Welcome to EDXLSharp

What is EDXLSharp?

EDXLSharp is a C# / .NET 3.5 implementation of the OASIS Emergency Data Exchange Language (EDXL) family of standards. The purpose of these libraries are to allow developers to:

- Parse EDXL Messages from a string / underlying stream
- Programmatically create EDXL messages
- Validate EDXL Messages to the schema
- Validate that EDXL Messages conform to the additional business rules specified in the standards documentation
- Write EDXL messages to a string / underlying stream

This set of libraries supports CAP v1.2 and EDXL-DE v1.0.

Future released will include EDXL-HAVE v1.0, EDXL-RM v1.0, EDXL-SitRep v1.0, and EDXL-TEP v1.0.

This code was generated as part of the internally funded research & development program, the MITRE Innovation Program (MIP) as MITRE Sponsored Research (MSR) effort called IC.NET.

Getting Started

1. Install EDXL Sharp
    a. Download the latest version of the msi installer from the project download section
    b. Run the installer
2. Start a new project in Visual Studio
3. In your newly created project, right click on “References” and select “Add Reference”
4. In the Add Reference Window, select the “Browse” Tab
5. Navigate to the directory where EDXLSharp was installed and select all of the dll files
6. You are now ready to include the EDXLSharp namespace in your project

Tutorial: How to Use EDXLSharp

Programmatically Creating a Message & Writing it to a File

1. Begin by completing the steps in the Getting Started Section. For this example you can create a simple C# console application.
2. For this example we will programmatically create an EDXL-DE message. First, include the namespaces for EDXLSharp:

```
using EDXLSharp;
using EDXLSharp.EDXLDELib;
```

3. In your Main function create and instantiate a new empty EDXL-DE object. In this example we have called our EDXLDE object 'DEXMLObj':

```
EDXLDE DEXMLObj = new EDXLDE();
```

4. The EDXL-DE standard has a number of required elements. The sample code below will initialize the required elements through the EDXLDE public accessors:

```
DEXMLObj.DistributionStatus = StatusValue.Test;
DEXMLObj.DistributionType = TypeValue.Report;
DEXMLObj.CombinedConfidentiality = "Unclass";
DEXMLObj.DistributionID = "EDXL Sharp Test Message";
DEXMLObj.SenderID = "test@edxlsharp.codeplex.com";
DEXMLObj.DateTimeSent = DateTime.UtcNow;
```

5. Now let's add some content. Let's say we have an XML file on our disk called "test.xml". We want to add this as a contentobject in the EDXL-DE message:

```
ContentObject obj = new ContentObject();
obj.XMLContent = new XMLContentType();
XElement xe = XElement.Parse("test.xml");
obj.XMLContent.AddEmbeddedXML(xe);
```

6. Now we have a simple EDXL-DE message. Let's write it to an underlying stream; in this case a filestream to the file "DEXMLObj.xml":

```
StreamWriter sw = new StreamWriter("DEXMLObj.xml");
sw.Write(DEXMLObj.WriteToXML());
sw.Flush();
sw.Close();
```

Parsing a Message From an Underlying Stream

1. Begin by completing the steps in the Getting Started Section. For this example you can create a simple C# console application.
2. For this example we will read an EDXL-DE message from an underlying stream. First, include the namespaces for EDXLSharp:

```
using EDXLSharp;
using EDXLSharp.EDXLDELib;
```

3. We will parse an EDXL-DE message from a file called "DEExample.xml"

```
StreamReader sr = new StreamReader("..\\..\\..\\TestData\\DEExample.xml");
EDXLDE de = new EDXLDE(sr.ReadToEnd());
sr.Close();
```

4. Alternatively let's say you had an EDXL-DE message stored in a string called "s" and wanted to read it in:

```
EDXLDE de = new EDXLDE();
de.ReadXML(s);
```