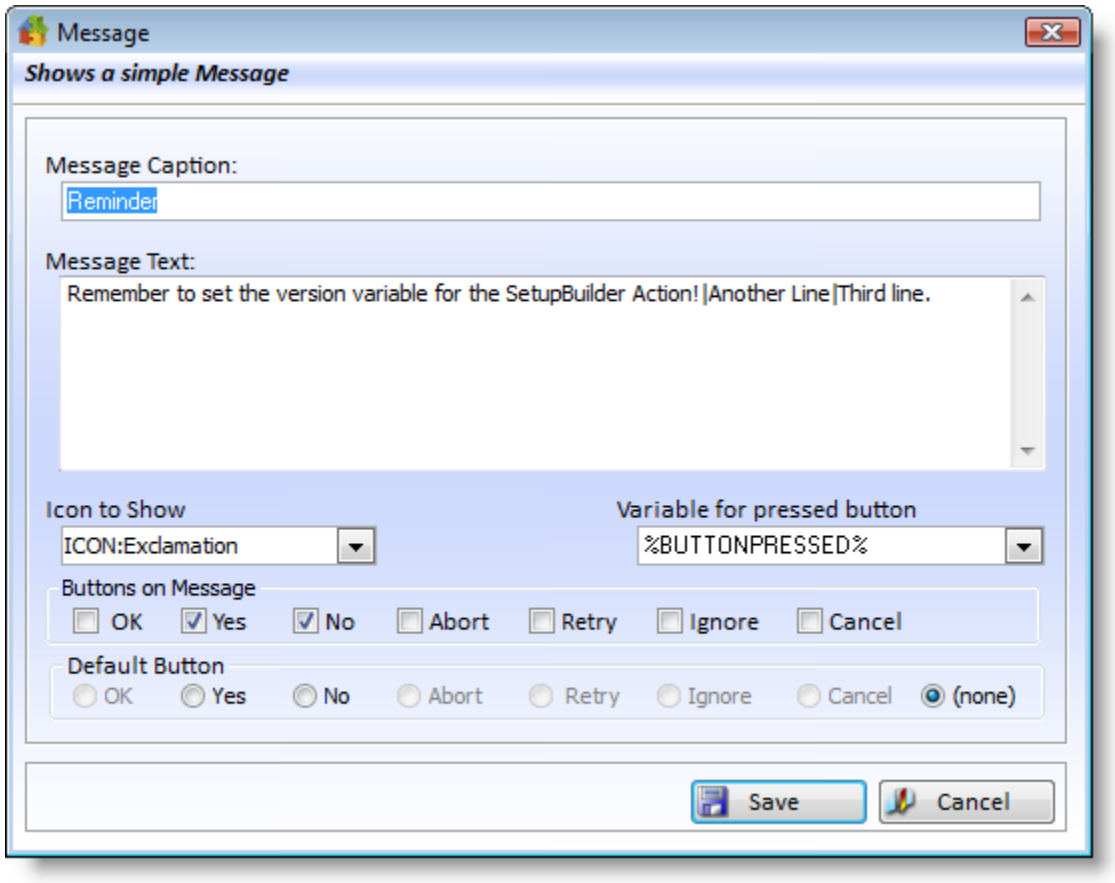


Properties	Explanation

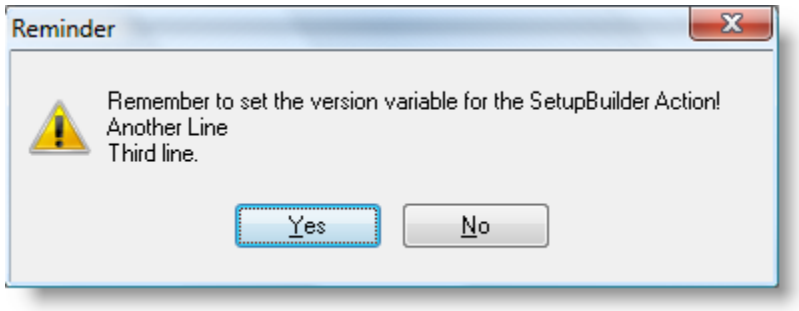
Comment Text	Text to put as a comment into the script.

4.7.3 Message Notification Action

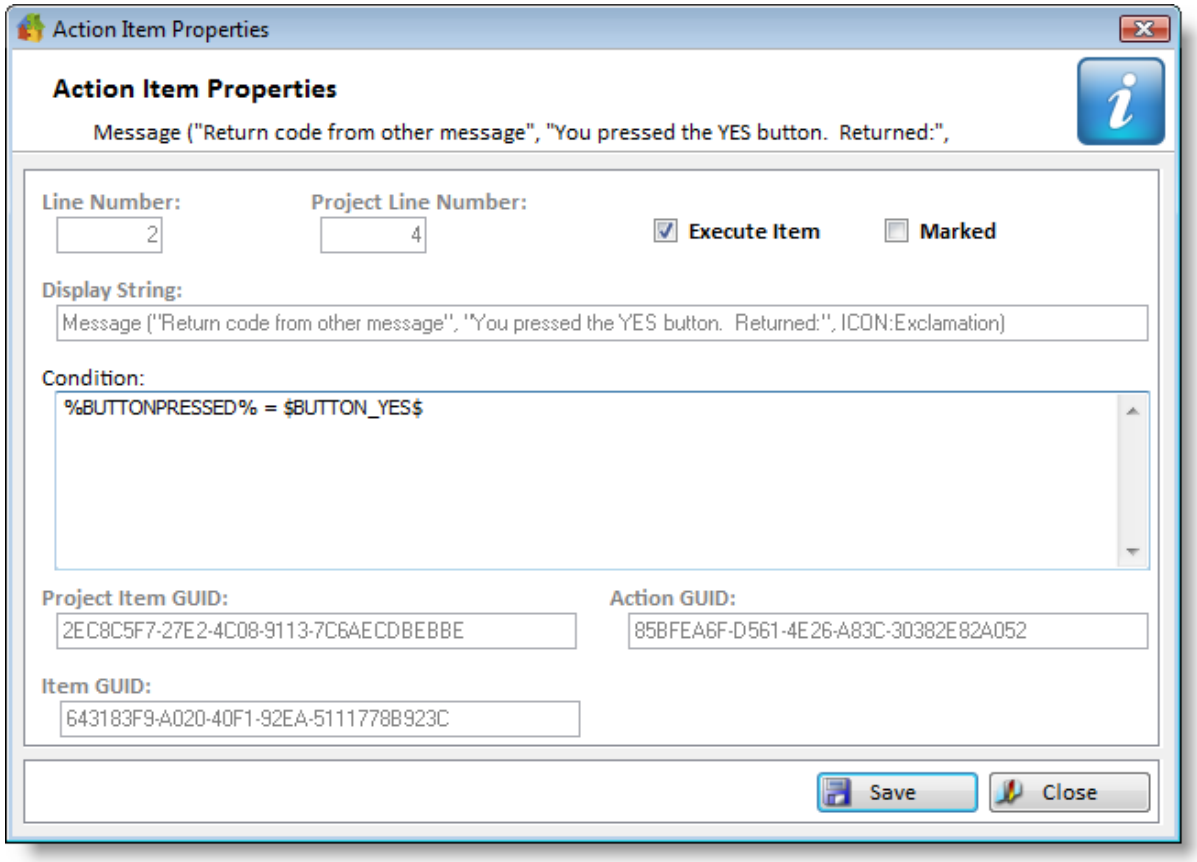
This is a very simple Message action that can come in handy to put reminders into your script. You can set the caption, the main text and the icon being displayed.



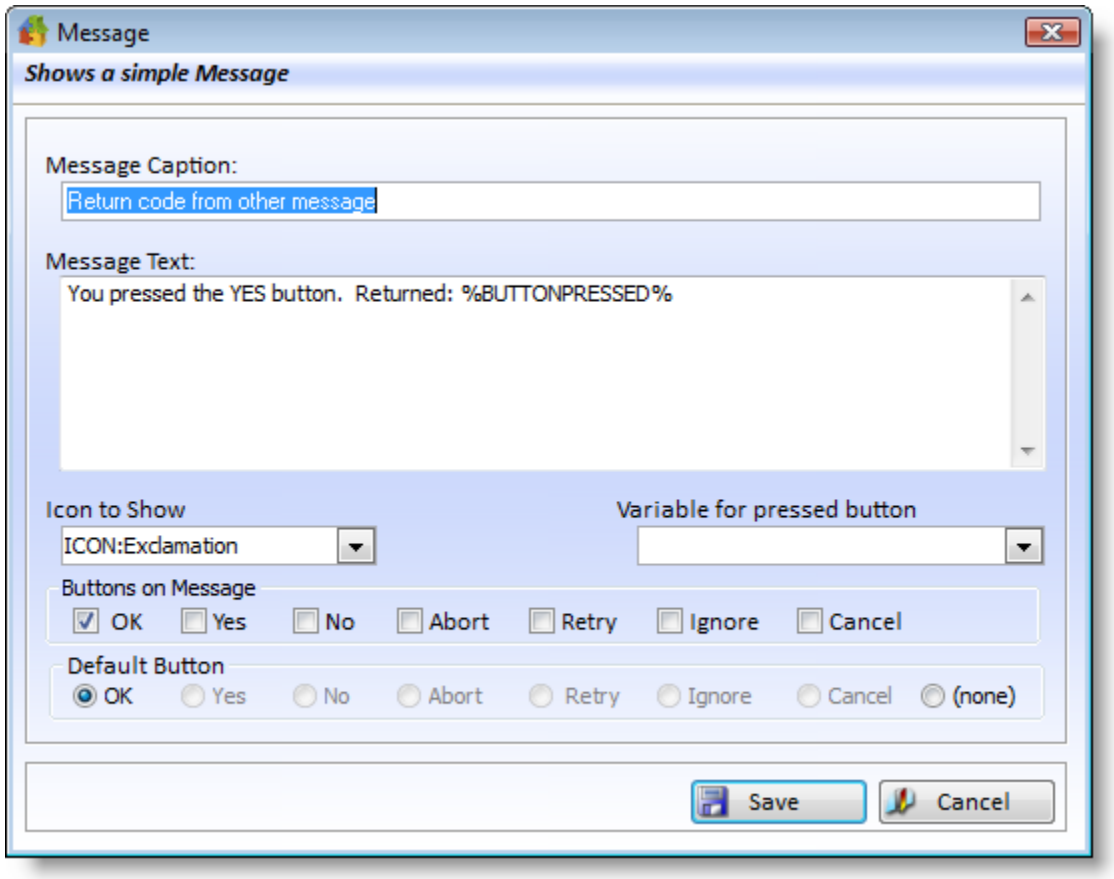
This results in the message window shown below:



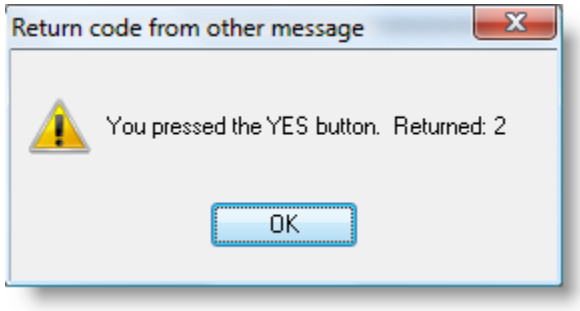
You can use the button variable, in this case %BUTTONPRESSED% in conditions in another action. Below is an example screenshot of an action properties:



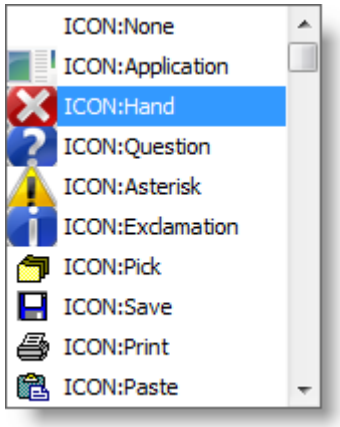
The $BUTTON_YES$ and other button constants are declared as System Variable 135 so you can test the results from a message against them. This condition belongs to another Message action that looks like this:



When you run this, it will show up like this, ONLY when you click on the Yes button in the first message:



Properties	Explanation
Message Caption	The caption of the message window.
Message Text	The main message text. Use pipe characters to add an empty line to the message. For example "One|Two" will result in two lines in the message. Using a line break in the text does the same thing. Note that you can use variables in the main text.

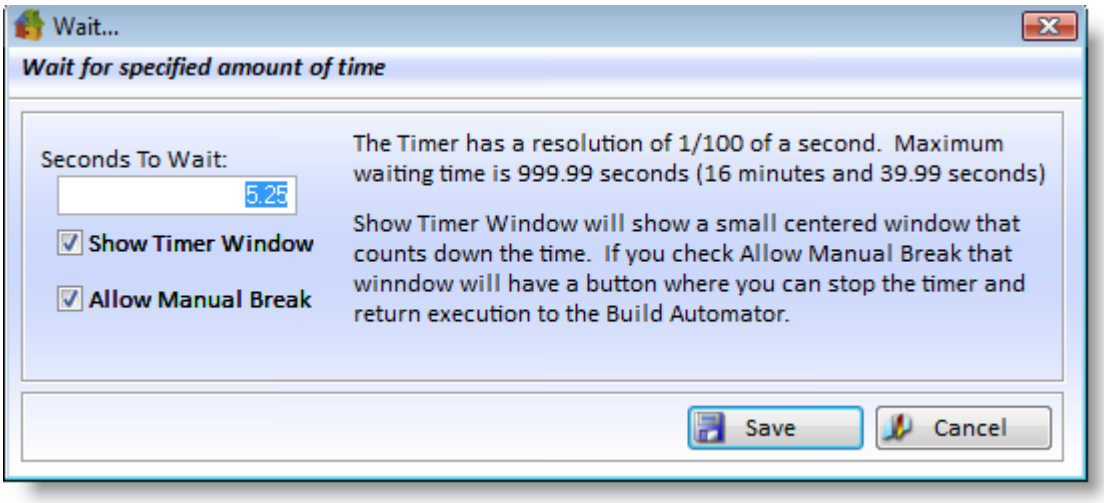
Icon to Show 	This lets you pick from several standard icons to use on the message window.
Variable for pressed button	Here you can select or enter a variable name to receive the value of the pressed button. You can then compare this to the various $BUTTON_$ constants that are declared as System Variables. See below.
Buttons on Message	Check all the applicable buttons. The buttons you click will all show up when the Message action is executed. When you check a button it also becomes enabled in the Default button option.
Default button	Select the button that you want to be the default button. Select None if you don't want any default button.

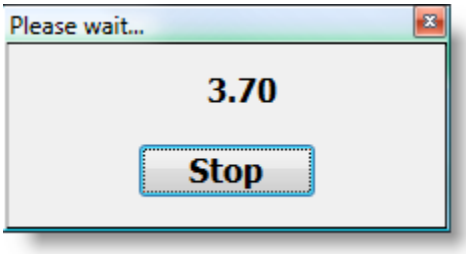
Available button constants for the Message action are:

Button Constant	Hex value	Decimal value
$BUTTON_OK$	0x01	1
$BUTTON_YES$	0x02	2
$BUTTON_NO$	0x04	4
$BUTTON_ABORT$	0x08	8
$BUTTON_RETRY$	0x10	16
$BUTTON_IGNORE$	0x20	32
$BUTTON_CANCEL$	0x40	64

4.7.4 Wait — Notification Action

This action waits for a specified number of seconds. It can optionally show a window that counts the time left waiting and also optionally show a button where the user can stop the timer at any time and return back to the execution of the script. This action can come in handy with programs that don't behave when they are set to wait but take a fairly fixed amount of time to finish their job. The execution of this action doesn't do anything except wait.

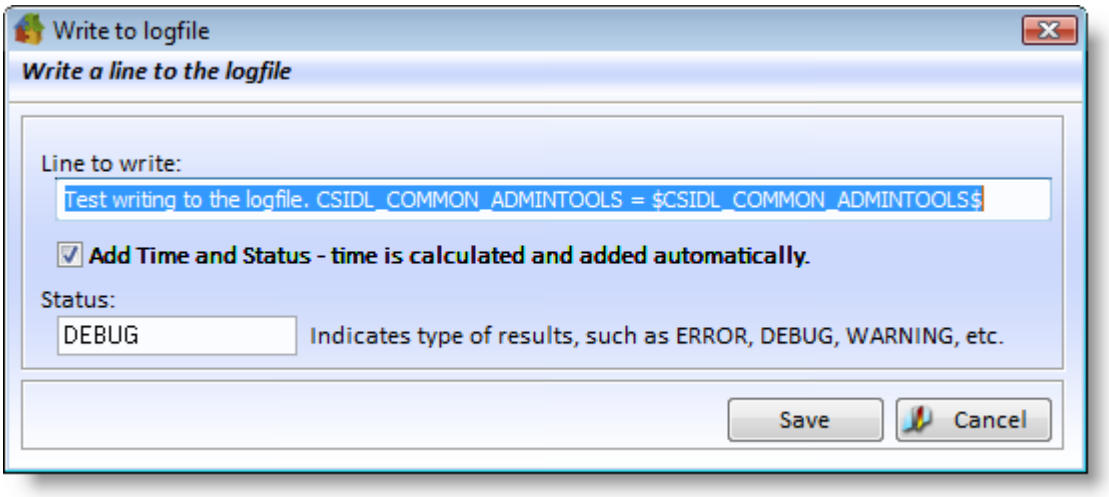


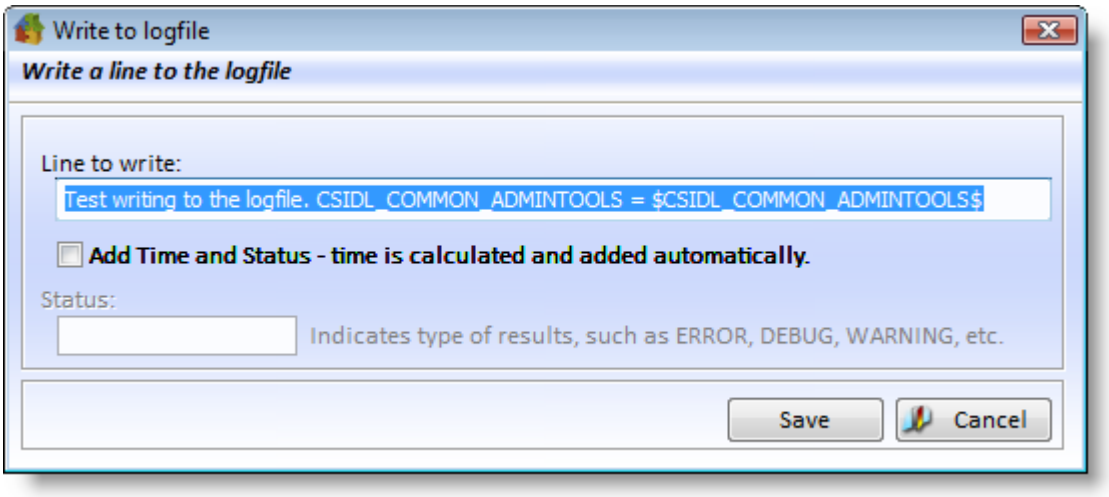


Properties	Explanation
Seconds to Wait	Enter the number of seconds to wait. You can enter values between 0.01 and 999.99, although practical lower limit is around 0.5 second.
Show Timer Window	Displays a window where the time is counted down. See screenshot above.
Allow Manual Break	Only available when Show Timer Window is checked. It enables a Stop button on the timer window. See screenshot above.

4.7.5 Write Line to Logfile — Notification Action

This action writes a single line to the log file. The line can be formatted or not formatted. If it is formatted the time of the write is added as well as an optional Status text. This is a great way to get debug information about variables.





Properties	Explanation
Line to write	The text you want to add to the log file.
Add Time/ Status	This formats the line. See below for examples.
Status	Enter optional status. This could be something like DEBUG, ERROR, WARNING, etc.

The logfiles are written each time a project or part of project is executed. The logfile writes are performed by two functions called *WriteLogLine* and *WriteFormatLogLine*. The difference is in how they write the line sent to them.

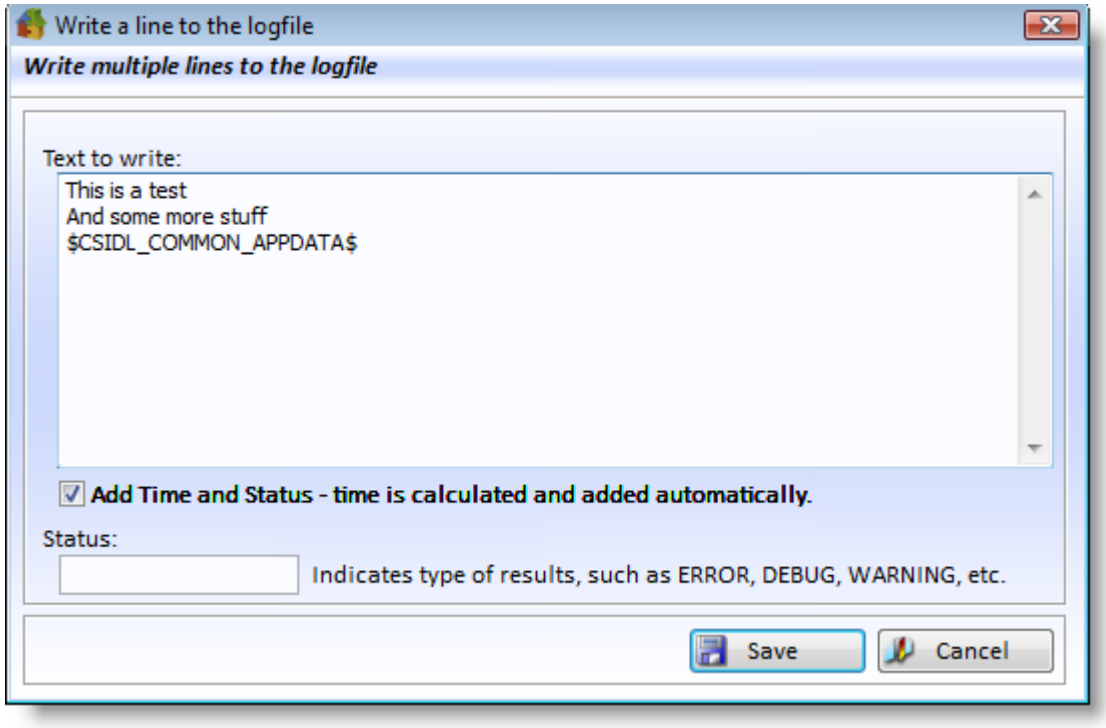
```
================================================================================
Log for project:         T:\Shared Documents\Build Automator Projects\test1.aprj
Project Name:            test1
Started on:              09/15/2015 16:24:10
================================================================================
```

```
Line     Time         Status      Information
-------------------------------------------------------------------------------
010 +Test Compiles
010-0005 16:24:11.153 *EXECUTE    Compile in Clarion 6.0: C:\Clarion\Apps\C63\Products\Automato
         16:24:11.480 INFO        Compiling 1 Clarion apps.
         16:24:11.668 INFO        Compile Started:  C:\Clarion\Apps\C63\Products\AutomatorDLL\B
         16:24:19.369 ERROR       Compile failed: C:\Clarion\Apps\C63\Products\AutomatorDLL\BAS
         16:24:19.384 CLAERROR      (BASup001.clw 12,2) Make error: Expected: <ID> & APPLICATIO
         16:24:19.394 CLAERROR      (BASup001.clw 13,2) Syntax error: Illegal data type: X
         16:24:19.404 CLAERROR      (BASup001.clw 13,5) Syntax error: Unknown attribute: BYTE
         16:24:19.412 CLAERROR      (BASup001.clw 14,4) Syntax error: Illegal data type: THIS
         16:24:19.441 INFO        Compiling complete for 1 apps.
-------------------------------------------------------------------------------

Execution ended on:      09/15/2015 16:24:19
Execution took:          8.830 seconds to execute the script.
```

In the example above the lines that are colored green **are formatted**, and the ones colored blue **are not formatted**.

4.7.6 Write Multiple lines to Logfile — Notification Action

This action works exactly the same as the Write to Logfile 131 action, except it can write multiple lines to the logfile.



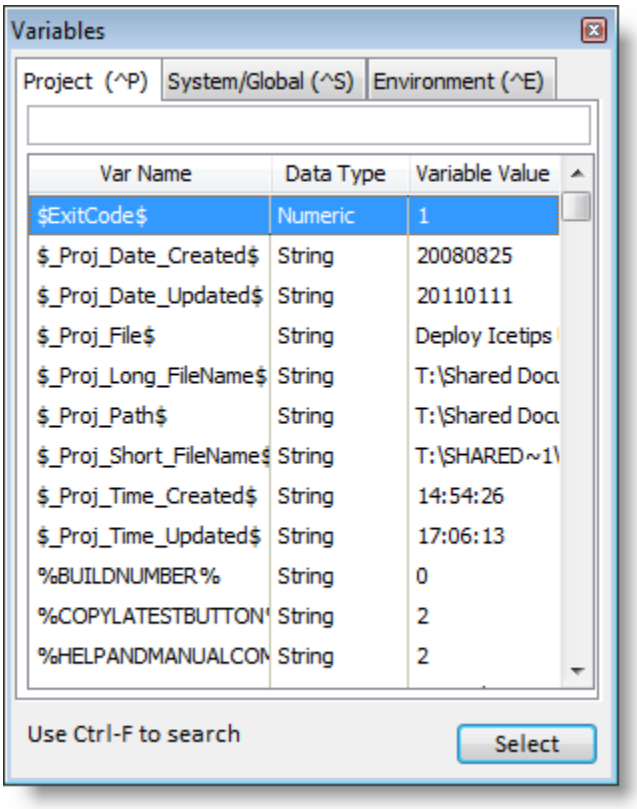
Properties	Explanation
Text to write	Enter text to write to the logfile. Each line is written separately to the logfile. The text in the screenshot will result in 3 lines in the logfile:

	17:10:37.203 This is a test 17:10:37.203 And some more stuff 17:10:37.203 C:\Documents and Settings\All Users \Application Data
Add Time/ Status	This adds the time and status option. Referring to the partial logfile in the "Text to write" above, this triggers the time to be written to the logfile. Without it the text would start at the left margin.
Status	Use this to give an indication in the logfile if this is an ERROR, WARNING or a DEBUG entry.

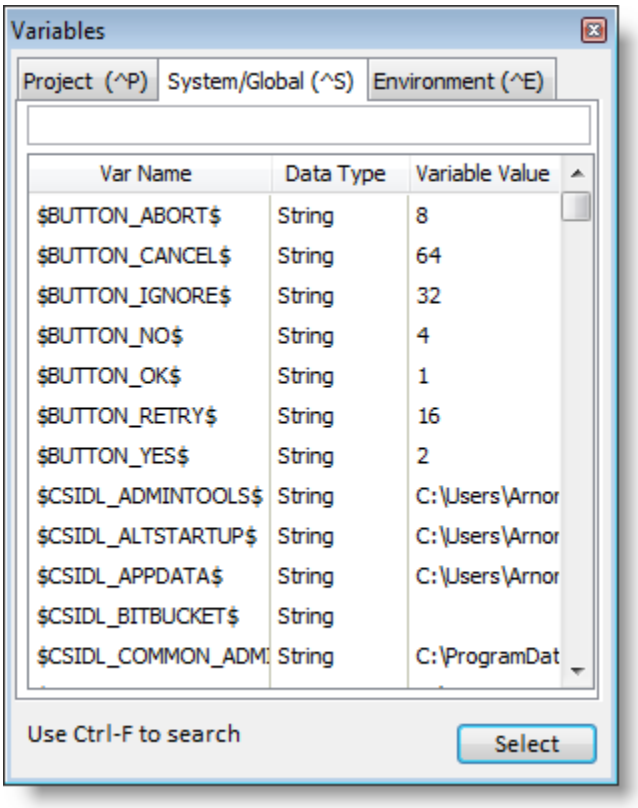
4.8 Select Variable

In the actions provided by the author of Build Automator™, Icetips Alta LLC, you can select variables into most entries on the Action windows. To select a variable hold down the Control Key (CTRL) and clicking with the right mouse button. Alternatively place the cursor where you want the variable to appear, hold down the Control key and press the down arrow key on the keyboard. If you want to close the window without selecting a variable, press the Esc key on the keyboard or click the red close button in the top right corner of the window.

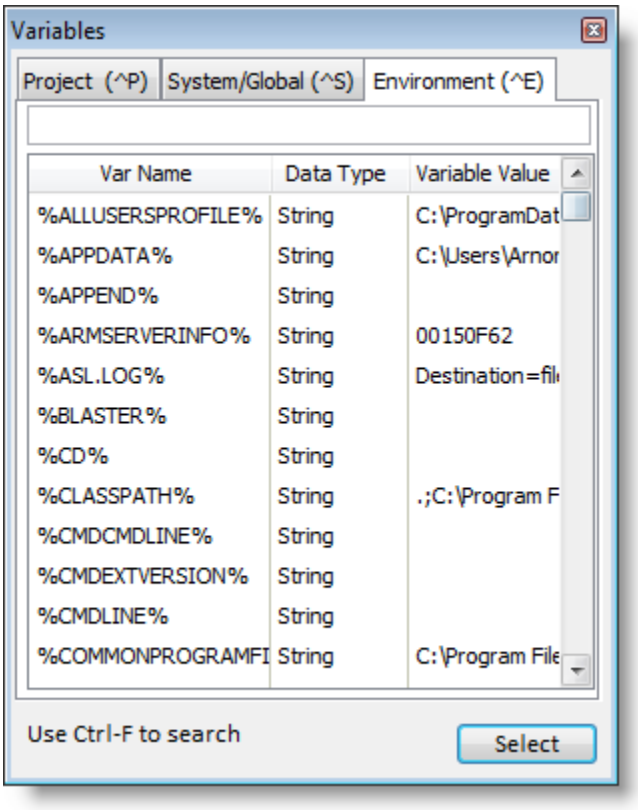
The Build Automator™ supports three kinds of variables, Project Variables, System/Global Variables and Environment Variables. Environment variables use % as name delimiters, just like Project Variables so you cannot use names for Project Variables that are the same as environment variables. The list of Environment variables contains both the environment variables that are normally defined by the operating system as well as those that are user defined.



The Project Variables 34 are related to the project and you can add new variables to the list, delete or change them. Please check the Project Variables 34 topic for more information about how to create variables. Variables that you create can have different data types, String, Decimal, Numeric, Date or Time. The data is always stored as strings (i.e. as plain text) but the data type defines how the data is formatted. Project variables can be changed and are saved to disk in separate file that uses the same name as the project, but with a **.avar** extension. The format of the .avar file is XML.

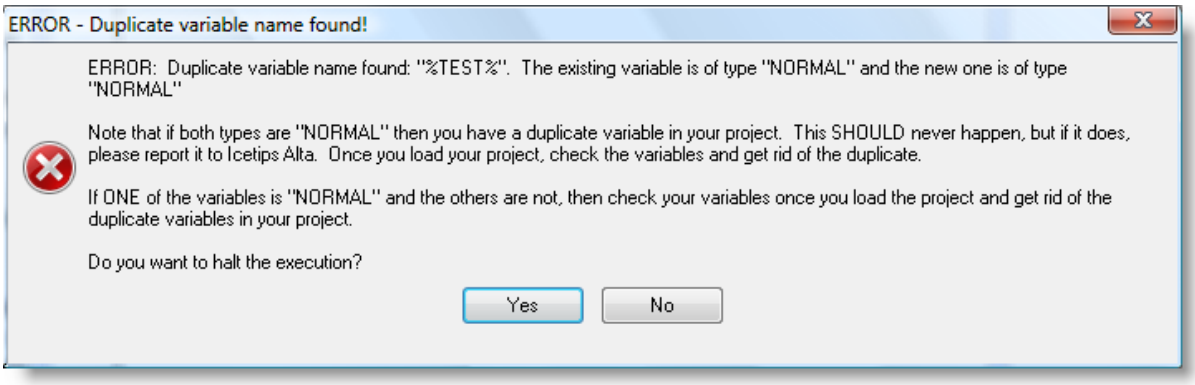


System or Global variables are variables that the system updates automatically. In the start of the beta we have all the CSIDL paths as well as the current date and time. System variables cannot be changed and are not saved to disk.



Environment variables are retrieved from the operating system environment. The data in each variable is automatically expanded so any environment variables inside the environment variable data is correct when Build Automator accesses it. The maximum size of the entire environment variable block is 32K (32,767 characters/bytes) and the maximum size of each entry is either 2K (2047 characters) or 8K (8191 character) depending on the operating system. For more information about the environment variables please visit http://support.microsoft.com/kb/830473. Environment variables cannot be changed and are not saved to disk.

Variables must have unique names and with the addition of Environment variables, which like the Project variables use % to enclose the variable name, there are possibilities that you may have two variables with the same name. Build Automator detects this when you load a project and will warn you if it finds duplicate variable names.



If you click Yes, Build Automator will close. If you click No, Build Automator will continue and you can go through your project and fix the duplicate name problem.

Note that the Variables window is resizable and will remember it's size and location next time you open it to make it more convenient for you to select.

Build Automator

Part V

Chapter 5 - License and Support

5 License and Support

In this chapter we will detail support information as well as any other relative information to the Build Automator™. This chapter also includes the software license agreement.

We offer a one year maintenance plan for the software. It includes all updates to the software for one year. Without the maintenance plan you are not entitled to updates to the software.

Currently we have a built-in module in the program that will provide you with on-line support. Please be aware that we may not be able to get to you right away so please leave a message if nobody picks up your support request. We do not have a large support staff but if you leave your name and email we will get back to you as soon as possible. We also have an online forum and blog where we post semi-regularly about what we are working on or other interesting stuff.

5.1 End-User License Agreement for the Build Automator™

End-User Licence agreement for Build Automator™

ICETIPS ALTA LLC ("ICETIPS") IS WILLING TO LICENSE THE SOFTWARE ONLY UPON THE CONDITION THAT YOU ACCEPT ALL OF THE TERMS CONTAINED IN THIS SOFTWARE LICENSE AGREEMENT. PLEASE READ THE TERMS CAREFULLY. BY SELECTING THE "I ACCEPT..." OPTION OR BY INSTALLING THE SOFTWARE, YOU WILL INDICATE YOUR AGREEMENT WITH THEM. IF YOU ARE ENTERING INTO THIS AGREEMENT ON BEHALF OF A COMPANY OR OTHER LEGAL ENTITY, YOUR ACCEPTANCE REPRESENTS THAT YOU HAVE THE AUTHORITY TO BIND SUCH ENTITY TO THESE TERMS, IN WHICH CASE "YOU" OR "YOUR" SHALL REFER TO YOUR ENTITY. IF YOU DO NOT AGREE WITH THESE TERMS, OR IF YOU DO NOT HAVE THE AUTHORITY TO BIND YOUR ENTITY, THEN ICETIPS IS UNWILLING TO LICENSE THE SOFTWARE, AND YOU SHOULD SELECT THE "I DO NOT ACCEPT..." OPTION AND THE DOWNLOAD OR INSTALL WILL NOT CONTINUE.

SOFTWARE LICENSE AGREEMENT

1. **PARTIES.** The parties to this Agreement are you, the licensee ("You") and Icetips. If You are not acting on behalf of Yourself as an individual, then "You" means Your company or organization.

2. **THE SOFTWARE.** The Software licensed under this Agreement consists of computer programs only in compiled, object code form, data compilation(s), and documentation referred to as Build Automator (the "Software").

3. **EVALUATION VERSION LICENSE GRANT.** If You have downloaded or otherwise received an evaluation version of the Software, You are authorized to use the Software on a royalty-free basis for evaluation purposes during the initial evaluation period of thirty (30) days. During the evaluation period, You may copy the Software for archival purposes, provided that any copy must contain the original Software's proprietary notices in unaltered form, and you may distribute and/or transmit as many copies to others as You wish. You have the option to register for full use of the Software at any time during the evaluation period by following the instructions in the accompanying documentation, including the payment of the required license fee. Registration will authorize You to use an unlocking key which will convert the Software to full use, in accordance with the terms and conditions provided below. Your use of the Software for any purpose after the expiration of the initial evaluation period is not authorized. Upon expiration of the limited evaluation period, the Software may automatically disable itself.

4**. PERPETUAL TERM FOR REGISTERED VERSION LICENSE.** The term of the license granted herein for the registered version of the Software shall be perpetual unless terminated by written notice by You for convenience or terminated by either party for material breach. Immediately upon termination of this license for any reason, You shall return to Icetips all copies of the Software and documentation. Provided that You have paid the applicable Maintenance fees, You will be entitled to receive any Maintenance Releases and/or Upgrades made generally available during the Maintenance Period for the Software licensed from Icetips by You and covered under a Maintenance and Support Plan. Any Upgrades released during the Maintenance Period shall be made available for electronic download by You. Icetips shall provide You with instructions regarding registration for such electronic downloads. When a Maintenance Release or Upgrade is available for download, You will receive an electronic communication from Icetips indicating the availability of electronic downloads of upgrades to the Software. Use of each Upgrade is subject to the terms of the license agreement for such Upgrade.

5. **REGISTERED VERSION LICENSE GRANT FOR SINGLE COPIES (NON-NETWORK USE).** If You are a registered user of the Software, You are granted non-exclusive rights to install and use the Software by a single person who uses the Software only on one or more computers or workstations. You may copy the Software for archival purposes, provided that any copy must contain the original Software's proprietary notices in unaltered form.

6. **REGISTERED VERSION LICENSE GRANT FOR NETWORK USE.** If You are a registered user of the Software, You are granted non-exclusive rights to install and use the Software and/or transmit the Software over an internal computer network, provided You acquire and dedicate a licensed copy of the Software for

each user who may access the Software concurrently with any other user. You may copy the Software for archival purposes, provided that any copy must contain the original Software's proprietary notices in unaltered form.

7. **RESTRICTIONS.** You may not: (i) permit others to use the Software, except as expressly provided above for authorized network use; (ii) modify or translate the Software; (iii) reverse engineer, decompile, or disassemble the Software, except to the extent this restriction is expressly prohibited by applicable law; (iv) create derivative works based on the Software; (v) merge the Software with another product; (vi) copy the Software, except as expressly provided above; or (vii) remove or obscure any proprietary rights notices or labels on the Software.

8. **PURCHASE OF ADDITIONAL LICENSES.** Registered users of the Software may purchase license rights for additional authorized use of the Software in accordance with Icetips's then-current volume pricing schedule. Such additional licenses shall be governed by the terms and conditions hereof. You agree that, absent Icetips's express written acceptance thereof, the terms and conditions contained in any purchase order or other document issued by You to Icetips for the purchase of additional licenses, shall not be binding on Icetips to the extent that such terms and conditions are additional to or inconsistent with those contained in this Agreement.

9**. TRANSFERS.** You may make a one-time permanent transfer of all of your license rights to the Software to another party, provided that all of the following conditions are satisfied: (a) you notify us in writing of your intent to transfer your license rights and identify the party receiving the Software with complete contact information; (b) the transfer must include all of the Software, including all its component parts, original media, printed materials and this License Agreement; (c) you do not retain any copies of any version of the Software, full or partial, including copies stored on a computer or other storage device; and (d) the party receiving the Software reads and agrees to accept the terms and conditions of this License Agreement. Notwithstanding the foregoing, we reserve the right to require the transfer of possession of all physical copies of the Software to us for purposes of re-issue of replacement copies to the party receiving the Software.

10. **OWNERSHIP.** Icetips and its suppliers own the Software, all physical copies thereof, and all intellectual property rights embodied therein, including copyrights and valuable trade secrets embodied in the Software's design and coding methodology. The Software is protected by United States copyright laws and international treaty provisions. This Agreement provides You only a limited use license, and no ownership of any intellectual property. We reserve the right to require you to transfer possession of all physical copies of the Software to us for purposes of re-issue of replacement copies.

11. **WARRANTY DISCLAIMER; LIMITATION OF LIABILITY.** ICETIPS PROVIDES THE SOFTWARE "AS-IS" AND PROVIDED WITH ALL FAULTS. NEITHER ICETIPS NOR ANY OF ITS SUPPLIERS OR RESELLERS MAKES ANY WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. ICETIPS AND ITS SUPPLIERS SPECIFICALLY DISCLAIM THE IMPLIED WARRANTIES OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, SYSTEM INTEGRATION, AND DATA ACCURACY. THERE IS NO WARRANTY OR GUARANTEE THAT THE OPERATION OF THE SOFTWARE WILL BE UNINTERRUPTED, ERROR-FREE, OR VIRUS-FREE, OR THAT THE SOFTWARE WILL MEET ANY PARTICULAR CRITERIA OF PERFORMANCE, QUALITY, ACCURACY, PURPOSE, OR NEED. YOU ASSUME THE ENTIRE RISK OF SELECTION, INSTALLATION, AND USE OF THE SOFTWARE. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL PART OF THIS AGREEMENT. NO USE OF THE SOFTWARE IS AUTHORIZED HEREUNDER EXCEPT UNDER THIS DISCLAIMER.

12. **LOCAL LAW.** If implied warranties may not be disclaimed under applicable law, then ANY IMPLIED WARRANTIES ARE LIMITED IN DURATION TO THE PERIOD REQUIRED BY APPLICABLE LAW. Some jurisdictions do not allow limitations on how long an implied warranty may last, so the above limitations may not apply to You. This warranty gives you specific rights, and You may have other rights which vary from jurisdiction to jurisdiction.

13. **LIMITATION OF LIABILITY.** INDEPENDENT OF THE FORGOING PROVISIONS, IN NO EVENT AND UNDER NO LEGAL THEORY, INCLUDING WITHOUT LIMITATION, TORT, CONTRACT, OR STRICT PRODUCTS LIABILITY, SHALL ICETIPS OR ANY OF ITS SUPPLIERS BE LIABLE TO YOU OR ANY OTHER PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY KIND, INCLUDING WITHOUT LIMITATION,

DAMAGES FOR LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER MALFUNCTION, OR ANY OTHER KIND OF COMMERCIAL DAMAGE, EVEN IF ICETIPS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY TO THE EXTENT PROHIBITED BY APPLICABLE LAW. IN NO EVENT SHALL ICETIPS'S LIABILITY FOR DAMAGES FOR ANY CAUSE WHATSOEVER, AND REGARDLESS OF THE FORM OF ACTION, EXCEED IN THE AGGREGATE THE AMOUNT OF THE PURCHASE PRICE PAID FOR THE SOFTWARE LICENSE.

14. **EXPORT CONTROLS.** You agree to comply with all export laws and restrictions and regulations of the United States or foreign agencies or authorities, and not to export or re-export the Software or any direct product thereof in violation of any such restrictions, laws or regulations, or without all necessary approvals. As applicable, each party shall obtain and bear all expenses relating to any necessary licenses and/or exemptions with respect to its own export of the Software from the U.S. Neither the Software nor the underlying information or technology may be electronically transmitted or otherwise exported or re-exported (i) into Cuba, Iran, Iraq, Libya, North Korea, Sudan, Syria or any other country subject to U.S. trade sanctions covering the Software, to individuals or entities controlled by such countries, or to nationals or residents of such countries other than nationals who are lawfully admitted permanent residents of countries not subject to such sanctions; or (ii) to anyone on the U.S. Treasury Department's list of Specially Designated Nationals and Blocked Persons or the U.S. Commerce Department's Table of Denial Orders. By downloading or using the Software, Licensee agrees to the foregoing and represents and warrants that it complies with these conditions.

15. **U.S. GOVERNMENT END-USERS.** The Software is a "commercial item," as that term is defined in 48 C.F.R. 2.101 (Oct. 1995), consisting of "commercial computer software" and "commercial computer software documentation," as such terms are used in 48 C.F.R. 12.212 (Sept. 1995). Consistent with 48 C.F.R. 12.212 and 48 C.F.R. 227.7202-1 through 227.7202-4 (June 1995), all U.S. Government End Users acquire the Software with only those rights as are granted to all other end users pursuant to the terms and conditions herein. Unpublished rights are reserved under the copyright laws of the United States.

16. **LICENSEE OUTSIDE THE U.S.** If You are located outside the U.S., then the following provisions shall apply: (i) Les parties aux presentes confirment leur volonte que cette convention de meme que tous les documents y compris tout avis qui siy rattache, soient rediges en langue anglaise (translation: "The parties confirm that this Agreement and all related documentation is and will be in the English language."); and (ii) You are responsible for complying with any local laws in your jurisdiction which might impact your right to import, export or use the Software, and You represent that You have complied with any regulations or registration procedures required by applicable law to make this license enforceable.

17. **SEVERABILITY.** If any provision of this Agreement is declared invalid or unenforceable, such provision shall be deemed modified to the extent necessary and possible to render it valid and enforceable. In any event, the unenforceability or invalidity of any provision shall not affect any other provision of this Agreement, and this Agreement shall continue in full force and effect, and be construed and enforced, as if such provision had not been included, or had been modified as above provided, as the case may be.

18. **ARBITRATION.** Except for actions to protect intellectual property rights and to enforce an arbitrator’s decision hereunder, all disputes, controversies, or claims arising out of or relating to this Agreement or a breach thereof shall be submitted to and finally resolved by arbitration under the rules of the American Arbitration Association ("AAA") then in effect. There shall be one arbitrator, and such arbitrator shall be chosen by mutual agreement of the parties in accordance with AAA rules. The arbitration shall take place in Port Angeles, Washington, USA, and may be conducted by telephone or online. The arbitrator shall apply the laws of the State of Washington, USA to all issues in dispute. The controversy or claim shall be arbitrated on an individual basis, and shall not be consolidated in any arbitration with any claim or controversy of any other party. The findings of the arbitrator shall be final and binding on the parties, and may be entered in any court of competent jurisdiction for enforcement. Enforcements of any award or judgment shall be governed by the United Nations Convention on the Recognition and Enforcement of Foreign Arbitral Awards. Should either party file an action contrary to this provision, the other party may recover attorney's fees and costs up to $1000.00.

19. **JURISDICTION AND VENUE.** The courts of Clallam county in the State of Washington, USA and the nearest U.S. District Court shall be the exclusive jurisdiction and venue for all legal proceedings that are not arbitrated under this Agreement.

20. **FORCE MAJEURE.** Neither party shall be liable for damages for any delay or failure of delivery arising out of causes beyond their reasonable control and without their fault or negligence, including, but not limited to, Acts of God, acts of civil or military authority, fires, riots, wars, embargoes, Internet disruptions, hacker attacks, or communications failures. Notwithstanding anything to the contrary contained herein, if either party is unable to perform hereunder for a period of thirty (30) consecutive days, then the other party may terminate this Agreement immediately without liability by ten (10) days written notice to the other.

21. **MISCELLANEOUS.** This Agreement constitutes the entire understanding of the parties with respect to the subject matter of this Agreement and merges all prior communications, representations, and agreements. This Agreement may be modified only by a written agreement signed by the parties. If any provision of this Agreement is held to be unenforceable for any reason, such provision shall be reformed only to the extent necessary to make it enforceable. This Agreement shall be construed under the laws of the State of Washington, USA, excluding rules regarding conflicts of law. The application the United Nations Convention of Contracts for the International Sale of Goods is expressly excluded. The parties agree that the Uniform Computer Transactions Act or any version thereof, adopted by any state, in any form ("UCITA"), shall not apply to this Agreement, and to the extent that UCITA may be applicable, the parties agree to opt out of the applicability of UCITA pursuant to the opt-out provision(s) contained therein.

5.2 7Zip license

7Zip is distributed with Build Automator. You are free to use it as you like by calling it from Build Automator. For more information about 7Zip, please visit the website at www.7-zip.org You can get more information about 7Zip as well as download the source code at the Open Source page on Sourceforge. 7-Zip is licensed under the GNU LGPL license.

The version distributed with Build Automator 4.9 is version 9.2 from November 2010. There are just two files, 7z.dll and 7z.exe, total of just about 1MB. Note that the 7z.exe is a **command line console program** only, it is NOT the full Windows version of 7Zip!

For information about the command line switches, please visit http://7zip.bugaco.com/7zip/MANUAL/syntax.htm

5.3 Online Resources

For support and information you can visit our website at www.buildautomator.com as well as our online forums at www.buildautomator.com/forum and our online blog at www.buildautomator.com/blog. If you have specific support questions you can always post on our forums. You are required to log in, but registration is completely free. We would appreciate if you used your real name so we know who you are and can contact you again:) From time to time we may post polls on our forums about what actions and features you would like to see to get a feel for what would be most appropriate for us to work on. You can also leave comments on our blog.

5.4 Data Conversions

At some points the data may need to be converted, due to changes in program and data structure. This is a list of the data conversions:

From Version 1.50.1212

This conversion is a **critical conversion**. It converts all the .PATN files that are in the project folder into the .aprj file which contains the project data. All string data is now stored in CDATA sections in the .aprj XML file and this prevents a lot of problems that we had with keeping all those files in perfect sync during operations.

What happens during this conversion?

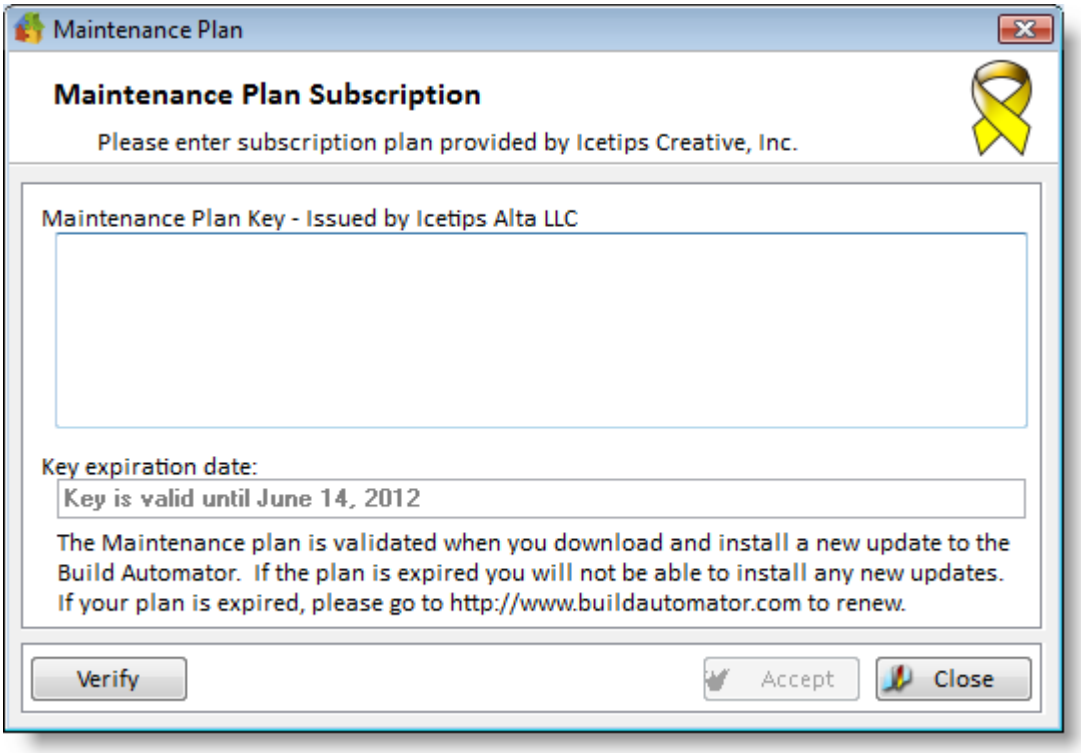
1. The .aprj file is copied to .aprj_v0 file indicating a backup file for version 0 to version 1 conversion.
2. The .avar file is copied to .avar_v0 file.
3. The .patn files in the data folder are combined into the .aprj file.

This process is NOT optional and is run when you open an existing project that hasn't been coverted and saved. If you do not save the project once it is converted it will be converted again. The process is completely automatic and no interaction is required.

5.5 Maintenance Plan

A Maintenance Plan is necessary to get new updates to the software. When you purchase the Build Automator you get a 60 day Maintenance plan included for free.

The Maintenance plan gives you free access to all updates and upgrades to the Build Automator and any plugins that you have purchased. It gives you direct access to download new updates for free during the subscription period.



A 1 year Maintenance Plan subscription is included for free with your purchase of the Build Automator. You will receive an email from Icetips Alta LLC with the Maintenance Plan Key, which you simply copy from the email and paste it into the Maintenance Plan Key text box on the Maintenance Plan window.

Note that when the program is running as demo, the word "DEMO" will appear in the Maintenance Plan Key box on the window. Once you purchase the Build Automator you will receive a Maintenance Plan Key via email, that you can paste into the Maintenance Plan Key box on the window. Until you have a valid Maintenance Plan you cannot download or install any updates to the software.

When you enter the Maintenance Plan code the code is verified immediately.

Once the Maintenance Plan expires, you will not be able to install any updates for the Build Automator. Apart from that, the software will function without any other limitation. This limitation applies to both installing the full version and also to updates from the website. Neither option will work if the Maintenance Plan is expired.

NOTE: Version 1.50.1210 or prior:
If you have entered a valid Maintenance Plan code in version 1.50.1210 or earlier, the "Key expiration date" field may be empty. Press the "Verify" button and the Maintenance Plan will be verified and the expiration date will be calculated and placed into the field.

5.6 Frequently Asked Questions

5.7 Future plans

Build Automator™

The Gold release of the 2014 Edition will be available on September 17, 2014. Included in our Maintenance Plan 148 are all updates for one year Registered users will receive emails when new releases are made available. We will also post on our forum when we are ready with new releases. The Build Automator™ can also check automatically for new updates every time it is started.

Please let us know what actions you would like to see in the upcoming releases and we will review each suggestion.

Some of the planned actions and additions for Build Automator include - in no specific order:

- Set File Attributes
- Set File date/time
- Variable pad
- Property pad
- User Tools menu
- Improved Drag and Drop support
- Support for various version control systems
- Upload files to FTP 121 server. **Included in version 1.50.1210**
- Download files from FTP server.
- Compare files in a FTP folder to a local folder.
- Compile Help and Manual projects.
- Send emails.
- Logical structures - IF/ELSE/END.
- Conditions for action execution (acts as a one action IF statement) **Included in version 1.30.100**
- Rename Files 111. **Included in version 1.10.100.**
- Delete Files 109. **Included in version 1.50.1210**
- Get version information from executable files.
- Create AutoRun.inf/AutoPlay.inf files for CD/DVD distribution or USB drive configuration.
- Create PAD files for software uploads.
- Advanced Copy action with wildcard and overwrite options if date/time/size/attributes are different between source and destination.
- Create Folders 110. **Included in version 1.10.100.**
- Delete Folders 109. **Included in version 1.50.1210**
- Zip/Unzip files/folders.
- Generate XP/Vista/Server 2008 manifest files 91. **Included in version 1.10.100.**
- Code sign executables.
- Check for code signing.
- Show code signing information from executable files.
- Set variables action to assign values returned from various functions to variables. **Included in version 1.30.100**

- Add option to check/uncheck all in Project Item list and Action Item list on the Project Window 30.
- Search locator on Variables 135 selection window. **Included in 2014**
- User formatting of variables.
- Copy/Paste variables from one project to another.
- Wait action 130 in case some programs need specific time to shut down or time out. **Included in version 1.10.100.**
- Compile multiple Clarion apps 81. **Included in version 1.30.100**
- Multi section/entry update of INI files.
- Multi key/value update of registry.
- Multi line write to log 133. **Included in version 1.10.100**
- Include projects. When included projects change, those changes are reflected in the execution of the main script. Developer Edition only.
- Action templates - selection of actions saved to a template that can be added at any point. Developer Edition only.
- Network protection of scripts so that only one user can modify a script over a network but any user could execute it. This also involves creating separate log files for each user.
- Improved protection of scripts that are opened in more than one window, by more than once instance of the program, or by more than one user.
- Scheduling of executing scripts.
- For Clarion developers:
  Register templates
  Export application to .txa
  Export dictionary to .txd
  Execute utility template
- Write one or more lines to a text files 114 with overwrite/append option. **Included in version 1.10.100.**
- Execute associate files 103 with options to OPEN, PRINT etc. **Included in version 1.10.100.**
- Write resource files for Visual Studio languages. Set the version to a variable and use that to set the version in the resource. (*See Write text to file* 114) **Included in version 1.60.1252.**
- Write resource files for Clarion, supporting Clarion 6 and older, Clarion 7 and Clarion#. Set the version to a variable and use that to set the version in the resource. (*See Write text to file* 114) **Included in version 1.60.1252.**
- Support for UNC paths for complete machine independence on networks.
- Delete Registry keys

The list goes on...

Build Automator

Part VI

Chapter 6 - Version History

6 Version History

The Build Automator has been released in the following versions:

Version 6.11.1366 - November 17, 2016

New features/changes:

July 14, 2016

- Format for Project Variable List 34 is now saved between sessions.

Fixes:

December 14, 2015

- Clarion 9.0 install was not always correctly detected in "Call MS Build" Fixed.
- If Clarion 10 was installed along with just one other Clarion build, Clarion 10 would not show up in the Clarion IDE version drop down. Fixed.

December 15, 2015

- Icons were missing from some buttons. Fixed.

January 23, 2016

- In some circumstances Write text to file 114 action with {#INCLUDE(Filename)} would erroneously add 2-5 characters at the end of the resulting file causing problems. Fixed.

February 13, 2016

- Some windows did not have the correct application icon. Fixed.

February 17, 2016

- Maintenance Plan verification was not working correctly. Fixed.
- Application icon was not set correctly so some windows did not show it. Fixed.
- The registration window was too short to show all the information properly. Fixed.

July 22, 2016

- Adding .prj files to compile in "Compile Clarion 76" and "Compile Multiple Clarion Apps 81" did not work. Fixed.

July 28, 2016

- Some windows did not resize correctly. Fixed.

November 7, 2016

- Executing a project from the command line would crash the program. Fixed.

Version 5.12.1344 - December 8, 2015

New features/changes:

August 25, 2015

- Implemented full support for Clarion 10.

October 13, 2015

- Implemented full support for Setup Builder 10.
- Build Automator is now code signed with both SHA1 and SHA2 certificate. This makes it compliant with MS Windows SHA2 code signing requirements being enforced in 2016.
- Generate Manifest 91 is now compatible with Windows 10

December 7, 2015

- Drag and Drop from Windows Explorer added to most entry and list controls that accept files or folders.

Fixes:

January 22, 2015

- On some computers the Project type drop down on Call MSBuild update did not fill in correctly. Fixed.
- Warning message was shown when a project file no longer exists, even if the project filename was empty. Fixed.

January 26, 2015

- Log file viewer was limited to 1,000,000 byte log files. Changed so that it is now completely dynamic and only limited by available memory. Fixed.

March 16, 2015

- The text for the main menu didn't always show up. Fixed.

June 16, 2015

- Compiling with Setup Builder could result in error being reported when the project compiled correctly.

August 25, 2015

- Call MSBuild did not correctly detect version 4 and 4.5 of .NET. Fixed.

September 28, 2015

- When adding a Call MS-Build 70 action, an error message would show up about the Project File not being found. Fixed.

October 13, 2015

- Resizing on some windows was not working correctly after conversion to Clarion 10. Fixed.
- Setup Builder 10 was not recognized correctly. Fixed.
- Setup Builder 10 file extension was not recognized correctly. Fixed.

November 18, 2015

- The message action window 127 did not resize correctly. Fixed.
- Multiple debug entries showing in DebugView were still showing. Fixed.

November 21, 2015

- Changing font size for project and action item lists did not change the line height in the lists. It now changes the row height and sets the width of the second column (checkbox) accordingly. Fixed.
- Generate Manifest was not compatible with Windows 10. Fixed.
- Version Resource templates were missing from build process. Fixed.

Version 4.9.1326 - September 18, 2014

New features/changes:

December 7,2013

- Added INCLUDE() syntax for the Write Text To File 114 action. By using {#INCLUDE ("FileNameToInclude")} in the text to write to file you can now include one or more files.

December 2, 2013

- Added drag & drop from windows explorer to Compile Multiple Clarion action.
- Added drag & drop from windows explorer to Copy Multiple files action list and destination entry.

November 5, 2013

- Added Debug logging. This sends ton of debug output to the default debug output and also to a log file. Best way to capture it is by using DebugView. Download DebugView from Microsoft Windows Sysinternals website.

- Verified that Copy/Paste are present in the popup menu on the Action Item list.
- Copy/Paste, Duplicate and several other issues related to positions have been fixed.

- **April 5, 2014**
- Added option to the Run Program 102 action to to run programs as Administrator/elevated. This option will prompt for administrator authorization during execution.
- Added option to cause Compile SetupBuilder 86 to terminate the script if an error occurs.
- Added [Search/Replace] buttons as well as [Remove All] button to the Search and Replace 112 action. G#26

July 13, 2014
- Added GetFileDate 56, GetFileTime 56 and GetFileSize 56 to the functions available in evaluative assignments.

August 2, 2014
- Implemented C and U to comment and uncomment single/multiple rows in the Action item list.
- Added Comment/Uncomment to the popup menu for the Action item list.

August 21, 2014
- Changes to splash window and program wallpaper.
- Added Drag/Drop from File Explorer to Copy Files 108 action.
- Changed some general visual elements.

Fixes:

March 30, 2014
- Problem with Compile Setup Builder where it could report success even though the compiler failed. Fixed. G#42

April 5, 2014
- Run Program 102 and Run Associated File 103 did not return an error if the program or file to run didn't exists. Fixed. G#44
- If no plugin DLL was found in the plugin folder, the program would end up in an endless loop. Fixed. G#5
- If a wildcard was used in Copy Multiple Files 105 it could fail in some circumstances. Fixed. G#38
- Extra button on Set Variable 94 window was removed. G#37
- File list in FTP Upload did not respond to double click and keyboard shortcuts for insert/edit/delete/ move etc. Fixed. G#23
- In some cases the wrong action update window was called when updating an action. Fixed. G#15
- If parameters in the Run Program 102 action were entered on separate lines the action could fail to execute the program correctly. Fixed. G#14
- Using Ctrl-D would sometimes duplicate the wrong action in the Action Item List. Fixed. G#4
- Duplicating an action could throw "Variable.Load called without variable filename being specified!" error. Fixed. G#3
- Using Copy/Paste in the Action Item List could be fragile. This was caused by the visual selection not always updating the selection array. Fixing this fixed several problems related to selecting from

the list, see points 7 and 9. Fixed. G#1

- In some circumstances Build Automator would hang if strings were not enclosed in double quotes or if double percent (%%) was detected. Fixed. G#10
- If a project file (.aprj) was copied but not the variable file (.avar) Build Automator would open the project without warning. Fixed. (G#25)

May 12, 2014

- Generating Clarion 9 solutions would fail. Fixed. (W#1083)

August 3, 2014

- Various paths and registry keys were still named "Icetips Creative" They have now all be changed to "Icetips"
- Installer modified to copy files and registry settings based on name changes above.
- Program code all changed to reflect changes to file and registry name changes above.

August 21, 2014

- If a Setup Builder project file did not exist, Setup Builder would not report an error. Fixed.
- Minor memory leak could occur in the Copy Multiple files 105 action after dragging and dropping files onto the folder entry. Fixed.
- If a Setup Builder project didn't exist no error was reported. Fixed
- Return value from Compile with Setup Builder 86 was not always reliable. Fixed.

August 23, 2014

- If a Solution or Project file no longer exists, there was no warning in the "Call MS-Build" action. Fixed.
- Call MS-Build did not fully support Clarion 9.1. Fixed.
- Call MS-Build did not support /ConfigDir option for Clarion. Fixed. NOTE: This is ONLY supported in Clarion 9.1 and later.

August 31, 2014

- Project information variables, i.e. those that started with $_ were not being evaluated at all. Fixed.
- Debugging was not activated in the variable parsing part of the environment. Fixed.

September 3, 2014

- Under certain circumstances return value from Compile with Setup Builder 86 was not correct. This was related to item 21 above. Fixed
- $ToDay$, Now and $ExistCode$ were being inserted in upper case. Fixed.

Version 2.0 was released into beta testing in 2012.

New features/changes:

October 21, 2011

- Variable handling has been completely rewritten to fix vulnerabilities in variable expansion.
- Select Variables 135 window now also has Environment Variables.
- Added a locator to Select Variables 135 window.
- Added a button to Copy Files 108 action to paste destinations directly into the queue. If a single line is added, it is opened in the edit window after adding it to the list.
- Implemented Ctrl-V to paste a destination into the destination list in the Copy Files 108 action. If a single line is added, it is opened in the edit window after adding it to the list.
- Implemented Shift-Delete to delete ALL destinations in the Copy Files 108 action.
- **Warning message about Delete from lists now defaults to Yes** in Copy Files 108 action. Warning message to Delete ALL entries from listboxes defaults to No.
- Implemented Shift-Delete to delete ALL files in the Copy Multiple Files 105 action.
- Implemented Ctrl-V to paste a file list from the clipboard into the file list in Copy Multiple Files 105 action.
- **Warning message about Delete from lists now defaults to Yes** in Copy Multiple Files 105 action. Warning message to Delete ALL entries from listboxes defaults to No.
- Improved logging for Copy Files 108 action.
- Implemented locator on picklist. It works on the filename only (not the path) and will find any occurance of the text in the filename. I.e. "my" will be found in "myfile.aprj" and "ismyfile.aprj" The picklist is filtered so it is easy to find files. A "Clear" button returns the list to the original. Use Ctrl-F to jump to the locator entry. Use Ctrl-R to remove the locator.

January 24, 2012

- Added Ctrl-R to execute selected item(s) without opening the log file viewer 48 at the end. This makes it easier to step through scripts while debugging them without having the log window open at the end of each one. Ctrl-E works as before executing the item(s) and then opening the log file viewer 48.

April 1, 2012

- Added option to copy selected project action items to the clipboard as text. Implemented Ctrl-T as hot key for it.

Fixes:

June 17, 2011

- Call MSBuild 70 could list "DF" as .NET version. Fixed.
- Call MSBuild 70 listed .NET versions that did not have MSBuild installed. Fixed.
- Compile Clarion 76 allowed the user to select an app file for Clarion 7 or Clarion 8. Fixed.
- Compile Multiple Clarion Applications 81 allowed the user to select an app file for Clarion 7 or Clarion 8. Fixed.
- Text on Compile SetupBuilder 86 action mention "SetupBuilder 5 or 6" Fixed.
- Text on Compile Clarion 76 and Compile Multiple Clarion Applications 81 was running into "Currently running" text with information about currently running version of Clarion. Fixed.

October 21, 2011

- Variable expansion was not always reliable and would fail in some lists, such as "Copy Multiple Files" Fixed.
- Build Automator exe and plug-in dll,BuilAuto1.dll, used two different windows to select variables. Combined to use one window. Fixed.

October 22, 2011

- Insert destination folders to Copy Files 108 action allowed duplicate entries. Fixed.
- Copy Files 108 did not have Insert/Enter/Delete keys active to edit the list. Fixed.
- Execute Selected button in Project window executed only one item. Ctrl-E and Execute from popup menu worked as expected. Fixed.
- CHM documentation was not being compiled and distributed.
- PDF documentation was excluded from web update.

November 3, 2011

- Deleting a file from the Picklist 11 did not refresh list correctly. Fixed.
- The Picklist window 11 caused problem with window list menu. Fixed.
- Build date and time was not updated correctly during build process and not shown correctly on the splash screen 23.
- When locator on Picklist was accepted the focus did not return to the list. Fixed.

November 4, 2011

- Cancel button on Variables window 34 did not move horizontally when window was resized. Fixed.
- Selection on Variables list on Variables window 34 was not activated when clicking on the list with the right mouse button. Fixed.

November 5, 2011

- Server update form was inconsistent between the IDE and the FTP Server update. Fixed.
- The format of Project Item list and Action Item list was not completely consistent. Fixed.
- Action tree did not load with the top node visible. Fixed.
- Build Automator would crash if the version of Setup Builder indicated in the Compile Setup Builder 86 action didn't exist on the machine where the Build Automator project was opened on. Fixed.

November 17, 2011

- Call MSBuild 70 action would incorrectly indicate that any project or solution would be a Clarion project or solution. Fixed.
- Call MSBuild 70 action would incorrectly indicate that any solution was a Clarion solution just if Clarion 7 or 8 was installed. Fixed.
- Call MSBuild 70 action did not correctly determine if a solution files was a Clarion solution. Fixed.
- Call MSBuild 70 could not specify what Clarion version compiler to use to compile the project. I.e. compiling using Clarion 6.3 compiler in Clarion 8 was not possible. Fixed.

November 18, 2011

- Call MSBuild 70 would show versions of Clarion.NET available when compiling in Clarion. Removed.
- Call MSBuild 70 would not indicate error or warning lines inside the included MSBuild log. Fixed.
- Compile with SetupBuilder 86 sometimes causes a failed compile. Reason is unknown and does not appear with any other program. Implemented a 1 second timer delay before and after the Compile with SetupBuilder 86 action executes. Hope fully that will resolve this.

November 19, 2011 - Build 2.0.1300

- Further to adding a timer to Compile with SetupBuilder 86, we have added enumeration of the SB7 processes before and after the call to SetupBuilder. It has been experiences that the SB7.exe process is left in memory for unknown reason. If the number of SB7.EXE processes before and after does not match an error is added to the Build Automator log. Additional debug statement left in there for beta testing (visible in DebugView)
- Call MSBuild 70 would not generate log file if "Log to Build Automator log" was unchecked and could cause the Call MSBuild 70 action to erroneously report a failure. Fixed.

November 19, 2011

- When opening a second or subsequent Project windows the new window would sometimes open under the last active project. Fixed.

November 21, 2011

- Added option to Compile SetupBuilder 86 to terminate the execution of the Build Automator project if an error occurs.
- Added an option to Compile SetupBuilder 86 to include the SetupBuilder log into the Build Automator log. If an error occurs during a SetupBuilder compile, Build Automator will add the SetupBuilder error log into the Build Automator log if this is checked. If it is not checked, Build Automator will only add lines with "ERROR" or "WARNING"

March 27, 2012

- Variables could not be created in a new project. Fixed.
- Project information variables were not always created properly. Fixed.

Version 1.70.1277 BETA 9 - June 14, 2011

New features/changes:

- Logfile viewer now can be filtered to show errors, warnings, information or all. Color coding of errors/warnings etc. also implemented to make errors more visual.
- Servers can now contain both FTP and SMTP servers in preparation for emailing actions and options in the Build Automator.

- Run Program 102 and Run File 103 action: parameters can now be broken up into lines to make them easier to edit. The linebreak is interpreted as a non character, i.e. it is replaced with nothing before passing the parameters to the run action.
- Run Program 102 and Run File 103 update windows is now resizable. This, along with item 3, makes it much easier to construct long parameter lists.
- Option to minimize the program to the system tray when executing. **Rejected.**
- The project script can now be printed.
- Project is no longer stored in multiple files. The project is now stored in the .aprj file and the .avar files for the project and the variables. The program automatically converts the projects to the new format.
- Execution can now be stopped much more easily. In the previous version it was very difficult to break the execution loop and stop the script if needed. This new build starts a new thread for each execution and the main loop waits for each execution to finish before moving on. This makes the execution window much more responsive.
- Added global variables that contain information about the project file. Create date, time, last update date, time. Project path, filename and full path + filename. Also the short path+filename for the project.
- Call MS-Build 70 now supports compiling Clarion 7 solutions and projects.
- Get Version Information action added.
- CompareVersionStrings function added.
- Import file list to "Copy multiple files 105" option added
- Error/warning list from Setup Builder is now imported into the Build Automator log.

Fixes:

- The Help would not open on the correct topic for the Terminate Script action 68 update window. Fixed.
- Target file was not automatically added to the "Compile Clarion 76" action. Fixed.
- Under some circumstances the "Compile Clarion 76" action could hang without running the Clarion IDE. Fixed.
- "User Registraion" was misspelled on the "Enter Registration Key" window. Fixed.
- "Create Folders" action did not report folder names that it could not create. Fixed.
- "Create Folders" action would sometimes not create folder(s). We have discovered that the API that we use does not support periods in the folder names that it creates. For now we are classifying this as **limitation** and we will get back to it later. The action will log each such folder into the logfile as an ERROR.
- In some cases newly created action item would not be saved correctly. Fixed
- Regression: Copying and pasting action items from one project item to another would fail in some cases. Fixed
- In the Run Program action 102 update window, tabbing through controls on the window was fairly random. Fixed.
- FTP Server update window had a typo in the titlebar - "Upate" instead of "Update" Fixed.
- Run Program 102 action did not show parameters in the Action Items list 40. Fixed
- "Execute All / Execute Project" option in Project Window was no longer used. Removed.
- The execution progress window had some overlapping text. Fixed.

- Typo in log entry for Message action "butten" instead of "button". Fixed.
- Regression: "Stop on Error" option in Compile Clarion 76 and Compile multiple Clarion apps 81 actions did not work after changes in logic to retrieve error messages from the Clarion compiler window. Fixed
- Hotkeys were not implemented on Search and Replace 112 action. Fixed
- Parameters in Run File are not active and not used. Fixed.
- Installer did not use internal integrity check which meant that download could be corrupt and the installer would not detect it. Fixed.
- Picklist did not sort properly. It now lists the last opened project on the top and the projects are in the correct opening order. Fixed.
- Conversion of Server.tps to new format which now supports email servers would fail. Fixed.
- "Enter Registration Key 29" window was not documented at all. Fixed.
- When the Armadillo keycode is entered, the program needs to be restarted for it to KNOW that it is no longer a demo program. 2008-08-20: Added code to restart the program. Fixed.
- Inserting Project Item would always push the bottom project item down making adding multiple project items literally backwards. Fixed.
- When action items were selected and a new project item was selected the selection was not cleared. Fixed.
- When copying action items from one project to another the variables used in the action items were not created in the target project. Fixed.
- The Create Folders 110 action would sometimes not create the folder. We have discovered that the windows API that we use to create the folders does not support periods in the folder name. For now we are classifying this a **limitation** and we will get back to it as soon as we can.
- Copying and pasting action items from one project item to another would fail in some cases. Fixed.
- Sometimes the execute checkbox and the background color of the line number and execute column would not be drawn correctly when inserting a new action item. Fixed.
- Deleting multiple action items did not work. Fixed.
- If system variables, such as $ExitCode$ was selected in the variable list, it was not possible to insert a new variable. Fixed.
- Executing multiple action items was not possible. Fixed.
- FTP Uploads would timeout after 90 seconds. Fixed.
- Regression - action items that were not checked to execute would stop the script execution. Fixed.
- Variables in Destination in the Copy Multiple Files 105 action could cause invalid paths to be created. Fixed.
- Logging for Copy Multiple Files 105 was too verbose. Fixed.
- In some circumstances expanded variables could contain trailing spaces. This could cause the invalid paths described in item 34. Fixed.
- Total execution time in the Project Log was formatted as seconds and fractions of a second rather than hh:mm:ss.nn. Fixed.
- Project window 30 documentation showed old images for the toolbar execution buttons. **Item in review.**
- "Allow full Execute" defaulted to being unchecked when creating a new project. This was confusing. Fixed.
- If no Action Items 40 had been added to a Project Item 39, execution could stop on the empty

project item. Fixed.

- Search and Replace 112 does not report error if it cannot write the file back to disk. Fixed.
- Search and Replace 112 did not list the replace string in the log file, only the search string. Fixed.
- Search and Replace 112 did not add to the logfile if the search string was not found in a file. Fixed.
- Create Folders 110 would not create directories with period (.) in the foldernames. Fixed.
- Create Folders 110 did not create the start directory if it didn't exists. Fixed.
- There was no option to save the log file to a new file. Fixed.
- Adding files to the Compile Multiple Clarion apps 81 action didn't add the files in the correct order. Fixed. If the last item is selected the new item is added last. If other item is selected the new item is added after it.
- Target filename was not retrieved from the application when adding applications to Compile Multiple Clarion apps 81. Fixed.
- Target filename was not retrieved from the application when adding applications to Compile Clarion 76. Fixed.
- Deleting multiple project action items did not work correctly. Warning message only warned about "this action item" Fixed.
- Double clicking on a Search and Replace 112 action item would not open the update window. Fixed.
- If the project window was bigger than the program window, the program window could scroll and hide the top parts of the project window. Fixed.
- Statusbar still said "2008 Edition" Fixed.

Fixes for Beta 3:
Note that case numbers are now noted with fixes to reported bugs.

- **Case #64**: If project file was read-only it would show a rather cryptic message not detailing what the problem was and aborted opening the project. Fixed for Beta 3,
- **Case #62 & 65**: If one or more project items were unchecked the execution would stop. Fixed for Beta 3.
- **Case #66**: The Stop button on the Compile multiple Clarion apps 81 action did not work properly. Fixed for Beta 3,
- **Case #63**: When the "Stop with message on fatal errors" in the Compile Clarion 76 and Compile Multiple Clarion Apps 81 is checked the script terminates without an option to continue. Fixed for Beta 3.
- It was not possible to select a source folder in the Copy Files 108 action, only a file. Fixed for Beta 3.
- Source file entry on Copy Files 108 did not resize correctly. Fixed for Beta 3,
- **Case #60:** Wildcard handling in Source file entry on Copy Files 108 was not correct. Fixed for Beta 3.
- When selecting a single application file for Compile Multiple Clarion apps using the Add button, the target file was not always retrieved correctly. Rejected: Problem is in the FileDialog which sometimes shifts the filenames in the file entry for unknown reasons.
- Added a window to enter/paste multiple application names for Compile Multiple Clarion Apps 76. Application order is preserved when importing.
- Target filenames were still not always retrieved from the application in Compile Clarion 76 and Compile Multiple Clarion Apps 81. The reason was that in some cases there can be more than one project section in the application file. Fixed for Beta 3.

Fixes for Beta 4:

- Call MS-Build 70 window would jump when opened. Fixed for Beta 4.
- Call MS-Build 70 did not compile Clarion 7 solutions or projects. Fixed for Beta 4.
- Internal: Action Editor was not handling calling the update correctly. Fixed for Beta 4.
- Create Shortcut button did not have tooltip. Fixed for Beta 4.
- **Case #70:** When executing a project from shortcut the logfile window was different. Fixed for Beta 4.
- Clarion.NET projects (*.cnproj files) were not available in the Call MS-Build 70 action. Fixed for Beta 4.
- Call MS-Build 70 did not compile Clarion.NET solutions or projects. Fixed for Beta 4.
- Option to minimize the program when executing. This also minimizes all programs called from the Build Automator. This still has issues with focus being grabbed during execution. This is not finished for Beta 4.
- Option to "Run Minimized" added to Run File 103, Run Armadillo 93, Run Program 102, Call MS-Build 70, Compile SetupBuilder 86, Compile Inno 78, Compile Setup Factory 89 and Compile MSI Factory 90 actions.
- New function added to the SetVariable 94 action - **FileExists** - that can be used to check if file or folder exists. The function returns 1 if the file or folder exists and 0 if it does not exist.

Fixes for Beta 5:

- New function added to the SetVariable action - ExpandEnv - that expands an environment variable, such as %PATH%. NOTE: The environment variable MUST be passed without the surrounding percentage characters ("%") or it will be treated as a Build Automator variable! So to expand %PATH% use ExpandEnv("PATH")
- Call MS-Build would not get correct .NET folder for versions above 2.0.
  Note that this requires you to go into each MS-Build action update and select the correct version again.
  Fixed.
- Call MS-Build did not log an error if it could not find MS-Build in the version folder specified. Fixed.
- New action to create version resource files for Clarion and Visual C++
- Copy Files 108 window did not have tooltips on the "Select Folder" and "Select File" buttons. Fixed.
- If "Select Folder" button Copy Files was used and the filename ended in *.* the *.* would be added again if the same folder was selected so the entry would end up with *.**.* etc. Fixed.
- Added *.sb7 file mask to the Compile SetupBuilder 86 action.
- SetVariable 94 update window did not open the help in context. Fixed.
- **Case #64**: The Search Replace in Textfiles 112 window did not resize properly. Fixed.
- The Create Folders 110 window did not resize properly. Fixed.
- Create Folders 110 action could fail at certain depth of recursion through the folders to create. Fixed
- Create Folders 110 action was limiting the data for the folders to create to 2K when executing and 10K when updating. Both increased to 100K. Fixed.

Fixes for Beta 6:

- Copy Multiple Files 105 did not support wildcards. Fixed.
- Compile Setup Builder 86 action window did not have a delete button to remove variables. Fixed.
- Project Item conditions did not work correctly. Fixed.
- Project Item conditions caused execution to stop. Fixed.
- The "Select Folder" option on the Delete Files 109 action did not save/restore last used location. Fixed.
- It was not clear that the Delete Files 109 action did not remove the parent folder. Fixed on window and in help.
- Copying multiple files with wildcard would copy the same files multiple times if there was more than one line with a wildcard in it. Fixed.
- Regression in FTP in Beta 6 caused it to fail. Fixed.
- FTP did not support wildcards in filenames. Fixed.
- View Files to delete 109 window did not expand variables in path name. Fixed.
- If text that could contain variables, i.e. %VAR% or VAR contained % or $ that were not part of variable names, the results could be unpredictable. For example if a parameter was being sent to a program with the Run Program action, that contained something like %p@G()$ in the parameters this text could be removed causing problems. Fixed.
- Template files for VC++ and Clarion version resource are not included in install. Fixed

Fixes for Beta 7:

- Redundant debug information was being sent to the log file from E_Search_and_Replace_in_Textfiles. Fixed.
- Clarion errors in log viewer were not color coded as errors. Fixed.
- Copying and pasting action items sometimes fails. Can no longer be reproduced despite numerous efforts. Will tackle it if it comes up again, but I'm calling it fixed.

1. Main window sometimes appears outside of the visible area of the monitor(s). Fixed.

Fixes for Beta 8:

- Copy Multiple Files action does not expand variables correctly. Fixed.
- Log File viewer had a one byte out of bounds memory problem. Fixed.

Fixes for Beta 9:

June 10, 2011

- MS-Build action modified for more flexibility.
- MS-Build action can now handle both Clarion 7 and Clarion 8.

June 11, 2011

- MS-Build action can now handle code generation in Clarion 8. Note that this is only enabled for Clarion 8 if a solution is selected. If a Clarion project file (*.cwproj) is selected, then it is available for Clarion 7 also, but this will not work in the Clarion 7.0 - Clarion 7.2 builds.
- Added Copy, Paste and Duplicate to the pop up menu on the Action Items list 40.
- Implemented a Duplicate option for the Action Items list 40. It can be activated by using the popup menu or by using Ctrl-D.

- Implemented Alt-Enter on Action Items list 40 to open the Action Items Properties 43 window.
- Hotkeys are now visible on popup menu on the Action Items list 40.

June 12, 2011

- Added Windows 7 compatibility to "Generate XP/Vista/Win7 Manifests 91" action.
- Added option to exclude dependency to "Generate XP/Vista/Win7 Manifests 91" action.
- Implemented Ctrl-E on Action Items list 40 to execute the selected action item(s).

Version 1.50.1212 - August 18, 2008

New features/changes:

- Added "Terminate Script Execution" action. Can be used with a condition to conditionally terminate the execution of a script.

Fixes:

- Maintenance Plan 148 was not verified when it was entered and the expiration date was not filled in. This could cause confusion if the plan had been correctly updated or not. Fixed.
- Vendor action file for "Compile Multiple Clarion applications 81" accidentally dropped from install. Fixed.
- Text at bottom on the update window for "Compile Clarion 76" action was cut off with certain screen resolutions. Fixed.
- Changing a Project item 39 name did not trigger the Save button to become enabled. Fixed.
- Regression in 1.50.1210, see fixed item number 7: Under some circumstances the Action item list 40 could get corrupt and only show one Action item in other Project items. This only happened in project with multiple project items when the project was saved. Fixed.
- Regression in 1.50.1210, see fixed item number 11: Adding a variable to an empty entry would fail. Fixed.
- Web update would not acknowledge change in minor version number. Fixed. Minor version changed from 5 to 50.
- Checking or unchecking "Show Splash Screen at startup" on the About window 23 was not saved. Fixed.
- It was not possible to scroll the picklist contents horizontally. Fixed.

Version 1.5.1210 - August 5, 2008

New features/changes:

- Version labeling changed. Previously we used 4 components in the versions, now we use 3, major version number, minor version number and build number.
- Added option to show various buttons on the "Message 127" Action.
- Added option to set a default button on the "Message 127" Action.
- Added option to save clicked button to a variable on the "Message 127" Action. This variable can

later be used in conditions.

- Error logging added to "Copy Files (Simple) 108" and "Copy Multiple Files 105" actions when creating folders.
- Logging of folders created added to "Copy Files (Simple) 108" and "Copy Multiple Files 105" actions.
- Added option to "Project Properties 33" to Explore the project folder.
- Logging has been improved so that it shows each line number being executed (as 001-0001 for project item 1, action item 1 or 004-0095 for project item 4, action item 95) along with the same information as appears in the Action item list 40. Action item conditions are also shown and if they passed or failed.
- Delphi 2007 (.dproj) compiling implemented in "Call MS Build 70" action.
- "Call DLL 100" action implemented.
- "FTP Upload 121" action implemented.
- Errors from Clarion compiles 76 are now added to the logfile so they are easy to track down.
- Action Editor button removed from toolbar and menu item from main menu.
- "Delete Files 109" action implemented.

Fixes:

- The "Run File" action did not have any default settings. Set to "Open" and "Wait for Program to Finish" by default. Fixed.
- Under rare conditions the line numbers for the Action Items would get messed up. Fixed.
- Last folder in folder structure being copied would not always be copied. Fixed.
- Source folders with no files in them were no created when copying. Fixed.
- Adding variables at the very end of text in an entry field or text field would sometimes fail when new text had been entered. Fixed.
- If Action Items containing "Set Variable" Actions were copied from one project to the other, the variable would not be created. Fixed.
- $EXITCODE$ as System variable was not reliable. It is now created as part of the Project Variables as an "Internal Variable" meaning it cannot be modified or deleted. Fixed.
- When compiling multiple Clarion Apps, if one of the apps failed it erroneously was reported succeeding and the **next** app failed. Fixed.

Version 1.30.150 - June 24, 2008

New features:

- Search and Replace option added to the "Compile Multiple Clarion apps" action.
- Select Variable is now available in the BASupport.dll for anyone to use in their plugins.
- Added /NL command line parameter to tell the software not to open the log after the end of a script execution. This caused problems if the project was run as part of a scheduled task or in a batch file because the Build Automator would wait for the View Log window to be closed. See fix number 15 below.

Fixes:

- Variable lookup was not available in the update window for "Compile Multiple Clarion apps" action. Fixed.
- Variable lookup was not available in Search/Replace window. Fixed.
- A variable could not be selected with the keyboard in the "Select Variable" window. Fixed.
- Copying Action Items from one project to another would sometimes fail. Fixed.
- Variables in "Compile Multiple Clarion apps" action were not expanded. Fixed.
- Select Variables in BASupport was not loading local variables. Fixed.
- If a variable was entered using %VarName% into variable drop-downs the trailing % was duplicated into %VarName%%. Fixed.
- If a Clarion app file was marked as read-only the compile process would not detect that it had finished compiling. Fixed.
- Project Window would sometimes not resize properly. Some "jerkyness" was observed when resizing the window horizontally (see also item 37 in version 1.30.100 169). Fixed.
- Changing Project properties 33 to a Demo project did not allow it to be changed back to normal project and nothing alerted the user that checking the "Demo project" was not reversible, making it possible to accidentally create a Demo project that could no longer be executed. Fixed.
- Folder for Vendor Action was erroneously set to CSIDL_COMMON_APPDATA instead of CSIDL_LOCAL_APPDATA. This caused newly inserted Actions not to show up until the program was restarted. Fixed.
- Vendor file was not created when a new name was typed in on the *Update Action File* window. It was only created when the "Update" button was pressed. Fixed.
- Right clicking on the Action Item list would not select the correct item from the list. Fixed.
- Install was not time stamped during code signing. Fixed.
- Running a script from the command line would stop at the end because of the View Log window. Fixed.
- When no items where in the Action Item list it was still possible to use the popup menu and some of the toolbar buttons that should have been disabled. Fixed.
- GetPluginFolder and GetActionFolder in Automator.dll did not return the correct paths after latest update in default paths. Fixed.
- Checking "Always start Picklist 11 at startup" option on the Picklist window 11 was not being saved properly. Fixed.
- In some cases application filenames for Clarion apps would be upper cased after they were compiled. This only affected filenames that were 8 characters long or shorter. Fixed.
- Copy - Paste failed in the same project. Regression in fix for item number 4 above. Fixed.
- When compiling Clarion apps the log file would always indicate that the .app file was read-only, even when it was not. Fixed.
- When inserting a new Project Item, the Action Item list was not immediately cleared. Fixed.
- Pasting Action items into a Project Item with no Action items did not work. Fixed.

Version 1.30.100 - June 3, 2008

Please see the Known Limitations 176 chapter for any limitations that may affect the Build Automator

New features:

- "Set Variable 94" action implemented.
- Font setting 17 for Project Items 39, Action Items 40 and Actions 42 on Project Window 30 can now be changed.
- "Save As" 7 added.
- Option to create shortcut 44 to the project in a selected destination folder. The shortcut can be set up to open the project or execute the project.
- File association option added in installer.
- Changes to command line parameters 22.
- Condition option on Action Items Properties 43 window is now active.
- Maintenance Plan 148 window is now active. Maintenance Plan keys will be sent out to all customers after this release and subsequent releases will require a valid Maintenance Plan.
- Search/Replace on "Copy Multiple Files 105" action.
- Data lookup option for "Set Variable 94" action. Option to select file, folder, color (Clarion LONG color value or HTML color) etc.
- Standard Edition and Developer Editions combined for the time being. We may separate them later on, but for now the Build Automator is only available in one edition that includes everything that was planned for the Developer Edition.

Fixes:

- Web update was not working reliably under Vista when called to get information about a new install. Fixed.
- Web update was requesting elevation under XP as well as Vista. Fixed.
- Clarion compiles 76 would not always work reliably and would report errors to log. Fixed.
- Clarion compile 76 action would not start the IDE specified if it was not already running. Fixed.
- Call MS-Build 70 did not log the actual command line. Fixed.
- "Write Text To File 114" limited the file length to 2048 bytes. Fixed.
- "Write to Text File 114" update window did not resize correctly. Fixed.
- When adding variables to "Write Text To File 114" the text may be truncated. Fixed.
- Variables were not being saved after a change in internal build 1.20.210. Fixed.
- If the "Save" button on Variables 135 window was pressed while variable was being entered or updated, the active variable was not saved. Fixed.
- Expanding variables in content that was shorter than the variable name would fail. Fixed.
- When selecting variable 94 it was not possible to use the keyboard to switch between project variables and system variables. Fixed.
- When a project item is deleted, the NEXT project item was deleted. Fixed.
- When a Project Item is created and is not at the end of the Project Item list, the Action Items were not cleared. Fixed.
- When a Project Item was created the Action Items for the next Project Item could get attached to the new item. Fixed.
- When a new project was created no default Project Item was created. Fixed.

- When project log file opened during execution, it would cause an error message about "No application active". Fixed.
- Removing a single project from the Picklist did not work. Fixed.
- Text at the bottom of the "Call Inno Setup Compiler" window was being cut off at the bottom. Fixed.
- When executing a single item, if there were multiple items above it in the Project Item, there could be a delay before the item was executed. Fixed.
- It was not possible to select a project to open from the picklist by using the keyboard. Fixed.
- Picklist was sorted in alphabetical order which was not appropriate. Fixed - now sorted in most recent on top, oldest at the bottom.
- Installer used two dialog windows to prompt for "Create Desktop Icon" and "Create Quick Launch Icon" Fixed - placed on one dialog window.
- Verbiage for entering the registration key in the Help menu 7 changed from "Enter Registry Key" to "Enter Registration Key". Fixed.
- Unchecked Project Items 39 could make the execution loop stop and not finish. Fixed.
- "Compile Setup Builder 86" action did not log down results. Fixed.
- Save button on Action Item Properties 43 window was not active. Fixed.
- If a single Action item was executed and that Action Item was set to not execute or had condition, the script would continue to run. Fixed.
- Apps Key was not implemented for popup menus on Project Item 39 and Action Item 40 lists. Fixed.
- If Project execution was started before midnight and ended after midnight, the time reported for the total execution time was wrong. Fixed.
- Font settings on certain control types were not always appropriate when using Small Fonts (96DPI). Fixed.
- Startup window would save and restore position rather than always being centered above the Build Automator main window. Fixed.
- Opening a project that was already open allowed both to be updated. Fixed. Now only the first opened project can be modified. Any subsequent opening of that project while it is still open are now opened as read-only where no modifications are allowed.
- When using the "Copy Multiple Files" action it was not possible to edit the source filename. Fixed.
- When using the "Copy Multiple Files" action and a source filename contained a variable, it was not expanded. Fixed.
- On the "Copy Files - Simple" action pressing the Enter key did not open the Edit window. Fixed.
- Splitter on Project window 30 would not resize the Action Item 40 list when sized horizontally, rather resize the Action Tree 42. This caused problems with the Action Tree 42 when the window was resized down to be narrower. Fixed.
- Opening in-use Project 15 file did not allow for opening or unlocking a file that was marked in use because it hadn't been saved properly. Fixed.

Unconfirmed problems:

- When double clicking on the Action tree it would open the update window on the last updated action item and not create a new Action Item. We have only had one report about this and we have not been able to duplicate this problem.

Version 1.20.200 - May 15, 2008

This version sees most of the Developer Edition development in the IDE finished.

New features:

- *Action Editor* separated from the main IDE. This completes the separation of the Standard and Developer Editions
- Dynamic loading of plug-in DLLs completed. All actions are now treated as plugins including the standard Build Automator actions. This allows execution of third party plugin actions.
- Dynamic loading of plugin DLL from *Action Editor* completed. This was needed for creation of *Vendor Action files*.
- New Action: Compile multiple Clarion 81 apps. This action is only available in the Developer Edition. It lets you pick multiple Clarion apps to compile.
- Resizing added to several action update windows that use text boxes or listboxes: Compile Setup Builder, Comments, Copy Files, Copy Multiple Files, Create Folders, Message, Write lines to Logfile, Write text to a file.
- Plug-in DLLs are now loaded when the program starts. This reduces the time each action takes to execute as before the DLLs where loaded only when the actions were called either to update or execute.
- Vendor file and Actions are now moved automatically to the user application data folder instead of being shared. This allows writing to the action folder without elevating the program when it's running under Vista, which is not desirable. The mechanism for this will be improved in the coming weeks.
- Splitters are now complete and functional on the Project Window 30.

Fixes:

- When an action item was inserted and then canceled an extra confirmation message popped up asking if the action item should be deleted. Fixed.
- Folder location was not being saved/restored: Added to following actions: Compiling multiple Clarion apps, Call MS Build (manual selection of MS Build), Compile single Clarion app, Compile Inno, Compile Setup Builder (Project file), Copy Multiple Files, Create Folders, Generate Manifest, Get data from INI, Rename File, Run Armadillo (Project and file to protect), Run Associated File, Run Program, Update INI and Write Text to file.
- Save button on Project Variables 34 window did not close the window. Fixed.
- Adding Variables to Compile Setup Builder would only accept ONE variable.
- Project Item GUID was being created with curly braces. Fixed.
- Text at the bottom of the update window for "Compile Inno" was cut off in some cases. Fixed.
- "Compile Clarion" did not close after compile was done (new method). Fixed
- "Compile Multiple Clarion apps" did not advance between apps or close after a compile was done (new method). Fixed.
- Move up/down buttons on Project Window behaved erratically and could potentially corrupt script. Fixed.
- Move up/down buttons did not work with the Project Items list, only the Action Items. Fixed.
- The "Compile Setup Builder" window as too big to fit on small resolution monitors. Fixed.
- While Setup Builder was compiling there is no interaction. While we cannot fix this since it's inside Setup Builder, we have added a window that shows during the Setup Builder compile process with "Please wait..." Setup Builder 7 will have a progress bar that shows during the compile process. Please Note: If your Setup Builder script uses FTP to upload the new build, it can take a while to

finish and return to the Build Automator.

- On the update window for "Compile Setup Builder" if the variable was being edited and the Save button (not Apply) button was pressed, the variable was not saved. Fixed.
- Data that the Action Editor needs to access was being installed into common data causing permission problems. Program now checks and copies this data to CSIDL_LOCAL_APPDATA \Icetips Creative\Build Automator when the program starts. Fixed.
- Save Button on update form for multiple clarion compiles did not have appropriate icon. Fixed.
- The **/E** command line parameter 22 was not working properly. Fixed.

1. Move up/down buttons on Project window caused potential corruption problem in the Project Item list if the item was moved both up and down. Fixed.

Version 1.10.100 - May 2, 2008

Gold release for the Standard Edition, Beta 2 release for the Developer Edition.

Added since Initial release, Beta 1:

- Added action to Run a file with associated program with optional verbs to open, print, etc.
- Added hot key (Ctrl-Shift-F12) to all action windows that shows the GUID for the vendor action and optionally copies the GUID to the clipboard.
- Added action to write text to a file. The text is either appended to an existing file or overwrites an existing file.
- Added action to generate XP or Vista Manifest files.
- Added action to wait for predetermined number of seconds (ranging from 0.01 to 999.99 seconds).
- Added action to Rename a single file. Renaming multiple files with wildcards etc. will be added in May.
- Added action to write multiple lines to the logfile.
- Added window to edit entries in the folder list in Copy Files (Simple) This gives you an option to use variables such as current date in the destination folder - great for incremental daily backups!
- Added a simple Message action. It will wait until the OK button is pressed. You can set the caption text, the message text and select an icon from a dropdown.

Fixes since Initial release, Beta 1:

- Standard buttons were just a bit too narrow. Changed the standard button width in the entire program from 40 dialog units to 45. Fixed.
- Save buttons did not have icons. Fixed.
- Key to show the variable selection window was set to the down key. Didn't work so well in text boxes! Set to Ctrl-Down key. Fixed.
- When Project Variables list is empty the System Variables should be automatically selected. Fixed.
- Under some circumstances the correct GUID was not retrieved in the *Action Editor*. Fixed.
- If an insert was attempted for an action, where the vendor action file didn't exist, anywhere except at the end of the Action item list it could cause the linenumber order to fail and subsequent loadings of the project to halt at that line number. Fixed.
- Variables were not expanded in the Source for Copy Files (Simple) 108. Fixed.
- The Build Automator would crash on dual/multi-core machines. Fixed.

- The Compile SetupBuilder 86 action did not update a changed variable name. Fixed.
- File names in the error log for Copy Files (simple) included trailing spaces. Fixed.

Version 1.00.000 - April 24, 2008

Initial release.

Build Automator

Part VII

Chapter 7 - Known Limitations

7 Known Limitations

This section applies to the latest public release, version 5.12

- 5.11: Drag and drop from Windows Explorer to some windows that have file or folder entries may not work. This should **not** affect any of the action windows and only happens if the entries are on tabs. This is a bug in the Clarion runtime library that we have not been able to work around so far.
- UNC filenames are not specifically supported.
- The check if a project that is being opened is already in use does not currently work across networks. Therefore it is possible for two users to open the same project and make changes to it at the same time. Concurrency checks are **not** applied by the Build Automator so it is possible that both users could save the project and overwrite each other's modifications.
- Setup Builder: There was a bug in Setup Builder versions prior to Setup Builder 6.5 build 2000 making it impossible to reliably set the output file (EXENAME).

Index

- A -

- B -

- C -

- D -

- E -

- F -

- G -

- H -

- I -

- K -

- L -

- M -

- N -

- O -

- P -

- Q -

- R -

- S -

- T -

- U -

- V -

- W -

- X -

- Y -

- Z -