Understanding Rocket U2 XML Features

ROBERT WENDL
An overview of XML

What is XML? XML stands for eXtensible Markup Language, and is a specification that was developed by the World Wide Web Consortium (W3C®). It allows application developers to create their own customized tags, enabling the definition, transmission, validation, and interpretation of data between applications and between organizations.

XML is a very flexible text format that is derived from SGML. Originally designed to meet the challenges of large-scale electronic publishing, XML is now playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere.

Some key points of XML are:

- Tags can mean anything, describing what is relevant in your data.
- An XML document is highly structured.
- The structure is a hierarchy of nested data elements.
- XML efficiently provides description for one-to-one or one-to-many relationships.
- It is a perfect fit in the MultiValue world.

XML documents produced are most often "well-formed" to show both logical and physical structure. The structure is defined in the XML specification, which can be found on the W3C website. XML is an open and evolving standard which is designed to be easy to create and use.
Documents are described by the use of a document type definition (DTD) or XML schema (XSD), which are XML documents that describe the data contained within the primary document. Applications use DTDs and XML schemas to "map" relevant data within XML documents to their local database.

The Document Object Model (DOM) provides a standard way to manipulate XML documents. The DOM API, explained later in this article, may be used to delete, remove, and update an XML document, as well as create new XML documents.

The DOM represents a document as a tree of nodes. Each node has a parent (except for the "root" node), and optional children. The DOM provides functions to traverse and manipulate these nodes. Another technology, XPath (also supported in the XML for U2 BASIC API), provides the ability to locate nodes in the DOM based on search criteria.

Unlike traditional first normal form databases, U2 data files are three dimensional in nature. A data record contains a unique item ID (record key) and fields that can repeat horizontally or vertically. System delimiters are used to separate the item ID from the data section, and to separate the data elements themselves. This type of data model is known as Multi-Value or Nested Relational.

Figure 1 below illustrates the contents of a CUSTOMER record in the HS.SALES account shipped with UniVerse.

Figure 1. Tree view of an XML document

In Figure 1 above, the item ID "2" is displayed at the top of the record. Each attribute (field) is numbered to make it more human-readable, and each value within each attribute is separated by a system delimiter. If this record contained sub-valued data, it would contain different delimiters to separate the
sub-values within each value. This form of data storage saves file space and allows related data to be stored together, improving retrieval performance.

XML is a natural partner for U2 databases because of its nested nature and ease of use.

Figure 2. Multivalued data in a U2 file

You can display similar information in U2, as shown in Listing 1 below.
Listing 1. Dictionary listing of a UniData demo file

:LISTDICT STUDENT
SORT DICT STUDENT TYP LOC CONV MNAME FORMAT SM ASSOC BY TYP BY LOC BY @ID
11:06:10 Dec 15 2005

@ID.......... TYP LOC........... CONV MNAME............ FORMAT SM ASSOC.....
@ID........... D 0 STUDENT............. 12R### S

ID............. D 0 STUDENT............. 12R### S

@ID............ D 0 STUDENT............. 12R### S

LNAME........... D 1 Last Name........... 15T S
FNAME........... D 2 First Name........... 10L S
MAJOR........... D 3 Major................... 4L S
MINOR........... D 4 Minor................... 4L S
ADVISOR......... D 5 Advisor................. 8L S
SEMESTER....... D 6 Term..................... 4L MV CGA
COURSE NBR..... D 7 Crs #................... 5L MS CGA
COURSE GRD..... D 8 GD..................... 3L MS CGA
COURSE _HOURS.. I TRANS('COURSE S',COURSE NBR ,CREDITS,'X') Hours........... 5R MS CGA
COURSE _NAME... I TRANS('COURSE S',COURSE NBR ,'NAME','X') Course Name..... 25L MS CGA
CGA............. PH SEMESTER COUR SE NBR COUR SE NAME COUR SE _GRD COUR SE _NAME COUR SE _GRD COUR SE _GRD COUR SE _GRD COUR SE _GRD TEACHER

@ORIGINAL......... SQ @ID
@SYNONYM......... SQ ID
GPA1............. V SUBR('GPA1',C MD3 GPA........... 5R S

TEACHER......... V TRANS('COURSE S',COURSE NBR ,'TEACHER','X ') Teacher........ 10L MS CGA

17 records listed:

U2 XML features

The primary goal for XML support within UniData and UniVerse is to provide a standard means of sharing information between external systems. U2's XML support aims to:

- Produce XML documents from SQL and UniQuery (UniData) and RetriVe (UniVerse).
- Provide a command that updates U2 files from XML documents.
- Provide a command that produces XML documents from U2 files.
- Help you to create robust applications using powerful APIs built on open source standards.
- Work within client/server applications, such as with UniObjects for Java, UniObjects for .NET or U2 Toolkit for .NET.
• Provide a simple means of mapping U2 data to XML documents through the use of an Eclipse-based graphical tool.

XML in U2 is managed through the use of handles, similar to files and sockets. The structures used by the XML APIs remain internal to the database run machine. They provide greater performance over arrays due to their advanced design. Functions are included to copy XML documents to and from these structures, including writing them to disk.

U2 provides an XML repository in each new account. For UniData, this file is a directory called _XML_, while in UniVerse the directory is called <XML>. These names remain in line with their respective historical naming conventions. Each new account created in UniData will have a directory called _XML_ installed automatically. UniVerse will create its &XML& the first time an XML function or command references it, whether expressed or implied.

To help control the footprint of user processes, the XML libraries are dynamically loaded on both UNIX and Windows systems. They will load when needed, thus a user process that does not execute an application or command leveraging U2 XML capabilities will not load the libraries. If you are planning to leverage this technology on UNIX systems, you will need to set an environment variable to enable the use of shared libraries. The typical methods of transport of XML from U2 BASIC are callHTTP and SOAP (SOAP is described later in this article).

Because the XML capabilities are built directly into each U2 database, the power of XML can be more efficiently realized.

Using UniQuery and RetriveVe to produce XML

With the release of UniData V6.0 and UniVerse V10.0, the first U2 XML capabilities were introduced. A significant part of the offering was the introduction of several keywords that enable the production of XML documents from UniData UniQuery, UniVerse RetriveVe, and from the SQL engines of both databases.

Using the UniQuery/RetriveVe LIST and SORT statements, you can generate XML documents and DTDs. This is called "XML OUT," meaning that U2 queries can output in XML format. UniData V6.1 and subsequently UniVerse V10.1 added the ability to produce XML schemas, as well as BASIC APIs, which will be discussed in a later section of this article.

The following is a list of query keywords added to UniData at 6.1 and UniVerse at 10.1, and present in all subsequent releases. They may be used with LIST, SORT, and SQL SELECT statements.

- TOXML -- directs processing to the XML parser
- ELEMENTS -- directs the XML parser to construct the document as "element-centric" (the default is attribute-centric)
- WITHDTD -- precedes the XML document with a DTD
- WITHSCHEMA -- precedes the XML document with an XML schema
- XMLDATA -- use in a SQL statement to specify an extraction file
- XMLMAPPING -- instructs the parser to look for a user-specified XMAP file
- TO (pre-existing for UniData, new for UniVerse) -- redirects output to a file (the _XML_ or &XML& file by default)
Listing 2 below illustrates how to produce a DTD and XML document from a LIST statement.

**Listing 2. Generating XML from a LIST statement**

```xml
:LIST STUDENT ALL WITH FNAME = "Harry" TOXML WITHDTD
<?xml version="1.0"?>
<!DOCTYPE ROOT[
<!ELEMENT ROOT (STUDENT*)>
<!ELEMENT STUDENT (CGA-MV*)>
<!ATTLIST STUDENT
   ID CDATA #REQUIRED
   LNAME CDATA #IMPLIED
   FNAME CDATA #IMPLIED
   MAJOR CDATA #IMPLIED
   MINOR CDATA #IMPLIED
   ADVISOR CDATA #IMPLIED
]>  
<!ELEMENT CGA-MV (CGA-MS*)>
<!ATTLIST CGA-MV
   SEMESTER CDATA #IMPLIED>

<!ELEMENT CGA-MS EMPTY>
<!ATTLIST CGA-MS
   COURSE_NBR CDATA #IMPLIED
   COURSE_GRD CDATA #IMPLIED>
]>  
</ROOT>

<ROOT>
<STUDENT ID = "521814564" LNAME = "Smith" FNAME = "Harry" MAJOR = "CH" MINOR = "PY" ADVISOR = "Carnes">
   <CGA-MV SEMESTER = "FA93">
      <CGA-MS COURSE_NBR = "CS130" COURSE_GRD = "A"/>
      <CGA-MS COURSE_NBR = "CS100" COURSE_GRD = "B"/>
      <CGA-MS COURSE_NBR = "PY100" COURSE_GRD = "B"/>
   </CGA-MV>
   <CGA-MV SEMESTER = "SP94">
      <CGA-MS COURSE_NBR = "CS131" COURSE_GRD = "B"/>
      <CGA-MS COURSE_NBR = "CS101" COURSE_GRD = "B"/>
      <CGA-MS COURSE_NBR = "PE220" COURSE_GRD = "A"/>
   </CGA-MV>
</STUDENT>
</ROOT>
```

The DTD in the above example allows external applications to understand the contents of the XML document that follows. It is identifiable with the comment tag "DOCTYPE ROOT [" and closed by "]>" just prior to the actual XML document.

The format of the example XML document is attribute-centric, meaning the document is not nested with respect to the U2 data contained within it. Element-centric documents (like the one shown in Listing 3 below) are formatted to nest the elements using separate tags. This is useful when the focus is on multi-valued data and not single-valued data.
Listing 3. Example of an XML document in element-centric format from UniData

```
:LIST STUDENT ALL WITH FNAME = "Harry" TOXML ELEMENTS
<?xml version="1.0"?>
<ROOT>
  <STUDENT>
    <ID>521814564</ID>
    <LNAME>Smith</LNAME>
    <FNAME>Harry</FNAME>
    <MAJOR>CH</MAJOR>
    <MINOR>PY</MINOR>
    <ADVISOR>Carnes</ADVISOR>
    <CGA-MV>
      <SEMESTER>FA93</SEMESTER>
      <CGA-MS>
        <COURSE_NBR>CS130</COURSE_NBR>
        <COURSE_GRD>A</COURSE_GRD>
      </CGA-MS>
      <CGA-MS>
        <COURSE_NBR>CS100</COURSE_NBR>
        <COURSE_GRD>B</COURSE_GRD>
      </CGA-MS>
      <CGA-MS>
        <COURSE_NBR>PY100</COURSE_NBR>
        <COURSE_GRD>B</COURSE_GRD>
      </CGA-MS>
    </CGA-MV>
    <SEMESTER>SP94</SEMESTER>
    <CGA-MS>
      <COURSE_NBR>CS131</COURSE_NBR>
      <COURSE_GRD>B</COURSE_GRD>
    </CGA-MS>
    <CGA-MS>
      <COURSE_NBR>CS101</COURSE_NBR>
      <COURSE_GRD>B</COURSE_GRD>
    </CGA-MS>
    <CGA-MS>
      <COURSE_NBR>PE220</COURSE_NBR>
      <COURSE_GRD>A</COURSE_GRD>
    </CGA-MS>
  </STUDENT>
</ROOT>
```

The following is an example of how to create a simple XML document from a UniVerse SQL statement
Listing 4. XML output using UniVerse SQL

```sql
:SELECT FNAME, LNAME, MAJOR, MINOR, SEMESTER FROM STUDENT
WHERE FNAME = 'Harry' TOXML;

<?xml version="1.0" encoding="UTF-8"?>
<ROOT>
  <STUDENT FNAME = "Harry" LNAME = "Smith" MAJOR = "CH" MINOR = "PY">
    <STUDENT_CGA_MV SEMESTER = "FA93"/>
    <STUDENT_CGA_MV SEMESTER = "SP94"/>
  </STUDENT>
</ROOT>

Unlike UniData, UniVerse treats sub-values in queries similar to multi-values. Because of this, UniVerse needs an additional mapping file to create a properly formed XML document from a query where sub-values are to be displayed. If only multi-values are required in the query, a mapping file is optional.

DB.TOXML command

UniData and UniVerse can populate data files from XML documents through the use of a command called DB.TOXML. This is a program that uses the XML capabilities to extract data from a U2 file and generate an XML document for use by external applications.

The syntax for the DB.TOXML command is:

```
DB.TOXML "xml_doc_filename" "xmap_filename" "condition"
```

where xml_doc_filename is the name of the XML document to be generated, xmap_filename is the name of the mapping file (also known as the XMAP file) which maps the data to an XML document, and condition is any conditional query phrase, such as WITH SCHOOL = "CO002".

The DB.TOXML command queries the database using any optional conditions and generates an XML document. To perform this action, DB.TOXML creates space in memory to hold the XML document it will create, and also loads in memory the mapping file to be used to format the XML document.

Listing 5. DB.TOXML example in UniData

```
:DB.TOXML PUPIL.XML PUPIL.MAP "WITH FNAME = 'Harry'"
DB.TOXML success
:CT _XML_ PUPIL.XML
_XML_: PUPIL.XML
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<ROOT>
  <STUDENT ADVISOR="Carnes" FNAME="Harry" ID="521814564" LNAME="Smith" MAJOR="CH" MINOR="PY">
    <CGA_MV SEMESTER="FA93">
      <CGA_MS COURSE_GRD="A" COURSE_NBR="CS130"/>
      <CGA_MS COURSE_GRD="B" COURSE_NBR="CS100"/>
      <CGA_MS COURSE_GRD="B" COURSE_NBR="PY100"/>
    </CGA_MV>
</ROOT>
```
In the above example, the XMAP file "PUPIL.MAP" was created using the Rocket U2 XML/DB Tool (XMAP and the Rocket U2 XML/DB Tool described later in this article) and an XML schema named DW.xsd. Listing 6 shows how the schema was created.

**Listing 6. XML schema example**

```xml
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      account: C:\U2\ud82\demo
      command: SORT STUDENT ALL TOXML WITHSCHEMA TO PUPIL
    </xsd:documentation>
  </xsd:annotation>
  <xsd:element name="ROOT">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="STUDENT" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="STUDENT">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="CGA_MV" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="ID" type="xsd:string"/>
      <xsd:attribute name="LNAME" type="xsd:string"/>
      <xsd:attribute name="FNAME" type="xsd:string"/>
      <xsd:attribute name="MAJOR" type="xsd:string"/>
      <xsd:attribute name="MINOR" type="xsd:string"/>
      <xsd:attribute name="ADVISOR" type="xsd:string"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="CGA_MV">
    <xsd:complexType>
      <xsd:attribute name="SEMESTER" type="xsd:string"/>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```
XML.TODB command

DB.TOXML was designed to provide an easy way to produce XML documents from the U2 query engines. Conversely, a means of populating the database from XML documents is also important. With this in mind, the XML.TODB command was created to read XML documents and write the appropriate content to U2 files.

The syntax for the XML.TODB command is:

XML.TODB <xml_document> <xmap_file> where "xml_document" is the XML document name to be read and "xmap_file is the mapping file to be used.

XML.TODB reads the specified XML and XMAP files from disk and creates internal structures for them. Using the information stored in the XMAP file, XML.TODB determines the U2 file to open. The command then finds and extracts each record in the XML document and writes them to the opened file. This is accomplished by using the U2 XML APIs, XDOM, and XMAP.

Listing 7 demonstrates the use of the XML.TODB command. Everything it needs to perform its intended actions exists in the XML document and XMAP file supplied by the user, providing a simple means of updating U2 records and files from XML.

Listing 7. XML.TODB example

LIST STUDENT 10:32:37 Dec 29 2017 1
2017 1
STUDENT.....

291-22-2021
424-32-5656
414-44-6545
221-34-5665
978-76-6676
5 records listed

:XML.TODB STUDENT.XML PUPIL.MAP
XML.TODB Success
:

In the above example, the XMAP file "PUPIL.MAP" was created using the Rocket U2 XML/DB Tool (XMAP and the Rocket U2 XML/DB Tool described later in this article).

XML in U2 BASIC

Both Rocket U2 UniData and UniVerse have extensive APIs for creating and shredding XML documents. In this section, the XDOM, XMAP, and SOAP client functions are reviewed, with sample code to demonstrate how to use them.
**XDOM API**

XDOM is short for XML Document Object Model, and is the name of a set of functions you can use to create, copy, traverse, and otherwise manipulate an XML document from within U2 BASIC. This is a very powerful API. Although there is some coding involved when creating or importing XML documents over loading an existing document, program execution is often faster than coding plain text, due to the improved efficiency of the XDOM memory management. Complete information on the use of each XDOM function may be found in the UniVerse 11.3.1 BASIC Extensions guide or the UniData 8.2.1 UniBasic Extensions guide.

The XDOM API allows you to:

- Traverse the DOM tree in any direction.
- Search the tree for starting elements using the XPath language.
- Cut and paste nodes, and create sub-trees.
- Create new DOM trees.
- Clone the DOM in any granularity: single element, sub-tree, or the entire DOM.
- Append to any part of the DOM.
- Copy the DOM to a variable or to a file.

The following list represents the supported DOM node types:

- Document
- DocumentFragment
- DocumentType
- EntityReference
- Element
- Attribute
- ProcessingInstruction
- Comment
- Text
- CDATASection
- Entity
- Notation

The functions and descriptions are located in the BASIC Extension guides listed above, and as such this article focuses more on examples of how one might use them.

**XMAP API**

XMAP, sometimes referred to as U2XMAP, is a set of functions that allow XML documents to relate to U2 data files. It allows XML to be shredded and input into U2, as well as allowing U2 data structures to be mapped to elements within an XML document. UniData V6.1 and UniVerse V10.1 and higher versions support unlimited levels of nesting within an XML document, going beyond their native three-dimensional data model. Mapping files, combined with the XMAP API, let you logically relate one or more U2 files to a single XML document.
The U2 XMAP API:

- Provides the ability to convert XML data into U2 records (and vice versa).
- Uses U2XMAP internal structure to store mapping information.
- Integrates with U2 XDOM API to provide simple yet powerful tools for working with XML.

UniData and UniVerse supports three modes for mapping data to XML. They are attribute-centric, element-centric, and mixed.

For specific information regarding this API, consult the UniVerse 11.3.1 BASIC Extensions guide or the UniData 8.2.1 UniBasic Extensions guide.

**XMLExecute function**

With the release of UniVerse V10.1 and subsequently UniData V6.1, a new function was added to U2 BASIC that permits the execution of XML-based queries similar to those that can be executed from ECL or TCL. XMLExecute allows a program to send an XML query to a dynamic array. It takes 4 arguments, which are:

- **Cmd** -- the query to execute (including options, such as WITHSCHEMA, WITHDTD, and so on).
- **Options** -- field mark (@FM) delimited list of some global element settings from U2XMAP. (If element values are to change, then a value mark will separate the element name from the new value. For example: root="mystudent")
- **Return_var** -- the XML document created by Cmd is stored in this variable.
- **XMLSchema** -- this contains the schema generated if WITHSCHEMA is part of Cmd.
Listing 8. XMLExecute example

*******************************************************************************
* Program XMLEXECUTE.EXAMPLE
*******************************************************************************
* Author: rwendl
* Created on: Dec 29, 2017 10:52:06 AM
* Description: Example UniBasic program to demonstrate the XMLExecute
* Function using BASICTYPE 'U'
*******************************************************************************
$INCLUDE INCLUDE XML.H
CMD = "LIST STUDENT LNAME SEMESTER COURSE_NBR '414446545'"
OPTIONS = 'WITHDTD':@FM:'XMLMAPPING=STUDENT.map'
XMLSCHEMA = ''
MYXML = ''
STATUS = XMLExecute(CMD,OPTIONS,MYXML,XMLSCHEMA)
IF STATUS # XML.SUCCESS THEN
    STATUS = XMLGetError(code,msg)
    PRINT "Error code    : " : code
    PRINT "Error Message : " : msg
    PRINT "XMLExecute failed"
END
PRINT XMLSCHEMA
PRINT MYXML
PRINT "XMLExecute finished"
END

RUN XML.BP XMLEXECUTE.EXAMPLE
<!-- DTD for account: C:\U2\ud82\demo
   command: LIST STUDENT LNAME SEMESTER COURSE_NBR '414446545' TOXML WITHDTD
--> 
<!ELEMENT ROOT (STUDENT*)>
<!ELEMENT STUDENT ( CGA_MV* )>
<!ATTLIST STUDENT
     _ID CDATA #REQUIRED
     LNAME CDATA #IMPLIED
>
<!ELEMENT CGA_MV ( CGA_MS* )>
<!ATTLIST CGA_MV
     SEMESTER CDATA #IMPLIED
>
<!ELEMENT CGA_MS EMPTY>
<!ATTLIST CGA_MS
     COURSE_NBR CDATA #IMPLIED
>
<ROOT>
<STUDENT _ID = "414446545" LNAME = "Offenbach">
   <CGA_MV SEMESTER = "FA93">
      <CGA_MS COURSE_NBR = "CS104"/>
      <CGA_MS COURSE_NBR = "MA101"/>
      <CGA_MS COURSE_NBR = "FA100"/>
   </CGA_MV>
   <CGA_MV SEMESTER = "SP94">
      <CGA_MS COURSE_NBR = "CS105"/>
</STUDENT>
U2 XML/DB tool

As of the writing of this article, there have been two releases of the Rocket U2 XML/DB tool. The first release was made available with both the UniData V6.1 and UniVerse V10.1 Client media. The next generation of the XML/DB tool was released with UniData V7.1, with most features backward-compatible with UniData V6.1 and UniVerse V10.1.

Both the original and later versions of the XML/DB tool leverage UniObjects for Java (UOJ) to connect to either UniData or UniVerse. They allow developers to create mapping files using information from U2 files, DTDs, and XML schemas. Using a graphical point-and-click environment, developers can create a map file within minutes, potentially saving a great deal of time. In this article, the focus will be on the UniData 8.2.1 version.

Figure 3. U2 XML/DB tool released with UniData V8.2.1

For more information about how to use this tool, see the UniVerse 11.3.1 BASIC Extensions guide or the UniData 8.2.1 UniBasic Extensions guide.

Additional XML BASIC Functions

UniVerse 10.0 introduced the first round of XML functions in U2 BASIC. They exist in all subsequent U2 releases. These functions are:
• PrepareXML -- Allocates memory for the XML document, opens the document, determines the file structure of the document, and returns the file structure (a handle is assigned). A DTD is required.
• OpenXMLData -- Opens the document.
• ReadXMLData -- Extracts the data from the document and places into a dynamic array.
• CloseXMLData -- Closes the document.
• ReleaseXML -- Releases the memory allocated for the XML document.
• XMLError -- Retrieves the error text from last XML function used.

Summary of XDOM API update for UniVerse 11.2 and UniData 7.2:

1. XDOMLocate: added an optional argument. The last argument is optional. If specified, it can be XML.SINGLE (0, default), or XML.MULTI (1)
2. XDOMLength: new API
   1. This function determines the number of nodes in the list. The range of valid child node index is 1 to length inclusive.
3. XDOMItem: new API
   This function returns the index-th item in the list. If index is less than 1 or greater the number of nodes in the list, XDOMItem() returns XML.ERROR, use XMLGetError(errroCode, errorMessage) to return error message “index out of bounds”.
4. XDOMGetElementById: new API, a convenience API which is implemented on XDOMLocate
   This function finds the first element with the specified id. This behaves same as XDOMLocate(xmlHandle, “//[@ID='idstr' or @id='idstr’]", “", XML_SINGLE).

For example, for above sample XML document, suppose xmlHandle points to the document root, after call to XDOMGetElementById(xmlHandle, “id2”, nodeHandle), nodeHandle should point to element <ENTRY ID="id2" NAME="bookentry">.

5. XDOMGetElementsByTagName: new API, a convenience API which is implemented on XDOMLocate
   This function tries to find all elements with the specified name. This behaves same as XDOMLocate(xmlHandle, “//[@NAME='namestr’ or @name='namestr’]", “", XML.MULTI).

For example, for above sample XML document, suppose xmlHandle points to the document root, after call to XDOMGetElementsByTagName(xmlHandle, “bookentry”, nodeHandle), nodeHandle should point to elements <ENTRY id="id1" name="bookentry"> and <ENTRY ID="id2" NAME="bookentry">.
6. **XDOMGetElementsByTagName**: new API, a convenience API which is implemented on XDOMLocate

   This function tries to find all elements with the specified tagname. This behaves same as XDOMLocate(xmlHandle, "//tagname", "", XML.MULTI).

   Tagname can be one of following format:
   
   - Local_name
   - Prefix:local_name:namespace_uri
   - Prefix:local_name:namespace_uri

   The input handle xmlHandle acts as the context when resolving the namespace uri from the prefix or resolving the prefix from the namespace uri.

   For example, for above sample XML document, suppose xmlHandle points to the document root, after call to **XDOMGetElementsByTagName** (xmlHandle, “PHONENUM”, nodeHandle), nodeHandle should point to all PHONENUM elements.

7. **XDOMGetChildNodes**: new API, a convenience API which is implemented on XDOMLocate

   This function returns all child nodes of xmlHandle. This behaves same as XDOMLocate(xmlHandle, "*", "", XML.MULTI).

   For example, for above sample XML document, suppose xmlHandle points to <ENTRY id="id1" name="bookentry">, after call to **XDOMGetChildNodes** (xmlHandle, nodeHandle), nodeHandle should point to all child nodes, i.e. <NAME>, <ADDRESS>, 3 <PHONENUM>’s, and <EMAIL>.

8. **XDOMQuery**: new API, XQuery support

   This function executes xquery on the current document or document node specified xmlHandle. Depending on the xqueryLocation, xquery contains the query if xqueryLocation is XML.FROM.STRING; or xquery is a file name which contains the query if xqueryLocation is XML.FROM.FILE. The output itemListHandle is the result item lists.

   For simple applications, these functions may be useful for developers. The XDOM and XMAP APIs are more robust, therefore are more often used by developers today. For more information about these BASIC functions, see the UniVerse 11.3.1 BASIC Extensions guide or the UniData 8.2.1 UniBasic Extensions guide.
Programming examples

This section contains some code samples. Map files, XML documents, and some dictionary listings are provided for reference. These programs will generally work for UniVerse 11.3.1 and UniData 8.2.1 releases.

*U2 XMAP example (UniData)*

**Listing 9. XML document**

```
<?xml version="1.0" ?>
<ROOT>
  <SCHOOL SCHOOLID="CO001" NAME="Fairview" SCHOOL_DISTRICT="VBSD">
    <CLASS CLASSOF="2004">
      <STUDENT ADVISOR="Carnes" FNAME="Harry" ID="521814564" LNAME="Smith" MAJOR="CH" MINOR="PY">
        <CGA_MV SEMESTER="FA93">
          <CGA_MS COURSE_GRD="A" COURSE_NBR="CS130"/>
          <CGA_MS COURSE_GRD="B" COURSE_NBR="CS100"/>
          <CGA_MS COURSE_GRD="B" COURSE_NBR="PY100"/>
        </CGA_MV>
        <CGA_MV SEMESTER="SP94">
          <CGA_MS COURSE_GRD="B" COURSE_NBR="CS131"/>
          <CGA_MS COURSE_GRD="B" COURSE_NBR="CS101"/>
          <CGA_MS COURSE_GRD="A" COURSE_NBR="PE220"/>
        </CGA_MV>
      </STUDENT>
    </CLASS>
  </SCHOOL>
  <SCHOOL SCHOOLID="CO002" FNAME="Golden" SCHOOL_DISTRICT="ACSD">
    <CLASS CLASSOF="2004">
      <STUDENT ADVISOR="Eades" FNAME="jojo" ID="291222021" LNAME="Smith" MAJOR="CS" MINOR="FA">
        <CGA_MV SEMESTER="SP94">
          <CGA_MS COURSE_GRD="B" COURSE_NBR="FA100"/>
        </CGA_MV>
      </STUDENT>
    </CLASS>
    <CLASS CLASSOF="2005">
      <STUDENT ADVISOR="Fried" FNAME="Sally" ID="424325656" LNAME="Martin" MAJOR="PY" MINOR="EG">
        <CGA_MV SEMESTER="SP94">
          <CGA_MS COURSE_GRD="C" COURSE_NBR="PY100"/>
          <CGA_MS COURSE_GRD="C" COURSE_NBR="PE100"/>
        </CGA_MV>
      </STUDENT>
    </CLASS>
  </SCHOOL>
</ROOT>
```

**Listing 10. XMAP file**

```
<?xml version="1.0" ?>
<!DOCTYPE U2XMAP SYSTEM "U2XMAP.DTD" -->
<U2XMAP Version="1.0" Name="XMAP1">
  <!-- Table/Class map -->
</U2XMAP>
```
Listing 11. TEST.XMAP program

*****************************************************************************
**
* Program TEST.XMAP
*
* Copyright (C) 2018
*
*****************************************************************************
**
* Author: rwendl
* Created on:  Jan 2, 2018 11:34:13 AM
* Description: Example UniBasic program to demonstrate the U2 XMAP
* Functions using BASICTYPE 'U'
*****************************************************************************

$INCLUDE INCLUDE XML.H
* Parse XML document and build DOM tree in memory
  STATUS = XDOMOpen('STUDENT2.XML',XML.FROM.FILE,domH)
  IF STATUS <> XML.SUCCESS THEN
    * We hit an error
    CRT 'Error with XDOMOpen().'
    GOSUB PrintError
    STOP
  END
* Position at a specific node, I.E., the second School in the XML documents
  STATUS = XDOMLocate(domH,'ROOT/SCHOOL[2]', '', domHandle)
  IF STATUS <> XML.SUCCESS THEN
    * We hit an error
    CRT 'Error from XDOMLocate().'
    GOSUB PrintError
    STOP
  END
* Open XMAP dataset for reading
  U2XMAP.FNAME = 'STUDENT2.MAP'
  STATUS = XMAPOpen(domHandle, XML.FROM.DOM, U2XMAP.FNAME, XML.FROM.FILE, Xfile)
  IF STATUS <> XML.SUCCESS THEN
    * We hit an error
    CRT 'Error with XMAPOpen().'
    GOSUB PrintError
    STOP
*****************************************************************************
OPEN 'STUDENT' TO F1 ELSE STOP 'Error opening file STUDENT.'
* Read records from XMAP dataset, write to STUDENT file
Record = ''
LOOP
  STATUS = XMAPReadNext(Xfile, 'STUDENT', Record)
  UNTIL STATUS = XML.EOF DO
    IF STATUS <> XML.SUCCESS THEN
      CRT 'XMAPReadNext() error.'
      GOSUB PrintError
      STOP
    END
    ID = Record<1>
    Rec = FIELD(Record, @FM, 2, 999)
    WRITE Rec ON F1, ID ON ERROR STOP 'Write to file STUDENT failed.'
    CRT 'ID ':ID:' written to the STUDENT file from STUDENT.XML.'
  REPEAT
    IF STATUS = XML.EOF THEN
      CRT 'EOF reached in document.'
    END
  END
* Close all handles before exiting
STATUS = XMAPClose(Xfile)
IF STATUS <> XML.SUCCESS THEN
  * We hit an error
  GOSUB PrintError
  STOP
END
STATUS = XDOMClose(domHandle)
IF STATUS <> XML.SUCCESS THEN
  * We hit an error
  CRT 'Error with XDOMClose().'
  GOSUB PrintError
  STOP
END
STATUS = XDOMClose(domH)
IF STATUS <> XML.SUCCESS THEN
  * We hit an error
  CRT 'Error with XDOMClose().'
  GOSUB PrintError
  STOP
END
CLOSE F1
STOP
PrintError:
* Using XMLGetError() we can find out the very last error code and
* its corresponding message. This allows us to display a descriptive
* reason for the problem.
Tmp = XMLGetError(Code,Msg)
CRT
CRT 'The error description and code:'
CRT '------------------------------------------'
CRT 'Error desc:' ; CRT Msg
CRT 'Error code: ':Code
CRT '------------------------------------------'
RETURN
END
Listing 12. Running TEST.XMAP

:COUNT STUDENT
COUNT STUDENT

4 record(s) counted.
:RUN XML.BP TEST.XMAP
ID 291222021 written to the STUDENT file from STUDENT.XML.
ID 424325656 written to the STUDENT file from STUDENT.XML.
EOF reached in document.
:COUNT STUDENT
COUNT STUDENT

6 record(s) counted.

Listing 13. STUDENT file listing

:LIST STUDENT ALL "291222021""424325656"
LIST STUDENT ALL "291222021""424325656" 14:46:44 Jan 17 2018 1
STUDENT..... Last Name...... First Name Major Minor Advisor. Term Crs # GD.

<table>
<thead>
<tr>
<th>ID</th>
<th>Last Name</th>
<th>First Name</th>
<th>Major</th>
<th>Minor</th>
<th>Advisor</th>
<th>Term</th>
<th>Crs</th>
<th>GD</th>
</tr>
</thead>
<tbody>
<tr>
<td>291-22-2021</td>
<td>Smith</td>
<td>jojo</td>
<td>CS</td>
<td>FA</td>
<td>Eades</td>
<td>SP94</td>
<td>FA100</td>
<td>B</td>
</tr>
<tr>
<td>424-32-5656</td>
<td>Martin</td>
<td>Sally</td>
<td>PY</td>
<td>EG</td>
<td>Fried</td>
<td>SP94</td>
<td>PY100</td>
<td>C</td>
</tr>
</tbody>
</table>

2 records listed:

The TEST.XMAP program above uses several XDOM functions, including one that uses XPath for locating a specific node, and several XMAP functions. They combine to open the example XML document, locate a specific node, open an XMAP file, loop through the document starting at the found node, remove the data from the document, and write it to a U2 file called XSTUDENT.

_U2 XDOM example (UniVerse)_

Listing 14. Sample XML document

+xml version="1.1" encoding="UTF-8" standalone="no" ?>
<ADDRBOOK cmt="my address book">
  <ENTRY ID="id1">
    <NAME>Name One</NAME>
    <ADDRESS>101 Some Way</ADDRESS>
    <PHONENUM DESC="Work">303-555-1111</PHONENUM>
    <PHONENUM DESC="Fax">303-555-2222</PHONENUM>
    <PHONENUM DESC="Pager">303-555-3333</PHONENUM>
    <EMAIL>name.one@some.com</EMAIL>
  </ENTRY>
  <ENTRY ID="id2">
    <NAME>Name Two</NAME>
  </ENTRY>
</ADDRBOOK>
<ADDRESS>202 Some Way</ADDRESS>
<PHONENUM DESC="Work">303-555-1111</PHONENUM>
<PHONENUM DESC="Fax">303-555-2222</PHONENUM>
<PHONENUM DESC="Home">303-555-3333</PHONENUM>
<EMAIL>name.two@some.com</EMAIL>
</ENTRY>
</ADDRBOOK>

Listing 15. TEST.XDOM program

*******************************************************************************
*       Program TEST.XDOM
*       Copyright (C) 2018
*******************************************************************************
*  Author: rwendl
*  Created on: Jan 17, 2018 2:55:30 PM
*  Description: Example UniVerse Basic program to demonstrate the U2 XDOM
*******************************************************************************
$INCLUDE UNIVERSE.INCLUDE XML.H
* Program to demonstrate XDOM
* Open our sample document as a DOM tree
doc = 'SAMPLE.XML'
Status = XDOMOpen(doc, 0, domHandle)
Gosub checkStatus
Print 'DOM open status = ':Status
* Locate the ENTRY node, passing a string to XPath, the node with ID = "id1"
Status = XDOMLocate(domHandle, '/ADDRBOOK/ENTRY[@ID="id1"]', '', nodeHandle)
Print 'DOM locate status = ':Status
Gosub checkStatus
* Create a new element node at the current location in the tree
nodeName = 'PHONENUM'
Status = XDOMCreateNode(nodeHandle, nodeName, '', XDOM.ELEMENT.NODE, newElem)
Print 'New element node create status = ':Status
Gosub checkStatus
* Create the attribute for the new element node
Attr = 'DESC'
attrValue = 'Cell'
Status = XDOMCreateNode(nodeHandle, Attr, attrValue, XDOM.ATTR.NODE, newAttr)
Print 'New attribute node create status = ':Status
Gosub checkStatus
* Create a new text node at the current location in the tree
dataString = '444-555-4444'
Status = XDOMCreateNode(nodeHandle, '', dataString, XDOM.TEXT.NODE, newText)
Print 'Create text node status = ':Status
Gosub checkStatus
* Add the new node as the next sibling in the DOM tree using XPath to find
* the new location of where we want to place the node
Status = XDOMAppend(nodeHandle, 'PHONENUM[2]', '', newElem, XDOM.NODUP)
Print 'DOM append completed node = ':Status
Gosub checkStatus
* Add the text node as the next child after the attribute note using newElem
* as the handle to which the text will added
Status = XDOMAddChild(newElem, '', '', newText, XDOM.NODUP)
Print 'DOM add child status = ':Status
Gosub checkStatus

* Add the attribute to our node by using XDOMAppend, because attributes are
* added with this function. We just append to the element node we are
* building.
Status = XDOMAppend(newElem, '', '', newAttr, XDOM.NODUP)
Print 'DOM append completed attr node = ':Status
Gosub checkStatus

* Write the modified document to the &XML& file
xmlString = 'OUT.xml'
Status = XDOMWrite(domHandle, xmlString, XML.TO.FILE)
Print 'DOM write to file = ':Status
Print 'DOM ':xmlString:' written to &XML& file'
Gosub checkStatus
Stop

checkStatus:
* Test for an error and display if found
If Status <> XML.SUCCESS Then
    Status = XMLGetError(errCode, errMsg)
    Print errCode:' ':errMsg
    Stop 'Stop the program'
End
Return
End

Listing 16. XDOM.TEST output

> RUN XML.BP TEST.XDOM
DOM open status = 0
DOM locate status = 0
New element node create status = 0
New attribute node create status = 0
Create text node status = 0
DOM append completed node = 0
DOM add child status = 0
DOM append completed attr node = 0
DOM write to file = 0
DOM OUT.xml written to &XML& file

Listing 17. OUT.xml written to &XML&

<?xml version="1.1" encoding="UTF-8" standalone="no" ?>
<ADDRBOOK cmt="my address book">
    <ENTRY ID="id1">
        <NAME>Name One</NAME>
        <ADDRESS>101 Some Way</ADDRESS>
        <PHONENUM DESC="Work">303-555-1111</PHONENUM>
        <PHONENUM DESC="Fax">303-555-2222</PHONENUM>
        <PHONENUM DESC="Cell">444-555-4444</PHONENUM>
        <PHONENUM DESC="Pager">303-555-3333</PHONENUM>
        <EMAIL>name.one@some.com</EMAIL>
    </ENTRY>
</ADDRBOOK>
In the example above, the TEST.XDOM program inserted a node at a specific location found by an XPath string. It begins by opening an XML document located in the local BP file, and creates a DOM structure from the document. The new DOM is given a handle. Using XPath, it locates the first node and assigns the found node a handle of its own. While the internal pointer is at the found node, the program then creates a new element called PHONENUM, followed by a new attribute called DESC with a value of Cell, then a new text node 444-555-4444. Next, the element is appended after the second occurrence of PHONENUM within the first ENTRY ID located earlier in the program. This is accomplished using the XDOMAppend function using an XPath string. The text node is added as a child to the new PHONENUM element node and then the attribute is appended to the node. The newly modified DOM structure is then written to a file called OUT.xml.

It is worthwhile to note that the XDOM API does not "beautify" XML documents, but the output is in accordance with XML specifications.

**U2 XDOMQuery example (UniVerse)**

**Listing 18. Sample XML document**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<result>
    <video_template>
        <title/>
        <genre>
            <choice>action</choice>
            <choice>comedy</choice>
            <choice>drama</choice>
            <choice>family</choice>
            <choice>foreign</choice>
            <choice>horror</choice>
            <choice>musical</choice>
            <choice>special</choice>
        </genre>
        <rating>
            <choice>G</choice>
            <choice>PG</choice>
            <choice>PG-13</choice>
        </rating>
    </video_template>
</result>
```
<choice>R</choice>
<choice>NC-17</choice>
</rating>
<user_rating>
<choice>1</choice>
<choice>2</choice>
<choice>3</choice>
<choice>4</choice>
<choice>5</choice>
</user_rating>
<summary />
<details />
<year />
<director />
<studio />
<runtime />
<vhs />
<vhs_stock />
<dvd />
<dvd_stock />
<beta />
<beta_stock />
<laserdisk />
<laserdisk_stock />
</video_template>
<actors>
<actor id="00000015">Anderson, Jeff</actor>
<actor id="00000030">Bishop, Kevin</actor>
<actor id="0000000f">Bonet, Lisa</actor>
<actor id="00000024">Bowz, Eddie</actor>
<actor id="0000002d">Curry, Tim</actor>
<actor id="00000033">Dullea, Keir</actor>
<actor id="00000042">Fisher, Carrie</actor>
<actor id="00000006">Ford, Harrison</actor>
<actor id="00000045">Foster, Jodie</actor>
<actor id="00000018">Ghigliotti, Marilyn</actor>
<actor id="0000000c">Hackman, Gene</actor>
<actor id="0000003f">Hamill, Mark</actor>
<actor id="00000027">Heames, Darin</actor>
<actor id="0000001e">Heche, Anne</actor>
<actor id="00000003">Jones, Tommy Lee</actor>
<actor id="00000039">Lockwood, Gary</actor>
<actor id="00000048">McConaughey, Matthew</actor>
<actor id="0000003c">Richter, Daniel</actor>
<actor id="00000021">Schwimmer, David</actor>
<actor id="00000009">Smith, Will</actor>
<actor id="0000001b">Spoonhauer, Lisa</actor>
<actor id="00000036">Sylvester, William</actor>
<actor id="0000002a">Todd, Antonio</actor>
<actor id="00000012">Voight, John</actor>
<actor id="0000004b">Woods, James</actor>
<actor id="1784027567">Lewis, Daniel</actor>
<actor id="2096814035">Manesse, Gaspard</actor>
<actor id="325442748">Feito, Raphael</actor>
<actor id="4231919377">Morier, Philippe</actor>
<actor id="2142583927">Racette, Francine</actor>
<actor id="916503204">The Beatles</actor>
<actor id="916503205">Neeson, Liam</actor>
<actor id="916503206">Pinocchio, Mr.</actor>
<actor id="916503207">Parillaud, Anne</actor>
<actor id="916503208">Pitt, Brad</actor>
<actor id="916503209">Freeman, Morgan</actor>
<actor id="916503211">Domingo, Placido</actor>
<actor id="916503210">Sharif, Omar</actor>
</actors>
<videos>
  <video id="id1235AA0">
    <title>The Fugitive</title>
    <genre>action</genre>
    <rating>PG-13</rating>
    <summary>Tommy Lee Jones and Harrison Ford are the hunter and the hunted in this fast-paced story of a falsely convicted man who escapes to find his wife's true killer.</summary>
    <details>Harrison Ford and Tommy Lee Jones race through the breathless manhunt movie based on the classic TV series. Ford is prison escapee Dr. Richard Kimble, a Chicago surgeon falsely convicted of killing his wife and determined to prove his innocence by leading his pursuers to the one-armed man who actually committed the crime.</details>
    <year>1997</year>
    <director>Andrew Davis</director>
    <studio>Warner</studio>
    <user_rating>4</user_rating>
    <runtime>110</runtime>
    <actorRef>00000003</actorRef>
    <actorRef>00000006</actorRef>
    <vhs>13.99</vhs>
    <vhs_stock>206</vhs_stock>
    <dvd>14.99</dvd>
    <dvd_stock>125</dvd_stock>
    <beta>1.03</beta>
    <beta_stock>12</beta_stock>
    <LaserDisk>12.00</LaserDisk>
    <LaserDisk_stock>10</LaserDisk_stock>
  </video>
  <video id="id1244100">
    <title>Enemy of the State</title>
    <genre>action</genre>
    <rating>R</rating>
  </video>
</videos>
<summary>After a chance meeting with an old pal, Robert Deal finds himself in possession of a disk that contains evidence of an assassination plot by the NSA.</summary>

<details>It's not paranoia if they're really after you. Will Smith (Men in Black) stars as attorney Robert Clayton Dean, a man at the center of a high-stakes pursuit when he inadvertently comes into possession of secret information implicating corrupt officials within the government.</details>

<year>1999</year>
<director>Tony Scott</director>
<br>studio>Buena vista</br>
<br>user_rating>5</br>
<br>runtime>113</br>
<br>actorRef>00000009</br>
<br>actorRef>0000000c</br>
<br>actorRef>0000000f</br>
<br>actorRef>00000012</br>
<br>vhs>16.99</br>
<br>vhs_stock>188</br>
<br>dvd>29.99</br>
<br>dvd_stock>353</br>
<br>LaserDisk>12.57</br>
<br>LaserDisk_stock>203</br>

<video id="id124E230">
<title>Clerks</title>
<genre>comedy</genre>
<rating>R</rating>
<br>summary>Spend a day at work with two friends and their quirky customers at a video shop and convenience store.</br>
<br>details>A day-in-the-life of two men that work as clerks in adjacent video and convenience stores. The story follows the two through an agonizing day at work, a surprise hit at the 1994 Cannes film festival.</details>

<year>1999</year>
<director>Kevin Smith</director>
<br>studio>Buena vista</br>
<br>user_rating>3</br>
<br>runtime>97</br>
<br>actorRef>00000015</br>
<br>actorRef>00000018</br>
<br>actorRef>0000001b</br>
<br>betta>23.97</br>
<br>betta_stock>419</br>

<video id="id124F050">
<title>Six Days Seven Nights</title>
<genre>comedy</genre>
<rating>PG-13</rating>
<br>summary>A classy magazine editor and her gruff pilot must put aside their mutual dislike for each other to survive on a deserted island.</br>
Robin Monroe, a New York magazine editor, headed to a paradise island for a photo shoot, and the gruff pilot, Quinn Harris, who is flying her there, must put aside their mutual dislike for each other when their small plane crashes on a deserted South Seas island. Now, at the mercy of the ocean and the island’s jungles, they must help each other survive.

<year>1998</year>
<director>Ivan Reitman</director>
<studio>Buena vista</studio>
<user_rating>2</user_rating>
<runtime>123</runtime>
<actorRef>00000006</actorRef>
<actorRef>0000001e</actorRef>
<actorRef>00000021</actorRef>
<vhs>16.99</vhs>
<vhs_stock>327</vhs_stock>
<LaserDisk>12.00</LaserDisk>
<LaserDisk_stock>10</LaserDisk_stock>
<beta>10.75</beta>
<beta_stock>251</beta_stock>

Grad students find a mannequin and their worst fears come true.

<year>1995</year>
<director>Vincent Robert</director>
<studio>Simitar Video</studio>
<user_rating>2</user_rating>
<runtime>96</runtime>
<actorRef>00000024</actorRef>
<actorRef>00000027</actorRef>
<actorRef>0000002a</actorRef>
<dvd>12.99</dvd>
<dvd_stock>309</dvd_stock>

The Muppets appear in a classic tale of treasure and pirates.

<year>1996</year>
<director>Brian Henson</director>
<video id="id125BCC0">
<title>2001: A Space Odyssey</title>
<genre>drama</genre>
<rating>PG</rating>
<summary>An epic tale of space travel and evolution.</summary>
<details>A mysterious monolith awakens the imagination of man's distant ancestors; A second one awaits his giant leap to the moon; And in orbit around Jupiter a third beckons man to transcend beyond the limits of his body and his machines. Stanley Kubrick's cosmic metaphor of human evolution remains an unfaded visual achievement.</details>
<year>1968</year>
<director>Stanley Kubrick</director>
<studio>Warner Studios</studio>
<user_rating>1</user_rating>
<runtime>148</runtime>
<actorRef>00000033</actorRef>
<actorRef>00000036</actorRef>
<actorRef>00000039</actorRef>
<actorRef>0000003c</actorRef>
<vhs>16.99</vhs>
<vhs_stock>120</vhs_stock>
<dvd>21.49</dvd>
<dvd_stock>54</dvd_stock>
<beta>32.55</beta>
<beta_stock>231</beta_stock>
</video>

<video id="id1256990">
<title>Star Wars</title>
<genre>action</genre>
<rating>PG</rating>
<summary>Luke embarks on an epic journey in a galaxy far far away.</summary>
<details>A young man on a desert planet joins a desperate rebellion against the evil galactic Empire after his aunt and uncle are murdered by Imperial stormtroopers.</details>
<year>1977</year>
<director>George Lucus</director>
<studio>Fox Films</studio>
<user_rating>5</user_rating>
<runtime>121</runtime>
<actorRef>0000003f</actorRef>
<actorRef>00000042</actorRef>
<actorRef>00000006</actorRef>
<video id="id1263D00">
<title>Contact</title>
<genre>drama</genre>
<rating>PG</rating>
<summary>A scientist makes contact with intelligent life and travels billions of miles to meet them.</summary>
<details>Devoted astronomer Dr. Ellie Arroway undertakes an emotional and spiritual journey after receiving the message she's waited for all her life—a mysterious signal beamed in from alien beings, who pass along instructions for building and piloting a craft that will presumably survive the passage from Earth to their home. While struggling to fund her mission, Arroway also struggles with her feelings about the nature of things, particularly after meeting a charismatic New Age believer who questions her disbelief in God.</details>
<year>1997</year>
<director>N/A</director>
<studio>Warner Studios</studio>
<user_rating>3</user_rating>
<runtime>150</runtime>
<actorRef>00000045</actorRef>
<actorRef>00000048</actorRef>
<actorRef>0000004b</actorRef>
</video>

<video id="3325584384">
<studio/>
<actorRef>325442748</actorRef>
<actorRef>2096814035</actorRef>
<actorRef>4231919377</actorRef>
<director>Louis Malle</director>
<title>Au Revoir Les Enfants</title>
<dvd>25.25</dvd>
<laserdisk>55.25</laserdisk>
<genre>foreign</genre>
<rating>G</rating>
<runtime>124</runtime>
<user_rating>5</user_rating>
<details>A young boy befriends a new student whom the others feel is different. When he discovers the new student is a Jew, he tells no one and remains a true friend. Tragedy strikes when a
school employee tells the Gestapo they are hiding Jews and the student is arrested and taken away.

<summary>A young boy befriends a new student whom the others feel is different.</summary>

<video id="647599245">
  <studio />
  <actorRef>916503204</actorRef>
  <director>George Dunning</director>
  <title>Yellow Submarine</title>
  <dvd>25</dvd>
  <laserdisk />
  <genre>musical</genre>
  <rating>G</rating>
  <runtime>100</runtime>
  <user_rating>4</user_rating>
  <details>This restored, animated valentine to the Beatles offers viewers the rare chance to see a work that's been substantially improved by its technical facelift, not just supersized with extra footage.</details>
</video>

<video id="647599246">
  <studio />
  <actorRef>916503205</actorRef>
  <director>David Breashears</director>
  <title>Everest</title>
  <dvd>27</dvd>
  <laserdisk />
  <genre>special</genre>
  <rating>G</rating>
  <runtime>110</runtime>
  <user_rating>4</user_rating>
  <summary>Originally shot in the IMAX format, this video shows spectacular shots of climbers on Mount Everest.</summary>
</video>

<video id="647599245">
  <studio />
  <actorRef>916503204</actorRef>
  <director>George Dunning</director>
  <title>Yellow Submarine</title>
  <dvd>25</dvd>
  <laserdisk />
  <genre>musical</genre>
  <rating>G</rating>
  <runtime>100</runtime>
  <user_rating>4</user_rating>
  <details>This restored, animated valentine to the Beatles offers viewers the rare chance to see a work that's been substantially improved by its technical facelift, not just supersized with extra footage.</details>
</video>

<video id="647599246">
  <studio />
  <actorRef>916503205</actorRef>
  <director>David Breashears</director>
  <title>Everest</title>
  <dvd>27</dvd>
  <laserdisk />
  <genre>special</genre>
  <rating>G</rating>
  <runtime>110</runtime>
  <user_rating>4</user_rating>
  <summary>Originally shot in the IMAX format, this video shows spectacular shots of climbers on Mount Everest.</summary>
</video>
Filmed in the IMAX format, this film had the luck (or lack thereof) to be shot during the same fateful and fatal climb of Mount Everest chronicled in Jon Krakauer's book, Into Thin Air, in which a group of rich hobby climbers found themselves trapped by a blizzard near the summit. The IMAX film contains footage of those people, but focuses on its own group, as they make their assault on the top of the world's highest peak.

---

This Disney masterpiece from 1940 will hold up forever precisely because it doesn't restrain or temper the most elemental emotions and themes germane to its story. Based on the Collodi tale about a wooden puppet who wants to become a real boy, Pinocchio is among the most magical, mythical, and frightening films to come from the studio in its long history.

---

This classic story of the all-time favorite puppet.

---

This film is a foreign language film and contains elements of violence.

---
<summary>Anne Parillaud plays a special agent in this fast-moving French thriller.</summary>

French director Luc Besson (The Fifth Element) broke the commercial taboo against female-driven action movies with this seminal, seductively slick film about a violent street punk (Anne Parillaud) trained to become a smooth, stylish assassin.

<details>
French director Luc Besson (The Fifth Element) broke the commercial taboo against female-driven action movies with this seminal, seductively slick film about a violent street punk (Anne Parillaud) trained to become a smooth, stylish assassin.
</details>

<vhs>15</vhs>
<beta_stock />
<year>1990</year>
<vhs_stock>88</vhs_stock>
<dvd_stock>22</dvd_stock>
<beta />
</video>

<video id="647599249">
<studio />
<actorRef>916503208</actorRef>
<actorRef>916503209</actorRef>
<director>David Fincher</director>
<title>Seven</title>
<dvd>22</dvd>
<laserdisk>24</laserdisk>
<laserdisk_stock>2</laserdisk_stock>
<genre>horror</genre>
<rating>R</rating>
<runtime>130</runtime>
<user_rating>3</user_rating>
<summary>A serial killer forces each of his victims to die by acting out one of the seven deadly sins.</summary>

<details>
The most viscerally frightening and disturbing homicidal maniac picture since The Silence of the Lambs, Seven is based on an idea that's both gruesome and ingenious. A serial killer forces each of his victims to die by acting out one of the seven deadly sins. The murder scene is then artfully arranged into a grotesque tableau, a graphic illustration of each mortal vice.
</details>

<vhs>15</vhs>
<beta_stock />
<year>1995</year>
<vhs_stock>88</vhs_stock>
<dvd_stock>22</dvd_stock>
<beta />
</video>

<video id="647599250">
<studio />
<director />
<actorRef>916503210</actorRef>
<title>National Geographic's Mysteries of Egypt</title>
<dvd>22</dvd>
<laserdisk />
<laserdisk_stock />
<genre>special</genre>
<rating>R</rating>
<runtime>110</runtime>
Experience the magnificence of Egypt like never before!

Legendary actor and native Egyptian Omar Sharif and Kate Maberly join National Geographic's award-winning filmmakers to weave a fascinating story of a grandfather who enchants his granddaughter with tales of tombs and treasures. Travel back more than 4,000 years to a time when the Great Pyramids of Giza were built.

** Listing 19. XDOMQUERY program  

*----------------------------------------------------------------------*
*  Program XDOMQUERY                                                 *
*  Copyright (C) 2018                                                 *
*----------------------------------------------------------------------*
*  Author: rwendl                                                  *
*  Created on: Feb 6, 2018 9:57:46 AM                               *
*  Description: XDOMQuery function runs xquery on the current document or
*  document node you specify with xmlHandle.
*****************************************************************************
$INCLUDE UNIVERSE.INCLUDE XML.H
CR          = CHAR(10)
*
XMLScen="videos.xml"
QURY="videos.q"
* XQuery FLWOR Expression
QURY.TEXT = "let $name :=//actor/text()":CR: "return $name"
OPEN "","&XML&" TO DATA.XML ELSE STOP "Can't open &XML&"
WRITE QURY.TEXT ON DATA.XML, QURY
*
* The XDOMOpen function reads an xmlDocument and creates DOM structure.
*
IF XDOMOpen(XMLScen, XML.FROM.FILE, hDOM) = XML.ERROR THEN
  GOSUB SHOW.ERROR
END
*
* query is in file, return items are all nodes
QURY_LOC=XML.FROM.FILE
GOSUB DO.QUERY
*
* XQuery FLWOR Expression query is in a string, return items are all simple type
QURY="let $title :=//video/title/text()":CR: "return $title"
QURY_LOC=XML.FROM.STRING
GOSUB DO.QUERY
*
* The XDOMClose function frees the DOM structure.
*
status=XDOMClose(hDOM)
STOP
*
* -------
DO.QUERY:
* -------
* The XDOMQuery function runs xquery on the current document or document node you
* specify with xmlHandle.
*
Stat = XDOMQuery(hDOM, QURY, QURY_LOC, hNDLIST)
*
IF Stat = XML.ERROR THEN
  CRT "xpath was NOT found for " : QURY : "."
END ELSE
  CRT "xpath was found " : QURY : "."
*
* The XDOMLength function determines the number of nodes in the list. The range of
* node index is to 1 to length, inclusive
*
Stat = XDOMLength(hNDLIST, len)
CRT "XDOMLength : " : len
FOR K = 1 TO len
*
* The XDOMItem function returns the index-th item in the list.
*
    Stat = XDOMItem(hNDLIST, K, datand, datatype)
IF datatype = XQ.ITEM.NODE THEN
    Stat = XDOMWrite(datand, ndtxt, XML.TO.STRING)
   CRT K:"-":"node=":ndtxt
END ELSE
   CRT K:"simple type=":datatype:, value=":datand
END
Listing 20. XDOMQUERY output

> XDOMQUERY
xpath was found videos.q.
XDOMLength : 38
1-node=Anderson, Jeff
2-node=Bishop, Kevin
3-node=Bonet, Lisa
4-node=Bowz, Eddie
5-node=Curry, Tim
6-node=Dullea, Keir
7-node=Fisher, Carrie
8-node=Ford, Harrison
9-node=Foster, Jodie
10-node=Ghigliotti, Marilyn
11-node=Hackman, Gene
12-node=Hamill, Mark
13-node=Heames, Darin
14-node=Heche, Anne
15-node=Jones, Tommy Lee
16-node=Lockwood, Gary
17-node=McConaughey, Matthew
18-node=Richter, Daniel
19-node=Schwimmer, David
20-node=Smith, Will
21-node=Spoonhauer, Lisa
22-node=Sylvester, William
23-node=Todd, Antonio
24-node=Voight, John
25-node=Woods, James
26-node=Lewis, Daniel
27-node=Manesse, Gaspard
28-node=Feito, Raphael
29-node=Morier, Philippe
30-node=Racette, Francine
31-node=Beatles, The
32-node=Neeson, Liam
33-node=Pinocchio, Mr.
34-node=Parillaud, Anne
35-node=Pitt, Brad
36-node=Freeman, Morgan
37-node=Domingo, Placido
38-node=Sharif, Omar
xpath was found let $title :=//video/title/text()
There are several ways to find and extract elements and attributes from an XML document. XQuery is probably more efficient for retrieving complex data from a large XML document using the powerful query language. In the example above, the XDOMQUERY program executes two XQuery. The first query is from a file which returns a list of actors. The second is from a string and returns a list of video titles.

**XQuery Basic Syntax Rules:**

- XQuery is case-sensitive
- XQuery elements, attributes, and variables must be valid XML names
- An XQuery string value can be in single or double quotes
- An XQuery variable is defined with a $ followed by a name, e.g. $bookstore
- XQuery comments are delimited by (: and :), e.g. (: XQuery Comment :)

**Summary**

XML is a continuously developing technology that is ideal for organizations to exchange information with one another. Rocket U2 provides the openness of XML so that developers can create powerful and efficient applications which can easily exchange information. U2 XML empowers businesses to share information according to individual business rules, yet remain open and application agnostic.

Combined with technologies as UniObjects for Java, UniObjects for .NET, and the U2 BASIC callHTTP and SOAP client APIs, U2 XML is the ideal choice for the format of data exchange, as it easily maps to U2's nested data structures.