Variables

As you promote your application through test, UAT, staging and production, there are going to be differences in:

- Database connection strings
- Application settings
- Web service URLs
- Many other parameters

To make it easy to support different environments without hard-coding these configuration values, the Deployment Manager web interface allows you to create *variables*.

These variables are used during your application deployment. For example, variables you define will be automatically substituted into XML configuration files, and made available to your PowerShell scripts. This allows you to create applications and deployment scripts that are agnostic of the target environment.

Escaping

The value of a variable should be the unescaped version. Quotes and other characters that would normally need to be escaped in XML should not be escaped in the Deployment Manager web interface. This is because they will be escaped when injected, and doing so could result in double escaping.

Contents

Scoping

A variable can be 'scoped' to zero or one of each of the following:

- An environment (most common)
- A specific target
- A specific package

For example, suppose these variables exist:

Name	Value	Environment	Target	Step
LogLevel	*Info*	All	All	All
LogLevel	*Warn*	Production	All	All
DBConnectionString	*Server=SQL-UAT1;Database=...*	UAT	All	All
DBConnectionString	*Server=SQL-PROD;Database=...*	Production	All	All

During deployment, Deployment Manager will try to select the most specifically scoped variable that applies. For example, when deploying to Production, the LogLevel property would be Warn. But to any other environment, it would fall back to the less-specific variable and have a value of Info instead.

Predefined variables

Deployment Manager automatically makes a number of common useful variables available during each deployment. All predefined variables have a *RedGate* prefix. The available predefined variables are:

Name	Example	Description
RedGateEnvironmentName	*Production*	The name of the environment that was selected when this deployment was scheduled.
RedGateMachineName	*ProdServer01*	The name assigned to the machine in the Deployment Manager Environments page in the web interface. May or may not be the hostname of the actual machine.
RedGateReleaseNumber	*1.9*	The release number entered when this release was created in the Deployment Manager web interface.
RedGatePackageName	*AcmeCorp.Web*	The name of the NuGet package being deployed.

RedGatePackageVersion	*1.9.3*	The version number of the NuGet package.
RedGatePackageNameAndVersion	*AcmeCorp.Web.1.9.3*	The above two values combined.
RedGateProjectName	*ACME Website*	The name of the current project, as defined in the Deployment Manager web interface.
RedGateTaskId	*157*	An integer used to identity the current task in Deployment Manager. You can use this if you need to record something unique about each deployment.
RedGateDeploymentAgentVersion	*2.0.0.0*	The version number of the Agent service.
RedGateReleasedBy	*Administrator*	The name of the user who created the release.
RedGateDeployedBy	*Administrator*	The name of the user who deployed the release.
RedGateProjectDescription	*My project description*	The description given when creating the project.
RedGateReleaseNotes	*My release notes*	The release notes added when creating the release.
RedGateDeploymentComments	*My deployment comments*	The comments added at time of deployment.
RedGateMaxConcurrentTargets	*5*	Limits the maximum number of targets that can concurrently be deployed to within a step. If this variable isn't set, the number of concurrent targets that a step can deploy to is unlimited. Deploying to a large number of databases without setting this variable may cause performance problems.

The following additional variables don't have a default value, but can be set by you to override Deployment Manager behavior:

Name	**Example**	**Description**
RedGatePackageDirectoryPath	*C:\MyApp*	A custom path that you can tell an Agent to extract your package to. If not set, it will have the value that an Agent extracted the NuGet package to.
RedGatePurgePackageDirectoryBeforeCopy	*False*	If you're using `RedGatePackageDirectoryPath`, set this variable to *True* to clean the target directory before copying.

IIS website variables

You can use the following variables when deploying an IIS website:

Name	**Example**	**Description**
RedGateWebSiteName	*MySite* or: *MySite/MyVDir*	The IIS site name or virtual directory to use when creating or updating an IIS website.
RedGateNotAWebSite	*True*	Set to *True* to skip the IIS convention (don't configure any IIS sites).
RedGateCreateWebSiteOnPort	*8080*	The port number to use when creating a new IIS website.
RedGateCreateWebSiteApplicationPool	*MySite*	The name of the application pool to create. You can only use this variable with IIS 7.
RedGateCreateWebSiteDotNetVersion	*v4.0*	The .NET Framework version to use for the application pool. You can only use this variable with IIS 7.

RedGateCreateWebSiteIdentityType	*ApplicationPoolIdentity*	The identity type to use for the application pool. Allowed values: • LocalSystem • LocalService • NetworkService • ApplicationPoolIdentity For more information, see Application Pool Identities (IIS documentation). Specific user identity is not currently supported. You can only use this variable with IIS 7.

For more information, see Deploying IIS websites.

Database variables

To control how Deployment manager deploys databases, for example by excluding certain database objects like users, see Database deployment options.

The following variables are set when you add a SQL Server to an environment through the web interface. You can use the variable names below to reference them in other steps:

Name	Example value	Description
RedGateDatabaseServer	*localhost*	The database server you want to deploy to.
RedGateDatabaseIntegratedAuthentication	*True*	If you want to use integrated authentication to connect to the SQL Server, specify the value *True*. If you use integrated authentication, you do not need to specify a username and password.
RedGateDatabaseUsername	*user*	The username for the database you want to deploy to.
RedGateDatabasePassword	*password*	The password for the database you want to deploy to.

The following variables are set when you configure a database package step through the web interface. You can use the variable names below to reference them in other steps:

Name	Example value	Description
RedGateDatabaseName	*WidgetShopStaging*	The database you want to deploy to.
RedGateDatabaseMode	*upgrade* *dropandcreatenew*	The type of upgrade you want Deployment Manager to perform. You can specify either *upgrade* if you want to upgrade the database, or *dropandcreatenew* if you want to drop and recreate the database.
RedGateDatabaseAllowDynamicUpgrade	*True*	If you want Deployment Manager to perform a dynamic database upgrade, specify *True*. See How Deployment Manager upgrades databases.
RedGateDatabaseAllowPreDeployValidation	*True*	If you want Deployment Manager to validate the upgrade before deployment, specify *True*. See Deployment validation.
RedGateDatabaseAllowPostDeployValidation	*True*	If you want Deployment Manager to validate the upgrade after deployment, specify *True*. See Deployment validation.

RedGateDatabaseAbortOnWarningLevel	*none* *high* *medium* *low* *info*	To control when Deployment Manager aborts a deployment if there are any warnings, set one of the following options: • *none* - never abort the deployment • *high* - abort only on high severity warnings • *medium* - abort on medium severity warnings or higher • *low*- abort on low severity warnings or higher • *info* - abort on any warnings, including informational messages. If this option is not set, it defaults to high.

For more information, see Deploying database packages.

Database deployment options

Database deployment options are variables that control how Deployment manager deploys databases, for example by excluding certain database objects like users or indexes. The variables have the prefix, *RedGateDatabaseOptions* and can be set to *true* or *false*.

For more information about database deployment options, including a list of variables and their descriptions, see Database deployment options.

If you don't set any database deployment options, Deployment Manager applies a number of default options. For more information, see Default database deployment options.

Referencing variables

Referencing variables within the same scope

To dynamically update a value on deployment, you can reference a variable using `$(variable)` notation. The variable is evaluated on deployment.

You can create custom variables and reference them, or you can reference built-in variables, such as the predefined variables, database variables and IIS website variables.

This will reference variables which are inside your current scope. To reference variables outside the current scope but inside the same project, see Referencing variables from a different step or target.

Examples

- When adding a database package step, the **Database name** is set to $(RedGatePackageName) by default. The database name will evaluate to the package name on deployment.
- You can specify a value for your package directory path that includes the package version number. This means on every deployment, the package will be extracted in a new folder with the package version number as its name.

Syntax

To reference a variable, specify `$(variable)`

To include the literal characters `$(` in a variable without referencing another variable, specify `$$(`

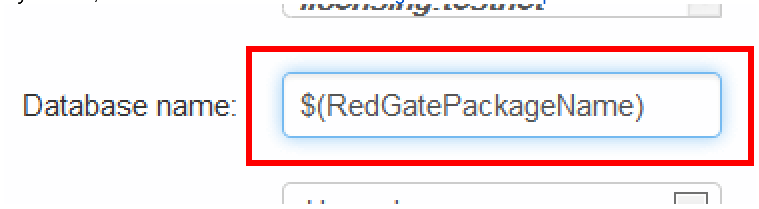
This is interpreted as a literal `$(` on deployment.

Syntax Examples

- To set the package directory path so packages are extracted in a folder for each version, set the variable RedGatePackageDirectoryPath to:

```
C:\MyApp\$(RedGatePackageVersion)
```

- By default, the database name when creating a database step is set to:



Database name: $(RedGatePackageName)

so the name for the database is the same as the package name.

Referencing variables from a different step or target

To reference a variable from a different step or target, you can use `::` notation inside the `$(variable)` notation. The step and target referenced must be in the same project.

Syntax

To reference a variable in a different step, specify `$(step::variable)`

To reference a variable in a different target, specify `$(step::`*target*`::variable)`, where *target* is the literal string target rather than its name.

This only references one target so the it must be the only one in the environment you are deploying to. If you deploy to more than one target, the reference won't be evaluated and the deployment will fail.

Example use case

We're deploying a web application that incorporates a website deployment step, called *WidgetWebsite*, and a database deployment step, called *WidgetDatabase*. The website needs to access the database, so the website step needs to use the user name and password from the database step.

In the website step, we create two custom variables:

Variable name	Value
DatabaseUsername	`$(WidgetDatabase::target::RedGateDatabasePassword)`
DatabasePassword	`$(WidgetDatabase::target::RedGateDatabaseUsername)`

The WidgetWebsite deployment step can now pass these values to the website so it can access the database. The database deployments most only contain one target per environment or the deployment will fail.