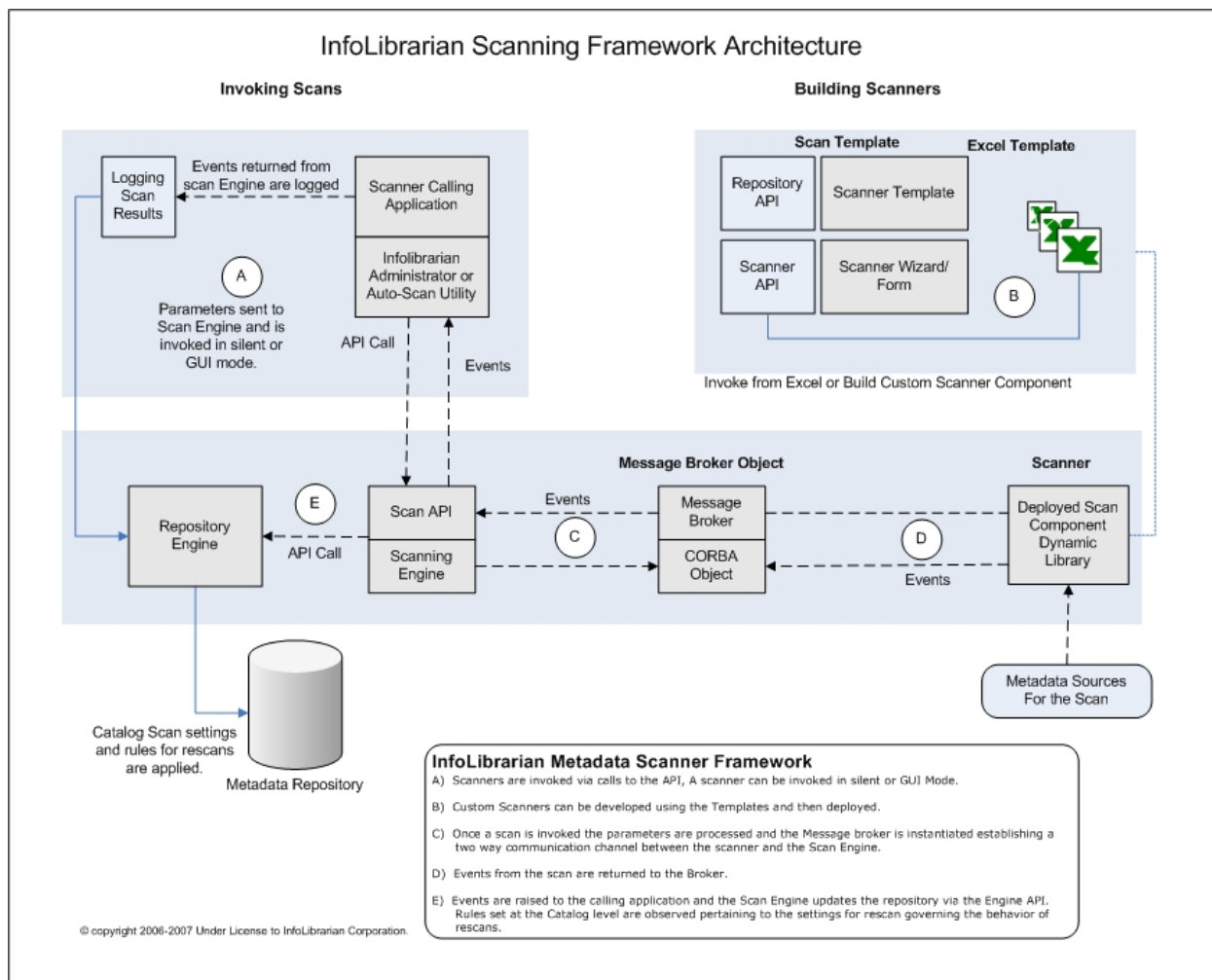
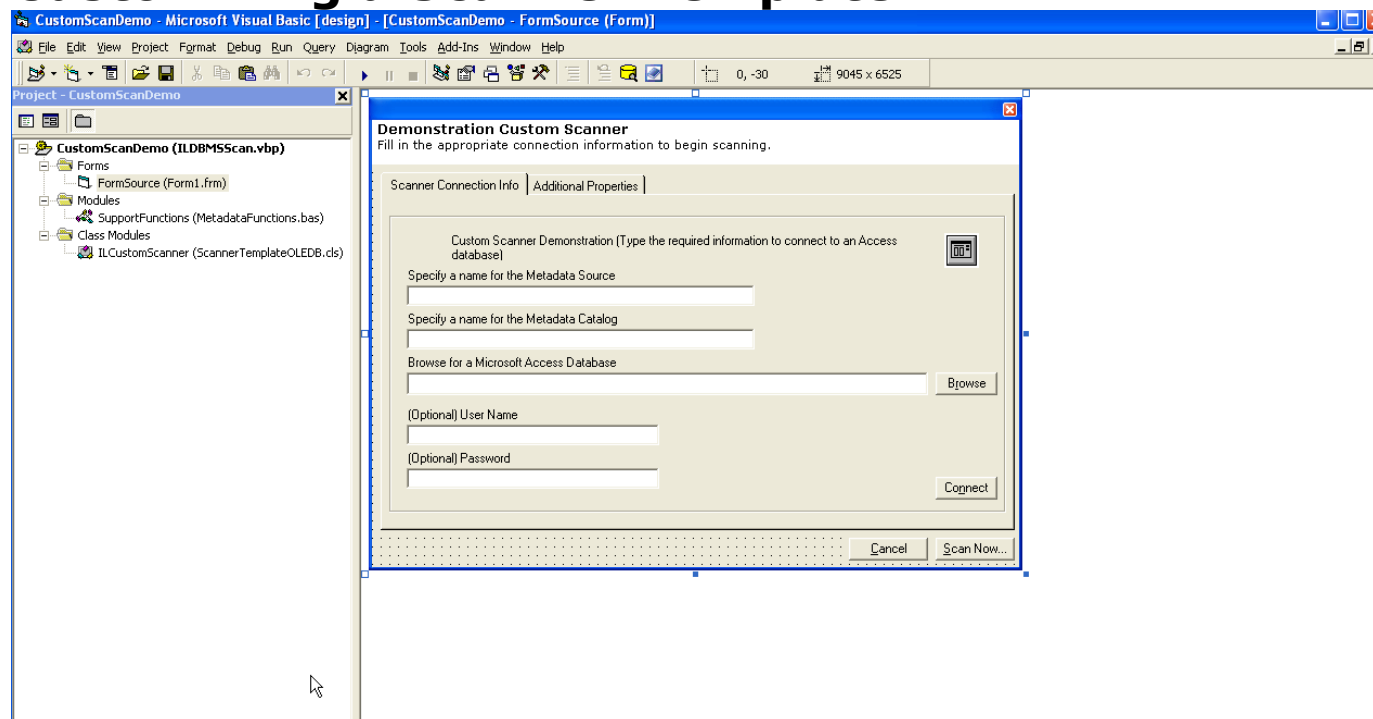


Custom Scanner Development



Customizing a Scanner Template



Scanner Form

You will notice a form within the scanner project; this form is setup for passing the scanner parameters into the main scan function. Modify this form to customize a new scanner. In most cases you can select a template and only modify the title etc... on this form to distinguish your custom scanner in your environment. But you will not normally need to change the form itself. Note: If you do customize this form be sure to code the appropriate handlers behind the form as per the code template.

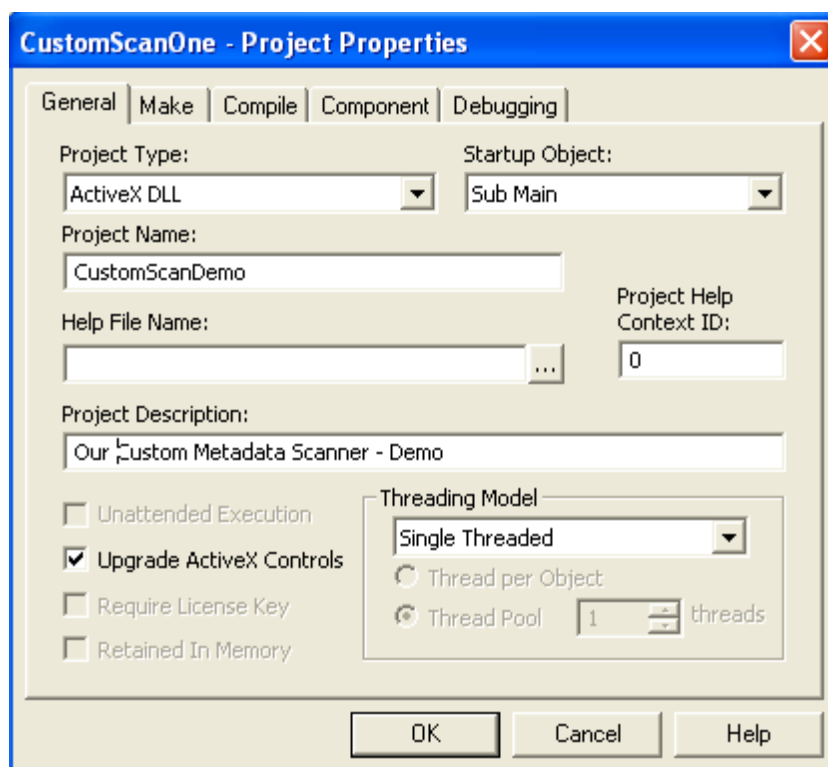
Scanner Code

The primary method of customizing a scanner involved modifying this function only:

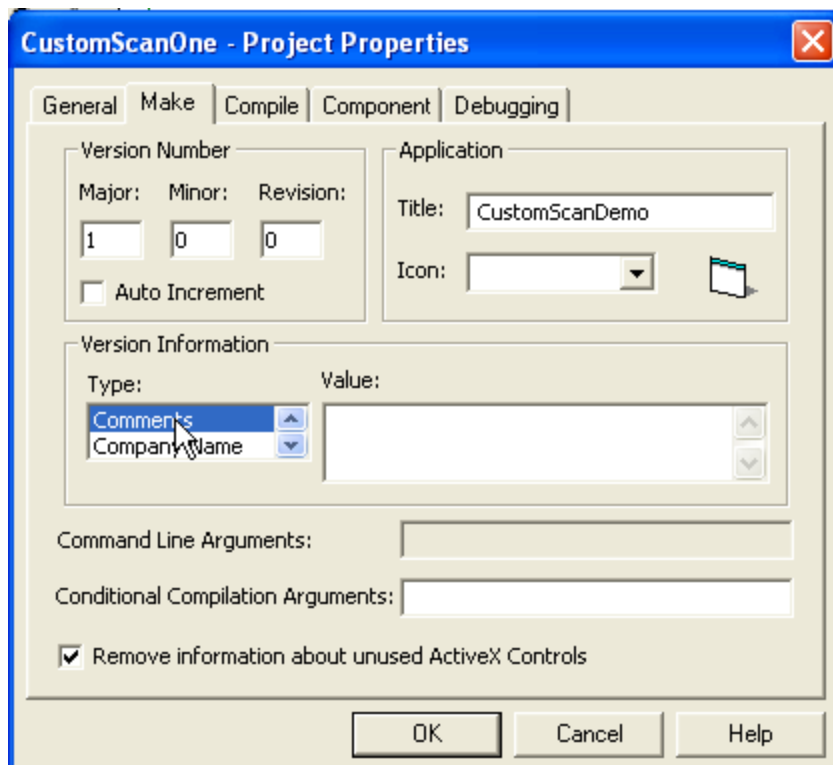
MapMetadata()

The MapMetadata function contains the code for mapping the source metadata to the repository utilizing the Scanner and Repository API function calls. Please refer to the SDK documentation for more information.

Registering a Custom Scanner Visual Basic Project Properties Dialog



Within the Project Properties Dialog, Provide a name and description for your custom scanner.



On the Make tab ensure that the title is the same as in the previous step.

```

CustomScanDemo - Microsoft Visual Basic [design] - [ILCustomScanner (Code)]
File Edit View Project Format Debug Run Query Diagram Tools Add-Ins Window Help
Ln 30, Col 50

Project - CustomScanDemo
  CustomScanDemo (ILDBMS5Scan.vbp)
    Forms
      FormSource (Form1.frm)
    Modules
      SupportFunctions (MetadataFunctions.bas)
    Class Modules
      ILCustomScanner (ScannerTemplateOLEDB.cls)

(General) ScanMetadata

'The Product is protected by copyright and other intellectual property laws and treaties. Brian Brewer own
'-----
' InfoLibrarian Standard Scanner COM API Template ()
'-----
'*** LOCAL VARIABLES ***
'-----

Private ObjectCount As Long           'Object Count returned by scan
Private arrCount As Long              'Used to parse array in Scanner Pump
Private SourcePathsArray() As Variant 'Array of paths to scan
Private SourceFileTypesArray() As Variant 'Array of File types filters
Private retval As Boolean

'Scanner Events Raised to the Client
Public Event ObjectCount(EventName As String, ByVal ObjectCount As Long)
Public Event ProgressPercent(EventName As String, ByVal Percent As Integer)
Public Event ResetProgress(EventName As String, ByVal Max As Integer)
Public Event ProgressMessage(EventName As String, ByVal Message As String)
Public Event cRaiseError(EventName As String, ByVal Message As String, ByVal Number As Long, ByVal Source As String)
Private WithEvents ScannerAPI As ILScannerAPI

'-----
' MAIN SCANNER FUNCTION
'-----
Public Function ScanMetadata(ScannerArguments() As Variant) ' Standard Scanner Entry Point Function

    Set ScannerAPI = New ILScannerAPI           'Creates new instance of Scanner API
    ScannerAPI.MetadataProvider = "CustomScanDemo" 'Change this to the scanner you create here
    ScannerAPI.CurrentObjectType = 0           'Set this to the object type being scanned in the catalog
    Call ScannerPump(ScannerArguments)

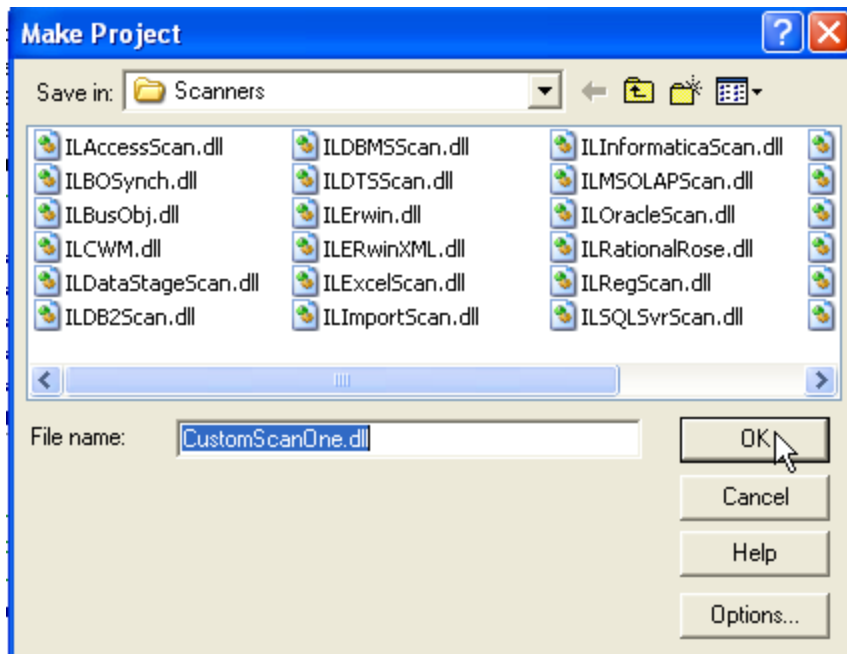
End Function

'-----
' MAIN METADATA MAPPING FUNCTION
'-----

```

Important, within the Class "CustomScanner", change this line of code to specify the scanner name provided in the previous 2 steps. Ensure that it is identical and it is case sensitive.

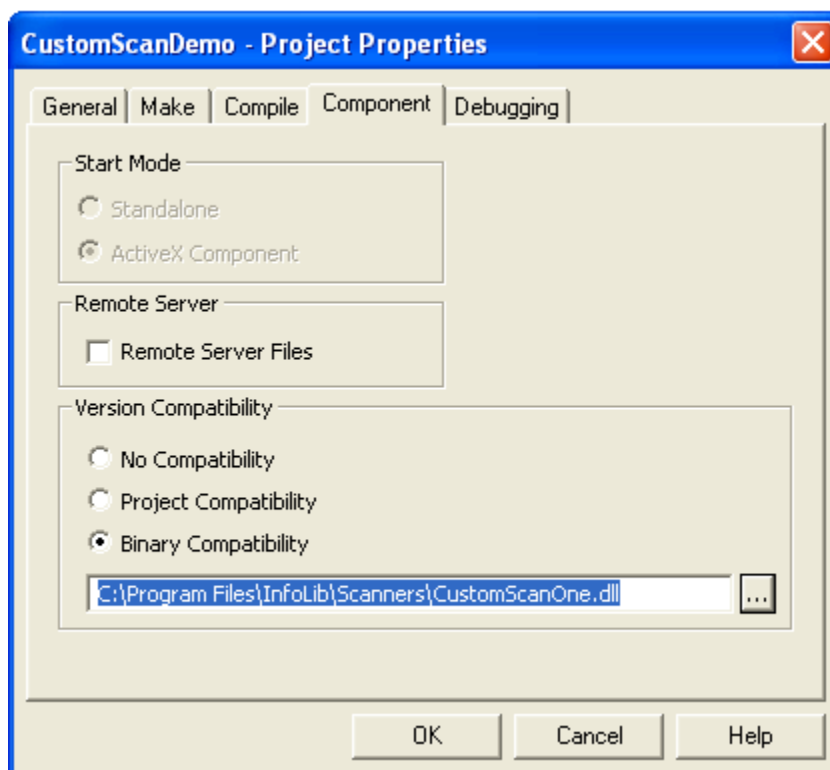
Building the DLL for the Scanner



In the file menu click "Make xxx.dll" to compile your dll.

Provide a unique dll name here.

Once the dll is initially built...



Within the project properties dialog, set the Binary Compatibility then rebuilt the dll.

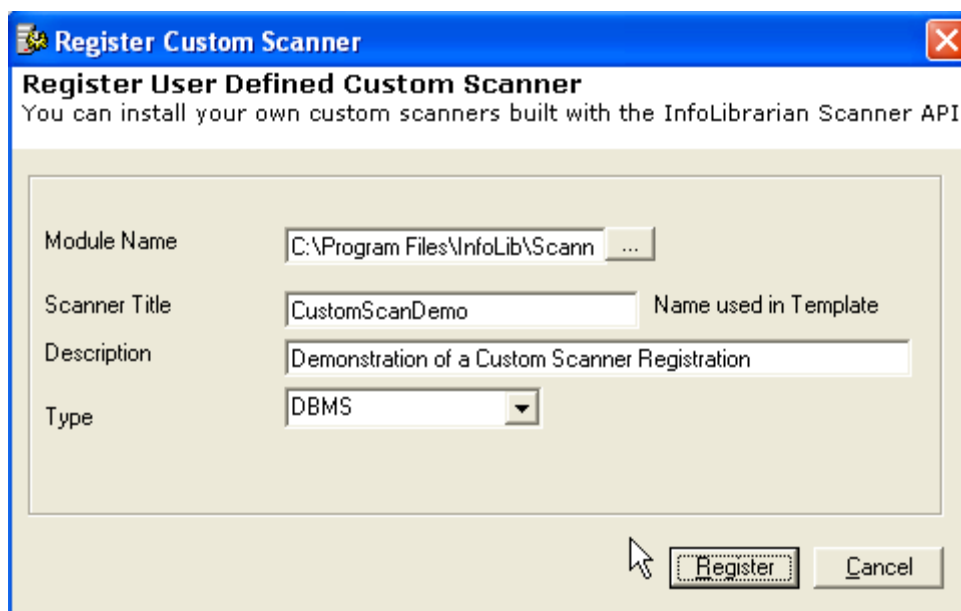
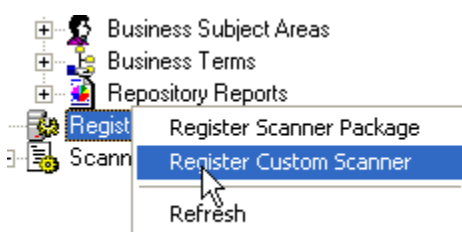
This will enable debugging and proper registration of the dll.

Note : When building a custom scanner for the first time, ensure that the binary compatibility is set to "No Compatibility" Build the dll then reset to "Binary Compatibility" then rebuilt the dll again before deploying.

Registering the Scanner

1. Copy the dll to the machine on which you wish to register the custom scanner.
2. Register the new scanner dll using regsvr32.exe
3. Open the InfoLibrarian Administrator

Right click on the "Register Custom Scanner" in order to register your new custom scanner.



In this dialog enter the information as follows:

Module Name: Full path to dll created

Scanner Title: Name specified in the project dialog and template

Any description you wish

Choose the type of scanner template that was utilized for building this custom scanner.

Running the Scanner

	InfoLibrarian Metadata	Scanner	Description	Version	Engine Build
	Metadata Semantics				
	Business Rules	Access Metadata Scanner	Scans Access Databases for Schema Metadata	4.1 Compatible	4.1
	Business Subject Areas	Ascential DataStage ETL Scanner	Scans Ascential Data Stage ETL Metadata	4.1 Compatible	4.1
	Business Terms	Business Objects Metadata Scanner	Scans Business Objects DBMS based Universes	4.1 Compatible	4.1
	Repository Reports	CustomScanDemo	Demonstration of a Custom Scanner Registration	CUSTOM	NA
	Registered Scanners	DB2 Metadata Scanner	Scans DB2 RDBMS Metadata	4.1 Compatible	4.1
	Scanning Logs	DTS Package ETL Scanner	Scans DTS Packages	4.1 Compatible	4.1
		Erwin Native Scanner	Captures Erwin Metadata from ER1 Files	4.1 Compatible	4.1
		ERwin XML Scanner	Scans Erwin XML Extracts for Model Metadata	4.1 Compatible	4.1
		Excel Metadata Scanner	Scans Excel Files for Schema Metadata	4.1 Compatible	4.1
		InfoLibrarian Business Objects Synchronizer	Updates Terms and Descriptions to Business Objects	4.1 Compatible	4.1
		Informatica Repository Scanner	Scans, maps metadata and lineages from Informatica	4.1 Compatible	4.1

Once you refresh the list you should see your custom scanner registered similar to the above. You will notice the version should indicate "CUSTOM"

Double Click the scanner to run it as with any other scanner.