WcfNQueueSMEx2 Visual Studio Solution Structure

George Stevens

|  |  |  |
| --- | --- | --- |
| **version** | **date** | **author** |
| 0.1 | 5/18/2015 | George Stevens |
| 0.2 | 5/19/15 | George Stevens |
|  |  |  |
|  |  |  |

# Introduction

The solution structure of folders, projects, and namespaces presented herein is based on that described by Monty Montgomery in the IDesign Forum thread “Namespace convention and namespace best practices”. Monty is a Master Architect at IDesign. This structure fulfills the following goals that serve to enable Service Oriented Apps:

* Use the IDesign Method practice of having one service per assembly, i.e. one Manager or Engine or ResourceAccessor per assembly. This facilitates the composition of larger services from smaller services, and reuse.
* Promote reuse by presenting a broad view of the service oriented solution in source control and the Solution Explorer. This minimizes the depth of solution folder hierarchies to allow developers to easily explore existing components.
* Partition assemblies so as to produce the smallest possible client side foot print.
* Allow easy finding of components when looking at an IDesign architecture diagram.

The above goals are realized in the following structure of the WcfNQueueSMEx2 Visual Studio Solution.

Please note that Service Oriented Apps are typically composed of a number of services, each of which may be hosted…. Well, where ever! In the data center, in the cloud, on several different virtual machines, with at times multiple instances of the same service alive at once and running in different hosts! And, microservices result in even more services and hosting options. This requirement results in the one assembly per service goal, in addition to the other above goals that make it easier for developers to deal with this requirement.

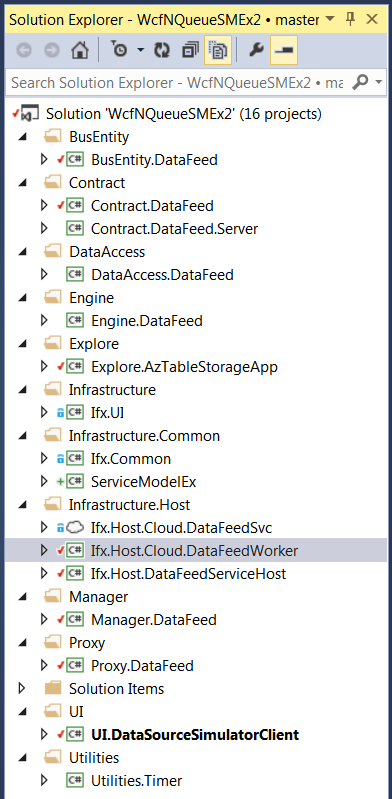
Here is Monty’s basic Namespace convention, with Product being optional:

<Company>.<Concept>.[<Product>].<Subsystem>

For example in the WcfNQueueSMEx2 project, GS.Manager.DataFeed is where the DataFeedManager project resides.

* Company is GS (abbreviation for my name for lack of anything better in this example).
* Concept is Manager.
* Product is omitted since there is only one product, however it could be something like Mobile if there were a mobile version.
* Subsystem is DataFeed.

Here is the solution structure as of 5-18-15, 1100 am. It shows the scheme described subsequently. 5-19 OUT OF DATE.



Note that <Company> is GS, George Stevens.

**Contract** folder – Contains data and service contracts shared by the client and server. And it also contains those that are used only on the server side (those for Engines and RAs). Thus, for any one subsystem, there will be 2 projects – GS.Contract.<Subsystem> and GS.Contract.<Subsystem>.Server. If you do not want something shared between client and server (like Engine and RA service and data contracts) move them to an Engine or RA project rather than put them here. The general rule – Things are always shared until consciously placed where they cannot be shared.

GS.Contract.DataFeed project

GS.Contract.DataFeed.Server project

GS.Contract.Admin project

GS.Contract.Admin.Server project

**DataAccess** folder – Contains projects for all the different subsystem’s ResourceAccessors. Each project will contain the RA/DA interfaces and the code that implements them, plus the DataContracts, the code defining the entities used, plus other classes used by the RA/DA. Note that some of these supporting entities and classes can possibly be put into an Infrastructure folder so long as they have absolutely NO BUSINESS LOGIC in them.

GS.DataAccess.DataFeed project

IngestedDataDA.cs

GS.DataAccess.Admin project

AdminDA.cs

**Engine** folder – Contains projects for all the different subsystem’s Engines. Each project will contain the Engine interfaces and the code that implements them, plus the DataContracts, code defining the entities used, plus other classes used by the Engine. Note that some of these supporting entities and classes can possibly be put into an Infrastructure folder so long as they have absolutely NO BUSINESS LOGIC in them.

GS.Engine.DataFeed project

FeedValidityEngine.cs

GS.Engine.Admin project

FeedStatisticsEngine.cs

**Manager** folder – Contains projects for all the different subsystem’s Managers. The Manager’s service and data contracts should be in the Contracts folder. They are shared between client and server. Each project will contain the other classes used by the Manager except for those classes, DataContracts, and interfaces defined by the RA/DAs and Engines used by the Manager.

GS.Manager.DataFeed project

DataFeedsManager.cs

GS.Manager.Admin project

AdminManager.cs

**Proxy** folder -- Contains projects for all the different subsystem’s Proxys.

GS.Proxy.DataFeed project

GS.Proxy.Admin project

**Infrastructure** folder – Contains projects various Infrastructure areas that do not warrant being broken out into separate folders.

Ifx.UI project – leave here

**Infrastructure.Common** folder – Things deployed to both sides of the wire

ServiceModelEx project, due to the GenericResolver, etc. Plus, this project needs to be in a place where it can easily be removed and added due to copyright restrictions.

Ifx.Common project

**Infrastructure.Core** folder or project – Server side non-service Ifx things like caches, config helpers, mappers, security items and non-SO helpers.

Infrastructure.Contract folder or project

Infrastructure.Proxy folder or project

Infrastructure.Server folder or project

**Infrastructure.Test** folder

**Infrastructure.Host** folder

Ifx.Host.Cloud.DataFeedSvc project

Ifx.Host.Cloud.DataFeedWorker project

Ifx.Host.DataFeedServiceHost project

Infrastructure.Clients folder or project

**Utilities** folder

Utilities.Notification

Utilities.Timer

**UI** folder

UI.DataSourceSimulatorClient

UI.SysOperationUIClient

**Explore** folder – Contains exploratory coding projects.

Explore.AzTableStorage project

Revision History

V 0.1, 5-18-15, George Stevens. Initial document using Monty’s posts in the above mentioned thread.

V 0.2, 5-19-15, George Stevens. Refined concepts and namespace/solution structure based on feedback and deeper understanding of the basic organizing principles involved.