Microsoft Business Solutions

.NET Development with Navision

Revision 1

Created May 7, 2004

SUMMARY

This article provides a list of suggestions and ideas to start a .NET development to access a Navision database with the SQL Server option through a web site portal. The recommended audience is .NET developers and Navision C/AL developers.

Code samples included in this white paper are written in .NET Visual Basic and Navision C/AL.

.NET Visual Basic code is quoted 
Navision C/AL code is quoted 

The article contains the following sections:

SUMMARY 2
DISCLAIMER 2
MORE INFORMATION 3
Code samples 3
Installing the code samples 3
Using the code samples 5
Accessing Navision data from .NET 10
One database / multiple companies 10
Identifier Conversion 11
Executing Navision business logic from .NET 13
Microsoft Message Queue 13
XML Message format 13
Navision application server 16
Integrating Navision security system with .NET 17
Data security 17
Process security 18
REFERENCES 20

DISCLAIMER

This document is provided "AS IS" with no warranties, and confers no rights.
Use of included script samples are subject to the terms specified at http://www.microsoft.com/info/cpyright.htm.

MORE INFORMATION

Code samples

Installing the code samples

Installing the required programs

The following programs are required to use the code samples:

Server:

- Microsoft Windows XP Professional
- Microsoft SQL Server 2000 SP3
- Microsoft Message Queuing
- Microsoft Internet Information Services
- Microsoft Business Solutions – Navision 3.70
- Microsoft Business Solutions – Navision Application Server 3.70
- Microsoft Business Solutions – Navision SDK 3.70 (Communication Component / Bus Adapter)
- Microsoft Visual Basic .NET – Development Environment 2002
- Microsoft Visual Basic .NET - .NET Framework 1.0

Client:

- Microsoft Internet Explorer 6.0

Please refer to the documentation of each program for further information on installation process. All Microsoft Business Solutions – Navision components can be installed from the Microsoft Business Solutions – Navision product CD. You need a Navision developer license to be able to code in Navision.

The Navision SQL database used in the code samples is the “Navision Demo Database” available in the Navision product CD (file:\\Product CD\Client\Program Files\Microsoft Business Solutions-Navision\Client\Navision Demo Database_Data.mdf).

Setting-up security

The code samples use Windows Authentication.
You must create a windows login to run the Navision Application Server. You can give the “SUPER” role access to this windows login inside Navision (using a Navision client, go to Tools / Security / Windows Logins / Roles).
You can use your own windows login to use the .NET application. You can give the “SUPER” role access to your windows login inside Navision (using a Navision client, go to Tools / Security / Windows Logins / Roles). You can give dbo access to your windows login to the Navision database inside SQL Server (use the Enterprise Manager).

Importing Navision objects

You must open the Navision database with a Navision client, and import the .fob file. The .fob file contains the following Navision objects:

Object Type	Object ID	Object Name	Version List
Codeunit	1	ApplicationManagement	NAVW13.70,NAVA13.70,MBS
Codeunit	80	Sales-Post	NAVW13.70,NAVA13.70,NAVAU3.70,MBS
Codeunit	70000	RunNavisionProcess	MBS
Codeunit	70001	WebServiceSecurity	MBS

Starting Navision application server

You must start the Navision application server with the following parameters.

Database Server Name	Your SQL Server name
Database	Navision Demo database
Company Name	CRONUS Australia Pty. Ltd.
OR	
Company Name	CRONUS New Zealand Pty. Ltd.
OR	
Company Name	CRONUS International Ltd.
Start-Up Parameter Value	WEBSERVICE

Installing the .NET projects

Stop the World Wide Web Publishing service on your computer.
Copy / paste the NavisionWebApplication and NavisionWebService folders in the Home Directory of your computer (by default C:\Inetpub\wwwroot).
Start the World Wide Web Publishing service.
In Internet Information Server, ensure that Windows authentication is supported by the NavisionWebApplication and NavisionWebService directories of the Default Web Site. Right click on these directories; go to Properties / Directory Security. Click Edit on Anonymous access and authentication control. Tick Digest authentication for Windows domain servers and tick Integrated Windows Authentication.

In Microsoft Internet Explorer 6.0, you can open the Navision web application at the following address: http://localhost/navisionwebapplication
In Microsoft Visual Basic .NET – Development Environment 2002, you can open the .NET projects at the following addresses:
http://localhost/navisionwebapplication/NavisionWebApplication.vbproj and
http://localhost/navisionwebservice/NavisionWebService.vbproj.

In the NavisionWebService .NET project, you can customize the “ConnectionString” and “ConvertIdentifiers” appSettings keys in the Web.config file if required.

Using the code samples

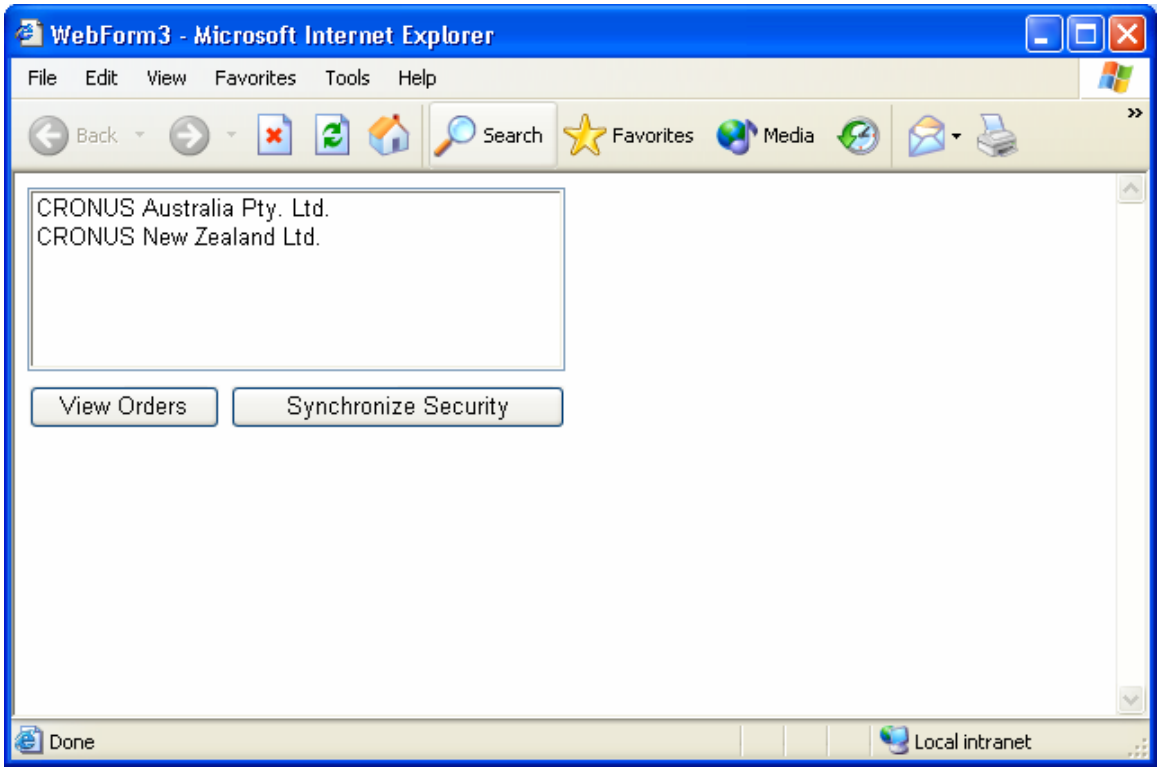
The purpose of the code samples is not to offer extensive functionalities, but to demonstrate the possible interactions between a Navision database and a .NET based development.

Code samples scenario

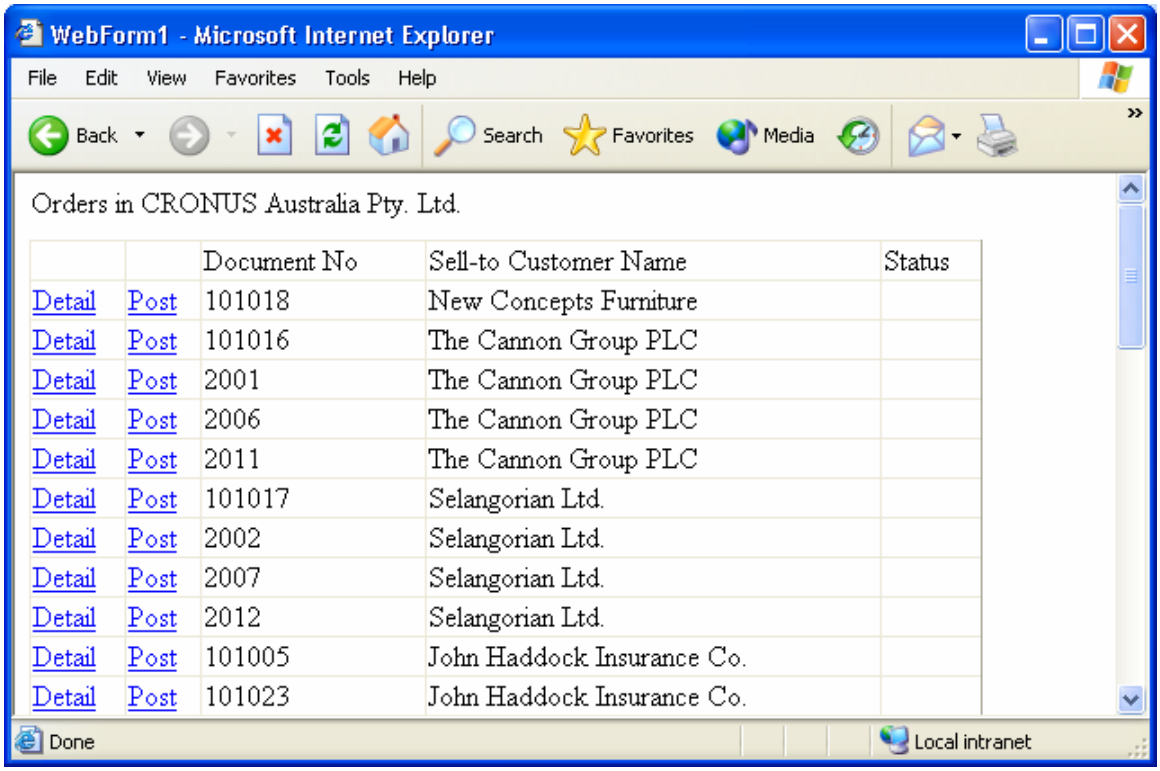
Sales persons work with laptops. Navision is not installed in their computers. But they need to be able to browse Sales Orders and to post them on behalf of their customers.

When starting the Web Application, you must “Synchronize Security” for each company you have started a Navision application server for.

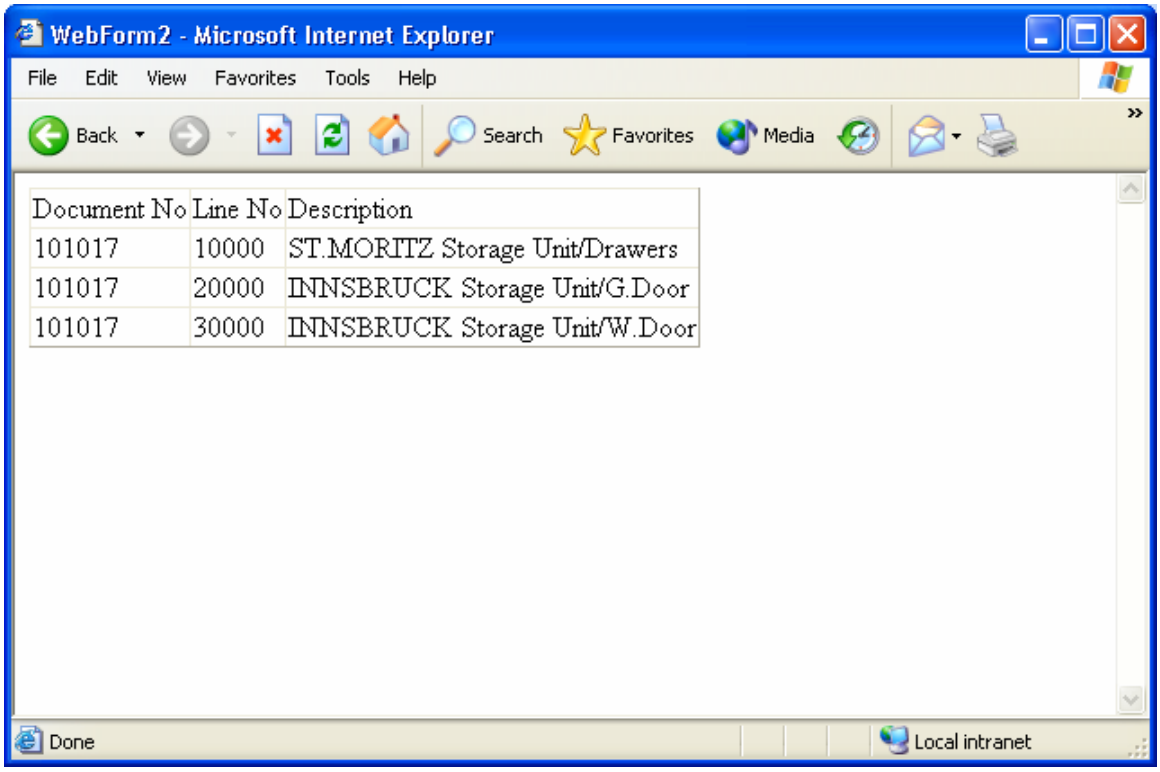
The sales persons can then choose the Navision company they want to work with.



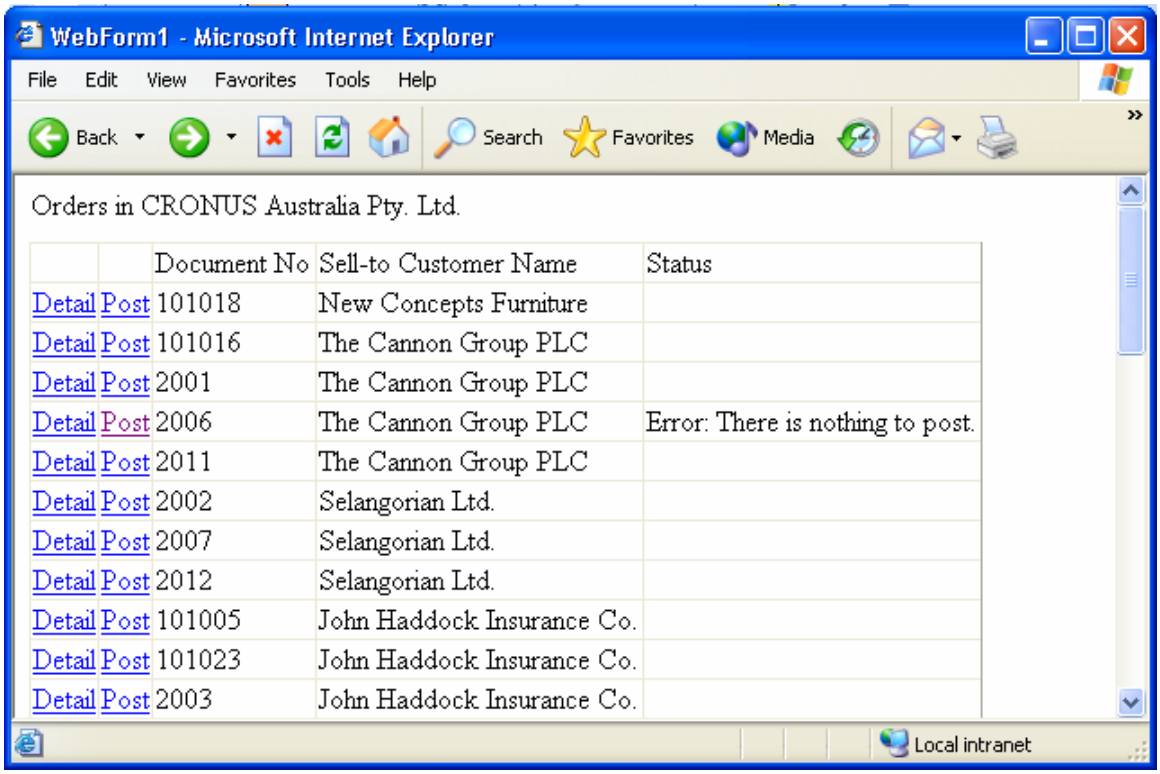
They can view all the sales orders in the company.



They can get the details (sales lines) of a specific order.

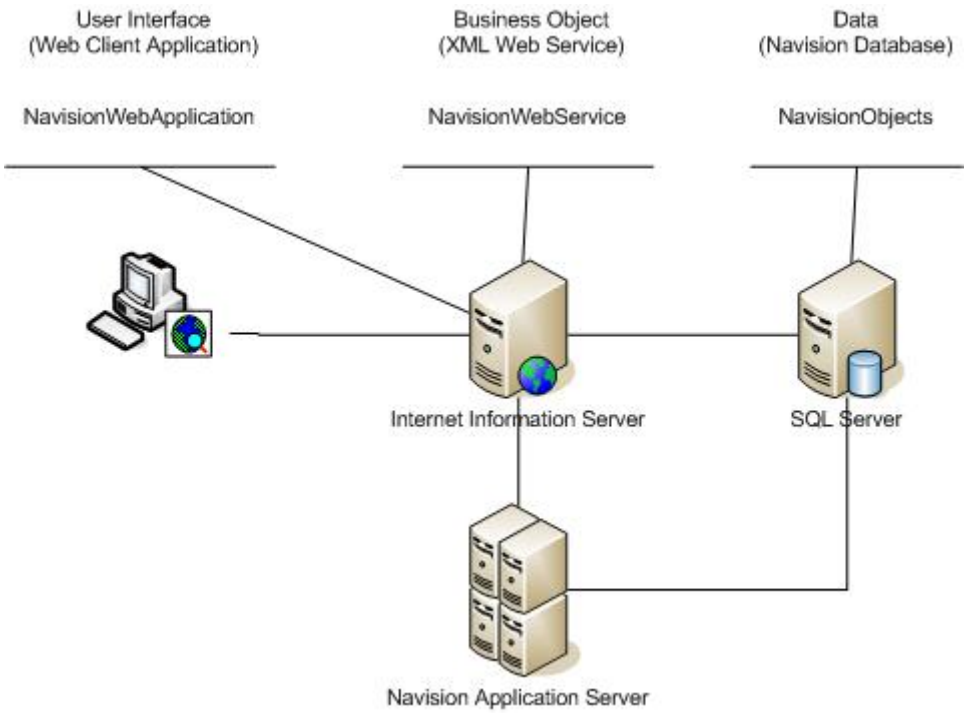


Finally, they can post a sales order and get the status of the order back.



		Document No	Sell-to Customer Name	Status
Detail	Post	101018	New Concepts Furniture	
Detail	Post	101016	The Cannon Group PLC	
Detail	Post	2001	The Cannon Group PLC	
Detail	Post	2006	The Cannon Group PLC	Error: There is nothing to post.
Detail	Post	2011	The Cannon Group PLC	
Detail	Post	2002	Selangorian Ltd.	
Detail	Post	2007	Selangorian Ltd.	
Detail	Post	2012	Selangorian Ltd.	
Detail	Post	101005	John Haddock Insurance Co.	
Detail	Post	101023	John Haddock Insurance Co.	
Detail	Post	2003	John Haddock Insurance Co.	

Code samples overview



User Interface (web client application: Navision Web Application)

- Visual Basic .NET Project Name: NavisionWebApplication
- URL: http://servername/NavisionWebApplication/default.aspx

Business Object (XML web service: NavisionWebService)

- Visual Basic .NET Project Name: NavisionWebService
- URL: http://servername/NavisionWebService/NavisionService.asmx
- Supported Operations:

  GetCompany (return the list of companies in the Navision database)

```
POST /NavisionWebService/NavisionService.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/GetCompany"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetCompany xmlns="http://tempuri.org/" />
  </soap:Body>
</soap:Envelope>
```

GetSalesDocument (return the list of sales orders in a company)

```
POST /NavisionWebService/NavisionService.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/GetSalesDocument"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetSalesDocument xmlns="http://tempuri.org/">
      <CompanyName>string</CompanyName>
    </GetSalesDocument>
  </soap:Body>
</soap:Envelope>
```

GetSalesDocumentDetail (return the list of sales lines in a sales order in a company)

```
POST /NavisionWebService/NavisionService.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/GetSalesDocumentDetail"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetSalesDocumentDetail xmlns="http://tempuri.org/">
      <CompanyName>string</CompanyName>
      <DocumentType>int</DocumentType>
      <DocumentNo>string</DocumentNo>
    </GetSalesDocumentDetail>
  </soap:Body>
</soap:Envelope>
```

RunNavisionProcess (run a Navision function in a company)

```
POST /NavisionWebService/NavisionService.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/RunNavisionProcess"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <RunNavisionProcess xmlns="http://tempuri.org/">
      <CompanyName>string</CompanyName>
      <FunctionName>string</FunctionName>
      <ParameterList>string</ParameterList>
    </RunNavisionProcess>
  </soap:Body>
</soap:Envelope>
```

GetCompany, GetSalesDocument and GetSalesDocumentDetail connect directly to the Navision database (SQL Server) to return the appropriate records.
RunNavisionProcess connects to the Navision Application Server. The Navision Application Server connects to the Navision database to run the defined function.

Accessing Navision data from .NET

One database / multiple companies

One Navision database can contain several companies. The data for each company are kept in different SQL tables.
When a SQL table contains the data for 1 company, its name is prefixed with the name of the company:

Navision Table	SQL Database	SQL Table
Misc. Article	Navision Demo Database	CRONUS Australia Pty_ Ltd_$Misc_ Article
		CRONUS New Zealand Ltd_$Misc_ Article

When a SQL table contains data common to all companies, its name is not prefixed.

To access the Navision database, you can build SQL Statements in .NET. You must not hard code the name of the tables. Otherwise, your .NET development will not work if a company is added to the database.

A user can only work in one company at a time. You can keep the name of the current company in an http cookie. At the beginning of a session, the user selects the company to work with, and the browser stores the name of the company in a cookie for the duration of the session.

NavisionWebApplication \ WebForm3.aspx

```
If Me.ListBox1.SelectedIndex >= 0 Then
   MyCookie.Value = Me.ListBox1.SelectedItem.ToString
   Response.Cookies.Add(MyCookie)
End If
```

The first parameter of any XML web service operations is always the company name. When calling an operation, the company name is retrieved from the cookie.

NavisionWebApplication \ WebForm1.aspx

```
SalesDocument.Merge(NavisionService.GetSalesDocument(Request.Cookies.Item("CompanyName").
Value.ToString))
```

Finally, the SQL Statement is populated in the XML web service using the company name to build the table names.

NavisionWebService \ NavisionService.asmx

```
MySQLCommand =
   "SELECT [Document Type], [No.] AS [Document No], [Sell-to Customer Name]
   FROM [" + CompanyName + "$Sales Header] WHERE ([Document Type] = 1)"
```

Identifier Conversion

By default, the “Identifier Conversion” option is enabled in Navision, and the following characters in the Navision object names (such as tables) are converted to an underscore in SQL Server: ."\/'
You can modify the list of identifiers to convert. Using a Navision client, go to Database / Alter… / Integration and modify the content of the of the “Remove Characters” text box.

Using “Identifier Conversion”, the name of the Navision company “CRONUS Australia Pty. Ltd.” will be converted to “CRONUS Australia Pty_ Ltd_” in SQL Server for example. As the name of the Navision company is used to generate the .NET SQL Statements, it is important to work around this issue.

The “Identifier Conversion” option can be disabled in Navision. Using a Navision client, go to Database / Alter… / Integration and un-tick “Convert identifiers”.

Whether this option is enabled or not cannot be retrieved dynamically from Navision by the .NET Visual Basic code. Therefore, to make your .NET solution as portable as possible, you can store whether the option is enabled or not in Navision in the web.config file of the XML web service, by adding a “Convert Identifiers” appSettings key.

NavisionWebService \ Web.config

```
<appSettings>
   <add key="ConvertIdentifiers" value=".\/'"></add>
</appSettings>
```

Note that you cannot specify to remove the “ symbol in the web.config file. Therefore, you must not use this character in the name of any Navision objects (company, tables, fields…).

You can use this key in the web.config file to build a .NET function to convert any characters to an underscore.

NavisionWebService \ NavisionNetFunction.vb

```
Public Function ConvertCharacters(ByVal MyString As String) As String
   Dim i As Integer
   Dim ReturnString As String
   Dim IdentifiersToConvert As String

   IdentifiersToConvert =
      ConfigurationSettings.AppSettings("ConvertIdentifiers")
   For i = 0 To MyString.Length - 1
      If InStr(1, IdentifiersToConvert, MyString.Chars(i), CompareMethod.Text) <> 0 Then
         ReturnString = ReturnString + "_"
      Else
         ReturnString = ReturnString + MyString.Chars(i)
      End If
   Next
   Return (ReturnString)
End Function
```

You can then use this function to build the SQL statements.

NavisionWebService \ NavisionService.asmx

```
MySQLCommand =
   "SELECT [Document Type], [No.] AS [Document No], [Sell-to Customer Name]
   FROM [" + CompanyName + "$Sales Header] WHERE ([Document Type] = 1)"
MySQLCommand = NavisionWebService.NavisionNetFunction.ConvertCharacters(MySQLCommand)
```

Executing Navision business logic from .NET

To execute the Navision business logic, you need to run the Navision application server. As a normal Navision client, the Navision application server can execute Navision business processes coded in Navision C/AL code. It runs as a service, and can only access one Navision company at a time. For a multi-company .NET solution, you need to run one Navision application server per company.

Microsoft Message Queue

The preferred method of communication with the Navision application server is Microsoft Message Queue. As the Navision application server is dedicated to one company only, you can create 2 queues per company: one queue to send instructions to the Navision application server (outgoing messages) and one queue to receive status messages from the Navision application server (incoming messages). Queues can be created automatically the first time a request is sent to the Navision application server.

NavisionWebService \ NavisionService.asmx

```
Dim QueueToNavision As String = ".\private$\" + CompanyName + " 1"
Dim QueueFromNavision As String = ".\private$\" + CompanyName + " 2"
If Not MyQueue2.Exists(QueueFromNavision) Then
   MyQueue2.Create(QueueFromNavision)
   MyQueue2.Label = QueueFromNavision
   MyQueue2.Authenticate = 0
   MyQueue2.MessageReadPropertyFilter.CorrelationId = True
   MyQueue2.UseJournalQueue = True
End If
If Not MyQueue1.Exists(QueueToNavision) Then
   MyQueue1.Create(QueueToNavision)
   MyQueue1.Label = QueueToNavision
   MyQueue1.Authenticate = 0
   MyQueue1.UseJournalQueue = True
End If
```

XML Message format

The label of the messages sent to the Navision application server must be “Navision MSMQ-BA”.

NavisionWebService \ NavisionService.asmx

```
MyMessage1.Label = "Navision MSMQ-BA"
```

The label of the messages received from the Navision application server is “Navision MSMQ-BA-Reply”. The reply is generated automatically by the Navision “MS-Message Queue Bus Adapter” component. Please refer to the relevant documentation on the Navision product CD (file:\\Product CD\Client\Common\Navision\Communication Component\devguide.chm). This component is installed with the Navision SDK.

Codeunit 70000 RunNavisionProcess

```
MSMQ_BA.OpenReceiveQueue('.\private$\' + COMPANYNAME + ' 1',0,0);
MSMQ_BA.OpenReplyQueue('.\private$\' + COMPANYNAME + ' 2',0,0);
```

Messages sent to and from the Navision application server are XML documents.

XML documents to the Navision application server

You can use the following format for the documents sent to the Navision application server.

```
<?xml version="1.0"?>
<NavisionObject xmlns:xsd=http://www.w3.org/2001/XMLSchema
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <CompanyName>The company in which to run the function</CompanyName>
   <FunctionName>The Navision function to run</FunctionName>
   <ParameterList>The potential parameters to run the function</ParameterList>
   <UserName>The Windows login who requested the function to be run</UserName>
</NavisionObject>
```

In .NET, the documents can be generated automatically using a .NET class.

NavisionWebService \ NavisionObject.vb

```
<XmlRoot(Namespace:="urn:www-Navision-com:NavisionObject.xml",
ElementName:="NavisionObject")> _
    <XmlElement(ElementName:="UserName")> _
    Public Property UserName() As String
    <XmlElement(ElementName:="CompanyName")> _
    Public Property CompanyName() As String
    <XmlElement(ElementName:="FunctionName")> _
    Public Property FunctionName() As String
    <XmlElement(ElementName:="ParameterList")> _
    Public Property ParameterList() As String
End Class
```

In Navision, the documents can be read by using “Microsoft XML, v3.0 parser”.

Codeunit 70000 RunNavisionProcess

```
XMLDoc.loadXML(XMLText);
IF XMLDOMMgmt.FindNode(XMLDoc,'/NavisionObject/UserName',XMLNode) THEN
  UserName := XMLNode.text;
IF XMLDOMMgmt.FindNode(XMLDoc,'/NavisionObject/CompanyName',XMLNode) THEN
  CompanyName := XMLNode.text;
IF XMLDOMMgmt.FindNode(XMLDoc,'/NavisionObject/FunctionName',XMLNode) THEN
  FunctionName := XMLNode.text;
IF XMLDOMMgmt.FindNode(XMLDoc,'/NavisionObject/ParameterList',XMLNode) THEN
  ParameterList := XMLNode.text;
```

XML Messages from the Navision application server

Two types of messages are generated from the Navision application server: success and failure messages.

In case of success, the initial document can be returned.

Codeunit 70000 RunNavisionProcess

```
InStreamQueue := InMsg.GetStream();
InStreamQueue.READTEXT(XMLText);
…
XMLDoc.loadXML(XMLText);
…
OutMsg := InMsg.CreateReply
OutStreamQueue := OutMsg.GetStream();
XMLDoc.save(OutStreamQueue);
OutMsg.Send(0);
```

But you can also choose to send back a different document.

In .NET, you can use the correlation ID to match the outgoing message (to the Navision application server) with the incoming message (from the Navision application server).

NavisionWebService \ NavisionService.asmx

```
MyMessage2 = MyQueue2.PeekByCorrelationId(MyMessage1.Id, New TimeSpan(0, 1, 0))
```

In case of failure, a XML document is created automatically (by the InMsg.CreateReply function in Navision) containing the error message in the following format.

```
<?xml version="1.0" ?>
<NavisionErrorMsg xmlns="urn:www-Navision-com:NavisionErrorMsg.xml">
   <ErrorNo>The error number</ErrorNo>
   <ErrorMessage>The error message text</ErrorMessage>
   <ErrorSource>The application that caused the error</ErrorSource>
   <BusAdapterId>The ID of the bus adapter that returns the error</BusAdapterId>
</NavisionErrorMsg>
```

In .NET, the document can be read using a .NET class.

NavisionWebService \ NavisionErrorMsg.vb

```
<XmlRoot(Namespace:="urn:www-Navision-com:NavisionErrorMsg.xml",
ElementName:="NavisionErrorMsg")> _
Public Class NavisionErrorMsg
    <XmlElement(ElementName:="ErrorNo")> _
    Public Property ErrorNo() As String
    <XmlElement(ElementName:="ErrorMessage")> _
    Public Property ErrorMessage() As String
    <XmlElement(ElementName:="ErrorSource")> _
    Public Property ErrorSource() As String
    <XmlElement(ElementName:="BusAdapterId")> _
    Public Property BusAdapterId() As String
End Class
```

To distinguish the type of the incoming documents in .NET, you must use the XMLMessageFormatter.

NavisionWebService \ NavisionService.asmx

```
Dim MyFormatter = New XmlMessageFormatter(New Type() {GetType(NavisionErrorMsg)})
…
If MyFormatter.CanRead(MyMessage2) = True Then
```

Navision application server

Please refer to the relevant documentation in the Navision product CD to install Navision application server (file:\\Product CD\Doc\w1w1atas.pdf).

To start the Navision application server through the Microsoft Management Console, a new parameter must be added in function 99 NASHandler of codeunit 1 ApplicationManagement in Navision.

Codeunit 1 ApplicationManagement

```
CASE Parameter OF
   'WEBSERVICE':
      CODEUNIT.RUN(CODEUNIT::RunNavisionProcess);
   …
END;
```

The Navision application server will then wait for the messages to pop in the queue. According to the name of the function in the document received, specific Navision objects are run.

Codeunit 70000 RunNavisionProcess

```
CASE FunctionName OF
   'PostSalesDocument':
      BEGIN
         …
         IF EVALUATE(DocumentType,SELECTSTR(1,ParameterList)) THEN BEGIN
            IF SalesHeader.GET(DocumentType,SELECTSTR(2,ParameterList)) THEN BEGIN
               SalesHeader.Ship := TRUE;
               SalesHeader.Invoice := TRUE;
               CODEUNIT.RUN(80,SalesHeader);
            END;
         END;
      END;
   ...
END;
```

In Navision, you must customize some of the standard functionalities to disable the messages displayed on the screen (such as progress bars) if the code is run through the Navision application server.

Codeunit 80 Sales-Post

```
IF GUIALLOWED THEN
   Window.OPEN(
      '#1##################################\\' +
      Text002 +
      Text003 +
      Text004 +
      Text005)
```

Integrating Navision security system with .NET

Navision uses its own security system. Thus, it uses an Application role to access the Navision SQL Database (ndoshadow). The role has full permissions on all objects in the Navision SQL database. By default, the Public role has no permissions. Therefore, you can only access the Navision SQL database through the Navision client.

Inside Navision, you can create database logins or give access to specific Windows logins, using Windows authentication. You can assign Navision roles to logins, and give users access to data and processes for specific companies in the database.

Data security

In .NET, you want the users to access the data in the SQL tables directly (without going through the Navision client). You can "duplicate" the Navision security permissions to SQL Server.
For example, you give to a user access to table "Sales Header" inside Navision for all companies. To use the XML web service to access Navision, you need to give to the same user access to table "CRONUS Australia Pty_ Ltd_$Sales Header" and table "CRONUS New Zealand Ltd_$Sales Header" inside SQL Server.

To keep the security consistent, permissions must be managed centrally inside Navision. You can develop in Navision a small component to synchronize the Navision security permissions with SQL Server.

In the code samples, you can run the synchronization through the XML web service. When it runs, permissions for all SQL database users are first reset (deleted and created again).

Codeunit 70001 WebServiceSecurity

```
IF WindowsAccessControl.FIND('-') THEN BEGIN
   REPEAT
      SQLStatement :=
         STRSUBSTNO(
            'USE [%1] EXEC sp_revokedbaccess [%2] EXEC sp_grantdbaccess [%2]',
            SQLDatabase,
            WindowsLogin.ID);
      SQLServer.ExecuteImmediate(SQLStatement);
   …
   UNTIL WindowsAccessControl.NEXT = 0;
…
END;
```

Based on the Navision security permissions, the corresponding permissions are created in SQL Server for the SQL database users.

Codeunit 70001 WebServiceSecurity

```
IF Permission.FIND('-') THEN BEGIN
   REPEAT
      …
      GivePermission(
         SQLDatabase,
         Object."Company Name",
         Object.ID,
         Object.Name,
         WindowsLogin.ID,
         (Permission."Read Permission" =
          Permission."Read Permission"::Yes),
         (Permission."Insert Permission" =
          Permission."Insert Permission"::Yes),
         (Permission."Modify Permission" =
          Permission."Modify Permission"::Yes),
         (Permission."Delete Permission" =
          Permission."Delete Permission"::Yes));
   …
   UNTIL Permission.NEXT = 0;
…
END;
```

Process security

When accessing Navision through a XML web service, processes inside Navision are run through the Navision application server. The Navision application server is started as a Windows service. By default, all the XML web service users will get the permissions given inside Navision to the Windows login used to start the Navision application server. Indeed, the Navision application server does not support security account delegation. A workaround has to be implemented to manage permissions per user on business processes.

The suggested workaround uses Windows logins. A client sends its Windows credential together with its requests to run Navision business objects.

NavisionWebApplication \ WebForm1.aspx

```
Dim NavisionService As New NavisionWebApplication.localhost.NavisionService()
NavisionService.Credentials = System.Net.CredentialCache.DefaultCredentials
…
NavisionService.RunNavisionProcess(Request.Cookies.Item("CompanyName").Value.ToString,
   "PostSalesDocument", e.Item.Cells(2).Text + "," + e.Item.Cells(3).Text)
```

The XML web service gets the Windows credential, and forwards it through to the Navision application server together with the name of the company and the process to run.

NavisionWebService \ NavisionService.asmx

```
Dim QueueToNavision As String = ".\private$\" + CompanyName + " 1"
Dim MyNavisionObject1 As NavisionObject = New NavisionObject()
…
MyNavisionObject1.UserName = Me.User.Identity.Name()
MyNavisionObject1.CompanyName = CompanyName
MyNavisionObject1.FunctionName = FunctionName
…
MyMessage1 = New Message(MyNavisionObject1)
…
MyQueue1.Send(MyMessage1)
```

The Navision application server gets the Windows credential, and checks it against the Navision security permissions before running the process.

Codeunit 70000 RunNavisionProcess

```
InStreamQueue := InMsg.GetStream();
InStreamQueue.READTEXT(XMLText);
…
XMLDoc.loadXML(XMLText);
…

IF XMLDOMMgmt.FindNode(XMLDoc,'/NavisionObject/UserName',XMLNode) THEN
  UserName := XMLNode.text;
IF XMLDOMMgmt.FindNode(XMLDoc,'/NavisionObject/CompanyName',XMLNode) THEN
  CompanyName := XMLNode.text;
IF XMLDOMMgmt.FindNode(XMLDoc,'/NavisionObject/FunctionName',XMLNode) THEN
  FunctionName := XMLNode.text;
CASE FunctionName OF
   'PostSalesDocument':
      BEGIN
         WebServiceSecurity.CheckPermission(UserName,CompanyName,0,36,1,0,1,1,0);
         …
         CODEUNIT.RUN(80,SalesHeader);
      END;
   ...
END;
```

REFERENCES

Microsoft .NET Framework SDK: QuickStarts, Tutorials and Samples (Microsoft Knowledge Base Article – 316736)

Microsoft Visual Basic .NET on-line help: Walkthrough: Creating a Distributed Application

Navision Development Guide for Communication Components (Navision Product CD)

Installation & System Management: Microsoft® Business Solutions–Navision® SQL Server Option (Navision Product CD)

Installation & System Management: Business Solutions–Navision® Application Server (Navision Product CD)