

ChemADVISOR Integration Project

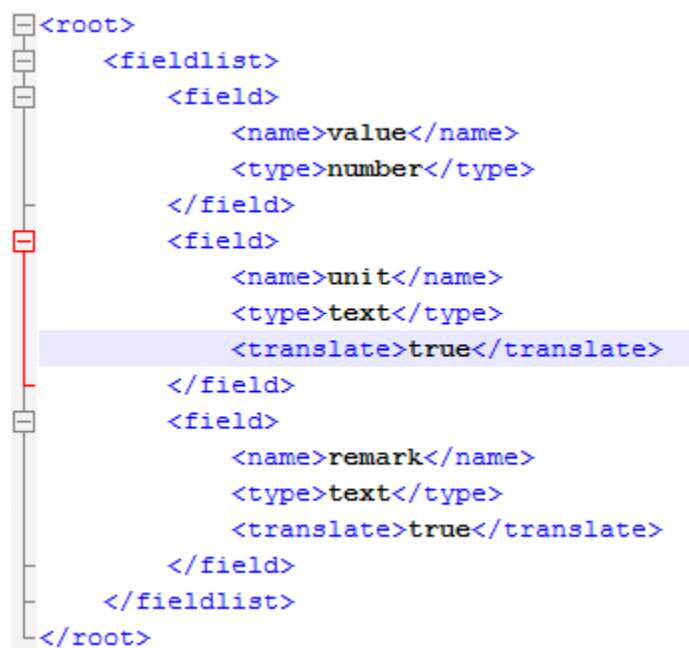
Purpose

The purpose of this document is to describe a library to aid application developers with processing the xml field of the ListData Table of a Version 4.X LOLI Database. Importantly, this document does not describe the regulatory meaning of any information in LOLI, nor does it describe the ListData Table in general. Contact ChemADVISOR Support (support@chemadvisor.com) or refer to LOLI Technical Documentation with any questions regarding this document.

Motivation for this library

A key benefits of using XML is that standard parsers are supplied with mainstream programming languages and the XML Document Object Model (DOM) is already familiar to many developers. Using XML allows LOLI to store a variety of information using a single database schema. Well-formed XML, however, does not imply a universal semantic. Being able to parse an XML document is not the same thing as consuming the information contained therein.

In LOLI Version 4.x databases, all records in the ListData table for a given ListID, have the same XML structure. That structure is defined within the ListNames_Control table's XML_Admin field. The records are keyed by ListID. Below is an example of a common LOLI XML_Admin value.



```
<root>
  <fieldlist>
    <field>
      <name>value</name>
      <type>number</type>
    </field>
    <field>
      <name>unit</name>
      <type>text</type>
      <translate>true</translate>
    </field>
    <field>
      <name>remark</name>
      <type>text</type>
      <translate>true</translate>
    </field>
  </fieldlist>
</root>
```

The above example describes a list of fields designated by <field> elements with an indication of the given field's name (<name>) and data type (<type>). Below is an example of a ListData.XML value that corresponds to the above XML_Admin example.

```

<root>
  <row>
    <value>0.020</value>
    <unit>ppm</unit>
    <remark>final</remark>
  </row>
  <row>
    <value>0.020</value>
    <unit>ppm</unit>
    <remark>final</remark>
  </row>
</root>

```

LOLI allows more complex information to be stored in xml elements of type number. LOLI allows ranges and inequality operators. The following are all valid content in a LOLI xml number field*:

>10

<=20

10 - 20

-2 - -1

>= -2 - <= -1

* The '>' and '<' signs would be '>' and '<,' respectively in the actual xml, but it's easier to read within this document using the literals.

LOLI Integration

This project contains Factory classes for LoliNumber and LoliListDataXml along with the associated classes of object that are instantiated by the factory methods. These classes may be used by application developers to process LOLI ListData.XML.

LoliNumber

The LoliNumberFactory class is used to create an object representation of a valid LOLI number field.

LoliNumberFactory exposes a single shared function, namely TryParse(). This is useful when processing LOLI xml at the field-by-field (element-by-element) level. The signature for TryParse is:

```
Public Shared Function TryParse(ByVal s As String, ByRef exp As ILoliValidatableField
) As Boolean
```

It is modeled after the .NET method Double.TryParse(). If TryParse returns True, then the variable passed in as exp will contain a valid instance of either LoliNumberTerm or LoliNumberRange. Both inherit from LoliNumber and both implement the ILoliValidatableField and ILoliNumberType interfaces.

For an example of how to use TryParse, please refer to the ExampleUse function in LoliNumberTest.vb.

Note: When using this to parse distributed LOLI data, all number fields should always parse successfully. In other words, TryParse should always return true.

Ranges will always have the smaller value stored in Value1. For instances of LoliNumberTerm, LoliNumberTerm.IsRange() will return false. LoliNumberTerm instances will return null for both Operator2() and Value2(). The source code contains more details and a more formalized explanation.

LoliListDataXmlFactory

To process a single LOLI xml field as a .NET DataTable (one row for each <row> element) then use the LoliListDataXmlFactory.

Important: The LoliListDataXmlFactory turns a single record from ListData.Xml into a table. It does not create a table that represents a LOLI List. There are two distinct behaviors TryParse and TryParseWithNumberExpansion. The methods are overloaded to accept xml inputs as either string data or as an XmlDocument. They are provided both as shared (static) methods and instance methods. Both the shared and instance methods provide identical functionality, however, depending on the use case, one may be preferred over the other. All LoliListDataFactory functions work like the .NET Double.TryParse() method. They do not throw exceptions, rather they return True on success and False on error. The dataTable variable is passed in by reference and set to a valid instance of a DataTable.

Class Methods

To use the class methods, you will need to pass the ListNames_Control.XML_Admin record with each call. Use the class methods for processing only a few records. For instance, if an application retrieved the ListData for a single CAS record from a single list using the LOLI Rest Api.

There are four static methods are:

```
Public Shared Function TryParseWithNumberExpansion(ByVal xmlAdmin As String, listData
Xml As String, ByRef dt As DataTable) As Boolean
```

```
Public Shared Function TryParseWithNumberExpansion(ByVal xmlAdmin As XmlDocument, lis
tDataXml As XmlDocument, ByRef dt As DataTable) As Boolean
```

```
Public Shared Function TryParse(ByVal xmlAdmin As String, listDataXml As String, ByRe
f dt As DataTable) As Boolean
```

```
Public Shared Function TryParse(ByVal xmlAdmin As XmlDocument, listDataXml As XmlDocu
ment, ByRef dt As DataTable) As Boolean
```

TryParseWithNumberExpansion – creates a .NET DataTable representation of a ListDataXml record with all elements of type=Number expanded into 4 columns in addition to a column containing the element as stored in the records. The DataTable columns for the element <limit> of type=number would be: limit, limit_OP1, limit_NUM1, limit_OP2, limit_VAL2. So you get the field as named, plus appended with “_OP1, _NUM1, _OP2, _NUM2”. These correspond to data from using LoliNumberTryParse and are similar to the format used in the deprecated LOLI “enfielded” tables. The image below is how the table looks using the DataSet Visualizer from Microsoft Visual Studio.

	limit	limit_OP1	limit_NUM1	limit_OP2	limit_NUM2
	>=-3.2 - <=5	>=	-3.2	<=	5
	0.25		0.25		
	-10		-10		
	1		1		
	0.25		0.25		
	1		1		
	2.5		2.5		
	0.25		0.25		
	0.025		0.025		

TryParse – creates a .NET DataTable that does **not** expand the LOLI number fields found in ListData.XML. The image below is how the same input will look using the DataSet Visualizer from Microsoft Visual Studio. Note that the only difference is the absence of the expanded columns.

limit
>=-3.2 - <=5
0.25
-10
1
0.25
1
2.5
0.25
0.025

Instance Methods

To use the instance methods, create an instance of LoliListDataXmlFactory. The constructor requires the appropriate ListNames_Control.XML_Admin record. The TryParse calls do not require XML_Admin. The instance methods may be more appropriate for processing all the records associated with a given LOLI list. The output of these methods are the same as the class methods.

New is overloaded to accept either a string or an XmlDocument.

```
Sub New(xmlAdmin As String)
Sub New(doc As XmlDocument)
```

```
Public Function TryParseWithNumberExpansion(listDataXml As XmlDocument, ByRef dt As DataTable) As Boolean
Public Function TryParseWithNumberExpansion(listDataXml As String, ByRef dt As DataTable) As Boolean
```

```
Public Function TryParse(listDataXml As String, ByRef dt As DataTable) As Boolean
```

```
Public Function TryParse(listDataXml As XmlDocument, ByRef dt As DataTable) As Boolean  
TryParseWithNumberExpansion
```

Refer to the explanation of the class method of same name.

TryParse

Refer to the explanation of the class method of same name.

Source Code Provided

This project is provided in source code under The MIT License and is copyrighted by ChemADVISOR, Inc. A copy of The MIT License is included with each source file and may also be found at <http://opensource.org/licenses/MIT>. The source code includes both the library and its n-unit test cases. The nunit-lite dll is distributed with this project. It is copyrighted by Charlie Poole. A copy of the n-unit license is included in the project and may also be found at <http://nunit.org/nuget/nunitlite-license.txt>.