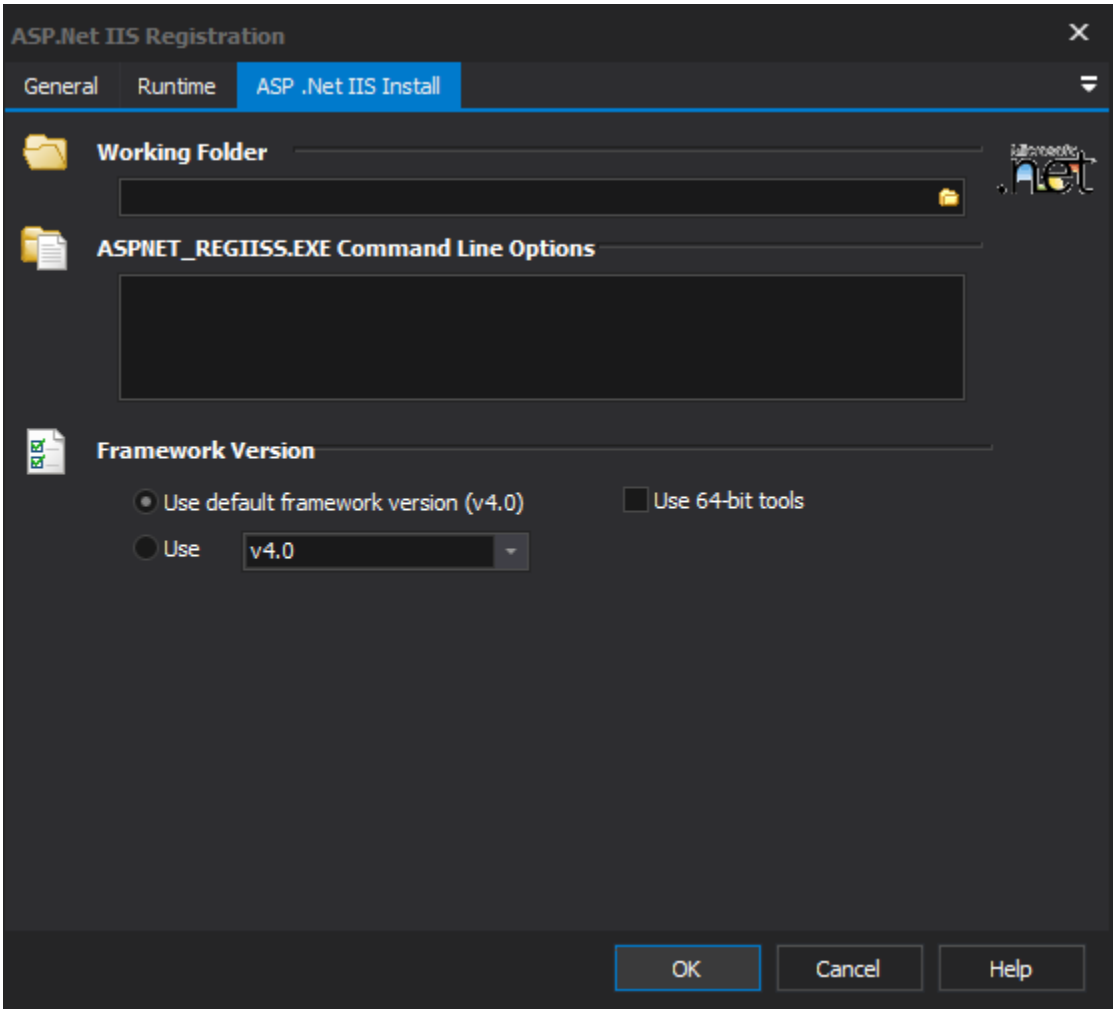
ASP.NET IIS Registration (ASPNET_REGIIS.EXE) Action

The ASP.NET IIS Registration action allows the registration or removal of ASP.NET applications within IIS. It can also create ApplicationPools or get the status of the installed ASP.NET framework.

ASP.NET IIS Registration



On this page:

Working Folder

The folder from which the ASPNET_REGIIS.EXE command will be run. This allows for easier referencing of source files in the AL command through relative paths.

ASPNET_REGIIS.EXE Command line options

The parameters to pass to ASPNET_REGIIS.EXE. This can take the following form:

- `[options]`

Due to the complexity of this tool and the number of options available we currently allow direct access to the command arguments. These can be found on the MSDN website located here ASPNET_REGIIS.EXE.

Framework Version

Use default framework version (vX.X) / Use vX.X

Allows the selection of the .NET version to use for the tool. The minimum is .NET v2.0.

Use 64-bit tools

This options specifies as whether to force the usage of 64-bit tools on a 64-bit system or not. As FinalBuilder is currently a 32-bit application it will default to running the 32-bit version on a 64-bit system. Check this option if you require the 64-bit version to be run.

Action Scripting

Toggle All | Expand All | Collapse All

Action Specific Scripting

Other

Script Property	Example (Javascript)	Valid Values	Default	Persisted	Description
`Options`	`Action.Options = '-s W3SVC/1/ROOT /WebsiteA';`	any text	''	On save	Specifies the parameters to pass to ASPNET_REGIIS. EXE.
`StartDir`	`Action.StartDir = 'C:\WebSiteA\';`	any text	''	On save	Specifies the working directory for ASPNET_REGIIS. EXE to be called with.

Common Scripting Properties

Behaviour

Script Property	Example (Javascript)	Valid Values	Default	Persisted	Description
`Enabled`	`Action. Enabled = false;`	***true***, ***false***	true	On save	Whether the action is enabled or disabled. Disabled actions are not considered part of a script. When the script is run they are ignore completely. Has to be set prior to the action being selected to run. This means the latest this can be set is in the actions parent.
`IgnoreFailure`	`Action. IgnoreFailure = true;`	***true***, ***false***	false	On save	When set to true, the action will always report as having successfully completed. The actions run result is ignored. Even if the action run result is failure, the build will continue. Has to be set before the action is run to have any affect.
`PauseInterval`	`Action. PauseInterval = 2;`	***0*** to ***18000 000*** (5 hours)	0	On save	The number of milliseconds to pause after the completion of an action. The IDE allows users to skip pause intervals if required. The command line runner does not allow for the skipping of this interval. Setting this is preferable to sleeping in a script as the action continues to be responsive, and handles termination requests. Has to be set prior to the action run to have any affect.
`MaxRetryAttempts`	`Action. MaxRetryAttempts = 3;`	any positive ***32-bit integer***	0	On save	The number of times to retry an action when it fails. Each failed run of the action is counted as a retry attempt. Once all the retry attempts are exhausted the action will return the result of the last retry attempt. On a successful run this counter is reset for the next time the action is called. This is important for loops which contain actions with retry attempts set. Has to be set prior to the run of the action to have any affect.
`RetryPauseInterval`	`Action. RetryPauseInterval = 500;`	any positive ***32-bit integer***	1000	On save	The number of milliseconds to wait before running a retry of an action. This starts counting directly after the action fails and is not able to be skipped. Useful for waiting resources to become available, or locks to be released. Has to be set before to the run of the action to have any affect.

Description

Script Property	Example (Javascript)	Valid Values	Default	Persisted	Description
Comment	`Action.Comment = 'Loads config for build';`	***any text*** (single line shown)	[blank]	On save	Allows you to add documentation to the action instance.
Description	`Action.Description = 'Upload [ Installer ]';`	***any text*** (single line shown)	[action dependent]	On save	The text shown in the IDE for the action. Describes the purpose of the action. Clear this property to revert to an automatically generated description.

Identity

Script Property	Example (Javascript)	Valid Values	Default	Persisted	Description
ActionName	`Action.SendLogMessage (Action.ActionName, stInformation);`	***any text (read only)***	[action dependant]	No	The name of the action which is shown in the action types list. This is defined by the action and is not able to be changed. All actions of the same type will have the same name.
Package	`Action.SendLogMessage (Action.Package, stInformation);`	***any text (read only)***	[action dependant]	No	The filename of the package from which the action was loaded. All actions from the same package will have the same package filename.

Logging

Script Property	Example (Javascript)	Valid Values	Default	Persisted	Description
ActionLogTitle	`Action. ActionLogTitle = 'Upload [ Installer ]';`	***any text*** (single line shown)	Action. Description	On save	Sets a different description for the action for logging purposes. This description is only ever used in the log.
ExpandActionLogTitle	`Action. ExpandActionLogTitle = false;`	***true***, ***false***	true	On save	Enables variables in the ActionLogTitle to be expanded. The expansion occurs at the time of logging.
HideActionFromLog	`Action. HideActionFromLog = true;`	***true***, ***false***	false	On save	Hides the action from the log. If the action execution results in an error the action is logged, effectively ignoring this setting.
LogActionProperties	`Action. LogActionProperties = true;`	***true***, ***false***	false	On save	Records the properties of the action to the log before the action is run.
LogToVariable	`Action. LogToVariable = 'MyVariable';`	***text name of variable***	[blank]	On save	Specifies which variable should have the output of the action written to it. The selected variable is required to be available to the action, otherwise an error will be raised. The variable will contain the actions output after the action has run.
SuppressStatusMessages	`Action. SuppressStatusMessages = true;`	***true***, ***false***	false	On save	Stops the logging of all the actions status messages. This stops the action status messages from being generated, so OnStatusMessage events will not fire when this options is turned on.

Other

Script Property	Example (Javascript)	Valid Values	Default	Persisted	Description

ErrorCount	`if (Action.ErrorCount > 0)`
	
	`  Action. SendLogMessage('There were errors', stError);`
		any positive ***32-bit integer (read only)***	0	No	Returns the number of errors the action has encountered during its run. Some actions can encounter more than one error before failing.

Errors	`Action.SendLogMessage ('Errors encountered' + Action.Errors, stError);`	***any text (read only)***	[blank]	No	Returns the description of all the errors encounter by the action during it run. The error descriptions are concatenated with new lines between each entry.
Locked	`Action.Locked = true;`	***true***, ***false***	false	On save	Locks the properties on the action so that they can not be changed through the action dialog. When turned on, the only property which can be altered is the locked property itself. Turning this off again means all other properties can be altered through the actions edit dialog.

TimedOut	if (Action.TimedOut) Action. SendLogMessage('Action timed out', stWarning);	***true***, ***false*** ***(read only)***	false	No	Set to true when the action has timed out waiting for the underlying tool to complete.

Execute Action Scripting Properties

Extra Command Line

Script Property	Example (Javascript)	Valid Values	Default	Persisted	Description
ExtraCmdLineParamsAtEnd	Action. ExtraCmdLineParamsAtEnd = '\s \p Text.txt';	***any text*** (validated by underlying tool)	[blank]	On save	Specifies additional command line parameters for the underlying tool. These are added after all command line parameters added by the action.
ExtraCmdLineParamsAtStart	Action. ExtraCmdLineParamsAtStart = '\q \ignore:3';	***any text*** (validated by underlying tool)	[blank]	On save	Specifies additional command line parameters for the underlying tool. These are added before all command line parameters added by the action.

Process

Script Property	Example (Javascript)	Valid Values	Default	Persisted	Description
UseErrorDialogMonitor	Action. UseErrorDialogMonitor = true;	***true***, ***false***	false	On save	Specifies whether to automatically watch for an error dialog which has stalled the underlying tool. The dialog will be closed, and if this is not possible the process will be terminated. This option does not work with all underlying tools.
ProcessPriority	Action. ProcessPriority = tpBelowNormal;	***tpIdle***, ***tpBelowNormal***, ***tpNormal***, ***tpAboveNormal***	tpNormal	On save	Specifies at what priority processes spawned by the action will have. Note that setting an idle priority will most likely mean the action will not progress. tpIdle is only included for completeness, typically it should not be used.
ProcessorAffinity	Action. ProcessorAffinity = 243;	***0*** (***unset***), ***1 to 255*** (***bit mask***)	0	On save	Specifies which processors spawned processes of the action will be allowed to run on. The value is in the form of a bit mask. The mask depends on the number of processors available on the running machine. A bit mask of 1101 1111 (= 223) for a six processor machine would mean spawned processes could run all but the 6th processor.

Run As User

Script Property	Example (Javascript)	Valid Values	Default	Persisted	Description
ImpersonateUserName	Action. ImpersonateUserName = 'domain/username';	[domain]/[username]	[blank]	On save	Specifies the user credentials with which to run processes spawned by the action. The user that FinalBuilder is running under, will require the permissions to elevate their rights and impersonate other users for this to be allowed. ImpersonateUser needs to be set to true for this to be used.
ImpersonateUseNetCredOnly	ActionImpersonateUseNetCredOnly = true;	***true***, ***false***	false	On save	Specifies whether the impersonation should only use the network component of the supplied users credentials. This allows for simpler access to network resources which have restricted access. ImpersonateUser needs to be set to true for this to be used.
ImpersonateUser	Action. ImpersonateUser = true;	***true***, ***false***	false	On save	Specifies whether processes spawned by the action should be run under a different set of user credentials.
ExpandImpersonateToken	Action. ExpandImpersonateToken = true;	***true***, ***false***	false	On save	Specifies whether to expand variables found within the impersonate user password. This is handy for setting the password to a variable and then loading from a secure file, or using a passed in variable when the script is run.
ImpersonateUserPassword	Action. ImpersonateUserPassword = '8as0dk9JLa!df';	***any text***	[blank]	On save	Specifies the password to use for the user who is being impersonated. Once set the value is encrypted for saving into the project (if its saved after this script is called).

Timeout

Script Property	Example (Javascript)	Valid Values	Default	Persisted	Description

`EnableTimeout`	`Action.EnableTimeout = true;`	***true***, ***false***	false	On save	Specifies whether the action should fail if the TimeoutLength expires. The TimeoutLength is counted from the start of the action running.
`TimeoutLength`	`Action.TimeoutLength = 2;`	any positive ***32-bit integer***	1	On save	Specifies the number of minutes to wait after the start of an action to complete. If the action does not complete before the timeout length has expired the action is given a failure result, and the TimedOut property is set to true.

.NET Scripting Properties

Other

Script Property	Example (Javascript)	Valid Values	Default	Persisted	Description
`OverrideSDKDefault`	`Action.OverrideSDKDefault = true;`	***true***, ***false***	false	On save	Specifies whether the default SDK version set in the FinalBuilder options should be overridden for this action. Setting this to true will mean the SDK version will be taken from the SDKVersion property on the action.
`Architecture`	`Action.Architecture = ta32;`	***ta32***, ***ta64***	ta32	On save	Whether to use the 32-bit or 64-bit versions of the underlying tool. Note that some tools can not be run as the 64-bit version from a 32-bit process. As a result these tools will either report an error or simply link to the 32-bit version. Check each tools MSDN reference for more details.
`SDKVersion`	`Action.SDKVersion = "v4.0";`	***v2.0***, ***v3.0***, ***v3.5***, ***v4.0***, or any other installed .NET version	*set in options*	On save	Sets the overridden SDK version for the underlying tool. The specified SDK version needs to be installed on the machine otherwise an error will be raised saying the specific tool could not be found.

Scripting Events

Script Event	Parameters		Description
BeforeAction	`Action : <TAction>`	The instance of the current action. Allows access to the action properties and methods. All properties are set to the values provided by the action editor or their defaults.	Called before the action is executed. When called all properties on the action have been initialised to those provided in the action editor dialog. Properties which have not been provided are set to their defaults. Use this event to change anything about the action or perform operations which need to occur before the action is run. Also this event can be used to skip an action entirely. If the action is skipped it will report as such in the log, and no more processing of the action or its scripts will occur. Run-time errors in this script will stop the action from running and the action will report as failed.
	`SkipAction : Boolean`	An out parameter which allows for the action to be skipped during a build. Default is ***false***.	

<table>
<tr>
<td>After Action</td>
<td>
<table>
<tr><td><code>Action : <TAction></code></td><td>The instance of the current action. Allows access to the action properties and methods.

All properties are set to the values used during the run of the action.</td></tr>
<tr><td><code>ActionResult : boolean</code></td><td>Indicates if the action succeeded or failed. Allows for the handling or overriding of the actions status during a build.</td></tr>
<tr><td><code>Continue : boolean</code></td><td>Indicates if the build should continue after this action has completed or not.

Return false to stop the build, return true to ignore the build result and continue the build. Default is unset.</td></tr>
</table>
</td>
<td>Called after the action has executed. All properties on the action will be the same as when the action was run. Any properties which change during the run will be available at this point (e.g. Exit codes properties).

This event allows for the handling of action error states that where are not directly handled by the action itself. For example if an certain error code is acceptable it can be logged and ignored through this event.

Conversely a certain successful run condition that is not desirable could be reported through this event. To achieve these outcomes set the ActionResult and Continue parameters to values which reflect the true outcome of the actions run.

Run-time errors produced by this script will cause the action to report as failed. Setting of either the ActionResult or Continue parameters will not override this.</td>
</tr>
<tr>
<td>OnStatusMessage</td>
<td>
<table>
<tr><td><code>Action : <TAction></code></td><td>The instance of the current action. Allows access to the action properties and methods.

All properties are set to values used during the run of the action.</td></tr>
<tr><td><code>StatusMessage : TStatusMessage</code></td><td>The status message object contains all information relating to the message being logged (Lines, MessageText, MessageTitle, and Progress).</td></tr>
</table>
</td>
<td>Called whenever the action generates a log message.

Actions will generate this event when the action has received output from the underlying tool, or when the action itself has something to report. The StatusMessage may contain zero-to-many lines of text and is in the format directly seen in the log.

As formatting depends on the tool being used we advise using the RegExp function to parse the output if required. Also please review the TString object type for more information on how to access the contents of the messages.

Run-time errors produced by this script will cause the action to report as failed.</td>
</tr>
</table>