Rocket | U2

●●● white paper

Using U2 and Microsoft® .NET

Jianfeng Mao
Rajan Kumar

Rocket®

Rocket | U2

using U2 and Microsoft® .NET

introduction

This white paper discusses the interaction between Microsoft .NET and Rocket Software's UniData® and UniVerse™ databases. There are a number of options today for integrating UniData and UniVerse into the .NET framework. This document explores several of the available methods.

overview of Microsoft® .NET

Microsoft .NET is software that connects information, people, systems, and devices. It spans clients, servers, and developer tools, and consists of:

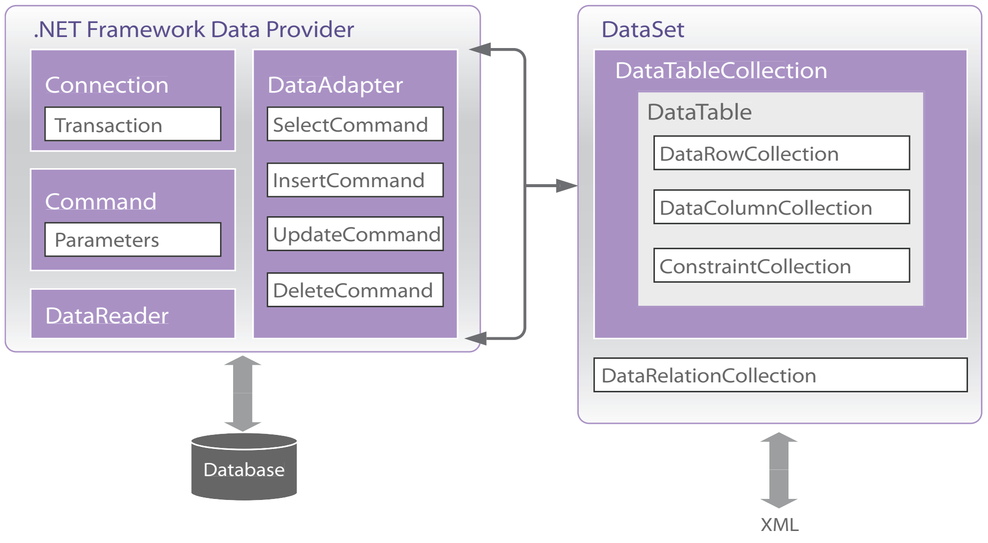
- The .NET Framework, used for building and running all kinds of software, including Web-based applications, smart client applications, and XML Web services—components that facilitate integration by sharing data and functionality over a network through standard, platform-independent protocols such as XML (Extensible Markup Language), SOAP (Simple Object Access Protocol), and HTTP (Hypertext Transfer Protocol).

- Developer tools, such as Microsoft Visual Studio, which provide an integrated development environment (IDE) for maximizing developer productivity with the .NET Framework.

- A set of servers, including Windows Server, SQL Server™, and BizTalk® Server, that integrate, run, operate, and manage web services and web-based applications.

- Client software, such as Windows, Windows Mobile™, and Microsoft Office, that helps developers deliver a deep and compelling user experience across a family of devices and existing products.

what is the .NET framework?



The .NET Framework is an integral Windows component for building and running the next generation of software applications and Web services. The .NET Framework:

- Supports over 20 different programming languages. These include: C#, VB.NET, and IronPython and more.
- Manages much of the plumbing involved in developing software, enabling developers to focus on the core business logic code.
- Makes it easier than ever before to build, deploy, and administer secure, robust, and high-performing applications.

The .NET Framework is composed of the common language run time and a unified set of class libraries.

common language runtime (CLR)

CLR is Microsoft's Virtual Machine (equivalent to the Java VM). CLR is responsible for run-time services such as language integration and security enforcement, as well as memory, process and thread management. In addition, the CLR has a role at development time when features such as life-cycle management, strong type naming, cross-language exception handling, and dynamic binding reduce the amount of code that a developer must write to turn business logic into a reusable component.

Microsoft Intermediate Language (MSIL)

Microsoft Intermediate Language (MSIL) is the CPU-independent instruction set that is generated by the Microsoft .NET Framework compilers and consumed by the CLR. Before MSIL can be executed, it must be converted to native, CPU-specific code by the common language runtime (CLR).

Managed code

Managed code is code executed and managed by the .NET Framework, specifically by the .NET Framework's common language run time (CLR). Managed code must supply the information necessary for the CLR to provide services such as memory management, cross-language integration, code access security, and automatic lifetime control of objects. All code based on MSIL executes as managed code.

Unmanaged code

Unmanaged code is code that is executed directly by the operating system, outside the Microsoft .NET Framework's CLR. Unmanaged code must provide its own memory management, type checking, and security support, unlike managed code, which receives these services from the CLR. Unmanaged code must be executed outside the .NET Framework.

Class Libraries

- Microsoft .NET Framework *base classes* provide standard functionality such as input/output, string manipulation, security management, network communications, thread management, text management, and user interface design features.

- The ADO.NET classes enable developers to interact with data accessed in the form of XML through the OLE DB, ODBC, Oracle, and SQL Server interfaces.

- XML classes enable XML manipulation, searching, and translations.

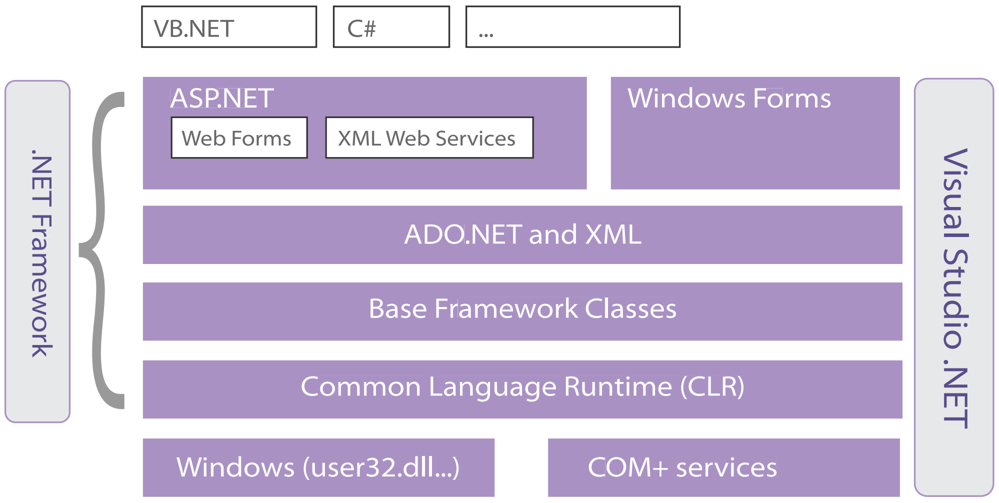
- ASP.NET classes support the development of Web-based applications and Web services.
- Windows Forms classes support the development of desktop-based smart client applications.

Together, the class libraries provide a common, consistent development interface across all languages supported by the .NET Framework.

.NET programming

The .NET development model is Object-Oriented (OO). All .NET programming languages (C#, VB.NET, etc.) are OO languages. Every type in .NET is a Class (this is similar to Java). Even an ASP.NET Web page is treated internally as a Class. Source code written in .NET programming languages is compiled into MSIL (Microsoft Intermediate Language) code, also called *managed code*. Standalone executables still have the file extension of .exe. Class libraries still have the extension of .dll, but they are called *assemblies* in .NET.

ADO.NET architecture



ADO.NET is the application-level interface for providing data access services in .NET. There are two core components in ADO.NET: dataset and data provider.

Dataset

- Stored in memory, disconnected, and best suited in to an n-tier environment
- Collection of tables and the relationships between the tables
- Can be populated with an XML document
- Can also be written out as an XML document
- Serialization format of Dataset is XML

Data Provider

- Connection object provides connectivity to data source
- Command objects execute commands in the data source and return results back
- DataReader provides forward only, read only access to data
- DataAdapter populates Dataset with its SelectCommand and resolves changes to Dataset back to data source with its InsertCommand, UpdateCommand, and DeleteCommand

The Microsoft Data Provider for OLEDB can be used with third party OLEDB providers such as UniOLEDB and the RedBack® RedPages DLL. The Microsoft Data Provider for ODBC can be used with third party ODBC drivers such as UniVerse ODBC and UniData ODBC.

.NET interoperability

The Microsoft .NET Framework promotes interaction with COM components, COM+ services, external type libraries, and many operating system services. Data types, method signatures, and error-handling mechanisms vary between *managed* and *unmanaged* object models. To simplify interoperation between .NET Framework components and unmanaged code and to ease the migration path, the CLR conceals from both clients and servers the differences in these object models through two different mechanisms.

Com Interoperability

Allows exposing COM objects to the .NET framework and .NET framework components to COM.

Exposing COM objects to .NET framework: There are several ways to expose COM objects to .NET framework: Visual Studio.NET project reference, Type Library Importer, TypeLibConverter class, and Custom COM wrappers. All of these tool sets generate a .NET assembly that wraps the exposed COM object so that it is accessible to .NET applications.

Exposing .NET framework components to COM: A managed component must be registered in the Windows registry before it can be activated from a COM client. The command-line tool called the Assembly Registration Tool (Regasm.exe) will register or unregister an assembly for use with COM. Regasm.exe adds information about the class to the system registry so COM clients can use the .NET class transparently. The RegistrationServices class provides the equivalent functionality.

Platform Invoke (P-Invoke)

Platform invoke is a service that enables managed code to call unmanaged functions implemented in dynamic link libraries (DLLs), such as those in the Win32 API. It locates and invokes an exported function and marshals its arguments (integers, strings, arrays, structures, and so on) across the interoperation boundary as needed.

using UniData and UniVerse with .NET

Since .NET is a set of added Windows components, it will not affect either UniData or UniVerse servers or clients because the Windows services and APIs that these products rely upon still exist. That means existing Windows applications based on UniData and UniVerse technologies will continue to run whether .NET is installed or not. Currently all, UniData and UniVerse products are in unmanaged code, for example, the native machine code.

using UniData and UniVerse clients with .NET

UniData and UniVerse provide several Windows-based APIs that an application can call to access UniData and UniVerse databases. These are: UniOLEDB, ODBC, UniObjects, InterCall, and UCI (for UniVerse). For them to work in .NET programs, different techniques can be used depending on each product's properties. The following table lists all these products and summarizes how to use them in .NET. For actual code samples, see **Appendices B through G.**

Interface	COM	Standard or U2 Native Interface	How to Use in .NET?
UniOLEDB	yes	standard	use with ADO.NET through Microsoft OLEDB data provider (COM interop is used)
ODBC	no	standard	use with ADO.NET through Microsoft ODBC data provider (P-invoke is used)
UniObjects	yes	native	use through COM interop
InterCall	no	native	use through P-invoke
UCI	no	native	use through P-invoke

UniData or UniVerse XML, Client APIs, and ADO.NET dataset

As discussed earlier, an ADO.NET Dataset is the convergence of XML and relational data. It can be populated by both relational data and XML, and can be manipulated as both relational data and XML. This feature of an ADO.NET Dataset really opens the door to how we can generate datasets in .NET from data stored in UniData or UniVerse databases. Users can use UniOLEDB and UniData or UniVerse ODBC together with Microsoft OLEDB Provider /ODBC Provider to generate a Dataset based on SQL statements, or any of the UniData and UniVerse Client APIs to use the XML capabilities of the UniData or UniVerse server to generate an XML document and feed it into a Dataset. Since all UniData and UniVerse Client APIs can call Basic subroutines with input/output parameters, subroutine calls seems to be the best means for this purpose. Moreover, with the UniData and UniVerse XML/DB data manipulation capability (XML.TODB, DB.TOXML), .NET applications can even synchronize any changes made to the Dataset back to the databases through XML. See the UniData or UniVerse documentation for more information.

UniXML class

UniObjects for .NET ships with an additional class called UniXML. This class defines several public methods that can be used to associate an XML document and its associated XML schema with U2 data files residing on the server. The UniXML class provides a Dataset type for the result set. As a Dataset, a document can easily be transformed and viewed from XML into a relational table, such as the .NET Dataset model.

appendix A

.NET Console Application in VB.NET- using UniOLEDB and the Microsoft OLEDB Provider

Below is sample code in VB.NET that connects to the UniVerse HS.SALES account and executes a 'SELECT' statement. It then creates a Dataset and prints field values, such as 'CUSTID' and 'LNAME'.

```
Imports System
Imports System.Data
Imports System.Data.OleDb

Public Class UniOLEDBSample
    Public Shared Sub Main()
        Dim nConn As OleDbConnection = New
OleDbConnection("Provider=IBM.UniOLEDB;
Location=HS.SALES; Data Source=localuv;user id=XXX;password=YYY;")

        Dim catCMD As OleDbCommand = nConn.CreateCommand()

        catCMD.CommandText = "SELECT * from CUSTOMER"
        nConn.Open()

        Dim myds As New DataSet
        Dim myadapter As New OleDbDataAdapter(catCMD)

        myadapter.Fill(myds)

        Dim mytable As DataTable
        Dim myrow As DataRow
        For Each mytable In myds.Tables
          For Each myrow In mytable.Rows
            Console.WriteLine(vbTab &"{0}"& vbTab &
"{1}",myrow.Item(0),myrow.Item(2))
          Next
        Next

        nConn.Close()
    End Sub

End Class
```

appendix B

.NET Console Application in C# - using UVODBC and Microsoft ODBC Data Provider

Below is sample code in VB.NET that connects to the UniVerse HS.SALES account and executes a 'SELECT' statement. It then creates a 'DataReader' object and prints a field value like 'CUSTID' and 'LNAME'.

```
using  System;
using  System.Data;
using  System.Data.Odbc;

class U2ODBCSample
{
    static void Main(string[] args)
    {
        OdbcConnection nConn = new OdbcConnection( "DRIVER={ UniVerse ODBC
Driver};Database=HS.SALES;Server=localuv;UID=XXX;PWD=YYY;");
        OdbcCommand catCMD =  nConn.CreateCommand();
        catCMD.CommandText = "SELECT * from CUSTOMER";

        nConn.Open();

        OdbcDataReader myReader = catCMD.ExecuteReader();

        while ( myReader.Read())
        {
            Console.WriteLine("\t" + "{0}" + "\t" + "{1}",
myReader.GetInt32(0), myReader.GetString(2));
        }
        myReader.Close();
        nConn.Close();
    }

}
```

Rocket | U2

appendix C

.NET console application in C# - using UniObjects and COM interop

Below is sample code in C#, that connects to the UniVerse HS.SALES account and reads the record ID called "RELLEVEL" from the VOC file.

```
using System;
using System.Data;
using UNIOBJECTSLib; // Use add the 'UniObjects Control 3.0' reference in
VStudio.NET COM references
class Sample
{
    public static void Main()
    {
        bool ret =false;
        UnioaifCtrl m_uniobj = new UnioaifCtrl();
        m_uniobj.UserName = "xxx";
        m_uniobj.Password = "yyy";

        m_uniobj.AccountPath = "HS.SALES";
        m_uniobj.HostName = "localhost";
        m_uniobj.ConnectionString = "uvcs";
        ret = m_uniobj.Connect();

        IUniFile d = (IUniFile)m_uniobj.OpenFile("VOC");
        d.RecordId = "RELLEVEL";
        d.Read();
        IUniDynArray s = (IUniDynArray)d.Record;
        string s1 = s._StringValue;
        Console.WriteLine("Reading UniVerse File VOC and Record
ID RELLEVEL" + "\n");
        Console.WriteLine (s1 + "\n");
    }
}
```

appendix D

VB.NET using UniObjects to call subroutine

Below is a code segment in VB.NET using UniObjects to call a subroutine that retrieves an XML document.

```
‘Add the 'UniObjects Control 3.0' reference in VStudio.NET COM references
Dim uo As UNIOBJECTSLib.UnioaifCtrl
        uo = New UNIOBJECTSLib.UnioaifCtrl
        uo.ExceptionOnError = True
        uo.HostName = "localhost"
        uo.AccountPath = "demo"
        uo.UserName = "xxx"
        uo.Password = "yyy"
        uo.DataBaseType = 2 ' UniData
        uo.Connect()

        Dim uosub As UNIOBJECTSLib.UniRPC
        uosub = uo.subroutine("STUDENTXML", 1)
        uosub.call()
        Dim xmlstr As String = uosub.getArg(0)
        ' populate a Dataset
        Dim ds As New DataSet
        Dim sr As New IO.StringReader(xmlstr)
        ds.ReadXml(sr, XmlReadMode.Auto)
     Console.WriteLine(ds.Tables.Item(0).ToString())
```

VB.NET using UniObjects to call subroutine

```
SUBROUTINE STUDENTXML(xmlvar)
   CMD="LIST STUDENT LNAME SEMESTER COURSE_NBR COURSE_GRD TOXML
   XMLMAPPING student.map"
   OPTIONS=""
   STATUS = XMLExecute(CMD,OPTIONS,xmlvar,schemavar)
END
```

Rocket | U2

appendix E

C# Example to work with UCI.DLL in .NET Application using DllImport

Overview

The following C# program demonstrates how to use C# DllImport to enable interop with uci.dll file.

Setup

Copy the following C# program in a file and save the file as "Program.cs". Modify the UCI Data Source Name defined in the uci.config file, User Name and Password in the file. Create a Directory called "U2UCITest". Put the "Program.cs" file into this directory"

Compilation

Go to Visual Studio .NET Command Prompt. Type the following command:

csc.exe Program.cs /platform:x86

where csc.exe is C# Compiler. This will produce 32-bit "Program.exe" program.

Run

Execute the program. "Program.exe".

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Runtime.InteropServices;

namespace ConsoleApplication_UCI
{
    /// <summary>
    /// Summary description for U2UCI.
    /// </summary>
    public class U2UCI
    {
       [DllImport("uci.dll", CallingConvention =
CallingConvention.Cdecl)]
       public static extern int SQLUseCfgFile(int henv,  int
option,string config);

       [DllImport("uci.dll", CallingConvention =
CallingConvention.Cdecl)]
       public static extern int SQLAllocConnect(int henv, ref
int phdbc);

       [DllImport("uci.dll",CallingConvention =
CallingConvention.Cdecl)]
       public static extern int SQLAllocEnv(ref int phenv);

       [DllImport("uci.dll", CallingConvention =
CallingConvention.Cdecl)]
       public static extern int SQLConnect(int hdbc, string
szDSN, int cbDSN, string szSchema, int cbSchema);

       [DllImport("uci.dll", CallingConvention =
CallingConvention.Cdecl)]
       public static extern int SQLDisconnect(int hdbc);

       [DllImport("uci.dll", CallingConvention =
CallingConvention.Cdecl)]
       public static extern int SQLFreeConnect(int hdbc);

       [DllImport("uci.dll", CallingConvention =
CallingConvention.Cdecl)]
       public static extern int SQLFreeEnv(int henv);

       [DllImport("uci.dll", CallingConvention =
CallingConvention.Cdecl)]
       public static extern int SQLSetConnectOption(int hdbc,
uint fOption, uint vParam, string szParam);

    }

    class Program
    {
        static void Main(string[] args)
         {
              int  m_penv=0;
             int  m_phdbc=0 ;
            // m_penv = Marshal.AllocHGlobal(100);
            int ret = U2UCI.SQLAllocEnv( ref m_penv);
            //m_phdbc = Marshal.AllocHGlobal(100);
            ret = U2UCI.SQLAllocConnect( m_penv,   ref m_phdbc);

        ret = U2UCI.SQLUseCfgFile(m_penv, 0, "uci.config");
        ret = U2UCI.SQLSetConnectOption(m_phdbc, 996, 0,
"xxx");
```

```
       ret = U2UCI.SQLSetConnectOption( m_phdbc, 995, 0,
"xxx");
       string uci_dsn = "localuv";
       string account = "HS.SALES";

       // make a connection
       //ret = U2UCI.SQLConnect(m_phdbc,
uci_dsn,uci_dsn.Length,account,account.Length);
       ret = U2UCI.SQLConnect(m_phdbc, uci_dsn,
uci_dsn.Length, account, account.Length);
       ret = U2UCI.SQLDisconnect(m_phdbc);
       // free resources
       U2UCI.SQLFreeConnect(m_phdbc);
       U2UCI.SQLFreeEnv(m_penv);
       if (ret ==0 )
       {
          Console.WriteLine("connected............");
       }
       else
       {
          Console.WriteLine("failed:"+ret);
       }

       }
    }
}
```

appendix F

C# example using the UniObjects for .NET UniXML class

(Note: for UniVerse, ensure that HS.SALES has BP, &XML&, and BP.O VOC items pointing to those directories)

Overview

The following C# program demonstrates the functionality of UniXML Test and related GetDataSet() function.

Setup

Copy the following C# program in a file and save the file as "Form1.cs". Modify Server Name, User Name and Password in the file. Create a Directory called "UniXMLTest". Put the following files into this directory"

1. Form1.cs
2. U2.Data.Client.dll

Compilation

Go to Visual Studio .NET Command Prompt. Type the following command:

```
csc.exe  Form1.cs /reference: U2.Data.Client.dll
```

where csc.exe is C# Compiler. This will produce "Form1.exe".

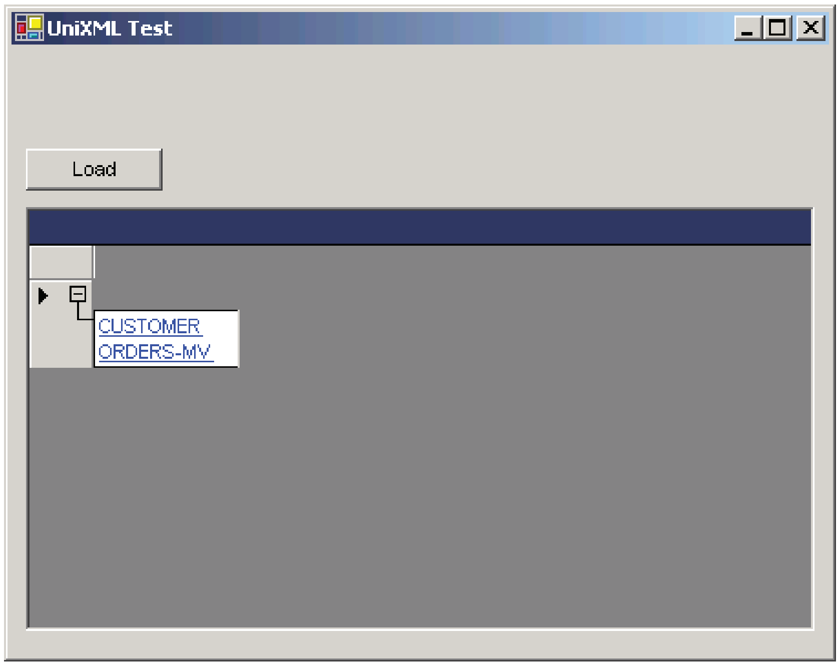
Run

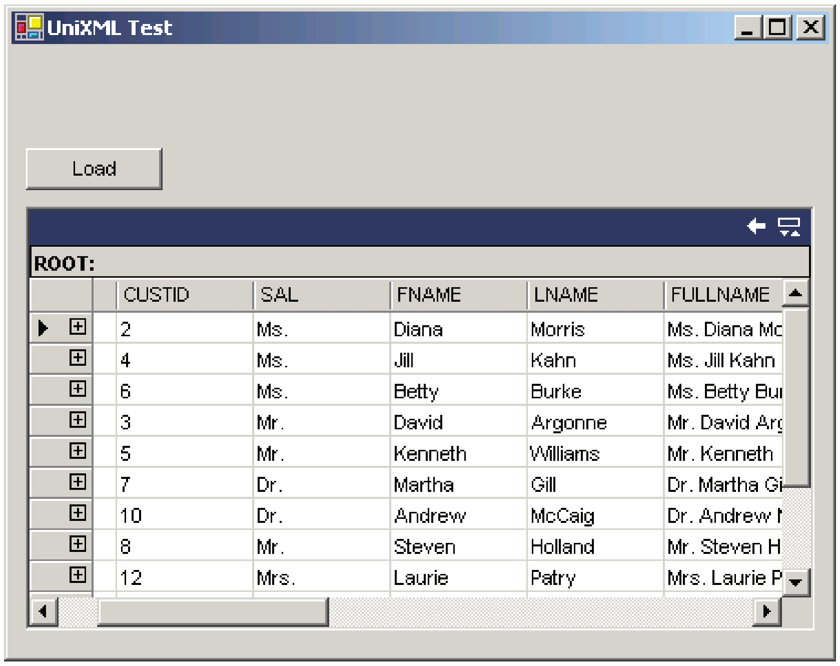
Execute the program. "Form1.exe".

Prerequisite

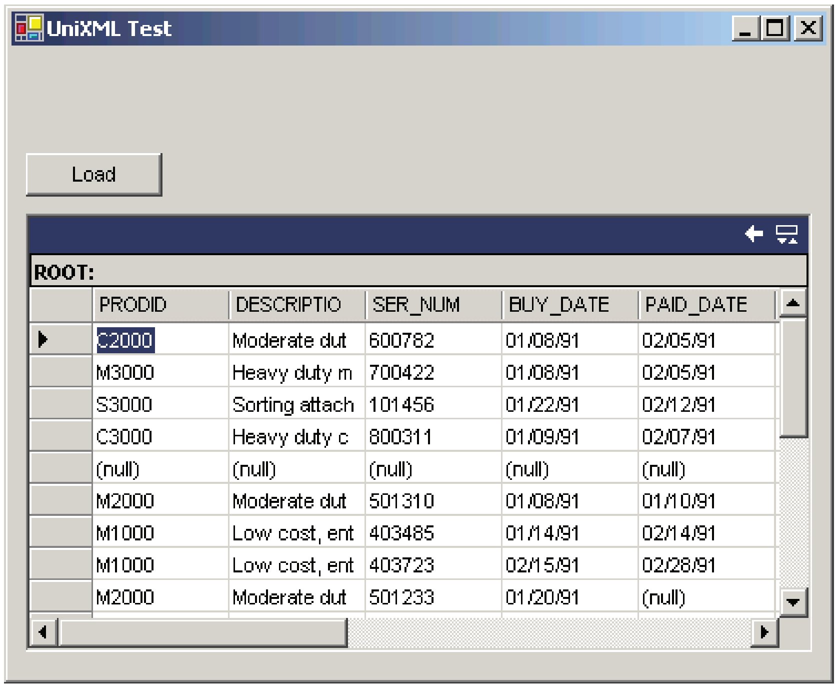
The Basic subroutine *GETXMLSUB must exist, be compiled and globally cataloged.

output





	CUSTID	SAL	FNAME	LNAME	FULLNAME
⊞	2	Ms.	Diana	Morris	Ms. Diana Mc
⊞	4	Ms.	Jill	Kahn	Ms. Jill Kahn
⊞	6	Ms.	Betty	Burke	Ms. Betty Bu
⊞	3	Mr.	David	Argonne	Mr. David Ar
⊞	5	Mr.	Kenneth	Williams	Mr. Kenneth
⊞	7	Dr.	Martha	Gill	Dr. Martha Gi
⊞	10	Dr.	Andrew	McCaig	Dr. Andrew I
⊞	8	Mr.	Steven	Holland	Mr. Steven H
⊞	12	Mrs.	Laurie	Patry	Mrs. Laurie P



PRODID	DESCRIPTIO	SER_NUM	BUY_DATE	PAID_DATE
C2000	Moderate dut	600782	01/08/91	02/05/91
M3000	Heavy duty m	700422	01/08/91	02/05/91
S3000	Sorting attach	101456	01/22/91	02/12/91
C3000	Heavy duty c	800311	01/09/91	02/07/91
(null)	(null)	(null)	(null)	(null)
M2000	Moderate dut	501310	01/08/91	01/10/91
M1000	Low cost, ent	403485	01/14/91	02/14/91
M1000	Low cost, ent	403723	02/15/91	02/28/91
M2000	Moderate dut	501233	01/20/91	(null)

C# Source Code

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Text;
using System.IO;
using U2.Data.Client;
using U2.Data.Client.UO;

namespace WindowsApplication5
{
    /// <summary>
    /// Summary description for Form1.
    /// </summary>
    public class Form1 : System.Windows.Forms.Form
    {
        private System.Windows.Forms.DataGrid dataGrid1;
        private System.Windows.Forms.Button button1;
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.Container components = null;
        public Form1()
        {
            //
            // Required for Windows Form Designer support
            //
            InitializeComponent();

            //
            // TODO: Add any constructor code after InitializeComponent call
            //
        }

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        protected override void Dispose( bool disposing )
        {
            if( disposing )
            {
                if (components != null)
                {
                    components.Dispose();
                }
            }
            base.Dispose( disposing );
        }

        #region Windows Form Designer generated code
        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
        this.dataGrid1 = new
System.Windows.Forms.DataGrid();
        this.button1 = new System.Windows.Forms.Button();

    ((System.ComponentModel.ISupportInitialize)(this.dataGrid1)).BeginInit();
        this.SuspendLayout();
        //
        // dataGrid1
        //
        this.dataGrid1.DataMember = "";
        this.dataGrid1.HeaderForeColor =
System.Drawing.SystemColors.ControlText;
        this.dataGrid1.Location = new System.Drawing.Point(8, 88);
        this.dataGrid1.Name = "dataGrid1";
        this.dataGrid1.Size = new System.Drawing.Size(432, 232);
        this.dataGrid1.TabIndex = 0;
        //
        // button1
        //
```

```
            this.button1.Location = new System.Drawing.Point(8, 56);
            this.button1.Name = "button1";
            this.button1.TabIndex = 1;
            this.button1.Text = "Load";
            this.button1.Click += new
System.EventHandler(this.button1_Click);
            //
            // Form1
            //
            this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
            this.ClientSize = new System.Drawing.Size(448, 333);
            this.Controls.Add(this.button1);
            this.Controls.Add(this.dataGrid1);
            this.Name = "Form1";
            this.Text = "UniXML Test";

    ((System.ComponentModel.ISupportInitialize)(this.dataGrid1)).EndInit();
                this.ResumeLayout(false);

            }
            #endregion

            /// <summary>
            /// The main entry point for the application.
            /// </summary>
            [STAThread]
            static void Main()
            {
                Application.Run(new Form1());
            }
            private void button1_Click(object sender,
System.EventArgs e)
            {
                U2Connection u2connect = new U2Connection();
                UniSession us = null;

                string con_str =
"Server=localhost;Database=HS.SALES;UserID=xxx;Password=xxx;Po
oling=False;AccessMode=Native;ServerType=UNIVERSE;RpcServiceTy
pe=uvcs";
                u2connect.ConnectionString = con_str;
                u2connect.Open();
                us = u2connect.UniSession;

                UniXML x = us.CreateUniXML();
                x.GenerateXML("LIST CUSTOMER");
                DataSet ds2 = new DataSet();
                ds2= x.GetDataSet();
                this.dataGrid1.DataSource = ds2.Tables[0].DefaultView;
            }
        }
    }
```

C# example using connection pooling in UniData

```
using System;
using System.Threading;
using U2.Data.Client;
using U2.Data.Client.UO;

namespace CPTest
{
    class CPTest
    {
        [STAThread]
        static void Main (string[] args)
        {
            U2Connection con = new U2Connection();
            UniSession us1=null;
            try
            {
                U2ConnectionStringBuilder conStr = new
U2ConnectionStringBuilder();
                conStr.UserID = "xxx";
                conStr.Password = "xxx";
                conStr.Server = "localhost";
                conStr.Database = "HS.SALES";
                conStr.ServerType = "UNIVERSE";
                conStr.AccessMode = "Native";
                conStr.RpcServiceType = "uvcs";

                // Pool
                conStr.Pooling = true;
                conStr.MinPoolSize = 1;
                conStr.MaxPoolSize = 10;
                con.ConnectionString = conStr.ToString();
                con.Open();
                us1 = con.UniSession;

                UniCommand cmd = us1.CreateUniCommand();
                cmd.Command="SSELECT STATES";
                cmd.Execute();
                string response_str = cmd.Response;
                Console.WriteLine(" Response from UniCommand :"+response_str);
            UniSelectList sl = us1.CreateUniSelectList(0);
            while (!sl.LastRecordRead)
            {
                string s = sl.Next();
                if (s != "")
                {
                    Console.WriteLine(" Record ID : "+s);
                }
            }
        }
        catch(Exception e)
                {
                    if (con != null && con.IsOpen)
                    {
                    con.Close();
                    }
                    Console.WriteLine("");

    Console.WriteLine(Thread.CurrentThread.Name +" : Connection failed in 
Test Program : " + e.Message +e.StackTrace);

    Console.WriteLine("=============================================================
==========================");

    Console.WriteLine("=============================================================
======================");

            }
                finally
                {
                    if (con != null && con.IsOpen)
                    {
                        Console.WriteLine("");
    Console.WriteLine(Thread.CurrentThread.Name +" : Connection Passed in 
Test Program");

    Console.WriteLine("=============================================================
======================");

    Console.WriteLine("=============================================================
======================");
                        con.Close();
                    }
                }
            }
        }
}
```

Rocket | U2

white paper

Rocket

u2.rocketsoftware.com
u2@rocketsoftware.com
twitter.com/rocketu2
facebook.com/rocketu2
youtube.com/rocketu2
www.rocketsoftware.com/u2/linkedin