FTP Build and Deployment: Development Guide

# Document History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Changes** | **Changed By** |
| 15/06/2012 | 0.1 | Initial draft started. | Rik Locke |
| 03/08/2012 | 0.2 | More draft contents completed | Rik Locke |
| 15/09/2012 | 1.0 | Updates to reflect enhancements to build definition templates. | Rik Locke |
| 03/04/2013 | 1.1 | Reviewed prior to Handover of DevInt automated refresh by Dev teams | Rik Locke |

# Document Overview

This document is intended to provide details about how to manage FTP projects in build and deployment terms.

It provides details on changes required when adding to FTP solutions and projects and how to maintain deployments for changing rig configurations.

# Related Documents

This document supports and extends the “FTP Automated Team Functional Build & Deployment Guide” and presumes familiarity with that document.

Contents

[Document History 1](#_Toc335648769)

[Document Overview 1](#_Toc335648770)

[1 Process Overview 3](#_Toc335648771)

[2 Automated Build & Deployment 4](#_Toc335648772)

[3 Manual Deployment 7](#_Toc335648773)

[3.1 Manual Rig Spinning 7](#_Toc335648774)

[3.2 Get appropriate code version of deployment scripts 7](#_Toc335648775)

[3.3 Setup-DeploymentFromConfig 7](#_Toc335648776)

[3.4 Deploy-RigFromConfig 7](#_Toc335648777)

[4 Post-Deployment Testing 5](#_Toc335648778)

[4.1 Deployment Verification Test (all rigs) 5](#_Toc335648779)

[4.1.1 Manual Test Execution (mstest) 5](#_Toc335648780)

[4.1.2 Manual Test Execution (console app) 6](#_Toc335648781)

[4.2 End to End Integration Smoke Test 6](#_Toc335648782)

[5 Release 7](#_Toc335648783)

[5.1 Release Notes 9](#_Toc335648784)

[6 Deployment Configuration 10](#_Toc335648785)

[6.1 Configuration 10](#_Toc335648786)

[6.2 Machine 10](#_Toc335648787)

[6.3 Server Roles 11](#_Toc335648788)

[6.3.1 ServiceDeploy 11](#_Toc335648789)

[6.3.2 WebDeploy 12](#_Toc335648790)

[6.3.3 IISSetup 13](#_Toc335648791)

[6.3.4 FileSystem 14](#_Toc335648792)

[6.4 Database Roles 14](#_Toc335648793)

[6.4.1 FromConfig 15](#_Toc335648794)

[6.4.2 AspNetMembership 15](#_Toc335648795)

[6.4.3 ServerDeploymentAccount 16](#_Toc335648796)

[6.5 Common Roles 16](#_Toc335648797)

[7 Application Configuration 17](#_Toc335648798)

[7.1 Database Projects 17](#_Toc335648799)

[7.1.1 Publish profiles 17](#_Toc335648800)

[7.2 Pre and Post-Deployment Scripts 17](#_Toc335648801)

[7.2.1 File Inclusion 17](#_Toc335648802)

[7.2.2 SqlCmd Variables 18](#_Toc335648803)

[7.3 App.config (Windows Services) 18](#_Toc335648804)

[7.4 Web.config (Web applications) 18](#_Toc335648805)

[7.5 Adding a Project to a Solution 18](#_Toc335648806)

[7.6 Adding SqlCmd Variables to DB Projects 19](#_Toc335648807)

[7.7 Adding WCF Service to Project 19](#_Toc335648808)

# Process Overview

The overall process for build with automated deployments is covered by the accompanying guide but is summarised as follows:-

* Build
  + Software Build ->Build Output directory (TFS Build Definition)
  + Spin up LabManager Rig (Deploy-LabManager.ps1)
  + Output Directory -> Deployment Server (Setup-Deployment.ps1)
* Deploy
  + Deployment Server -> Propagation in Rig (Deploy-Rig.ps1)
    - Web Server Setup (TFL.IISSetup.ps1)
    - Web Site and Service Deployment (TFL.WebDeploy.ps1)
    - Windows Service Deployment (TFL.ServiceDeploy.ps1)
    - Database Deployment (TFL.DataDeploy.ps1)
    - File System setup (TFL.FileSystem.ps1)
    - MSI Installation (TFL.MISDeploy.ps1)
* Test
* Release

# Automated Build & Deployment

Build Definitions and running builds are covered in the FTP Team Build Guide.

Some aspects relevant to development are;

Workspace:

If the solution contains references or dependencies outside of the existing workspace, these will need to be added here.

Remember to also update the workspace if the branch being built has changed. (Note any testsettings references will also have to be updated).

Deploying to existing rigs:

Remove the LabManager Script reference in the build definition, Process section 7.

Change the -LabManagerConfigName parameter from '$BuildNumber' to the name of the rig you wish to deploy to.

*Note: copying names from the browser control panel often introduces spaces at the start and/or end of the name which cause a failure to deploy).*

Building without deploying software:

Removing the Deployment Script reference to Setup-Dployment.ps1 will stop the automatic deployment of software after a build.

# Post-Deployment Testing

At present, automated post-deployment and smoke testing is executed (kicked off) manually.

A development VM is used to run the Common\Main\Code\Integrate visual studio solution which connects to the target integration rig by virtue of host names which need to be entered into the development VM's hosts file (as the dev VM's sit in the FAE domain and the rigs are in their own FAELAB domains).

There are two automated testing options for new rigs; Deployment Verification Tests and End 2 End Smoke Tests.

## Deployment Verification Test (all rigs)

The Deploy and Build solution contains a TestRigFromConfig test which determines what tests are required based on the configuration file used to deploy software on a rig.

### Manual Test Execution (mstest)

In Visual Studio, open the \FTPDev\Common\Main\Code\Integrate.sln solution.

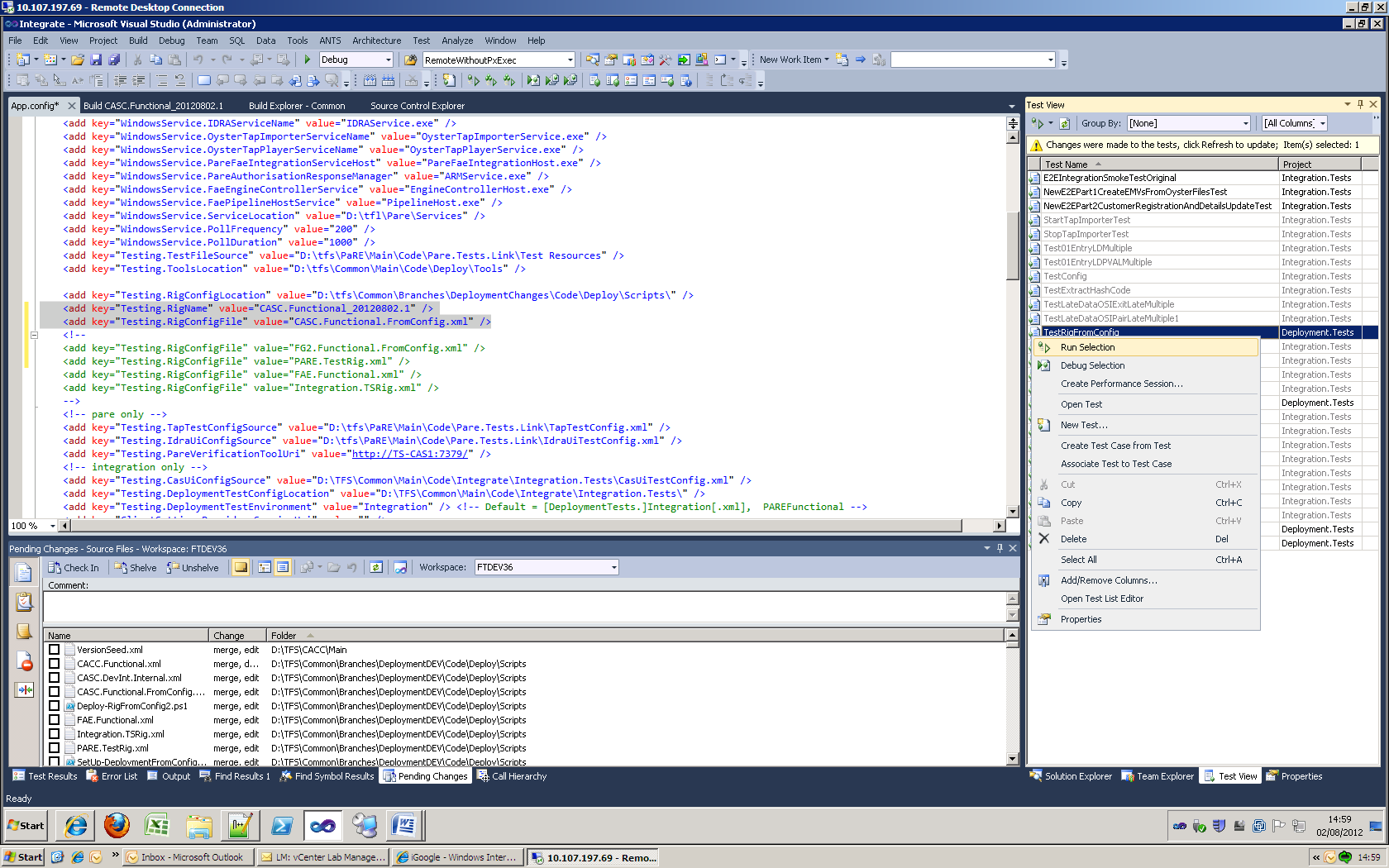
Update the app.config app settings:

<add key="Testing.RigName"value="CASC.Functional\_20120802.1" />

<add key="Testing.RigConfigFile"value="CASC.Functional.FromConfig.xml" />

From the Test (menu) select Windows > Test View.

Run the "TestRigFromConfig" test.



### Manual Test Execution (console app)

The Deployment Verification Tests (DVT's) have been factored out into an abject model which represents the environments and deployment configs. This enables the same tests to be run as mstests (see above) or independently of TFS/VS.

The utility to run independent tests is called 'DeploymentTool.exe' and is stored under Common\Main\ and developed under Common\DeploymentChanges) in a project of the same name.

Syntax;

DeploymentTool.exe -Type 'Pre/Post/PostLab' -ConfigFile 'c:\text.xml' -BuildLocation 'c:\Binaries' [-RigName 'MyRig']

Arguments;

-Type

There are three options for running the deployment tool.

Pre: Run pre deployment validation checks on the deployment 'package'.

Post: Run post deployment validation. Assumed the process the running 'in the domain' that the config is deployed to.

PostLab: Run post deployment checks on a LabManager rig.

-ConfigFile

Specifies the name of the xml rig config file.

-BuildLocation

The location of the source software build .

-RigName

If the RigName parameter is omitted then the utility will presume that it's being run within the target rig and therefore that machine names in the config file are resolvable without looking up the external IP addresses from LabManager.

If type is Post and not PostLab then this argument is not required.

## End to End Integration Smoke Test

In the same way as for post-deployment tests, run the E2EIntegrationSmokeTestOriginal.

There are currently some new E2E smoke being developed which will supersede the original one.

# Manual Deployment

Rigs are typically created and deployed to as part of an automated build process. The

This section details how to produce a rig manually should that be required.

## Manual Rig Spinning

1. Navigate to LabManager in browser;

https://tdc2vlm001.onelondon.tfl.local/LabManager/ControlPanel/Main.aspx

1. Navigate to Library (left hand menu under Build and Deploy)
2. Choose a suitable rig template; typically of the form "<team>.Functional.SQL2012Full" or "IntegrationRig" for cross-team builds.
3. Select 'Clone to Workspace...' from the context menu.

* Workspace: TFSWorkspace
* Name: clearly
* Select machines required (usually all)

1. **Change Owner: to TFSBuild**

This is important as automated deployments and tests need tfsbuild to be the owner to allow access to the configuration to acquire external IP addresses via the LabManager API.

## Get appropriate code version of deployment scripts

Deployment files:

D:\tfs\Common\Main\Code\Deploy\Scripts

Code to be deployed:

Copy build output directory to development server. This is most easily accessed by clicking the 'Open build folder' link in the TFS.

## Setup-DeploymentFromConfig

cmd prompt >

powershell D:\tfs\Common\Main\Code\Deploy\Scripts\Setup-DeploymentFromConfig.ps1

-RigRelativePath'Integration.TSRig.xml'

-DropFolder'D:\<copy of build output on machine this is running on>'

-LabManagerConfigName'Integration Rig - TS17\_20120413.5'

-DeployRig 1

*Note:Drop folder is absolute path on the local machine and without the trailing backslash.*

## Deploy-RigFromConfig

Corresponds to the step executed when -DeployRig 1 is specified in the build definition.

This step can is not required if the Setup-Deployment is run with DeployRig 1. Software is not copied to the deployment server so any changes to the build will not be re-deployed.

i) Connect to Deployment Server (typically TS-DB1)

cmd prompt >powershell D:\deploy\scripts\Deploy-RigFromConfig.ps1 -RigRelativePath 'Integration.TSRig.xml'

This can be run from a development VM via PsExec by prefixing the above command with

psExec.exe \\ts-cas1 -u FAELAB\tfsbuild -p LMTF$Bu1ld <above powershell command>

# Release

When automated builds, that create new rigs, are successfully completed the build process will automatically undeploy and move the rig to the Functional workspace in LabManager.

Ad-hoc development and testing rigs can be left in the TFSWorkspace but functional and integration test rigs (which will be cloned for many testers) should be moved to another workspace.

1. Shut down and then un-deploy the TS rig in the TFSWorkspace workspace.

**NOTE:** Do not use 'Un-deploy/discard state' or 'Power Off' unless a rig is going to be deleted.

2. Move the rig to the 'source' workspace

Each sprints Integration build (TS*n*) is kept in the Integration workspace.

Each team’s current functional testing rig is kept in the NightlyBuild workspace.

Leave this un-deployed as clones are taken for testing purposes.

## Release Notes

Take a copy of “SiteA<sprint#>.xlsx”

Update the Version tab with deployed software branch names

Update the Notes tab if required

If any new WCF services have been deployed, add them to the WCF Services tab

Replace contents of the Release Notes tab;

There is a spreadsheet which is used to extract this info from TFS, “ReleaseNoteTFSReport.xlsx”. Just update the sprint in the combo box and you can copy and paste the PBI information relevant to the released build straight into the Release Notes spreadsheet.

# Deployment Configuration

Both automated and manual deployments are driven by a deployment configuration file. What follows is a description of the structure of a deployment configuration file.

The full formal schemas can be found in (relative to TFS)

$\Deployment\Main\Code\Build\Deployment.Logic\Xml\Schemas\\*.config.xsd

## Configuration

|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Occurrences** | **Attributes** | **Child Nodes** |
| Configuration | 1 | Id  Name  Environment | Machine  CommonRoles |

<configuration Id="1" Name="Integration.TSRig" Environment="TSRig">

<machine Id="1" Name="TS-CAS1" ExternalIP="1">

<machine Id="n" Name="TS-DB1" ExternalIP="2">

...

The configuration section is the root element of the config file

|  |  |  |
| --- | --- | --- |
| Attributes | Type | Description |
| Id | string | Optional. Arbitrary (not currently used) |
| Name | string | Optional. Arbitrary (not currently used) |
| Environment | string | Required. Environment Tag, used to filter environment specific configuration files and publish profiles |

## Machine

|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Occurrences** | **Attributes** | **Child Nodes** |
| Machine | 0+ | Id  Name  ExternalIP  Role | ServerRole  DatabaseRole  ServerDeploymentAccount |

<machine Id="1" Name="TS-CAS1" ExternalIP="1" Role="Text description">

<ServerRole Name="TfL.<rolename>" Description="Text description" />

...

Each machine to be deployed to must have a machine configuration section added.

|  |  |  |
| --- | --- | --- |
| Attribute | Type | Description |
| Id | string | Optional. Arbitrary (not currently used) |
| Name | String | Optional. Name is the machine name (as it appears in Lab Manager or the name of the machine in DNS for machines outside Lab Manager) |
| ExternalIP | string | Optional. The external IP address of the machine.  (The External IP is ignored if a Lab Manager Rig name is supplied as the machines external IP address is looked up from the rig (based on the Name attribute) via the LabManager) |
| Role | string | Optional. Descriptive. |

Note: If no machine name is specified the external IP will be used directly. This also applies to the post-deployment unit tests.

## Server Roles

Each machine may have one or more Server Role, defined by a ServerRole section in the configuration file. The server role defines the software to be deployed to the machine.

|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Occurrences** | **Attributes** | **Child Nodes** |
| ServerRole | 1+ | Name  Environment | ServiceDeply  WebDeploy  FileSystem  IISSetup  MSIDeploy |

<ServerRole Name="TFL.ServiceDeploy" Description="FAE Controller">

<ServerRole>

|  |  |  |
| --- | --- | --- |
| Attribute | Type | Description |
| Name | String | Name defines the type of server role (see following sections) |
| Description | string | The description is used for logging. |

### ServiceDeploy

A ServiceDeploy section defines the deployment of Windows Services using a Windows installer. A single installer can be used to deploy a number of services. ServiceDeploy also allows the inclusion of credentials for running services under a Windows Active Directory account.

<ServiceDeploy Name="FAEEngineController">

<MSI>

<id>EED8A1A9-891E-4D78-9C56-532939489846</id>

<name>Controller.msi</name>

<installlocation>d:\tfl\FAE\Controller</installlocation>

</MSI>

<Services>

<Service>

<Name>EngineControllerHost.exe</Name>

<Credentials>FAEServiceAccount</Credentials>

</Service>

</Services>

<configs>

<config name="EngineControllerHost.exe.config" target="\tfl\FAE\Controller"/>

</configs>

</ServiceDeploy>

#### MSI Section

|  |  |
| --- | --- |
| Simple Child Elements | Description |
| Id | Windows Installer product ID. When specified the previous version of the product will be uninstalled. |
| Name | Path of the installer – relative to the build output folder |
| Installlocation | Installation Path of the product |

#### Services

This section is used as a container for the services

##### Service

|  |  |
| --- | --- |
| Simple Child Elements | Description |
| Name | Name as shown in Services Control Manager |
| Credentials | Tag name of the active directory account to run the service under. This must correspond to an account specified in the ServiceAccount configuration file |

### WebDeploy

A WebDeploy section defines the deployment of a web application using MSDeploy. WebDeploy provides the creation of Application Pools and Sites in IIS.

Any machine that is the target of WebDeploy should have IIS and MS WebDeploy installed, this can be done during deployment by including the IISSetupServerRole (see later section)

|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Occurrences** | **Attributes** | **Child Nodes** |
| WebDeploy | 0+ | Name | Site  Package  TestInfo |

<WebDeploy Name="CSC Web">

<Site>

<Name>CustomerPortal</Name>

<Port>80</Port>

<PhysicalPath>D:\tfl\CACC\CustomerPortal</PhysicalPath>

<ApplicationPool>ASP.NET v4.0</ApplicationPool>

</Site>

<Package>

<Name>CSC Web</Name>

</Package>

<TestInfo>

<EndPoint></EndPoint>

</TestInfo>

</WebDeploy>

#### Site

|  |  |  |
| --- | --- | --- |
| Simple Child Elements | Type | Description |
| Name | String | Name of the website to be created in IIS. |
| Port | String | Port to be bound to the site. |
| PhysicalPath | String/Int | The physical path of the site in IIS. |
| ApplicationPool | string | The AppPool the site should run under. |

There are some optional additions for setting up IIS sites.

##### Adding a Virtual Directory

<VirtualDirectory>

<Name>email</Name>

<PhysicalPath>C:\CACC\emailqueue</PhysicalPath>

</VirtualDirectory>

##### Adding an Application

This is equivalent to specifying "Convert site to application" in IIS Manager

<Application>

<Name>Selenium</Name>

<PhysicalPath>D:\tfl\CACC\Selenium</PhysicalPath>

</Application>

#### Package

|  |  |  |
| --- | --- | --- |
| Simple Child Elements | Type | Description |
| Name | string | Required. The Name of the Visual Studio project for the web application. |

This name is used to resolve the package name from the Build output folder. By default a Visual Studio Web project called MyWebApp will produce a publish package at the following path relative to the build output folder:-

\PublishedWebSites\MyWebApp\_Publish\

#### TestInfo

##### In WebDeploy

|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Occurrences** | **Attributes** | **Description** |
| EndPoint | 0+ | Value only | none |

If specified, each endpoint will trigger a web test against http://<hostname>[:port]/<endpoint> (port is omitted if it is 80).

If omitted a single test will be run against http://<hostname>[:port].

##### In ServiceDeploy

|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Occurrences** | **Attributes** | **Description** |
| Ignore | 1 | Bool | Suppresses a service test from being performed. |

### IISSetup

|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Occurrences** | **Attributes** | **Child Nodes** |
| IISSetup | 0+ | Name  Description | none |

<ServerRole Name="TfL.IISSetup" Description="IIS" />

This role configures IIS on machines that will have web components deployed to them. The Microsoft WebDeploy software is also installed for use during web deployments.

### FileSystem

|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Occurrences** | **Attributes** | **Child Nodes** |
| FileSystem | 0+ | none | CreateFolder  CopyItem |

<FileSystem>

<CreateFolder TargetPath="\d$\OysterTapImporter"/>

<CopyItem Source="\Software\DropFolder\BaseData" Target="\d$\tfl\FAE\BaseData"

Recurse="true" Filter="\*.\*" Replace="true"/>

Note: The paths are remote references which get prefixed by the host name so they do need to start with the leading back-slash and a reference to a valid share name.

#### CreateFolder

|  |  |  |
| --- | --- | --- |
| Attributes | Type | Description |
| TargetPath | string | Required. This is appended in powershell to a string to produce a UNC. Therefore not the use of "\D$\" and not "D:\" |

#### CopyItem

|  |  |  |
| --- | --- | --- |
| Attributes | Type | Description |
| Source | string | Required. Relative to deploy directory on Deployment server. |
| Target | string | Required. Again, note use of \D$\ not D:\ |
| Recurse | bool | Required. |
| Filter | string | Required. Typical file/directory name filter with \* wildcards. |
| Replace | bool | Required. |

## Database Roles

Database source code is managed using SQL Server Data Tools in Visual Studio. These projects produce DACPAC packages which are in turn deployed to SQL Server. The DataBaseRole section defines the specifics for this process.

|  |  |  |  |
| --- | --- | --- | --- |
| **Element** | **Occurrences** | **Attributes** | **Child Nodes** |
| DatabaseRole | 0+ | Name  Description  IncludeTFSBuild  RemoteWithoutPsExec |  |

|  |  |  |
| --- | --- | --- |
| Attributes | Type | Description |
| Name | string | Required. Either "FromConfig" or "AspNetMembership" |
| Description | string | Required. Environment Tag, used to filter environment specific configuration files and publish profiles |
| IncludeTFSBuild | bool | Optional. Forces creation of a tfsbuild SQL Server login. This is intended for remote testing against servers that are purely integrated authentication. |
| RemoteWithoutPsExec | bool | Deprecated. This is only applicable for 'remote' database deployments which are no longer supported. |

### FromConfig

|  |  |  |
| --- | --- | --- |
| Simple Child Elements | Type | Description |
| ProjectStub | string | Required. Name is arbitrary (not currently used) |
| TargetDatabase\* | string | Name of the target database |
| DatabaseInstance\* | string | Name of the database instance |
| PublishProfile | string | The path to a the database publish profile – relative to the Build output folder |
| PostDeployment | string | The name of a sql script (external to the dacpac) that will be run after dacpac deployment. |
| AlwaysCreateNewDatabase\* | string | True – the database will be dropped before the new schema is applied  False – an attempt is made to deploy only changes to the database |

\* AspNetMembership role uses only these Elements.

|  |  |  |
| --- | --- | --- |
| Child Elements | Occurrences | Description |
| EnableAspnetSqlCacheDependency | 0-1 | Optional. |
| TestInfo | 0-1 | See section 6.3.2.3 |

<DatabaseRole Name="FromConfig" Description="CACC Database">

<ProjectStub>CSC SQL Server Schema</ProjectStub>

<TargetDatabase>CSCWeb</TargetDatabase>

<DatabaseInstance>Inst3</DatabaseInstance>

<PublishProfile>CSCWebDb.$Environment.publish.xml</PublishProfile>

<PostDeployment></PostDeployment>

<AlwaysCreateNewDatabase>false</AlwaysCreateNewDatabase>

<EnableAspnetSqlCacheDependency>

<OnTable>SecurityQuestion</OnTable>

<OnTable>Country</OnTable>

<OnTable>CustomerType</OnTable>

<OnTable>PaymentCardType</OnTable>

<OnTable>PaymentCardScheme</OnTable>

<OnTable>Title</OnTable>

</EnableAspnetSqlCacheDependency>

<TestInfo UserName="CSCWeb" Password="CSCWeb" />

</DatabaseRole>

#### EnableSqlCacheDependency

Tables listed under this section will have the ASP.NET Cache dependency changes applied.

### AspNetMembership

Creates the ASPMembership schema objects in the specified database, for example:-

aspnet\_regsql.exe -S $Datasource -d $TargetDatabase -A all –E

|  |  |  |
| --- | --- | --- |
| Simple Child Elements | Type | Description |
| TargetDatabase\* | string | Name of the target database |
| DatabaseInstance\* | string | Name of the database instance |
| AlwaysCreateNewDatabase\* | string | True – the database will be dropped before the new schema is applied  False – an attempt is made to deploy only changes to the database |

<DatabaseRole Name="ASPMembership"Description="CACC ASP.NET Membership">

<TargetDatabase>CSCWeb</TargetDatabase>

<DatabaseInstance>Inst3</DatabaseInstance>

<AlwaysCreateNewDatabase>true</AlwaysCreateNewDatabase>

</DatabaseRole>

### ServerDeploymentAccount

Defines the server that the deployment software will be copied to and deployed from.

## Common Roles

Server roles can be grouped under the CommonRoles section and referred to from one or more machine section.

The 'include' attribute should be used to link entries under machine to the corresponding ServerRole in CommonRoles.

<ServerRole Name="TFL.ServiceDeploy" Description="FAE Engine" Include="<Name>" />

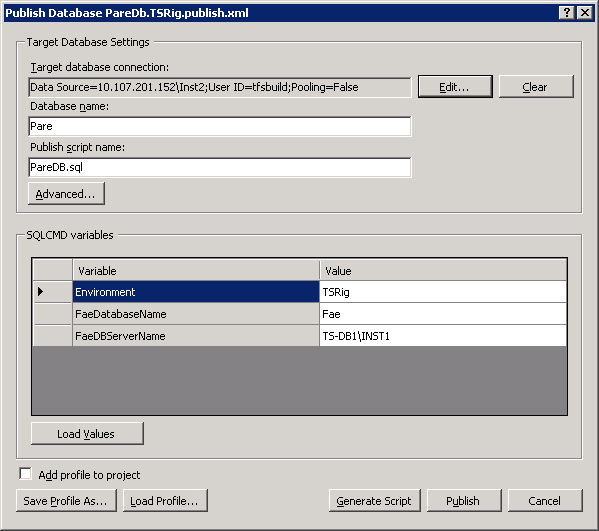
# Application Configuration

## Database Projects

### Publish profiles

We are working on removing these from the deployment.

(X.<env>.publish.xml)



## Pre and Post-Deployment Scripts

Stored in <DB Project>\Scripts\, these should have the properties;

* Build Action: PostDeploy (or PreDeploy)
* Copy to output directory: Do Not Copy.

They are included in the dacpac and are run by the msdeploy installer.

They should not be confused with scripts referenced in config files by <PostDeploymnetScript> which are run by the setup process.

### File Inclusion

:r ./<filename>.sql

e.g. :r.\AddBaseData.sql

Included files need to have the properties;

* Build Action: None
* Copy to output directory: Do Not Copy.

### SqlCmd Variables

These are used in pre- and post-deployment scripts

$(<var\_name>)

e.g. IF('$(Environment)'= 'TSRig')

T-SQL Statement;

## App.config (Windows Services)

In a Project:

app.config Contains the basic (default) config settings

app.<env>.config Contains xslt mappings applied to app.config depending on the enviroment

In a deployment:

%dropfolder%/Configuration/<env>/<application>.exe.config

## Web.config (Web applications)

This is only required if the values in the web.config need to vary by environment.

In a Project:

Configuration Parameters\<Project>.<env>.xml

Contains: (<setParameters>)

This file contains the specific values for the <env> environments version of this web.config entry.

NOTE: These files need to be included in the project and be set to;

* + Build Action: None (this may default to 'Content' for new files)
  + Copy to Build Output: Copy Always

In a deployment:

ConfigurationParameters\

ConfigurationParameters

## Adding a Project to a Solution

Adding Windows Service

Add Role to deploy config .xml file

Add exe.config to configs.config section

Windows command line executable or Gui program

Use a TFL.ServiceDeploy role (with an empty <Service /> block) and an msi installer.

WCF Service or Web site

Add a TFL.WebDeploy role to the config file.

## Adding SqlCmd Variables to DB Projects

SqlCmd variables are used pre- and post-deployment scripts and are added via the DB project properties.

When these are added, they must also be defined in each publish profile.

## Adding WCF Service to Project

Add to web.config

Add to parameters.xml

Add to ConfigurationParameters\<proj>.<env>.xml

# Manual Deployments

# Environment Pre-requisites