Chem4Word Build Primer

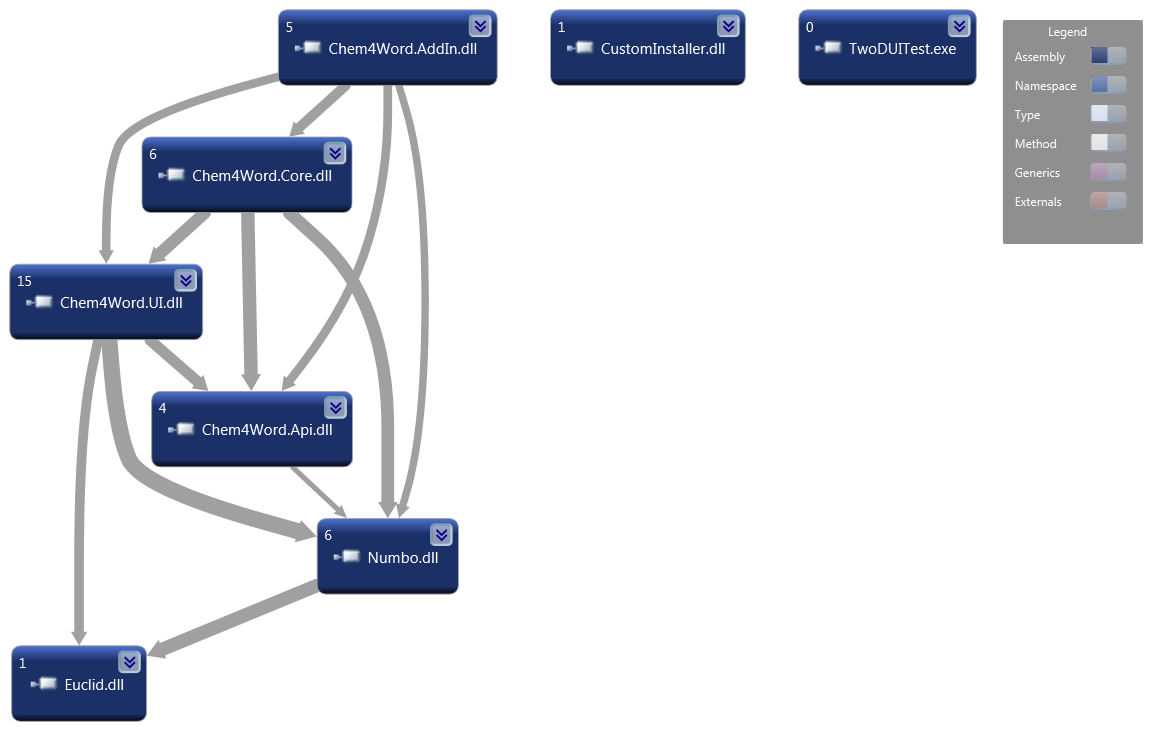
# Clyde Davies, 23/04/2012

# Getting Started

Chem4Word is a VSTO based add in for Word 2007 and later. It is written in C#. It consists of the following assemblies:

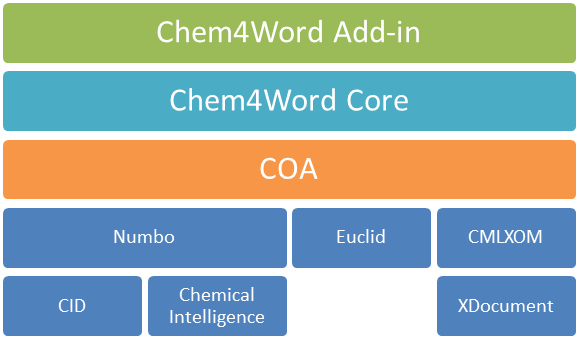
|  |  |
| --- | --- |
| Name | Purpose |
| Chem4Word.AddIn | Main add-in Module |
| Chem4Word.API | Supporting Library |
| Chem4Word.Core | Supporting Library |
| Chem4Word.Installer.CustomActions | Custom forms for Installer |
| Chem4Word.UI | Supporting Library |
| Euclid | Supporting Library: geometry services |
| Numbo | Supporting Library: chemistry services |
| TwoDUITest | Test Harness |
| Chem4Word.Setup | Windows Installer Project |

These assemblies interrelate as shown in the diagram below (generics and externals omitted for clarity):



### Basic Architecture

The architecture of Chem4Word is layered: various services reside in assemblies that are concerned with various kinds of information. At the bottom level are geometry and chemical services (Numbo and Euclid), and at the top level are services concerned with connecting to the application itself.



# Getting the source code

The project has its own [CodePlex project](http://chem4word.codeplex.com/). You can download all the source code from there as a ZIP file and unzip it to its own directory.

If you wish you can [connect directly to the source code repository](http://codeplex.codeplex.com/Wikipage?title=Source%20Control%20Clients). The location of the repository is at [**https://hg.codeplex.com/chem4word**](https://hg.codeplex.com/chem4word)**.**

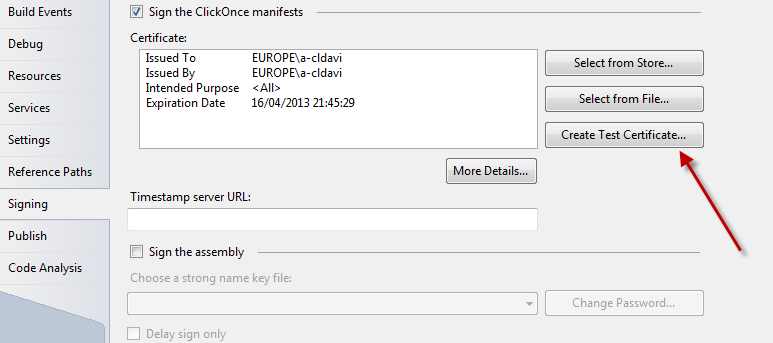
# Build the project

To open it you will need to use Visual Studio 2010. Locate the **Chem4Word.sln** file and double click it. Then attempt to build the solution.

## Possible issues

Before building the system you will need to install **Chem4Word.Font.DLL**. Download this from <http://research.microsoft.com/en-us/downloads/afbe4907-66bc-4d02-95e2-263c2bbea341> and copy the DLL to the **Chem3Word.AddIn/bin/debug** subfolder.

You may also get the error: ‘Unable to find manifest signing certificate in the certificate store’. This error does not prevent the solution from compiling but it does prevent the installation package from being built. You can circumvent this error is to create a temporary certificate by right clicking on the Chem4Word.AddIn project, then selecting the Signing tab:



Right click on the solution file in the Solution Explorer and select **Clean Solution** and then **Build Solution** (not Rebuild). The solution will now build successfully.

# System Components

A class-by-class account of the system is out of scope of this document. Instead, we’ll give a high level view of what each assembly does.

The best way to explore Chem4Word’s code is to open the solution and click **Architecture** 🡪 **Generate Dependency Graph**🡪**By Assembly**. This will give you an interactive graph within which you can ‘drill-down’ to various classes, methods and even code.

## Chem4Word.AddIn

This is the main add-in module created by Visual Studio Tools for Office. This contains some top-level events that are used to initialise the plug in. *ThisAddIn.cs* hosts these events and the initialisation code. *ChemistryRibbon.cs* defines the Word Ribbon associated with Chem4Word.

## Chem4Word.Core

This contains several classes, but by far the most important are **CoreClass**, ChemistryDocument, and **ChemistryZone**. **CoreClass** acts as the ‘controller’ in a relationship between the Word API and the Chem4Word UI. As such it also manages the relationship between the underlying **ChemistryDocument** and the Word Document.

**ChemistryDocument** manages the interplay between the Word Document’s hosted Content controls and the underlying context objects.

**ChemistryZone** is the presentation of 1-D (textual) or 2-D (structural) chemistry within a document. Each **ChemistryZone** is associated with an underlying **ChemistryDocument** and an overlying content control.

## Chem4Word.API

This library provides interface definitions for consumption in other components. There is also a **ChemistryZoneProperties** class which persists information about a given **ChemistryZone**.

## Chem4Word.UI

This is a large library and deals with many of the user-interface operations in Chem4Word, save for those embodied in the Ribbon.

The **ChemCommands** class provides basic structural manipulation. These operations are used in the chemical editor.

In the Converters namespace, are a set of classes that provide conversions for WPF data bindings. These are now not used and may be disregarded as far as current functionality goes.

The Import namespace contains a set of classes associated with importing CML. The most important of these is the **ImportMediator**. Importing chemistry in Chem4Word is a two stage process. Firstly, the foreign format is converted into [*CML-Lite*](http://www.jcheminf.com/content/3/1/39)*[[1]](#footnote-1).* The CML-Lite is then validated and stored natively: there is no internal ‘model’ other than the actual CML itself.

The **Navigator** namespace hosts one of the most important UI components. The Chem4Navigator is a WPF user control that displays structures in a single column format. The code-behind file therefore has methods that allow this class to be ‘wired up’ to the active Word document and to respond to selection and manipulation of chemistry objects.

The **TwoD** namespace is concerned with the presentation of 2-D chemistry, that is, chemical structures. Most of the classes are concerned with the visual display and manipulation of chemistry within a two-dimensional WPF environment. Many classes inherit from **AbstractNodeControl** (for atoms) or **AbstractEdgeControl** (for bonds).

There are also **PeriodicTableElementChooser** and **PeriodicTableChooser** XAML components to allow selection of atoms visually.

The final class of interest is the **DocumentHostControl.** Thisis a user control which hosts a Canvas component that allows chemical objects to be arranged on its surface directly.

## Euclid

As its name suggests, Euclid is concerned with geometry. It is a low-level library which does not recognise chemical concepts. Its main purpose is to provide services to Numbo.

## Numbo

Numbo (.NET Jumbo) is an adaptation of the the *Java Universal Molecular Browser for Objects*. Numbo overlaps with JUMBO to a degree: it contains a subset of JUMBO but also adds some extra functionality. Numbo consists of two main namespaces:

### CML

This namespace contains a set of objects that wrap the undying CML (persisted in an XDocument object).

### COA

This namespace contains some classes critical to Chem4Word. **ChemicalIntelligence** is a query interface over a chemical object. For instance, it allows one to determine whether certain chemical operations are possible. **CID (Chemical Interface Definition)** then allows manipulation of the chemistry. All operations are carried out directly on the underlying CML.

**ContextObject** is a wrapper around the XDocument that stores the CML for a given chemical entity. The **DepictionOption** **class** thengoverns how the context object is presented to via the user interface: as a 1-D or 2-D representation.

## Chem4Word.Setup

This is the installer package which depends upon the output from **Chem4Word.AddIn** and **Chem4.Installer.CustomActions.** The latter library contains custom forms used by the Windows Installer.

There is also a set of CML files included in this package. These are associated with the Smart tag and contain a series of predefined molecules for the Smart Tag to recognise. Adding a new Smart tag to recognise involves creating the CML file and then amending the **smart-tag-dict.xml** thus by adding a Term element:

 <Term MoleculeID="newcomp" Value="abcamine" />

if the CML file added is called **newcomp.cml.**

1. CMLLite is a collection of definitions and processes which provide strong and flexible validation for a document in Chemical Markup Language (CML) [↑](#footnote-ref-1)