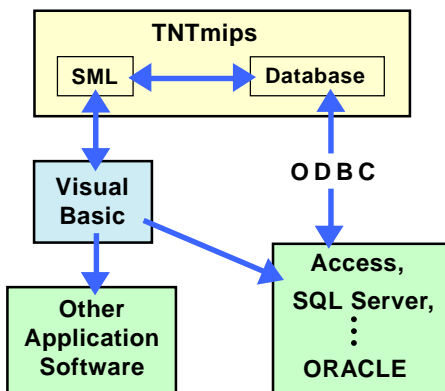


Communicate with Visual Basic Programs using SML

Your SML (Spatial Manipulation Language) scripts in TNTmips can now use Microsoft Windows ActiveX technology to launch and communicate with component software programs created in Visual Basic, C++, or Java. An SML script can directly access component class structures (data structures and methods), open a component dialog window (form), and exchange data between SML and the dialog window. Any spatial or attribute data that the SML script reads from your TNT objects can be transmitted to the component program. Any information processed by the component program can be transmitted to other application programs or sent back to SML to be written to your Project File data.

To implement a Visual Basic (VB) component for use with TNTmips, take these steps:

- design a VB form (dialog window) if a user interface is required
- create properties, methods, and data structures in VB as needed
- write an SML script that "imports" the VB component and handles data transfer between VB and your TNT spatial data and attached database attributes
- use the Visual Studio 6.0 Tools / Package and Deployment Wizard to build an installation package to allow others to install and use the component program



The Visual Basic demonstration component described here lets you modify records in a database table stored in a TNTmips Project File. But that table could instead be an ODBC link to a table maintained in Access or other database software. Either way, the appropriate records are updated and available to subsequent TNT operations. You could also write your Visual Basic application to take the database information provided by TNTmips and communicate directly with Access or any other ActiveX-aware software.

Example: Database Entry using Visual Basic Form

MicroImages has created a sample application to demonstrate how an SML script can access program actions and dialogs created in Visual Basic. This example uses an SML Tool Script to provide interactive selection of vector elements in a View window in TNTmips. When you run the script and select a vector polygon representing a land parcel, a form dialog created in Visual Basic opens for you to enter owner name information to be added to the database record for the parcel. Functions and data structures registered with the Visual Basic application are directly accessible in SML and are used by SML to send data to and retrieve data from the Visual Basic form.

The sample SML toolscript and Visual Basic code are shown on the other side of this page. You can download these and other files required to run this demonstration from: microimages.com/freestuff/smlscripts.htm. After downloading and unzipping the VBDEMO file, take the following steps:

1. Run the Setup program to register the Visual Basic component program.
2. In TNTmips, display the vector object Parcel from the Parcels Project File.
3. Install the ParcelToolVB toolscript button on the View window (View window's Options / Customize / Tool Scripts...).

4. Run the *ParcelToolVB* SML toolscript; the script imports the Visual Basic class named *VBTestForm*.

5. Left-click to select a parcel polygon. The toolscript passes the parcel ID from the vector's *ParcelInfo* table to the *VBTestForm* data structure.

The toolscript calls the *VBTestForm* class method that opens the Test Form dialog window created in Visual Basic.

6. Enter names into the First Name and Last Name fields in the Test Form dialog and press [OK].

The toolscript retrieves the names from the *VBTestForm* data structure and writes them to the appropriate fields in the *ParcelInfo* table in the vector's polygon database.

ParcelID	OwnerLastName	OwnerFirstName
SC182395		
SC203845	Chandler	Raymond
SC218977		
SC223403	Fowles	John
SC245903		
SC253218	Turow	Scott

Visual Basic Source Code for VBDEMO (MicroImages_SML_OLE_Demo.VBTestForm)

```

Option Explicit
Private mElemID As String
Private mFirstName As String
Private mLastName As String

Public Sub Clear()
    mFirstName = ""
    mLastName = ""
    mElemID = ""
End Sub

Public Property Get ElemID() As String
    ElemID = mElemID
End Property
Public Property Get FirstName() As String
    FirstName = mFirstName
End Property
Public Property Get LastName() As String
    LastName = mLastName
End Property
Public Property Let ElemID(ByVal str As String)
    mElemID = str
End Property
Public Property Let FirstName(ByVal str As String)
    mFirstName = str
End Property
Public Property Let LastName(ByVal str As String)
    mLastName = str
End Property

Private Sub Class_Initialize()
    Clear
    Debug.Print "Initialize Thing, ID=" & mElemID & " Name=" & mLastName & ", " & mFirstName
End Sub

Private Sub Class_Terminate()
    On Error Resume Next
    Debug.Print "Terminate Thing, ID=" & mElemID & " Name=" & mLastName & ", " & mFirstName
End Sub

Public Sub ShowDialog()
    Dim mdlg As dlgDemo
    Set mdlg = New dlgDemo
    With mdlg
        .txtDemo.Text = mElemID
        .txtFirstName = mFirstName
        .txtLastName = mLastName
        .Caption = "Test Form"
        .Show vbModal
        If .WasOK Then
            mFirstName = .txtFirstName
            mLastName = .txtLastName
            mElemID = .txtDemo
        End If
    End With
End Sub

```

define private data variables

define class method to clear the fields in the Test Form dialog window

define read/write properties of variables used for fields in Test Form

initialize the component class

terminate the component class

define class method to open the Test Form dialog window

NOTE: to enable multiple instances of the component class to be used in SML, build your Visual Basic project as an ActiveX DLL rather than an ActiveX EXE.

called when OK button is pressed

transfer dialog field values to private data variables

NOTE: The dialog window (form) is constructed in Visual Basic using the graphic Form Designer or the Application Wizard

Excerpt of Parcel Tool Script (ParcelToolVB.sml)

(use the Edit icon button in the Customize Tool Script window to view the script)

```

$import MicroImages_SML_OLE_Demo.VBTestForm

class VECTORLAYER vectorLayer;
class Vector targetVector;
class GROUP activegroup;
class VBTestForm form;
class Database ParcelDB;

func OnLeftButtonPress () {
    if (checkLayer()) {
        local class POINT2D point;
        local numeric elementNum;
        local string parcelID$;

        point.x = PointerX;
        point.y = PointerY;
        point = TransPoint2D(point, ViewGetTransViewToScreen(View, 1));
        point = TransPoint2D(point, ViewGetTransMapToView(View, vectorLayer.Projection, 1));

        elementNum = FindClosestPoly(targetVector, point.x, point.y, GetLastUsedGeorefObject(targetVector));

        if (elementNum > 0)
            vectorLayer.Poly.HighlightSingle(elementNum);
            parcelID$ = targetVector.Poly[elementNum].ParcelInfo.ParcelID$;

            form.Clear();
            form.ElemID = parcelID$;
            form.ShowDialog();

            targetVector.Poly[elementNum].ParcelInfo.OwnerLastName$ = form.LastName;
            targetVector.Poly[elementNum].ParcelInfo.OwnerFirstName$ = form.FirstName;
        }
    }
}

```

use preprocessor keyword to "import" the Visual Basic component class *VBTestForm*. Note that preprocessor command lines must NOT end in a semicolon (;).

variable declarations

declare instance of imported VB class *VBTestForm*

called when user presses 'left' pointer/mouse button

if the selected layer is valid, proceed

Set local variables

Get screen coordinates from cursor and transform to map coordinates

highlight polygon

get the parcel ID for the polygon from the database

clear fields in VB Dialog window if it has already been opened

pass the parcel ID to the *VBTestForm* data structure to show in the form

open the Test Form dialog window

read the strings entered in the name fields in Test Form and write them to the record in the *ParcelInfo* table

end of OnLeftButtonPress()

After you have opened the script in the SML Editor window and checked syntax, the imported class appears in the Insert Class window

