

# XML and Revelation: A White Paper

Mike Ruane and Sean FitzSimons  
Revelation Software

## XML and Revelation

This white paper is intended as an introduction to XML for Revelation Software developers and users. It will inform the reader of:

- What is XML
- Why Use XML?
- Terms and Definitions
- Using XML in OpenInsight
- XML Tools in OpenInsight

## What is XML?

XML stands for Extensible Markup Language. It is based upon an earlier markup language called SGML, or Standardized General Markup Language. A markup language is a special set of indicators, called tags, that indicate how information should be interpreted. XML is designed to describe data. XML uses a DTD or Schema to describe data. An XML file is a Document, made up of Tags that describe elements, that may or may not have Attributes.

## Why Use XML?

First, a few facts about XML. It is not a programming language. It is a markup language only, used to describe data. In order to manipulate XML one needs to use a programming language, such as Basic+. Also, XML will not replace HTML. HTML is used to mark up documents for display purposes. XML is used to markup documents for data purposes only.

Why should Revelation developers and users be concerned about XML? There are a number of reasons.

- Structured text Format.
- Designed with the Internet in mind: As more and more business is run across the internet, and as more information exchange occurs, XML will probably be the format of choice, instead of CSV files, EDI layouts and the like.
- Processing technology is widespread and cheap.
- Human Readable: If you've ever looked at a flat file, but you don't know the file layout, it is very difficult to determine where one field ends, and where the next begins. It's also difficult to determine which field is used for what purpose. That's not true with XML.
- Schema Agreement allows exchangeable documents.
- Enables E-business.
- Easily Internationalized.
- Easy for MV Developers – Hierarchical  
XML can represent our Multivalued data very well.
- XML and MV: Both delimiter-based.

- XML Schema and MV Dictionary – Both Metadata.
- XML embraces Multivalued Data.
- MV syntax ideal for manipulating XML.
- Industry is spending billions to be in the delimited solution space and we are there now.

### **XML and Revelation Developers**

XML is easy for Revelation developers. XML is self-describing data. It is a series of definitions of data, and then the data - sound familiar? It's the same model used by Revelation Software since its inception over 20 years ago. XML will not replace databases. In fact, there are products made specifically to be databases for XML. Revelation is an excellent database for XML.

## **Terms and Definitions**

There are a number of key terms that must be understood when working with XML.

- Element
- Attributes
- Document
- NameSpaces
- Schema
- DTD
- SAX
- DOM

### **Element**

An element is the basic unit in an XML document. It contains a Start Tag, an End tag, and data between the two.

E.g.        <tag> data </tag>

Tag Names are Case Sensitive

E.g.        <City> does not equal <CITY> does not equal <city>

Tags can be empty

E.g.        <due\_date/> instead of <due\_date></due\_date>

## Attributes

An attribute is a name-value pair separated by an equal sign '='. Attributes are optional; they don't need to be a part of an XML file. Attributes can accept default values; Elements cannot.



```
- <row ID="392">
  <author>John Galsworthy</author>
  <book_id>392</book_id>
  <check_out_date />
  <due_date />
  <Qtitle>Four Short Plays and a TD</Qtitle>
</row>
- <row ID="402">
  <author>John Galsworthy</author>
  <book_id>402</book_id>
  <check_out_date />
  <due_date />
  <Qtitle>Dream a Little Dream of me</Qtitle>
</row>
```

## Document

A Document is an XML element that may have nested XML elements. It must be well formed. By well formed, we mean it must always have a matching closing tag, and it must be completely nested.

So

“<foo>..<bar>..</bar>..</foo>” works and is well-formed, but the following is not

“<foo> <bar> </foo> </bar>”

The image below shows an example of a well-formed XML document.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <CATALOG>
- <CD>
  <TITLE>Empire Burlesque</TITLE>
  <ARTIST>Bob Dylan</ARTIST>
  <COUNTRY>USA</COUNTRY>
  <COMPANY>Columbia</COMPANY>
  <PRICE>10.90</PRICE>
  <YEAR>1985</YEAR>
</CD>
</CATALOG>
```

The next image, however shows XML that is not well-formed. Note that the final CD and YEAR tags are out of order.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<CATALOG>
<CD>
  <TITLE>Empire Burlesque</TITLE>
  <ARTIST>Bob Dylan</ARTIST>
  <COUNTRY>USA</COUNTRY>
  <COMPANY>Columbia</COMPANY>
  <PRICE>10.90</PRICE>
  <YEAR>1985</CD></YEAR>
</CATALOG>
```

When you look at this document in Internet Explorer, the following error appears:

The XML page cannot be displayed

Cannot view XML input using XSL style sheet. Please correct the error and then click the [Refresh](#) button, or try again later.

---

**End tag 'CD' does not match the start tag 'YEAR'. Error processing resource 'file:///C:/temp/xample2.xml'. Line 9, Position 15**

```
<YEAR>1985</CD></YEAR>
-----^
```

## Namespace

A Namespace is a collection of names that can be used as Element or Attribute names in an XML document. The Namespace is identified by a Uniform Resource Identifier (URI). A URI can be either a Uniform Resource Locator (URL) or Uniform Resource Number (URN). A URI can be real or made up - it does not really have to exist. Finally, a namespace can be declared explicitly or by Default.

## XML Example

```
<?xml version="1.0" encoding="windows-1252" ?> Prolog
- <BOOKS xmlns:rti="http://www.revelation.com/XMLSchema" rti:noNamespaceSchemaLocation="MCBOOKS_schema.xsd">
- <row ID="392">
  <author>John Galsworthy</author>
  <book_id>392</book_id> Elements
  <check_out_date />
  <due_date />
  <Qtitle>Four Short Plays and a TD</Qtitle>
</row> Attribute
- <row ID="402">
  <author>John Galsworthy</author>
  <book_id>402</book_id>
  <check_out_date />
  <due_date />
  <Qtitle>Dream a Little Dream of me</Qtitle>
</row>
```

## Schema

A Schema is an XML-based syntax for defining how an XML document is marked up. This is an external file; that is, it is not located within the XML file. A schema allows the use of Namespaces. Schemas support datatypes, and can define complex structures.

## Key Definitions: Schema Example

```
<?xml version="1.0" ?>
- <rti:schema xmlns:rti="http://www.revelation.com/XMLSchema" rti:noNamespaceSchemaLocation="c:\temp\ebooks_schema.xsd">
- <rti:annotation>
  <rti:documentation xml:lang="en">BOOKS schema created by the OpenInsight Schema Creator. Copyright 2002. All rights reserved.</rti:documentation>
</rti:annotation>
<rti:element name="BOOKS" type="OIRowType" />
- <rti:complexType name="OIRowType">
  <rti:element name="author" type="rti:string" />
  <rti:element name="book_id" type="rti:integer" />
  <rti:element name="check_out_date" type="rti:date" />
  <rti:element name="due_date" type="rti:date" />
  <rti:element name="Qtitle" type="rti:string" />
</rti:complexType>
</rti:schema>
```

## **DTD**

DTD stands for Document Type Definition. A DTD validates XML and SGML Documents. DTDs do not follow XML rules - they don't support DataTypes, and they can't specify complex relationships. In general, DTD is older technology that is falling out of use. OpenInsight's XML implementation does not use or support DTDs.

### **DTD Example**

```
<!ELEMENT mailAddress (name, address, zipcode)>  
<!ELEMENT name (#PCDATA)>  
<!ELEMENT address (#PCDATA)>  
<!ELEMENT zipcode (#PCDATA)>
```

## **SAX**

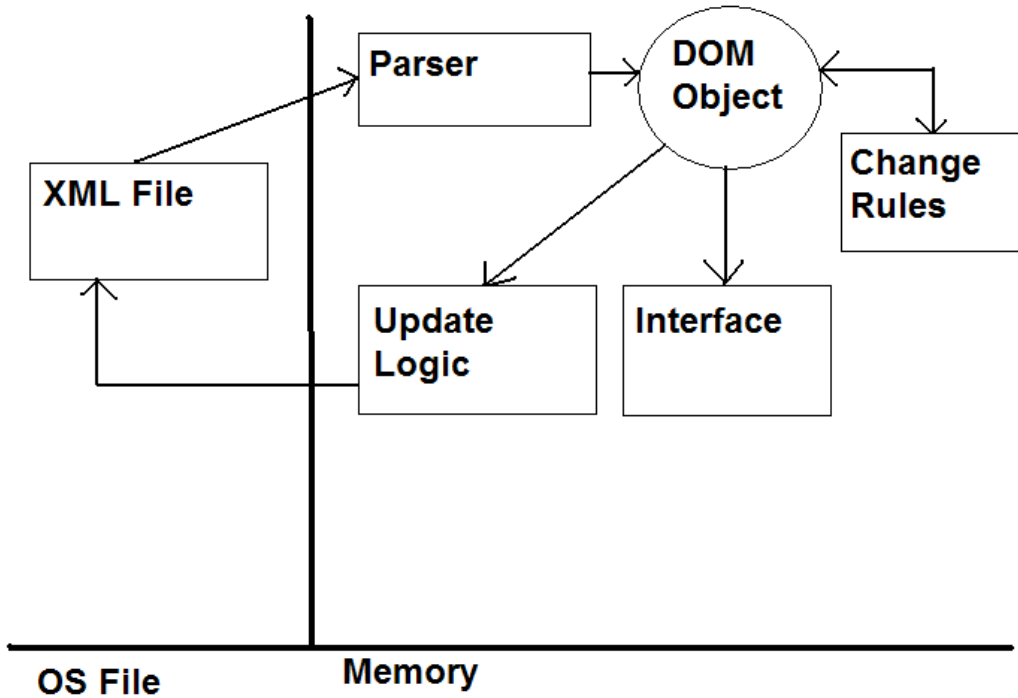
SAX stands for Simple API for XML. It is used by event-based applications. It's primarily a Java interface. With SAX there is no need to load whole XML file into memory. However, SAX is read only!

## **DOM**

DOM stands for Document Object Module. This is a process that converts an XML document into a collection of Objects. DOM treats a document as a Hierarchical structure. DOM is also known as 'Random Access' method. DOM is the method that is used by OpenInsight.

## XML Application Design: DOM

The flowchart below shows how DOM is implemented in OpenInsight. An XML file is read into memory and parsed into logical records. These records are then stored either in memory, or written to an OpenInsight file. Data from either this file, or other, existing OpenInsight files can then be written back out to an XML file.



## Using XML in OpenInsight

OpenInsight developers can create their own XML routines - it's fairly easy to do. Revelation Software has also provided a number of routines to assist in the importing and exporting of XML data. All of the programs documented below are provided, with source code, as of OpenInsight version 4.1.

### XML\_Importer

Instead of creating a template every time you want to import XML data into OpenInsight, you can call the XML\_IMPORTER routine directly. It is a function, and has the following syntax and parameters:

*XML\_Importer(parent, XML\_File, Batch\_Flag, Schema\_Name, key\_tag, target\_tags, Dict\_Mappings, OI\_Data\_File, Create\_Flag, Template\_Name, Storage\_Opts)*

Parameter	Description
<b>Parent</b>	The parent window of the calling process.
<b>XML_FILE</b>	The name of the operating system file you will be importing from.
<b>Batch_Flag</b>	A Boolean value indicating whether or not the program is being run in batch mode. In batch mode, the program will not display informational messages. Pass a 1 to indicate batch mode, and a 0 or null for interactive mode.
<b>Schema_Name</b>	The name of the Schema file associated with the XML file.
<b>Key_Tag</b>	The tag in the XML file that will be used as a key in the OpenInsight file.
<b>Target_Tags</b>	A value-mark delimited list of tags that will be extracted and put into the OpenInsight file.
<b>Dict_Mappings</b>	A value-mark delimited list of dictionary field names that the target tags will be mapped into. Based upon Storage_Opts, this parameter may be null.
<b>OI_Data_File</b>	The name of the OpenInsight file into which the XML data will be placed.
<b>Create_Flag</b>	A Boolean flag indicating whether or not the OI_Data_File should be created. Note: This functionality will not work in Runtime versions of OpenInsight.
<b>Template_Name</b>	The name of an import template that has been saved. If you want to run imports programmatically, provide this parameter, otherwise leave it null.
<b>Storage_Opts</b>	Indicates how data will be stored in fields in the OpenInsight file. It can have one of three possible values.
	C Create New 'F' type fields. Note: Cannot be implemented in Runtime versions of OpenInsight.
	E Map to Existing 'F' type fields.
	D Create a new 'F' type field with embedded with raw XML; also create new Symbolic fields to extract the XML data. Note: Cannot be implemented in Runtime versions of OpenInsight.

### **Extract\_XML\_Schema\_Name**

This program will extract the name of an XML file's schema when passed the XML file. It is a function, and has the following syntax and parameters:

*Extract\_XML\_Schema\_Name(XML\_File, XML\_Rec)*

<b>Parameter</b>	<b>Description</b>
<b>XML_File</b>	The XML file path and name.
<b>XML_Rec</b>	The XML file that has been read into memory.

### **Get\_XML\_Value**

This function is used to extract tag values from records that have raw XML data stored in them. It is a function, and has the following syntax and parameters:

*Get\_XML\_Value(Target\_Tag, XML\_Rec)*

<b>Parameter</b>	<b>Description</b>
<b>Target_Tag</b>	The tags within the XML from which you wish to extract values.
<b>XML_Rec</b>	The XML record containing the data you wish to extract.

### **Inet\_OI\_XML**

This program will create XML and XML Schema documents. It is a function, and has the following syntax and parameters:

*Function Inet\_OI\_Xml(Request, Cmd, Dosfile, Select Stmt, Template\_Name)*

<b>Parameter</b>	<b>Description</b>
<b>Request</b>	The CGI parameter passed when called through the Internet. If this program is not called via the Internet, this parameter should be null.
<b>Cmd</b>	A List statement containing the OpenInsight table and field names to be extracted from the database and published as XML.
<b>Dosfile</b>	The full path and filename of the XML file to be published.
<b>Select Stmt</b>	Optional. An RLIST select statement may be used to select subsets of the data to be extracted.
<b>Template_Name</b>	The name of an export (XML Publisher) template that has been saved. If you want to run exports programmatically, provide this parameter, otherwise leave it null.

## **Create\_XML\_XSD\_Schema**

This program will create will create XML Schema definition documents. It is a function, and has the following syntax and parameters:

*Create\_XML\_XSD\_Schema(TableName, Rpt\_Fields, Dos\_Path, DosFile)*

<b>Parameter</b>	<b>Description</b>
<b>TableName</b>	OpenInsight table to be published as XML.
<b>Rpt_Fields</b>	A value mark delimited list of dictionary field names to be included in the published document.
<b>Dos_Path</b>	Unassigned. Pass as a null parameter.
<b>DosFile</b>	The full path including filename of the XML file for which the XSD is being created.

## XML Tools in OpenInsight

As of OpenInsight version 4.1, a new XML Workspace was introduced into the toolset. The XML workspace consists of 5 tools: the XML Importer, the XML Publisher, the XML Explorer, the XPATH window, and the XML Configuration window. These tools are well documented in the OpenInsight Online Documentation, so we'll just describe them briefly here.

### XML Importer

The XML Importer is a front-end to the XML\_IMPORTER subroutine. It is used to create templates that are used to import XML data into OpenInsight files. When an XML import template is created, it can either be run, saved to file, or both.

Please note that this tool is only available on development copies of OpenInsight; it will not work on Runtime systems.

### The XML Importer Window:

The XML Importer window contains the following elements:

- Filename: [Text Input] [Options]
- Schema: [Text Input] [Options]
- Tag to Use as Key: [Dropdown Menu]
- OI Filename: [Text Input] [Options]
- Create New File
- Target Tags table:

	Tag Names	Mapped Dictionary Fields
1		
2		
3		
4		
5		
6		
7		
8		
- Data Storage:
  - Store XML, Create Symbolics
  - Store in Existing 'F' type Fields
  - Create New 'F' type Fields
- [Import]

<b>Control</b>	<b>Description</b>
<b>Filename</b>	The OS-based XML filename. Clicking on the Options button next to the edit line will display a dialog box allowing the user to choose an Operating System filename.
<b>Schema</b>	The name of the schema associated with the XML file that was entered in the Filename field. Clicking on the Options button next to the edit line will display a dialog box allowing the user to choose an Operating System filename.
<b>Tag to Use as Key</b>	The name of the tag in the XML file that will be used as a key in the OpenInsight file. If there is no tag in the XML file that can be used as the key, the user can choose 'Sequential Key Counter' from the drop down list.
<b>OI Filename</b>	The OpenInsight file into which the XML data will be stored. Clicking on the Options button next to the edit line will display a popup of existing OpenInsight tables available.
<b>Create New File</b>	When entering a filename that does not already exist, click on this checkbox. Doing so will create a new table in the current OpenInsight database being worked on and attach the file.
<b>Target Tags</b>	Edit table to determine which XML tags to import into the OpenInsight file. If the OpenInsight file is being created the Tag Name will be the only column displayed. This is because if it is a new table, there are no dictionary items to map to.
<b>Data Storage</b>	Indicates how the XML data will be stored in the OpenInsight files. The choices are: <ul style="list-style-type: none"> <li>• store raw XML in the first available field position, and create symbolic fields to display the data;</li> <li>• store the data in existing fields that have been mapped;</li> <li>• store the data in newly created F type fields.</li> </ul>

## **XML Publisher**

The XML Publisher is a front-end screen to the INET\_OI\_XML subroutine. It is used to create templates that are used to export data from OpenInsight files to XML files. When an XML Publisher template is created, it may be run, saved to file, or both.

Please note that this tool is only available on development copies of OpenInsight; it will not appear in Runtime systems.

### **The XML Publisher Window:**

The screenshot shows the XML Publisher window with the following components:

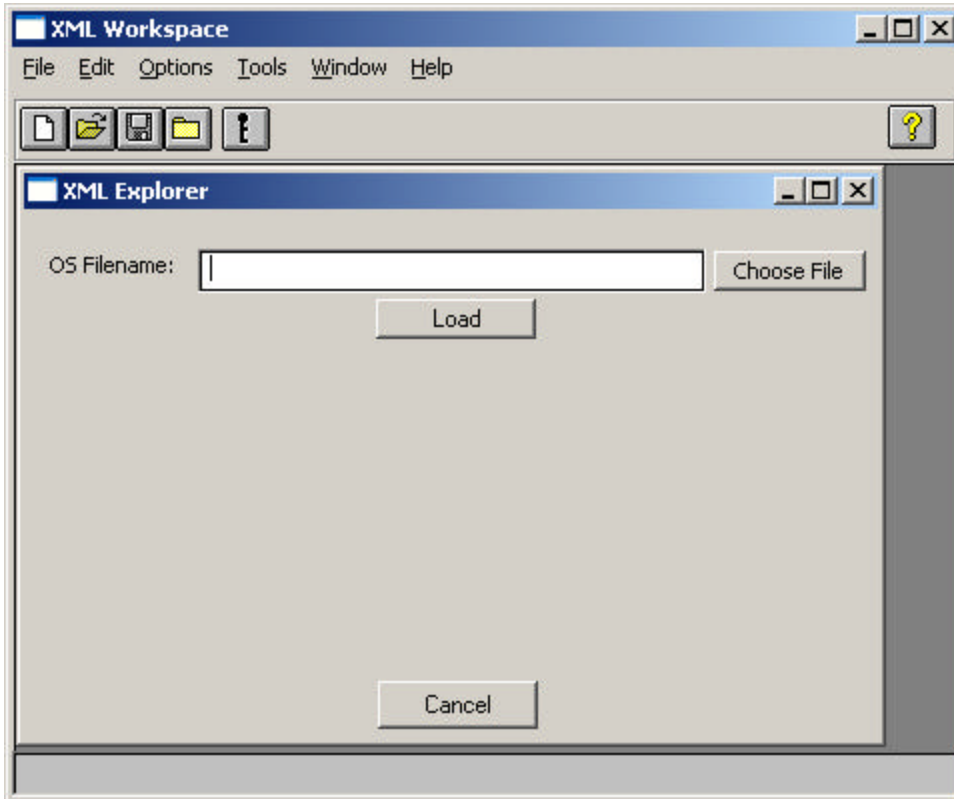
- Tablename:** A text input field followed by an **Options** button.
- Available Fields:** A large empty list box on the left.
- Selected Fields:** A large empty list box on the right.
- Navigation Buttons:** Four buttons between the list boxes: **>**, **>>**, **<**, and **<<**.
- Selection Criteria:** A section containing a **Select Statement** text box, a **Build Statement** button, and a **Check Syntax** button.
- OS Filename:** A text input field followed by a **Choose File** button.
- Bottom Buttons:** **Generate XML** and **Clear** buttons.

<b>Control</b>	<b>Description</b>
<b>Tablename</b>	The name of the OpenInsight file whose data will be exported.
<b>Available Fields</b>	A list of all fields as defined in the dictionary for the table in the Tablename field.
<b>Selected Fields</b>	A list of those fields, which will be exported to the XML File.
<b>Select Statement</b>	A select statement that will be used to select a subset of the file to be exported. This field is optional.
<b>Build Statement</b>	Clicking this button will call a window that will allow the user to create select statements.

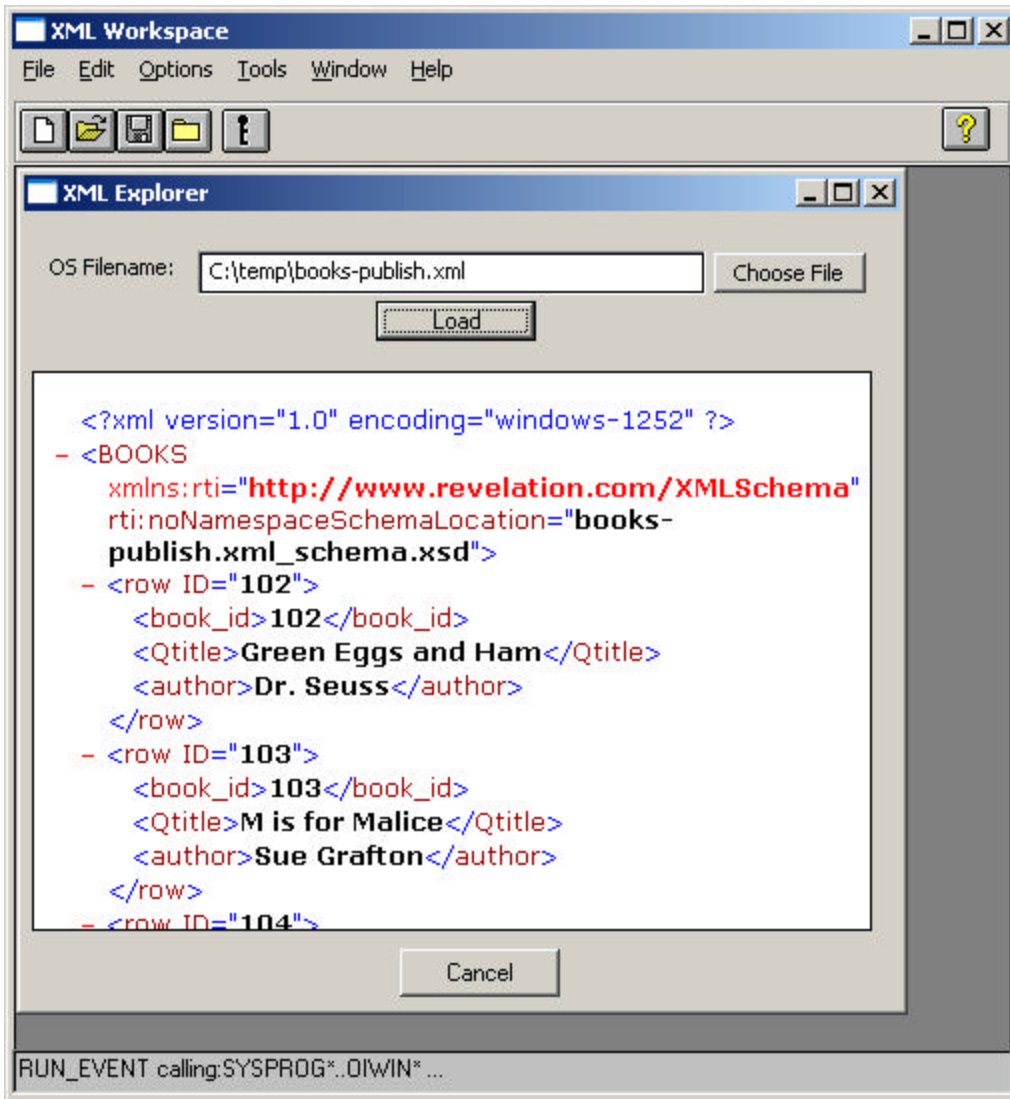
Control	Description
<b>Check Syntax</b>	Clicking this button will check the syntax of the Select statement, if there is one.
<b>OS Filename</b>	The DOS path and filename that will be created.
<b>Generate XML</b>	Clicking this button will generate the XML file.

### XML Explorer

This tool is used to view the hierarchical XML structure of an XML file from within an OI screen. Choosing XML Explorer from the Tools menu executes the XML Explorer window.



Clicking on the Choose File button will call up the Open File dialog box, listing XML files. Choose a file by clicking on it and clicking on the OK button. The filename will then be put into the OS Filename field. Clicking on the Load button will cause the explorer window to appear.

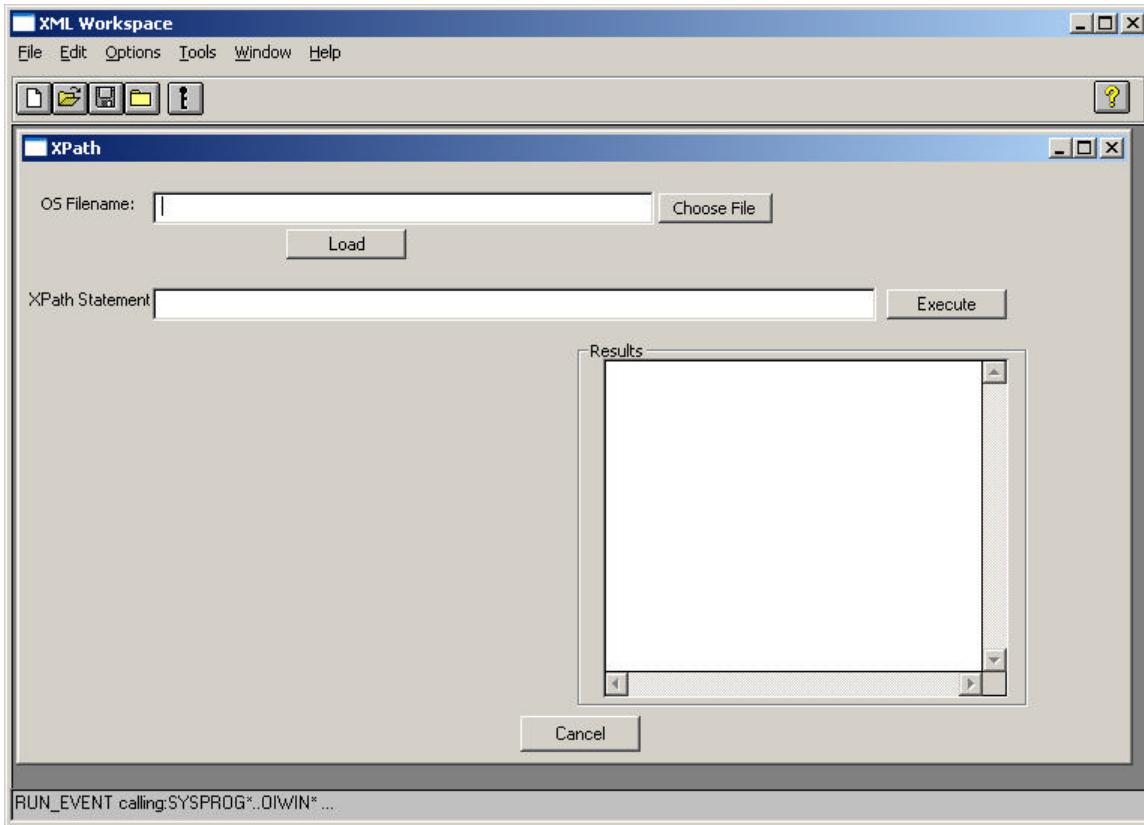


In hierarchical structures of XML, clicking on the plus ('+') or minus ('-') keys will make the listing expand or contract.

## XPath Window

The XPath window is provided to allow users to enter XPath statements and view the results. XPath statements are usually passed into an application via web requests. This window is provided to let users and developers experiment with and evaluate XPath commands.

Choosing XPath from the Tools menu executes the XPath window.

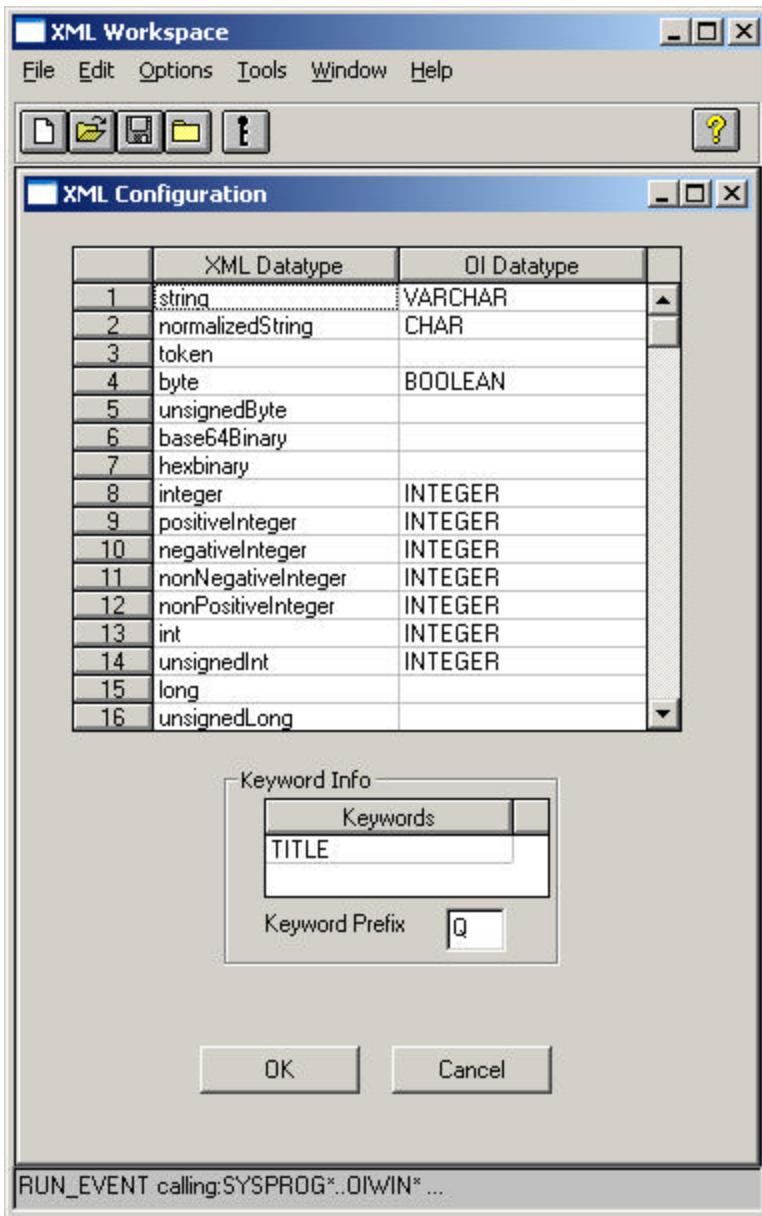


Control	Description
<b>OS Filename</b>	The name of the XML file that will be used. Clicking on the <b>Choose File</b> button will bring up the Open File dialog box.
<b>Load</b>	Clicking on this button will cause the XML file to appear for viewing.
<b>XPath Statement</b>	An editline for entering XPath commands.
<b>Execute</b>	Clicking on this button will run the command within the XPath Statement.
<b>Results</b>	Display the results of the XPath statement.

## XML Configuration Window

The XML configuration screen is used to enter and/or modify configuration settings for OpenInsight's XML functionality. It contains the datatype mapping that is used by OpenInsight when creating Dictionary items for new files.

Choosing XML Configuration from the Tools menu executes the XML Configuration window.



Control	Description
<b>XML Datatypes</b>	A list of all allowable XML datatypes.
<b>OI Datatypes</b>	The equivalent datatype for the XML datatypes. When no specific mapping is used, the system defaults to VARCHAR.

<b>Control</b>	<b>Description</b>
<b>Keywords</b>	A list of those words which may be imported or exported by an XML process, but may be reserved words.
<b>Keyword Prefix</b>	The character that is added to the beginning of the keywords.

## **XML Resources**

The following websites are very useful in gaining an understanding of XML:

[www.Biztalk.org](http://www.Biztalk.org) - Site from Microsoft that offers schemas from many industries

[www.w3schools.com/xml](http://www.w3schools.com/xml) - many sources, tutorials

[www.xml101.com](http://www.xml101.com) – Learning site

[www.xml.org](http://www.xml.org) – Industry Portal

[www.zvon.org](http://www.zvon.org) - The Guide to the XML Galaxy – tutorials, references.