

**Civica Housing**

**New Technologies Team**

**Cx Architectural Overview and Requirements**

Version: **1.0 Published**

Author: **Chris Townsend**

**Document Information**

|  |  |
| --- | --- |
| **Details** | |
| Document Type | Technical Architecture |
| Document Name | Cx Architectural Overview and Requirements |
| Document Location | <http://svr-tfs-1/Housing%20New%20Technology/Converged%20Housing/Shared%20Documents/Forms/AllItems.aspx?RootFolder=%2fHousing%20New%20Technology%2fConverged%20Housing%2fShared%20Documents%2fArchitecture&FolderCTID=&View=%7b731FEB01%2dD2FA%2d4038%2dB85B%2d9A771EA127BD%7d> |
| Document responsibility | Victoria Parkinson |

**Version History**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date Issued** | **Author** | **Reason** | **Sections**  **affected** |
| 0.1 Draft | 28/06/2012 | Vicki Parkinson | Created template |  |
| 0.2 Draft | 29/06/2012 | Chris Townsend | Specified technologies | All |
| 0.3 Draft | 03/07/2012 | Chris Townsend | Added additional rules and descriptions | 6, 7 |
| 0.4 Draft | 03/07/2012 | Chris Townsend | Updated diagram, added further acceptance criteria | 5, 7 |
| 0.5 Draft | 04/07/2012 | Vicki Parkinson | Added Exception Handling into Acceptance Criteria | 7 |
| 1.0 Published | 05/07/2012 | Vicki Parkinson | Published following approval from Richard Chambers |  |

**Related documentation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Document Title** | **Version** | **Date Issued** | **Author** | **Hyperlink to document** |
| Development Strategy | 1 | August 2011 | Richard Chambers | [http://svr-tfs-1/Housing New Technology/Shared Documents/Development Strategy/Development Strategy at August 2011/Civica Housing Development Strategy August 2011.docx](http://svr-tfs-1/Housing%20New%20Technology/Shared%20Documents/Development%20Strategy/Development%20Strategy%20at%20August%202011/Civica%20Housing%20Development%20Strategy%20August%202011.docx) |
| UI Review | 1.0 | April 2012  Updated version required | Matt Grey | <http://svr-tfs-1/Housing%20New%20Technology/Converged%20Housing/Shared%20Documents/User%20Interface/Cx%20UI%20Review%20v1.0.docx> |
| Coding Standards |  |  | Chris Townsend | Under development |
| Civica Utilities Package Document |  |  | Roger Adams | Under development |
| Data Import Framework |  |  | Roger Adams | Under development |

**Copyright**

|  |
| --- |
| © 2012  The copyright of this document is vested in Civica UK Ltd. No part of this document may be reproduced by any means, electronic or mechanical, including photocopying, recording or any information storage and retrieval system, without prior permission in writing from Civica UK Ltd.  All trademarks are hereby acknowledged. |

**Review and Approval**

|  |  |  |
| --- | --- | --- |
| **Reviewed By:** | **Name** | **Date** |
| *Development Team Leader* | *Chris Townsend* | 29/06/2012 |
| *Analyst Developer* | *Graham Copes* | 29/06/2012 |
| *Analyst Developer* | *Roger Adams* | 29/06/2012 |
| *Development Manager* | *Vicki Parkinson* | 03/07/2012 |
| **Approved By:** |  |  |
| *Development Director* | *Richard Chambers* | 05/07/2012 |

Warning:

This document is company confidential and may only be used for the purpose for which it was supplied. It must not be disclosed to any third party without prior permission in writing from Civica UK Ltd.

**Table of Contents**

[1 Intended Audience 5](#_Toc329173051)

[2 Purpose 5](#_Toc329173052)

[3 Original Requirements for Cx 5](#_Toc329173053)

[4 Overview of Technology Stack 6](#_Toc329173054)

[5 Architectural View 6](#_Toc329173055)

[6 Technology Overview 7](#_Toc329173056)

[6.1 Silverlight 7](#_Toc329173057)

[6.2 Prism 7](#_Toc329173058)

[6.3 Managed Extensibility Framework (MEF) 7](#_Toc329173059)

[6.4 RIA Services 7](#_Toc329173060)

[6.5 .NET 7](#_Toc329173061)

[6.6 NHibernate 8](#_Toc329173062)

[6.7 Fluent NHibernate 8](#_Toc329173063)

[6.8 SQL Server 8](#_Toc329173064)

[6.9 Log4Net 8](#_Toc329173065)

[7 Code / Technology Acceptance Criteria 8](#_Toc329173066)

[7.1 Separate business logic 8](#_Toc329173067)

[7.2 Client business logic should mirror server logic 8](#_Toc329173068)

[7.3 Separate data access 9](#_Toc329173069)

[7.4 Use MVVM pattern 9](#_Toc329173070)

[7.5 Minimise use of View code-behind 9](#_Toc329173071)

[7.6 Use Visual Studio linked classes 9](#_Toc329173072)

[7.7 Use Civica utilities 9](#_Toc329173073)

[7.8 Use NHibernate 9](#_Toc329173074)

[7.9 Use Fluent NHibernate 9](#_Toc329173075)

[7.10 Use Log4Net 10](#_Toc329173076)

[7.11 Include Exception Handling 10](#_Toc329173077)

[7.12 Use MEF 10](#_Toc329173078)

[7.13 Use C# 10](#_Toc329173079)

[7.14 Use a single database 10](#_Toc329173080)

[7.15 Use a single location for parameters 10](#_Toc329173081)

[7.16 Separate business modules 10](#_Toc329173082)

[7.17 Business classes should implement DIF interfaces 11](#_Toc329173083)

[7.18 Define unit tests 11](#_Toc329173084)

[7.19 Ensure UI conforms to UI specification 11](#_Toc329173085)

[7.20 Structure styles and themes 11](#_Toc329173086)

# Intended Audience

This document is pertinent to the following audience:

* The Civica Housing Development Director who is ultimately responsible for the business case, vision and strategy for the Cx product.
* The Civica Housing New Technologies development team
* The offshore development team, responsible for producing the project deliverables
* General Civica staff members who wish to know about the project

# Purpose

The purpose of this document is to provide a high-level description of the structure, architecture and technologies used in Cx, which is the brand name of the new Housing product which is being developed by Civica.

In addition, Chapter provides the developers working on Cx with a list of rules / constraints for development, which the Civica code reviews will be checking for compliance.

# Original Requirements for Cx

The following requirements have been taken from the Development Strategy document:

1. UI Separation – The user interface must be developed to ensure clear separation from the underlying business logic. Should presentation technologies change then it must be possible to replace the UI without the need to reengineer the main system components.
2. Business logic – The underlying business logic of the system should be able to be exposed via a web service layer with appropriate authentication and security controls. This will enable functionality to be available to intranets, extranets, mobile systems etc.

# Overview of Technology Stack

|  |  |  |
| --- | --- | --- |
| Technology | Version | Function |
|  |  |  |
| Silverlight | 4 | Client technology |
| Prism | 4.0 | Navigation, dynamic module loading |
| MEF | 2 | Dependency Injection |
| RIA Services | 1 SP1 | Data transfer |
| .NET Framework | 4 | Application framework |
| NHibernate | 3.2 | Object-relational mapping |
| Fluent NHibernate | 1.3 | Mapping definitions |
| SQL Server | 2008 R2 | Database |
| Log4Net | 1.2 | Logging |

# Architectural View



# Technology Overview

## Silverlight

The client architecture of Cx is written in Silverlight, Microsoft’s Rich Internet Application framework. The current version used in Cx is Silverlight 4.  
  
More information can be found here: <http://www.microsoft.com/silverlight/>

The client architecture uses the Model-View-ViewModel pattern (MVVM) to give clear separation of GUI and code.  
  
More information can be found here: <http://en.wikipedia.org/wiki/Model_View_ViewModel>

## Prism

Prism provides guidance and core principles for creating composite applications using Silverlight.

More information can be found here: <http://msdn.microsoft.com/en-us/library/ff648465.aspx>

## Managed Extensibility Framework (MEF)

The Managed Extensibility Framework is a framework that provides composition features in order to reduce coupling between components.

More information can be found here: <http://mef.codeplex.com/>

## RIA Services

RIA Services provides a means of communication between the Silverlight user interface, the presentation logic components and the Business Services layer.

More information can be found here: <http://msdn.microsoft.com/en-us/library/ee707344(v=vs.91).aspx>

## .NET

The .NET Framework is a Microsoft application development platform. The current version used on Cx is **.NET Framework 4.**More information can be found here: <http://www.microsoft.com/net>

The CLI language used in Cx is C#. The .NET Framework includes WCF which would generally be used to provide service endpoints for non-Silverlight clients.

## NHibernate

NHibernate is an object-relational mapping (ORM) solution for .NET, and provides the bridge between the database and the business logic. The current version used in Cx is **NHibernate** **3.2**.  
  
More information can be found here: <http://nhforge.org/Default.aspx>

## Fluent NHibernate

Fluent NHibernate provides a fluent way to define NHibernate mappings. It offers compile-time checking of mappings and removes the need for XML, as well as providing mapping conventions to reduce configuration time.  
  
More information can be found here: <http://www.fluentnhibernate.org/>

## SQL Server

SQL Server provides the database platform on which the Cx database resides. The current version on which the Cx database runs is SQL Server 2008 R2.

More information can be found here: <http://www.microsoft.com/sqlserver/en/us/editions/previous-versions.aspx>

## Log4Net

Log4Net provides a logging framework for outputting log statements to a variety of output targets.

More information can be found here: <http://logging.apache.org/log4net/>

# Code / Technology Acceptance Criteria

## Separate business logic

In order to promote code reuseability, all business logic should be contained within a separate layer and present on the server.

## Client business logic should mirror server logic

Business logic on the client is acceptable (for example, where form validation is required) but these rules should not exist solely on the client and should mirror the business logic rules defined in the Business Logic Layer (BLL) on the server.

## Separate data access

Data access code should be contained within a separate layer on the server. This promotes maintainability and enables the transferral to different datasources with relative ease and minimum impact on the rest of the system.

## Use MVVM pattern

The use of the Model-View-ViewModel (MVVM) pattern promotes separation of code and design and should be adhered to at all times to maintain separation of concerns.

## Minimise use of View code-behind

In following MVVM, the vast majority of code should be included in the ViewModel. If it is necessary to include code in the code-behind file of the View, this should be for presentation-level purposes only. For example, code to give a textbox focus when the View has loaded would be acceptable in the code-behind of the View, but code to determine the visibility of a control based on particular conditions would not – this should be bound to a property in the ViewModel.

## Use Visual Studio linked classes

In order to minimise the chance of developers mistakenly working on an automatically-generated version of a shared class which would then be overwritten at next compile, Linked Classes should be used so that only one class definition exists. This class definition exists at the server and is then referenced, rather than duplicated, at compile time.

## Use Civica utilities

The Civica Housing New Technologies Server Utilities (CHNTSU) package provides data access and persistence classes for working with NHibernate, and should be used to maintain consistency with other Civica Housing New Technologies projects.

## Use NHibernate

The CHNTSU is compatible with NHibernate 3.2, and so this version should be used when working with Cx applications to maintain developmental consistency.

## Use Fluent NHibernate

All NHibernate mappings should be defined using Fluent NHibernate to provide compile-time checking of mappings and minimise the use of XML, which is unchecked and prone to error.

## Use Log4Net

Logging should be performed using Log4Net, which is the standard logging application for Civica Housing New Technologies applications.

## Include Exception Handling

All expected exceptions should be dealt with programmatically including appropriate information being given to the user and recorded by logging.

Unexpected exceptions should always result in the current user being made aware of the general nature of the fault, whilst the full detail of the exception should be included in the server logs.  Where the exception occurs in the client code and logging facilities are not available the detail must be able to be captured by the user in a form that can be reported to Civica support.  If the exception occurs on a background thread without a current user interface then the exception must be logged as a minimum requirement.

Logging of exceptions must include the full message & stack trace of all embedded exceptions.

## Use MEF

Managed Extensibility Framework (MEF) should be used for all dependency injection requirements to maintain consistency with existing Civica technologies and simpler integration with Prism.

## Use C#

The CLI supports several languages, but C# is the language that is used in Cx and should be used for all Cx development where applicable for consistency and ease of support.

## Use a single database

All Cx modules should share a single database to ease installation.

## Use a single location for parameters

Parameters across all modules should be configurable from a single location to facilitate simple administration and setup as part of implementation.

## Separate business modules

Each business module should have its own specific business logic and separate corresponding persistence module, such that modules can be added/removed as required. In designing these corresponding components it is important to ensure that the interface specifications completely encapsulate the persistence mechanism. This will ensure that if future requirements demand a change of persistence strategy or repository it can be accomplished without affecting the Business Logic or clients.

## Business classes should implement DIF interfaces

In order to support the import of business classes via the Data Import Framework (DIF), all business classes requiring import should implement interfaces defined by the DIF. This documentation is currently underway.

## Define unit tests

Unit tests should be implemented for all testable entities, with a target code coverage of 75%.

## Ensure UI conforms to UI specification

In the interests of protecting Civica’s brand and ensuring good design principles are adhered to, all UI should conform to the provided UI specification.

## Structure styles and themes

Styles should be structured such that both colours and fonts can be changed with ease in a single location. This will also prevent areas of the system from retaining out-of-date colours and fonts should a change in theme be required during or following development.