CRM Build Manager

Document Version: 2016.06.30

Contact: Glen Wells, email: m@xim.is

About CRM Build Manager

CRM Build Manager is a tool to automate the processes typically followed when developing and deploying Dynamics CRM solutions.

CRM Build Manager offers a range of Actions that can be taken against one or more Organizations (CRM instances).

Run CRM Build Manager from the command line as follows:

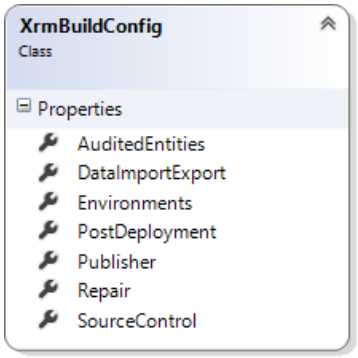
```
BuildManager.exe XrmBuildConfig.xml EnvironmentName Action [OrganizationName 1] [OrganizationName 2] [OrganizationName n]
```

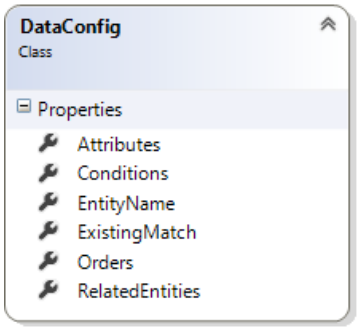
The XrmBuildConfig.xml file is deserialised into an XrmBuildConfig class instance and can therefore be constructed by creating an instance of that class, populating it as required and serialising to a file. The configuration values in this class instance are used by each Action to perform its operation.

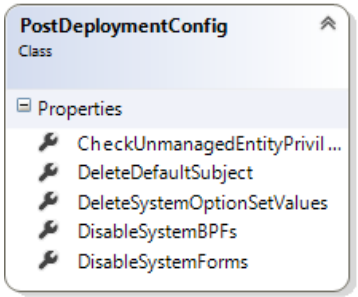
An XrmBuildConfig contains:

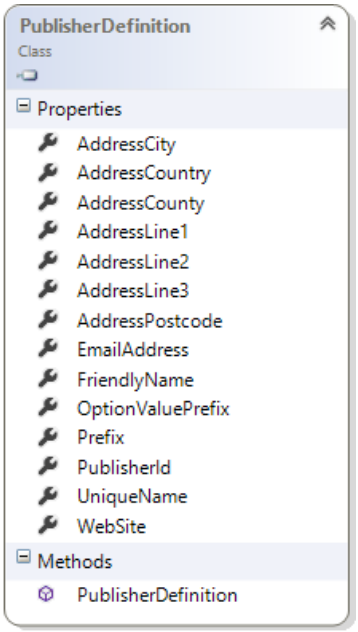
- Environments (one or more EnvironmentConfig classes)
- Publisher (details of the CRM Publisher record)
- SourceControl (configuration values for connecting to TFS or Git)

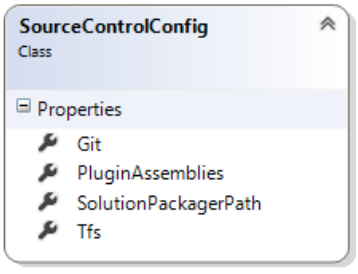
Plus other global configuration, including which Entities should be audited, whether data is imported/exported during a Repair (see below) and configuration of post-deployment actions.



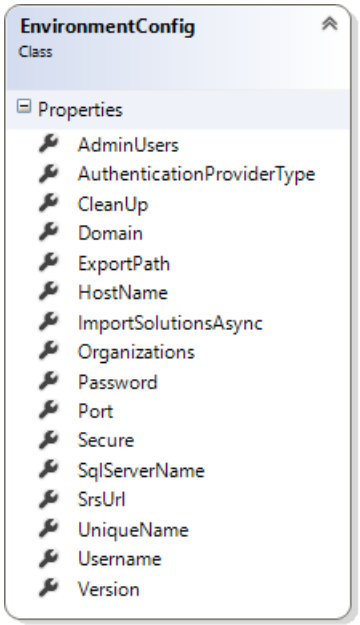


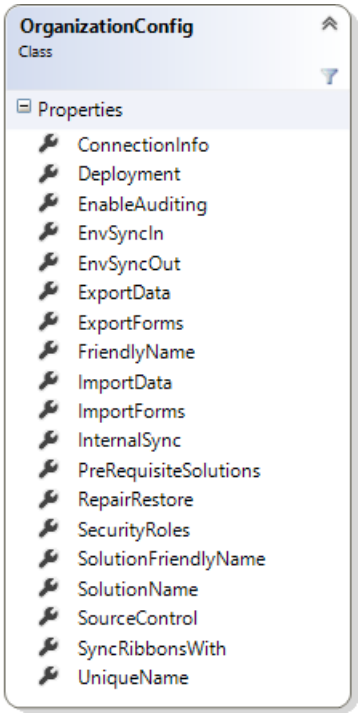






Each EnvironmentConfig contains details about a specific environment (e.g. Development, Staging, Production) and each Organization that exists within each Environment is defined using an OrganizatonConfig.





Available Actions

Actions are run against an Environment. They apply to ALL Organizations in the Environment, unless specific Organization UniqueNames are supplied as additional arguments to BuildManager.exe, for example:

```
BuildManager.exe XrmBuildConfig.xml Development DeleteOrgs Build Staging
```

The Actions available are as follows:

Organization Actions

CreateOrgs

Create the Organizations defined in the Environment (or just those supplied as optional arguments). This uses the CRM Deployment Service which must be running under an account with sufficient permissions (see *Setting Up the CRM Deployment Service)*.

DeleteOrgs

Delete the Organizations defined in the Environment (or just those supplied as optional arguments) and drop their associated databases.

SetupAdminUsers

Create Users in each Organization (or just those supplied as optional arguments) and grant each user the System Administrator role in each Organization. Users are specified as follows:

```
<EnvironmentConfig UniqueName="Development">
  <AdminUsers>
    <UserConfig Username="BEDROCK\fred" LastName="Flintstone" FirstName="Fred" />
    <UserConfig Username="BEDROCK\barney" LastName="Rubble" FirstName="Barney" />
  </AdminUsers>
```

```
</EnvironmentConfig>
```

SetupSolutions

Create a Solution in each Organization defined in the Environment (or just those supplied as optional arguments) using the Publisher defined in the top level XrmBuildConfig class.

InstallPreReqs

Install managed pre-requisite solutions in each Organization defined in the Environment (or just those supplied as optional arguments). This has a range of uses depending on the nature of the project. Pre-requisite solutions could be development tools, solutions developed by Microsoft partners or purchased from the Dynamics Marketplace. They are configured at Organization level:

```
<EnvironmentConfig UniqueName="Development">
  <Organizations>
    <OrganizationConfig UniqueName="Phase1Build">
      <PreRequisiteSolutions>
        <FullPath>C:\PreReqs\CrmUnitTest_2013_managed.zip</FullPath>
        <FullPath>C:\PreReqs\DynamicsXRMTools2015_3_0_0_0_managed.zip</FullPath>
      </PreRequisiteSolutions>
    </OrganizationConfig>
  </Organizations>
</EnvironmentConfig>
```

Initialise

Run the following Actions in order, to completely reset an Environment or create it from scratch:

1. DeleteOrgs
2. CreateOrgs
3. SetupAdminUsers
4. SetupSolutions
5. InstallPreReqs

As with all other Actions, this can be limited to specific Organizations if they are supplied as optional arguments.

Deployment Actions

Deploy

Deploy **to** the Organizations defined in the Environment (or just those supplied as optional arguments). The deployment source environment must also be accessible; therefore, this is typically used in a Development to Internal Staging scenario. Configuration as follows:

```
<EnvironmentConfig UniqueName="Staging">
  <Organizations>
    <OrganizationConfig UniqueName="UAT">
      <Deployment Mode="Unmanaged">
        <DeployFromOrgs>
          <UniqueName>Phase1Build</UniqueName>
          <UniqueName>Phase2Build</UniqueName>
        </DeployFromOrgs>
      </Deployment>
    </OrganizationConfig>
  </Organizations>
</EnvironmentConfig>
```

The Deploy Action also incorporates the PostDeployment Action, so this does not need to be run separately after Deploy.

DeployFromDisk

Deploy **to** the Organizations defined in the Environment (or just those supplied as optional arguments). This is typically used on a Production environment which is inaccessible from the development network. Solution zip files are generated using the ExportSolutions Action and transferred to the Production server for deployment. Solutions should be located in locally accessible folders using the pattern *[ExportPath]\Solutions\[DeployFromUniqueName]*. The name of the zip file within each folder does not matter. If there is more than one zip file, only the newest is installed. Configuration is defined as follows:

```
<EnvironmentConfig UniqueName="Production">
  <ExportPath>C:\SomeFolder</ExportPath>
  <Organizations>
    <OrganizationConfig UniqueName="CustomerName">
      <Deployment Mode="Managed">
        <DeployFromDisk>
          <UniqueName>Phase1Build</UniqueName>
          <UniqueName>Phase2Build</UniqueName>
        </DeployFromDisk>
      </Deployment>
    </OrganizationConfig>
  </Organizations>
</EnvironmentConfig>
```

This configuration would attempt to deploy:

- C:\SomeFolder\Solutions\Phase1Build\[zipfile].zip
- C:\SomeFolder\Solutions\Phase2Build\[zipfile].zip

The DeployFromDisk Action also incorporates the PostDeployment Action, so this does not need to be run separately after Deploy.

SetupAuditing

Enable and Disable Auditing at Organization and at Entity level. Organizations with EnableAuditing set to *true* will have Auditing enabled globally. Organizations without this (or set to *false*) will have Auditing disabled globally.

```
<EnvironmentConfig UniqueName="Production">
  <Organizations>
    <OrganizationConfig UniqueName="CustomerName">
      <EnableAuditing>true</EnableAuditing>
    </OrganizationConfig>
  </Organizations>
</EnvironmentConfig>
```

For each Organization, the Entities defined as being enabled for Auditing will be set as such, and those which are not in the list will have auditing disabled. This is defined globally for consistency across Environments.

```
<XrmBuildConfig>
```

```
  <AuditedEntities>
    <EntityName>account</EntityName>
    <EntityName>appointment</EntityName>
    <EntityName>...</EntityName>
  </AuditedEntities>
</XrmBuildConfig>
```

PostDeployment

Run a number of useful post-deployment tasks, as follows:

1. Disable out-of-the-box Business Process Flows.
2. Disable out-of-the-box Forms on Entities where a Custom Form has been added.
3. Delete unwanted options from out-of-the-box Option Sets.
4. Check for any Unmanaged Entities that no Unmanaged Roles have privileges against.
5. Delete the default Subject record.

Although defined as a standalone Action, it is typically only used when initially configuring post-deployment steps. The Deploy, DeployFromDisk and InternalSync actions all incorporate this Action.

Each task can be enabled or disabled as required. For unwanted Option Set values there is additional configuration to define these.

```
<XrmBuildConfig>
  <PostDeployment>
    <CheckUnmanagedEntityPrivileges>true</CheckUnmanagedEntityPrivileges>
    <DisableSystemBPFs>true</DisableSystemBPFs>
    <DisableSystemForms>true</DisableSystemForms>
    <DeleteDefaultSubject>true</DeleteDefaultSubject>
    <DeleteSystemOptionSetValues>
      <OptionSet EntityName="campaign" AttributeName="typecode">
        <UnwantedOptions>
          <OptionText>Event</OptionText>
          <OptionText>Co-branding</OptionText>
          <OptionText>Other</OptionText>
        </UnwantedOptions>
      </OptionSet>
      <OptionSet EntityName="contact" AttributeName="preferredcontactmethodcode">
        <UnwantedOptions>
          <OptionText>Mail</OptionText>
          <OptionText>Email</OptionText>
        </UnwantedOptions>
      </OptionSet>
    </DeleteSystemOptionSetValues>
  </PostDeployment>
</XrmBuildConfig>
```

Data Import/Export Actions

StaticDataExport and StaticDataImport

Export and Import data records which are required for the Application to function (e.g. configuration records, static data sets used as lookups, users, teams, products, etc.). Organizations are defined as being an export source, import target, or both. The entities and attributes that are exported/imported are defined globally.

```
<OrganizationConfig UniqueName="Phase1">
  <ImportData>true</ImportData>
</OrganizationConfig>
```

```
<OrganizationConfig UniqueName="Phase2">
  <ImportData>true</ImportData>
</OrganizationConfig>

<OrganizationConfig UniqueName="Build">
  <ExportData>true</ExportData>
</OrganizationConfig>

<XrmBuildConfig>
  <DataImportExport>
    <DataConfig EntityName="service">
      <Attributes>
        <Attribute>anchoroffset</Attribute>
        <Attribute>description</Attribute>
        <Attribute>duration</Attribute>
        <Attribute>granularity</Attribute>
        <Attribute>initialstatuscode</Attribute>
        <Attribute>isvisible</Attribute>
        <Attribute>name</Attribute>
        <Attribute>resourcespecid</Attribute>
        <Attribute>showresources</Attribute>
        <Attribute>strategyid</Attribute>
      </Attributes>
    </DataConfig>
  </DataImportExport>
</XrmBuildConfig>
```

StaticDataReset

Delete all records for all Entities which are defined as Static Data records, on all Organizations configured for either Static Data Export or Static Data Import (or just those supplied as optional arguments).

Development Actions

SetSolutionVersion

Set the Version Number of the Solution in each Organization defined in the Environment (or just those supplied as optional arguments). Version number format is yyyy.mm.dd.i, where “i" is an incremental number which resets to 1 each day.

SetupSecurityRoles

Synchronise privileges between Security Roles where roles should be considered sub-sets or super-sets of each other. In the following configuration, “Normal User 1” and “Normal User 2” would have all the privileges of “Baseline” plus their own, and “Power User” would have all the privileges of “Normal User 2” plus its own. Although not necessary in CRM (as users could be granted a combination of Roles) it makes Security Role maintenance a little easier.

```
<OrganizationConfig UniqueName="Build">
  <SecurityRoles>
    <SecurityRoleConfig Name="Baseline">
      <ChildRoles>
        <SecurityRoleConfig Name="Normal User 1" />
        <SecurityRoleConfig Name="Normal User 2" />
          <ChildRoles>
            <SecurityRoleConfig Name="Power User" />
          </ChildRoles>
        </SecurityRoleConfig>
      </ChildRoles>
```

```
    </SecurityRoleConfig>
  </SecurityRoles>
</OrganizationConfig>
```

FormsExport and FormsImport

Export and Import FormXML from each Organization defined in the Environment (or just those supplied as optional arguments). The following configuration could be used to update forms in a "Phase 1" Organization with their equivalents from "Phase 2".

```
<OrganizationConfig UniqueName="Phase1">
  <ImportForms>true</ImportForms>
</OrganizationConfig>

<OrganizationConfig UniqueName="Phase2">
  <ExportForms>
    <FormConfig EntityName="account" FormType="Main" />
    <FormConfig EntityName="contact" FormType="Main" />
    <FormConfig EntityName="lead" FormType="Main" />
    <FormConfig EntityName="opportunity" FormType="Main" />
  </ExportForms>
</OrganizationConfig>
```

ExportSolutions

Export both Managed and Unmanaged Solutions from each Organization defined in the Environment (or just those supplied as optional arguments).

SyncRibbons

Ensure that Entity ribbons are consistent across Organizations. Ribbon inconsistencies can occur on complex multi-solution deployments. This Action can be used to overwrite ribbons on a Build Organization if they differ from a specified Development Organization. This Action should only be used against a Solution using Unmanaged entities (i.e. a merged Build Organization).

```
<OrganizationConfig UniqueName="Build" SyncRibbonsWith="Phase2Dev">
...
</OrganizationConfig>
```

CleanUp

Remove old files from an Environment. Define the folders to be processed like this:

```
<EnvironmentConfig UniqueName="Development">
  <CleanUp>
    <CleanUpFolder Path="C:\BuildManager_Export\Solutions" FileAgeDays="14" />
    <CleanUpFolder Path="C:\DatabaseBackups" FileAgeDays="3" />
  </CleanUp>
</EnvironmentConfig>
```

EnvSyncIn and EnvSyncOut

Synchronise changes between Organizations in different Environments (e.g. a Development environment on a developer's PC and a master Development environment). EnvSyncIn pulls changes in from one or more other Organizations. EnvSyncOut pushes changes out.

The other Organizations that an Organization syncs with are defined as follows:

```
<OrganizationConfig UniqueName="LocalPhase2">
  <EnvSyncIn>
    <UniqueName>MasterPhase1</UniqueName>
    <UniqueName>MasterPhase2</UniqueName>
  </EnvSyncIn>
  <EnvSyncOut>
    <UniqueName>MasterPhase2</UniqueName>
  </EnvSyncOut>
</OrganizationConfig>
```

InternalSync

Synchronise solutions between Organisations in an Environment. Typically used to create a master build solution on a Build Organization for deployment. As an example, the following configuration would:

1. Export “Phase 1” into “Phase 2” as Managed
2. Export “Phase 1 and “Phase 2” into “Build” as Unmanaged and combine them (MergeUnmanaged)
3. Export “Build” into “Staging” as Managed

```
<OrganizationConfig UniqueName="Phase1">
  <InternalSync ExportPass="1" />
</OrganizationConfig>

<OrganizationConfig UniqueName="Phase2">
  <InternalSync ExportPass="1" ImportPass="1">
    </ImportManaged>
      <UniqueName>Phase1</UniqueName>
    </ImportManaged>
</OrganizationConfig>

<OrganizationConfig UniqueName="Build">
  <InternalSync ExportPass="2" ImportPass="1" MergeUnmanaged="true">
    <ImportUnmanaged>
      <UniqueName>Phase1</UniqueName>
      <UniqueName>Phase2</UniqueName>
    </ImportUnmanaged>
</OrganizationConfig>

<OrganizationConfig UniqueName="Staging">
  <InternalSync ImportPass="2">
    <ImportManaged>
      <UniqueName>Build</UniqueName>
    </ImportManaged>
</OrganizationConfig>
```

The InternalSync Action also incorporates the PostDeployment Action, so this does not need to be run separately after Deploy.

Repair

Completely rebuild an environment by performing the following actions in order:

1. StaticDataExport
2. *Export Solutions of Organizations with ExternalSync ExportPass = 1*
3. DeleteOrgs

4. CreateOrgs
5. SetupAdminUsers
6. SetupSolutions
7. *Import Solutions exported in step 2*
8. InternalSync
9. StaticDataImport

Source Control Actions

Defining Source Control Locations

For Source Control Actions to function, at least one Source Control Location must first be specified. A Location can be either a TFS Project Collection or a Git Repository. In both cases, these are given a LocationName to identify them.

TFS Project Collections are defined as follows:

```
<XrmBuildConfig>
  <SourceControl>
    <Tfs>
      <ProjectCollections>
        <ProjectCollection>
          <LocationName>mypc</LocationName>
          <LocalPath>C:\tfs\mypc</LocalPath>
          <PluginAssembliesPath>$/Project/Plugins</PluginAssembliesPath>
          <SolutionsPath>$/Project/Solutions</SolutionsPath>
          <ReportsPath>$/Project/Reports</ReportsPath>
          <ProjectCollectionUri>http://tfsserver:8080/</ProjectCollectionUri>
        </ProjectCollection>
        <ProjectCollection>...</ProjectCollection>
        <ProjectCollection>...</ProjectCollection>
      </ProjectCollections>
    </Tfs>
  </SourceControl>
</XrmBuildConfig>
```

Git Repositories are defined as follows:

```
<XrmBuildConfig>
  <SourceControl>
    <Git>
      <Repositories>
        <Repository>
          <LocationName>myrepo</LocationName>
          <BranchName>development</BranchName>
          <LocalPath>C:\git\myrepo</LocalPath>
          <EmailAddress>my.name@customer.co.uk</EmailAddress>
          <Username>myusername</Username>
          <Password>mypassword</Password>
          <Url>http://gitserver.customer.co.uk/crm/myrepo.git</Url>
        </Repository>
        <Repository>...</Repository>
        <Repository>...</Repository>
      </Repositories>
    </Git>
  </SourceControl>
</XrmBuildConfig>
```

Note the differences between Git and TFS:

- TFS does not require a username/password as it uses Windows Integration.
- TFS allows operations on subfolders so Plugin, Solution and Report paths are specified.

SolutionCheckIn

Extract a Solution into its component parts and commit/check in those parts to Source Control.

Build Manager must have access to the CRM Solution Packager tool, with its path defined as follows:

```
<XrmBuildConfig>
  <SourceControl>
    <SolutionPackagerPath>C:\SDK\Bin\SolutionPackager.exe</SolutionPackagerPath>
  </SourceControl>
</XrmBuildConfig>
```

Add a Source Control Location to an OrganizationConfig to include its Solution in the SolutionCheckIn Action:

```
<OrganizationConfig UniqueName="Phase1Build">
  <SourceControl>
    <SolutionLocation>myrepo</SolutionLocation>
  </SourceControl>
</OrganizationConfig>
```

The Action behaves slightly differently for TFS and for Git:

- TFS:
    - Extract the Solution to a folder called *SolutionName* in the path defined as SolutionsPath
    - Check in the SolutionsPath folder
- Git:
    - Extract the solution to a folder called Solutions*SolutionName* in the Git Local Path
    - Stage, Commit and Push the whole Repo

With Git, because the whole Repo has to be committed, be aware that it is possible other changed files will also be committed.

UpdatePluginAssemblies

Download Plugin Assembly Code from Source Control, generate Release Builds of Plugin Assemblies and update Assemblies in Organizations.

Plugin Assemblies, and the Nuget packages they require, are defined as follows:

```
<XrmBuildConfig>
  <SourceControl>
    <Nuget>
      <Package Id="Microsoft.CrmSdk.CoreAssemblies" Version="7.1.1" />
      <Package Id="Microsoft.CrmSdk.Workflow" Version="7.1.1" />
      <Package Id="Microsoft.IdentityModel" Version="6.1.7600.16394" />
    </Nuget>
    <PluginAssemblies>
      <PluginAssemblyConfig>
        <AssemblyName>Customer.Crm.Project.Plugins</AssemblyName>
        <LocationName>myrepo</LocationName>
        <ProjectName>Customer.Crm.Project.Plugins.csproj</ProjectName>
        <ProjectPath>Customer.Crm.Project.Plugins</ProjectPath>
        <ILMerge>
```

```
          <KeyFile>Customer.snk</KeyFile>
          <MergeAssemblies>
            <AssemblyFileName>Customer.Crm.Project.Common.dll</AssemblyFileName>
            <AssemblyFileName>Customer.Crm.Common.dll</AssemblyFileName>
          </MergeAssemblies>
        </ILMerge>
      </PluginAssemblyConfig>
  </SourceControl>
</XrmBuildConfig>
```

For TFS, the ProjectPath is relative to PluginAssembliesPath. For Git, it is relative to the Repo's LocalPath.

Plugin Assemblies are associated with Organizations as follows. Assemblies can be associated with multiple Organizations if required, and conversely each Organization can be associated with only a subset of Assemblies.

```
<OrganizationConfig UniqueName="Phase1Build">
  <SourceControl>
    <PluginAssemblies>
      <AssemblyName>Customer.Crm.Project.Plugins</AssemblyName>
      <AssemblyName>Customer.Crm.Project.Workflow</AssemblyName>
    </PluginAssemblies>
  </SourceControl>
</OrganizationConfig>
```

The Action behaves as follows:

- TFS: Get Latest on folder PluginAssembliesPath / Git: Pull the whole Repo
- Download Nuget packages if they are not already present
- Perform a Release Build of each Assembly
- Update each Assembly within each Organization

UpdateReports

Download SSRS RDL files from Source Control and update Reports in CRM.

Add a Source Control Location to an OrganizationConfig to synchronise Reports with Source Control:

```
<OrganizationConfig UniqueName="Phase1Build">
  <SourceControl>
    <ReportsLocation>myrepo</ReportsLocation>
  </SourceControl>
</OrganizationConfig>
```

The Action behaves as follows:

- TFS: Get Latest on folder ReportsPath / Git: Pull the whole Repo
- Update the Reports in the Organization with the versions downloaded from Source Cotnrol

Full Example Source Control Configuration

For reference:

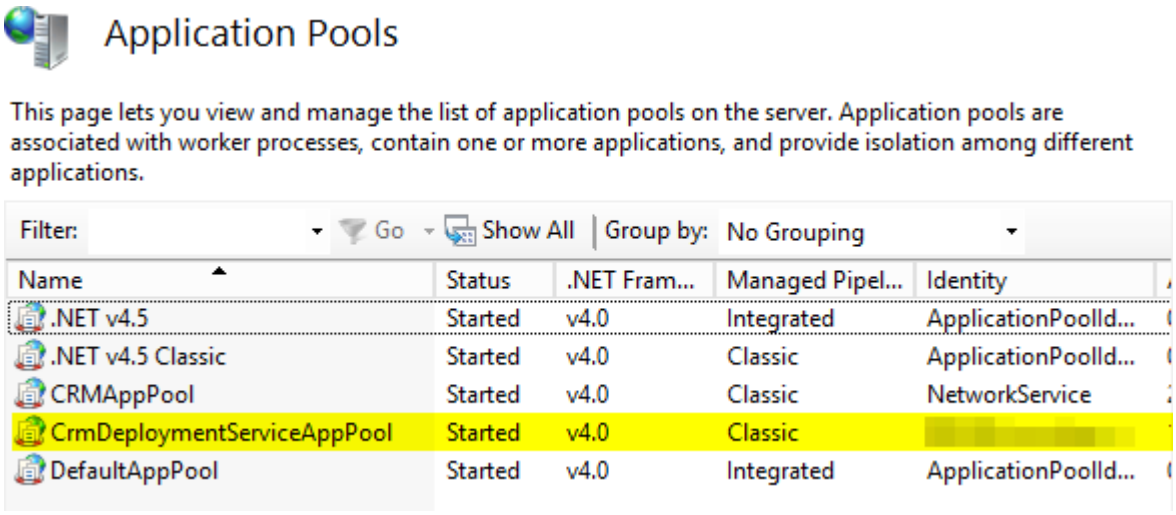
```
<XrmBuildConfig>
  <SourceControl>
    <Git>
      <Repositories>
        <Repository>
          <LocationName>myrepo</LocationName>
          <BranchName>development</BranchName>
          <LocalPath>C:\git\myrepo</LocalPath>
          <EmailAddress>my.name@customer.co.uk</EmailAddress>
          <Username>myusername</Username>
          <Password>mypassword</Password>
          <Url>http://gitserver.customer.co.uk/crm/myrepo.git</Url>
        </Repository>
        <Repository>...</Repository>
        <Repository>...</Repository>
      </Repositories>
    </Git>
    <Nuget>
      <Package Id="Microsoft.CrmSdk.CoreAssemblies" Version="7.1.1" />
      <Package Id="Microsoft.CrmSdk.Workflow" Version="7.1.1" />
      <Package Id="Microsoft.IdentityModel" Version="6.1.7600.16394" />
    </Nuget>
    <PluginAssemblies>
      <PluginAssemblyConfig>
        <AssemblyName>Customer.Crm.Project.Plugins</AssemblyName>
        <LocationName>myrepo</LocationName>
        <ProjectName>Customer.Crm.Project.Plugins.csproj</ProjectName>
        <ProjectPath>Customer.Crm.Project.Plugins</ProjectPath>
        <ILMerge>
          <KeyFile>Customer.snk</KeyFile>
          <MergeAssemblies>
            <AssemblyFileName>Customer.Crm.Project.Common.dll</AssemblyFileName>
            <AssemblyFileName>Customer.Crm.Common.dll</AssemblyFileName>
          </MergeAssemblies>
        </ILMerge>
      </PluginAssemblyConfig>
      <PluginAssemblyConfig>
        <AssemblyName>Customer.Crm.Project.Workflow</AssemblyName>
        <LocationName>myrepo</LocationName>
        <ProjectName>Customer.Crm.Project.Workflow.csproj</ProjectName>
        <ProjectPath>Customer.Crm.Project.Workflow</ProjectPath>
        <ILMerge>
          <KeyFile>Customer.snk</KeyFile>
          <MergeAssemblies>
            <AssemblyFileName>Customer.Crm.Project.Common.dll</AssemblyFileName>
            <AssemblyFileName>Customer.Crm.Common.dll</AssemblyFileName>
          </MergeAssemblies>
        </ILMerge>
      </PluginAssemblyConfig>
    </PluginAssemblies>
    <SolutionPackagerPath>C:\SDK\Bin\SolutionPackager.exe</SolutionPackagerPath>
  </SourceControl>
  <EnvironmentConfig UniqueName="Development">
    <Organizations>
      <OrganizationConfig UniqueName="Phase1Build">
        <SourceControl>
          <SolutionLocation>myrepo</SolutionLocation>
          <ReportsLocation>myrepo</ReportsLocation>
          <PluginAssemblies>
```

```
            <AssemblyName>Customer.Crm.Project.Plugins</AssemblyName>
            <AssemblyName>Customer.Crm.Project.Workflow</AssemblyName>
          </PluginAssemblies>
        </SourceControl>
      </OrganizationConfig>
    </Organizations>
  </EnvironmentConfig>
</XrmBuildConfig>
```

Setting up the CRM Deployment Service

For some of its Actions, the CRM Build Manager requires the CRM Deployment Service (see https://msdn.microsoft.com/en-us/library/gg327886.aspx).

Change the Identity of the CrmDeploymentServiceAppPool to a domain user. Make sure the user is a local Administrator and SysAdmin in SQL Server.



Use PowerShell to attempt to create an Organization as follows:

```
Add-PSSnapin Microsoft.Crm.Powershell

New-CrmOrganization -DisplayName "Test Org" -SQLServerName "sqlserver" -SrsUrl
"https://sqlserver/reportserver" -Name "testorg"
```

Resolve any permissions errors that appear and repeat until successful.

Source Code

Source code is available at: https://github.com/maximisltd/XrmBuild.