EMC²
where information lives

QuickScan 4.5

Exporter Software Development Kit

EMC² captiva

Copyright

© Copyright 2006 EMC Corporation. All rights reserved.

EMC^2, Captiva, ClaimPack, ISIS, PixTools, Claims Editor Professional, FormWare, and InputAccel Express are registered trademarks, and AdvantEdge, ClaimsEditor, CodeLink, Digital Mailroom, Dispatcher, Dispatcher for InputAccel, eInput, FirstPass, InputAccel for Invoices, InputAccel, PixTools/Distributed Imaging, and QuickScan Pro are trademarks of EMC Corporation. All other trademarks used herein are the property of their respective owners.

All information contained in this documentation is proprietary to EMC Corporation. Information in this document is subject to change without notice and EMC Corporation undertakes no commitment to update the document. The software described in this document is furnished under a license agreement. The document may not, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form without prior consent, in writing, from EMC Corporation.

Part Number: QuickScan_ExporterSDK.doc 4.5 (DA 6-June-2006)

Contents

Introduction 1

SDK Package Contents 2

Getting started 3

Using the Sample Exporter 3

Creating a custom QuickScan exporter 4

Creating an ActiveX Control Visual Basic project 4

Adding a Class module 5
Exposing an exporter 6

Adding Export Profile Editor tabs 8

Exposing a Configuration tab 8
Using the State property 9
Using the Configuration tab State string 10

Exporting using the State string 11

Exporting batch information 11

Using QSExportHelper1 12

To access QSExportHelper functions, see “Using the Indexer interface 12

Finishing your custom exporter 12

Distributing your exporter 14

Registering the QuickScan exporter 14

Understanding exporter interfaces 15

Overview 15

Using the QuickScan interfaces 15

Using the PixQSI.Exporter interface 15
Using the PixQSI.ExporterConfigPanel interface 17
Using the Indexer interface 18
Using the helper interfaces 19

Using the QuickScan batch information helper 22

Troubleshooting 28

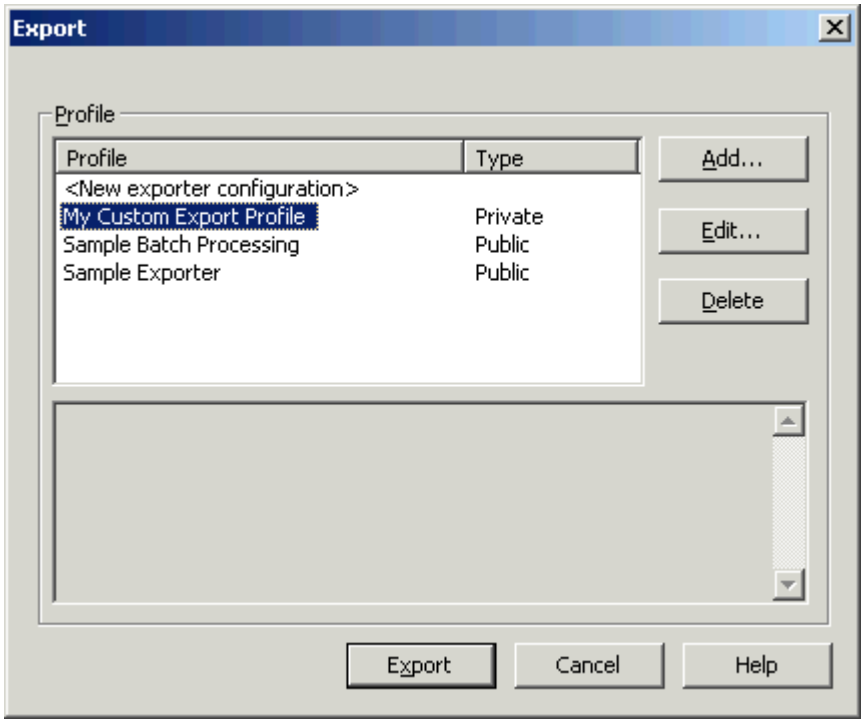
Introduction

A QuickScan Pro **exporter** is a framework that can be used for creating custom Export profiles in QuickScan Pro. Exporters can be designed to perform customized tasks, such as renaming image files based on barcode detection or exporting files to a database or other backend application, such as a document management system.

When you have finished creating an exporter as explained in this document, QuickScan Pro end users will be able to use that exporter to create custom Export profiles by selecting it from the Select an Exporter dialog.



After an end user creates a custom Export profile, it is available for use from the QuickScan Export dialog.



For details on creating custom Export profiles, refer to QuickScan help by selecting **Help > Contents** or pressing [**F1**].

SDK Package Contents

This SDK package contains the following:

- **QuickScan_ExporterSDK.pdf** - Explains how to make a custom QuickScan exporter.
- **PIXQSI.TLB** - Defines the interfaces required to make an exporter.
- **SAMPLES\My Exporter** - Source code for the samples are detailed in this document.
- **SAMPLES\Sample Exporter** - Source code for another sample exporter.

Redistribution of any files on this disk is strictly controlled by the accompanying license agreement. Please read the license agreement and contact EMC/Captiva if you have questions.

The redistribution of most files is not allowed without the execution of a specific Redistribution Agreement. IMPORTANT: Under no circumstances can you redistribute PixQSI.TLB.

Getting started

Using the Sample Exporter

A Sample Exporter is included with the Exporter SDK that demonstrates what a finished exporter project looks like and how it works. You can also use this sample as a starting point for creating your own exporter.

For instructions on building your own exporter from scratch, refer to "Creating a custom QuickScan exporter" on page 4.

To begin using the Sample Exporter:

1. Copy PixQSI.TLB to the Windows system directory on your development machine. For example, the Windows NT/2000 system directory will be winnt\system32.
2. Launch Visual Basic 6.0 and load the following project file: SAMPLES\Sample Exporter\SampleExporter.vbp.
3. Select **Project > References** and select **PixQSI.TLB** in the reference list. You may need to select **Browse** and locate this file in the Windows system directory.
4. Select **File > Make SampleExporter.ocx** to compile this project.
5. Run the Registry Editor application and navigate to the following path:

   [HKEY_LOCAL_MACHINE\SOFTWARE\EMC Captiva\QuickScan\Exporters]
6. In the above registry path, add the following string value: **SampleExporter.Exporter**. QuickScan will now show **Sample Exporter** in the list of available exporters in the Select an Exporter dialog.

Creating a custom QuickScan exporter

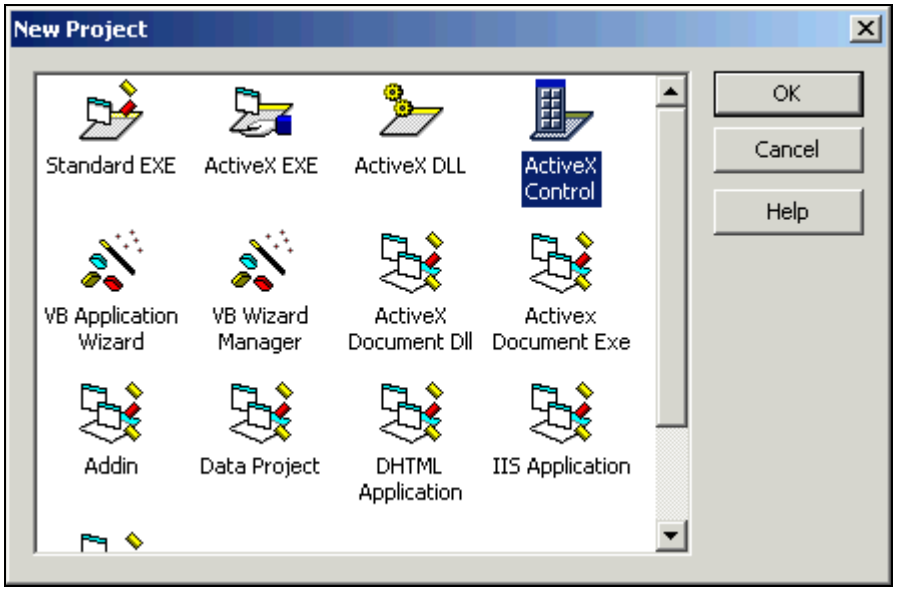
This section provides step-by-step instructions for creating a custom QuickScan Exporter using the Visual Basic IDE.

Creating an ActiveX Control Visual Basic project

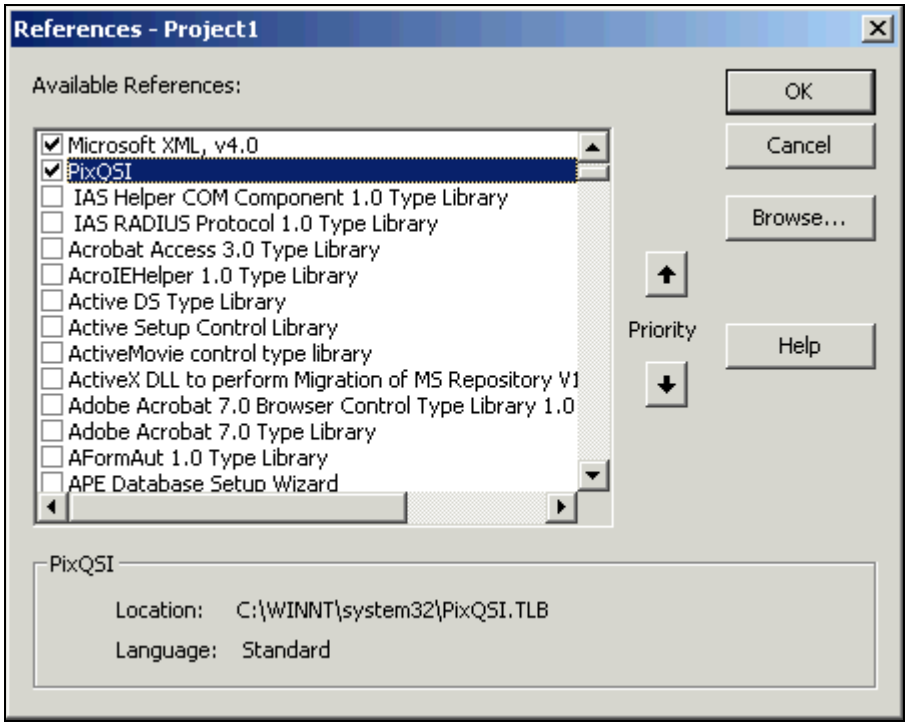
The first part of creating a custom QuickScan exporter is creating an ActiveX Control Project with the appropriate references.

To create an ActiveX Control project for a QuickScan exporter:

1. Make sure that PixQSI.TLB is copied to the Windows system directory on your development machine. For example, the Windows NT/2000 system directory will be winnt\system32.
2. Launch Visual Basic 6.0.
3. Select **File > New Project**. The New Project dialog displays.



4. Select **ActiveX Control**, and then select **OK**.
5. Select **Project > References**. The References dialog displays.



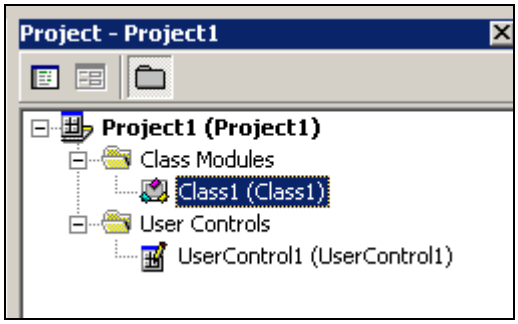
6. Select the following:
   - **PixQSI.TLB** (you may need to select **Browse** and locate this file in the Windows system32 directory)
   - **Microsoft XML, v4.0**

Adding a Class module

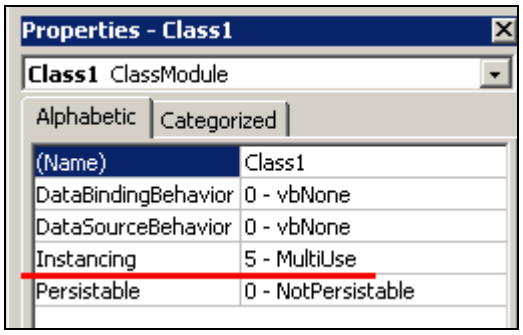
By default, the ActiveX Control Project Wizard added the **UserControl1** control to the project. We will use this control soon, but for now we will add a class module to the project.

To add a class module:

1. Select **Project > Add Class Module**. The Add Class Module dialog displays.
2. Select **Class Module**, and then select **Open**. You can name it anything you want. For this example, we will name it **Class1**.



3. In the Properties Window, set the Instancing property to **MultiUse**.



Exposing an exporter

- The PixQSI TLB provides the exporter interface **PixQSI.Exporter**. Using this interface, we will expose each exporter name, description, and configurable flag. For details about the PixQSI.Exporter interface, refer to "**Indexer** - Provides information about available Index profiles and their fields.

Using the PixQSI.Exporter interface" on page 15.

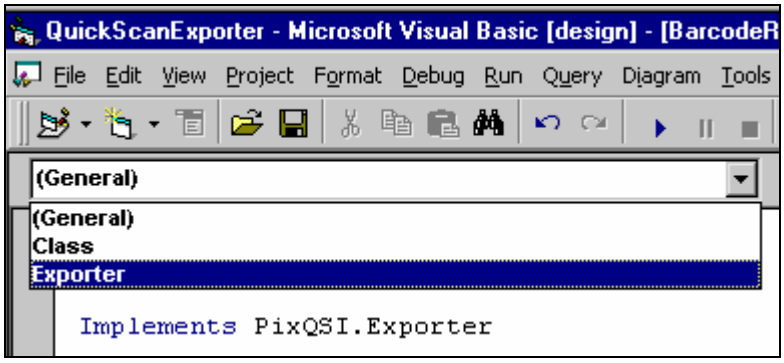
To implement the PixQSI.Exporter interface:

1. Open the Class1 source code and add this line:

   ```
   Implements PixQSI.Exporter
   ```

   If there is an Option Explicit line, add the Implements line after that.

2. From the Object selection drop-down menu, select **Exporter**.



3. Copy the following code into your sample project. QuickScan will use the Name and Description properties to properly list this exporter. The following code will expose one exporter named **My Exporter**.

   ```
   Private Property Get Exporter_Name() As String
    Exporter_Name = "My Exporter"
   End Property

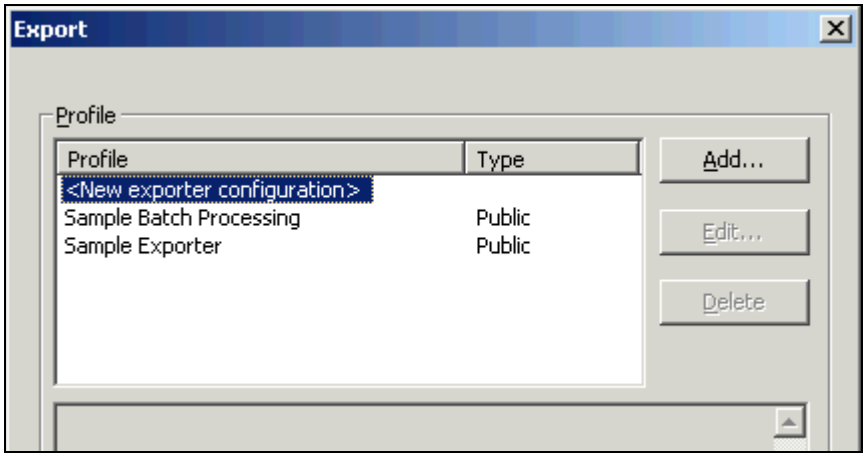
   Private Property Get Exporter_Description() As String
    Exporter_Description = "My Exporter Description"
   End Property
   ```

4. Create shell functions and properties (containing no code) for all other exporter functions and properties so that VB can compile it.

Testing the exporter

To test this exporter:

1. Select **File > Make Project1.ocx**. This automatically registers the OCX.
2. Register the OCX as a QuickScan exporter using the name **project1.class1** in the Registry Editor.
3. Launch QuickScan and select **File > Export**. The Export dialog displays.

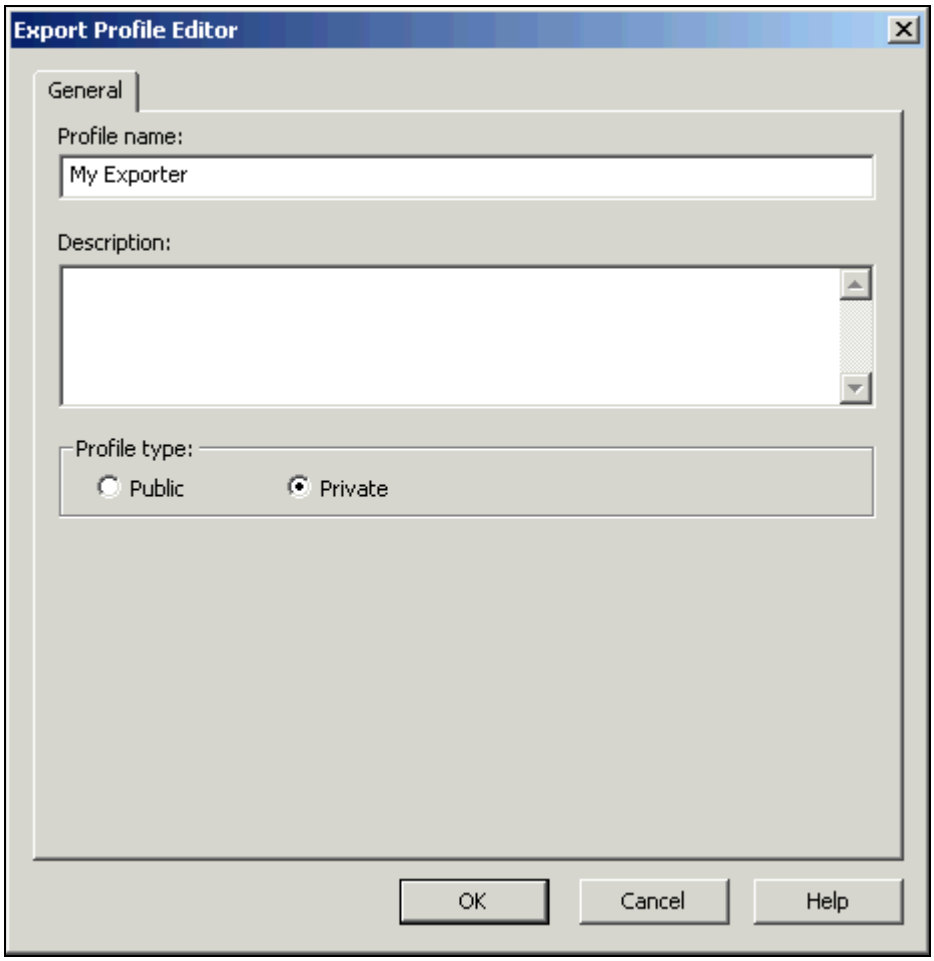


4. Select **<New exporter configuration>**, and then select **Add**. The Select an Exporter dialog displays. Your registered exporter should appear as **My Exporter**.



Adding Export Profile Editor tabs

By default, QuickScan generated the Name and Description Tab, which will contain the name and description for the configured exporter. This section explains how to add your own configuration tabs to the QuickScan Export Profile Editor.



Exposing a Configuration tab

The PixQSI TLB provides the exporter configuration panel interface **PixQSI.ExporterConfigPanel**. For details about the ExporterConfigPanel interface, refer to “Using the PixQSI.ExporterConfigPanel interface” on page 17.

Using this interface, we will do the following:

- Expose the exporter tab name.
- Set the GUI based on the state string.
- Set the state string based on the GUI.

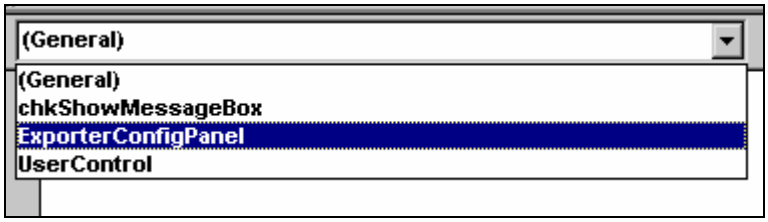
To implement PixQSI.ExporterPanelConfig:

1. Open the UserControl1 source code and add this line:

```
Implements PixQSI.ExporterConfigPanel
```

If there is an Option Explicit line, add the Implements line after that.

2. From the Object selection drop-down menu, select **ExporterConfigPanel**.



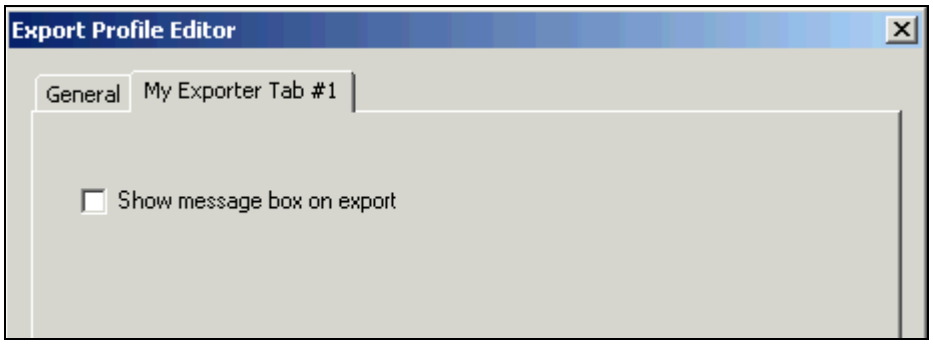
3. To register UserControl1, add the following function to Class1:

```
Private Function Exporter_AddConfigPanels(ByVal Panels As PixQSI.Panels) As
Long
  Call Panels.Add("Project1.UserControl1", "Configure")
  Exporter_AddConfigPanels = 0
End Function
```

4. To give the UserControl1 a tab title, add the following function to UserControl1:

```
Private Property Get ExporterConfigPanel_Name() As String
  ExporterConfigPanel_Name = "My Exporter Tab #1"
End Property
```

5. Create shell functions and properties (containing no code) for all other exporter functions and properties so that VB can compile it.
6. Add a check box to this panel with the following properties:
   - Name = chkShowMessageBox
   - Caption = Show message box on export
7. Compile and register your exporter as described in “Distributing your exporter” on page 14. In QuickScan, your Export Profile Editor tab should have a tab called “My Exporter Tab #1” with a **Show message box on export** check box.



Using the State property

The PixQSI.Exporter API provides the **PixQSI.Exporter.State** property, which is used to store and restore the configuration state of the exporter. QuickScan does not require an exporter to use the State string. These user-defined settings can be used to change the behavior of the exporter either at configuration or export time.

When the QuickScan Export Profile Editor displays, the exporter is given the last known saved State string.

This section describes how the exporter can receive the state string and also provide that state (or substate) to the other configuration tabs of the same exporter.

When the exporter is first loaded by QuickScan, it is given the state string last stored. The exporter class which implemented the PixQSI.Exporter will be given that State string via the State property. In our example, class1 implemented PixQSI.Exporter.

By adding the following code to class1 of our sample exporter, we can:

- Let and Get the state string provided from QuickScan (Let and Get Exporter_State)
- Let and Get the state string provided to UserControl1 (Let and Get UserControl1_State)

```
Private LocalState As String

Private Property Let Exporter_State(ByVal RHS As String)
  LocalState = RHS
End Property

Private Property Get Exporter_State() As String
  Exporter_State = LocalState
End Property

Public Property Get UserControl1State() As String
  UserControl1State = LocalState
End Property

Public Property Let UserControl1State(p As String)
  LocalState = p
End Property
```

Note: The Exporter_State property is a required API of PixQSI.Exporter.

Note: The UserControl1_State property is not a required API of PixQSI.Exporter, but it does facilitate the transfer of the state or substate between class1 and usercontrol1.

Using the Configuration tab State string

When a configuration tab is shown, it is given the object of the class that implemented the PixQSI.Exporter. In our example, class1 implemented PixQSI.Exporter. The above custom API (Let and Get UserControl1_State) can now be used by the Export Profile Editor tab to gain read/write access to the State String.

By adding the following code to UserControl1 of our sample exporter, we can:

- Get the class1 object (Let and Get ExporterConfigPanel_Exporter)
- Let and Get the state string provided from class1 (Let and Get UserControl1State)

```
Private gQSExporter As Class1

Private Property Set ExporterConfigPanel_Exporter(ByVal RHS As
PixQSI.Exporter)
  Set gQSExporter = RHS
End Property

Private Function ExporterConfigPanel_Hide() As Long
  gQSExporter.UserControl1State = chkShowMessageBox.Value
End Function

Private Function ExporterConfigPanel_Show() As Long
  If gQSExporter.UserControl1State = "1" Then
```

```
    chkShowMessageBox.Value = 1
  Else
    chkShowMessageBox.Value = 0
  End If
End Function
```

The above code will take get the State string from class1 and update the check box on UserControl1. When this configuration tab is hidden, QuickScan will call the ExporterConfigPanel_Hide function, thus allowing the current tab to save away its state.

Exporting using the State string

This section explains how to export using the State string collected in “Using the State property” on page 9.

When QuickScan performs an export operation, it does the following:

1. Provides the state string to class1 through Let Exporter_State
2. Calls Export in class1 defined by PixQSI.Exporter

Our sample exporter already handles step 1 by reusing the configuration code already done.

By adding the following code to our class1, we can handle step 2.

```
Private Function Exporter_Export(pxDoc As MSXML2.IXMLDOMDocument2) As Long
  If LocalState = "1" Then
    MsgBox "Hello from My Exporter"
  End If
End Function
```

Finally, compile this change and perform an export operation on a scanned image. If the Export Profile Editor had the check box checked (added in “Exposing a Configuration tab” on page 8), then the following message box will appear.



Exporting batch information

To access the Batch Information provided to the exporter, use the functions described in “Using the QuickScan batch information helper” on page 22.

Using QSExportHelper1

To access QSExportHelper functions, see "Using the Indexer interface

The Indexer interface provides information about available indexers and all associated index fields. Each indexer corresponds to one Index profile. Use the QSExporterHelper1 interface to get access to the indexer objects. The Indexer interface can be used to enumerate fields defined in this index.

Note: Use only the functions and properties in the table below. The Indexer interface also provides other functions and properties, but they are not implemented and will not work.

Name	VB Declaration	Public Property Get Name() As String
	Description	Provides the name of the indexer.
	Parameters	None
	Return Value	The indexer name
Description	VB Declaration	Public Property Get Name() As String
	Description	Provides the description of the indexer.
	Parameters	None
	Return Value	The indexer description
ValueNameCount	VB Declaration	Public Property Get ValueNameCount() As Long
	Description	Provides the number of fields defined in the indexer.
	Parameters	None
	Return Value	Number of fields
ValueName	VB Declaration	Public Function ValueName(index As Long) As String
	Description	Provides the name for the specified field.
	Parameters	Index = the field index
	Return Value	Field name

Using the helper interfaces" on page 18.

Finishing your custom exporter

QuickScan can now:

- Recognize the exporter
- Configure the exporter
- Export using the exporter

Adding more exporters to a single exporter OCX can be done as described in "Exposing an exporter" on page 6.

Adding more configuration tabs to a single exporter can be done as described in “Adding Export Profile Editor tabs” on page 8.

Distributing your exporter

To distribute a third-party exporter for use with QuickScan, your installer must do the following:

- Copy and register the OCX file on a machine running QuickScan Pro
- Register the QuickScan exporter

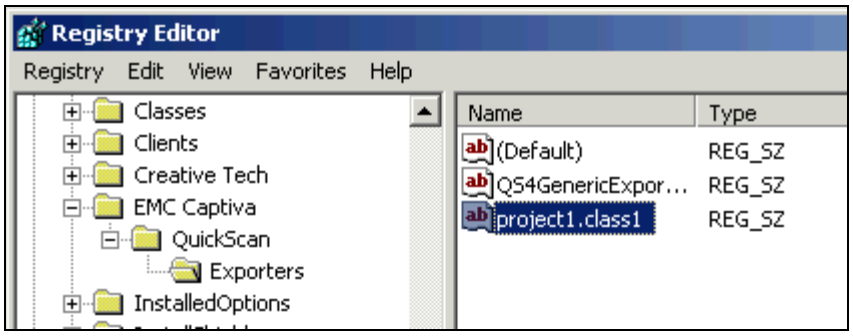
Registering the QuickScan exporter

In order for QuickScan Pro to show an exporter in the list of available exporters:

- The ocx must be registered with Windows.
- The project and class name (which Implements PixQSI.Exporter) must be listed below this registry path as a string name:

[HKEY_LOCAL_MACHINE\SOFTWARE\EMC Captiva\QuickScan\Exporters]

For example, our sample exporter (created as described in “Creating a custom QuickScan exporter” on page 4) would have the following entry in the Registry Editor.



Understanding exporter interfaces

Overview

The QuickScan Pro Exporter SDK allows third-party developers to create custom exporters. The SDK includes the following modules:

- QuickScan Interfaces Type Library (PixQSI.TLB)
- Sample exporters

Using the QuickScan interfaces

The QuickScan Interface type library (PixQSI.TLB) provides these Interfaces and Classes to support the export operations:

- **Exporter** – Provides the core API for an exporter.
- **ExporterConfigPanel** – Provides the API for an exporter configuration panel.
- **QSExportHelper1** – Provides the QuickScan access API for an exporter.
- **QSExportHelperCmdLine** – Provides access to a QuickScan command line for an exporter.
- **QSHelperVersionInfo** – Provides information about the QuickScan version.
- **Indexer** - Provides information about available Index profiles and their fields.

Using the PixQSI.Exporter interface

The PixQSI.Exporter Interface provides the core API for the exporter and allows you to do the following:

- Get the Name and Description values
- Get and Set the State value
- Define what configuration panels appear
- Validate the state
- Start an export operation

Name	VB Declaration	Public Property Get Name() As String
	Description	Used by QuickScan during the parent exporter selection process. The return value from this property is used as the name appearing in the list of available exporters.
	Parameters	None
	Return Value	VB String with the Name
Description	VB Declaration	Public Property Get Description() As String
	Description	Used by QuickScan during the exporter selection process. The return value from this property is used as the description appearing in the list of available exporters.
	Parameters	None
	Return Value	VB String with the Description
AddConfigPanels	VB Declaration	Public Function AddConfigPanels(ByVal Panels As Panels) As Long
	Description	Used by the QuickScan Export Profile Editor. This call gives the exporter the opportunity to add its own configuration tabs to the basic Export Profile Editor. Zero or more panels can be added in this function.
	Parameters	Panels — Panels provided by this exporter
	Return Value	0 = No error 1 = Error
Export	VB Declaration	Public Function Export(ByRef pxDoc As MSXML2.DOMDocument) As Long
	Description	Used by QuickScan to start the export operation for this exporter.
	Parameters	pDoc – XML Structure containing the complete batch information
	Return Value	0 = OK 1 = Cancelled -1 = Error
State (let)	VB Declaration	Public Property Let State(ByVal StateString As String)
	Description	Sets the state string this exporter will use during the export.
	Parameters	StateString – VB String with the State
	Return Value	None
State (get)	VB Declaration	Public Property Get State() As String
	Description	Gets the state string this exporter will use during the export.
	Parameters	None
	Return Value	VB String with the State
Validate	VB Declaration	Public Function Validate() As Long

	Description	Used by QuickScan to confirm the state of the exporter configuration before the Export Profile Editor disappears. This is an opportunity for the exporter to make a final error check on all panels if needed.
	Parameters	None
	Return Value	0 = No error -1 = Error
Reset	VB Declaration	Public Sub Reset()
	Description	Called by QuickScan during the configuration GUI to reset the exporter state. The exporter code will decide what this reset will do.
	Parameters	None
	Return Value	None
Helper	VB Declaration	Public Property Set Helper(Helper As ExporterHelper)
	Description	Called by QuickScan before the exporter configuration GUI appears and before export starts. The passed parameter provides the exporter access to Helper Interfaces (refer to “Using the QuickScan interfaces” on page 18).
	Parameters	Helper object provided by QuickScan
	Return Value	None

Using the PixQSI.ExporterConfigPanel interface

The PixQSI.ExporterConfigPanel Interface provides the configuration API for the exporter and allows you to do the following:

- Get the Name
- Hide and Show configuration GUI controls
- Validate the state

Name	VB Declaration	Public Property Get Name() As String
	Description	Used by QuickScan during the Export Profile Editor dialog. The return value from this property appears as the tab title to this panel appearing in the QuickScan Exporter Profile Editor dialog.
	Parameters	None
	Return Value	VB String with the Name
Exporter	VB Declaration	Public Property Let Exporter(exp As Exporter)
	Description	Passes a reference to the exporter for which this panel is created.
	Parameters	Exp = reference to the exporter
	Return Value	None
Helper	VB Declaration	Public Property Let Helper(hlp As ExporterHelper)
	Description	Passes a reference to the helper object.

	Parameters	Hlp = reference to helper
	Return Value	None
Hide	VB Declaration	Public Function Hide() As long
	Description	Used by QuickScan during the exporter configuration dialog. Informs the configuration tab that is being hidden.
	Parameters	None
	Return Value	0 = No error -1 = Error
Show	VB Declaration	Public Function Show() As Long
	Description	Used by QuickScan during the Export Profile Editor dialog. Informs the configuration tab that is being shown.
	Parameters	None
	Return Value	0 = No error -1 = Error
Validate	VB Declaration	Public Function Validate() As Long
	Description	Used by QuickScan during the exporter configuration dialog. Asks the configuration tab to validate its user inputs before switch away from this configuration tab.
	Parameters	None
	Return Value	0 = No error -1 = Error

Using the Indexer interface

The Indexer interface provides information about available indexers and all associated index fields. Each indexer corresponds to one Index profile. Use the QSExporterHelper1 interface to get access to the indexer objects. The Indexer interface can be used to enumerate fields defined in this index.

Note: Use only the functions and properties in the table below. The Indexer interface also provides other functions and properties, but they are not implemented and will not work.

Name	VB Declaration	Public Property Get Name() As String
	Description	Provides the name of the indexer.
	Parameters	None
	Return Value	The indexer name
Description	VB Declaration	Public Property Get Name() As String
	Description	Provides the description of the indexer.
	Parameters	None
	Return Value	The indexer description
ValueNameCount	VB Declaration	Public Property Get ValueNameCount() As Long

	Description	Provides the number of fields defined in the indexer.
	Parameters	None
	Return Value	Number of fields
ValueName	VB Declaration	Public Function ValueName(index As Long) As String
	Description	Provides the name for the specified field.
	Parameters	Index = the field index
	Return Value	Field name

Using the helper interfaces

The PixQSI Helper Interfaces provide a developer Query and Interface access into QuickScan. These interfaces include:

- PixQSI.QSExportHelper1 (page19)
- PixQSI.QSExportHelperCmdLine (page 20)
- PixQSI.QSHelperVersionInfo (page 21)

Understanding helper interface sample code

The PixQSI.Exporter API (see page 15) includes the property Helper. Through this property, the exporter can gain access to all the Helper Interfaces.

The following code shows how the Helper property receives all the helper interfaces (via the RHS parameter) and stores them into gHelper (the object containing all the interfaces). Individual Helper Interfaces are then cast from gHelper.

For example:

gParamHelper is cast from **gHelper** with **Set gParamHelper = gHelper**

gVersionHelper is cast from **gHelper** with **Set gVersionHelper = gHelper**

```
Private gHelper As PixQSI.ExporterHelper
Private gParamHelper As PixQSI.QSExportHelperCmdLine
Private gVersionHelper As PixQSI.QSHelperVersionInfo

Private Property Set Exporter_Helper(ByVal RHS As PixQSI.ExporterHelper)
 Set gHelper = RHS
 On Error Resume Next
 Set gParamHelper = gHelper
 Set gVersionHelper = gHelper
End Property
```

Using the PixQSI.QSExportHelper1 interface

The PixQSI.QSExportHelper1 Interface provides access to QuickScan indexer information including:

- Number of QuickScan Indexers
- Names of QuickScan Indexers
- Access to any instance of a QuickScan indexer PixQSI.Indexer (PixQSI.Indexer is outside the scope of this document)

IndexerCount	VB Declaration	Public Property Get IndexerCount() As Long
	Description	Provides the number of configured QuickScan indexers
	Parameters	None
	Return Value	The number of configured Indexer
IndexerName	VB Declaration	Public Property Get IndexerName(ByVal Index As Long) As String
	Description	Provides the name of a specific configured QuickScan indexer
	Parameters	Index = Index of the configured indexer
	Return Value	Name of the indexer at the index provided
Indexer	VB Declaration	Public Property Get Indexer(ByVal Index As Long) As Indexer
	Description	Provides an instance of the configured QuickScan indexer
	Parameters	Index = Index of the configured indexer
	Return Value	Instance of the configured QuickScan indexer
ChangeBehavior	VB Declaration	Public Function ChangeBehavior(ByVal Behavior As ExportHelperBehavior, ByVal Value As Long) As Long
	Description	Allows the exporter to control a set of QuickScan behaviors.
	Parameters	Behavior: 1 = enables the configuration dialog “OK” button
	Return Value	None

Using the PixQSI.QSExportHelperCmdLine interface

The PixQSI.QSExportHelperCmdLine Interface provides access to QuickScan command line information including:

- Command line mode
- Command line settings

InCommandLineMode	VB Declaration	Public Property Get InCommandLineMode() As Boolean
	Description	Informs the exporter that QuickScan runs in the command-line mode.
	Parameters	None
	Return Value	True = Command line mode is On False = Command line mode is Off
CommandLineSetting	VB Declaration	Public Property Get CommandLineSetting(ByVal Name As String) As String
	Description	Provides a single command line value referenced by the name passed in.
	Parameters	Name = Name of the command line parameter.
	Return Value	The value of the command line parameter

Using the PixQSI.QSHelperVersionInfo interface

The PixQSI.QSHelperVersionInfo Interface provides access to QuickScan version information including:

- Demo Mode
- Demo Type
- Level
- Pro Mode

Demo	VB Declaration	Public Property Get Demo() As Long
	Description	Provides the state of QuickScan Demo mode
	Parameters	None
	Return Value	0 = Non Demo -1 = Demo
DemoType	VB Declaration	Public Property Get DemoType() As Long
	Description	Provides the state of QuickScan Demo Type
	Parameters	None
	Return Value	0 = Not Applicable 1 = 30 Launch Demo 2 = Expiring
Level	VB Declaration	Public Property Get Level() As Long
	Description	Provides the state of QuickScan Level
	Parameters	None
	Return Value	-1 = Not Applicable (in Demo Mode) 0 = Level 0 (View only - no scanning) 1 = Level 1 2 = Level 2 3 = Level 3
Pro	VB Declaration	Public Property Get Pro() As Long
	Description	Provides the state of QuickScan Pro
	Parameters	None
	Return Value	0 = Non Pro -1 = Pro

Using the QuickScan batch information helper

The QuickScan Batch Information Helper module (BatchInfoHelper.bas) provides a number of functions to access the batch information provided by QuickScan.

The BatchInfoHelper.bas file is located in the Samples\Sample Exporter subfolder. If you include this file in your project, then you also need to include the following files in the same folder:

- CharacterEncoding.bas
- Localize.basn

FirstPage	VB Declaration	Public Function FirstPage(BatchInfo As MSXML2.DOMDocument40) As IXMLDOMNode
	Description	Returns the first page in the batch
	Parameters	XML Batch node
	Return Value	XML Page node
NextPage	VB Declaration	Public Function NextPage(BatchInfo As MSXML2.DOMDocument40, CurrentPage As IXMLDOMNode, Optional CrossDocumentBoundary As Boolean = True) As IXMLDOMNode
	Description	Returns the next page in the batch
	Parameters	XML Batch node, XML Page node of current page
	Return Value	XML Page node
PageFileName	VB Declaration	Public Function PageFileName(Page As IXMLDOMNode) As String
	Description	Returns the PageFileName for the given XML Page node
	Parameters	XML Page node
	Return Value	Page Filename
PageIndex	VB Declaration	Public Function PageIndex(Page As IXMLDOMNode) As Long
	Description	Returns the PageIndex for the given XML Page node
	Parameters	XML Page node
	Return Value	Page Index
PageDocumentIndex	VB Declaration	Public Function PageDocumentIndex(Page As IXMLDOMNode) As Long
	Description	Returns the DocumentIndex for the given XML Page node
	Parameters	XML Page node
	Return Value	Document Index
FindPageParameterValue	VB Declaration	Public Function FindPageParameterValue(Page As IXMLDOMNode, PageValueName As String) As String
	Description	Finds a Page level parameter value

	Parameters	XML Page node,Page Value Name being searched for
	Return Value	The value of the page level parameters
FirstIndexField	VB Declaration	Public Function FirstIndexField(Page As IXMLDOMNode, IncludeParents As Boolean) As IXMLDOMNode
	Description	Returns the first index field
	Parameters	XML Page node, IncludeParent: True = will include the parent index values False = will not include the parent index values
	Return Value	XML node containing the first index field
NextIndexField	VB Declaration	Public Function NextIndexField(Page As IXMLDOMNode, CurrentField As IXMLDOMNode, IncludeParents As Boolean) As IXMLDOMNode
	Description	Returns the next index field
	Parameters	XML Page node, IncludeParents CurrentField = the current index field
	Return Value	XML node containing the next index field
IndexFieldName	VB Declaration	Public Function IndexFieldName(Field As IXMLDOMNode) As String
	Description	Returns the index field name given the XML Page node
	Parameters	Field = XML Field node
	Return Value	Field Name
IndexFieldValue	VB Declaration	Public Function IndexFieldValue(Field As IXMLDOMNode) As String
	Description	Returns the index field value given the XML Page node
	Parameters	XML Field node
	Return Value	Field Value
FindIndexField	VB Declaration	Public Function FindIndexField(Page As IXMLDOMNode, FieldName As String) As IXMLDOMNode
	Description	Finds an index field node by index field name
	Parameters	XML Page node FieldName = name of the index field to find

	Return Value	If FieldName was found then
		return = XML Index Field node
		else
		return = nothing
		end

FindIndexFieldValue	VB Declaration	Public Function FindIndexFieldValue(Page As IXMLDOMNode, FieldName As String) As String
	Description	Finds an index field value by index field name

	Parameters	XML Page node FieldName = name of the index field to find
	Return Value	Index field value
FirstBarcode	VB Declaration	Public Function FirstBarcode(Page As IXMLDOMNode) As IXMLDOMNode
	Description	Returns the first XML Barcode node in the provided XML Page node
	Parameters	XML Page node
	Return Value	XML Barcode node
NextBarcode	VB Declaration	Public Function NextBarcode(Page As IXMLDOMNode, CurrentBarcode As IXMLDOMNode) As IXMLDOMNode
	Description	Returns the next XML Barcode node in the provided XML Page node starting after the provided XML Barcode node
	Parameters	Page As IXMLDOMNode, CurrentBarcode As IXMLDOMNode
	Return Value	XML Barcode node
BarcodeText	VB Declaration	Public Function BarcodeText(Barcode As IXMLDOMNode, Optional Default As String = "") As String
	Description	Returns the barcode text for the provided XML Barcode node
	Parameters	XML Barcode node
	Return Value	Barcode text
BarcodeType	VB Declaration	Public Function BarcodeType(Barcode As IXMLDOMNode, Optional Default As String = "") As String
	Description	Returns the barcode type of the provided XML Barcode node
	Parameters	XML Barcode node
	Return Value	Barcode type (full list not provided in this document)
BarcodeLeft	VB Declaration	Public Function BarcodeLeft(Barcode As IXMLDOMNode) As Long
	Description	Returns the left position of the provided XML Barcode node
	Parameters	XML Barcode node
	Return Value	Barcode left position
BarcodeWidth	VB Declaration	Public Function BarcodeWidth(Barcode As IXMLDOMNode) As Long
	Description	Returns the width of the provided XML Barcode node
	Parameters	XML Barcode node
	Return Value	Barcode width

BarcodeHeight	VB Declaration	Public Function BarcodeHeight(Barcode As IXMLDOMNode) As Long
	Description	Returns the height of the provided XML Barcode node
	Parameters	XML Barcode node
	Return Value	Barcode height
BarcodeTop	VB Declaration	Public Function BarcodeTop(Barcode As IXMLDOMNode) As Long
	Description	Returns the top position of the provided XML Barcode node
	Parameters	XML Barcode node
	Return Value	Barcode top position
FirstPatchcode	VB Declaration	Public Function FirstPatchcode(Page As IXMLDOMNode) As IXMLDOMNode
	Description	Finds the first patchcode on the provided page
	Parameters	XML Page node being searched
	Return Value	XML Patchcode node
NextPatchcode	VB Declaration	Public Function NextPatchcode(Page As IXMLDOMNode, CurrentPatchcode As IXMLDOMNode) As IXMLDOMNode
	Description	Find the next patchcode after the current patch code provided
	Parameters	XML Page node, XML Patchcode node
	Return Value	XML Patchcode node
PatchcodeType	VB Declaration	Public Function PatchcodeType(Patchcode As IXMLDOMNode) As String
	Description	Returns the patchcode type of the provided patchcode
	Parameters	XML Patchcode node
	Return Value	Patchcode type (full list not provided in this document)
FirstColorContent	VB Declaration	Public Function FirstColorContent(Page As IXMLDOMNode) As IXMLDOMNode
	Description	Finds the first Color Content Node
	Parameters	XML Page node
	Return Value	XML ColorContent node
ColorContentValue	VB Declaration	Public Function ColorContentValue(ColorContent As IXMLDOMNode, Optional Default As Single = 0) As Single
	Description	Returns the color content value
	Parameters	XML ColorContent node
	Return Value	Color content value
FirstBlankPage	VB Declaration	Public Function FirstBlankPage(Page As IXMLDOMNode) As IXMLDOMNode

	Description	Returns the first IP Blank page results for this page
	Parameters	XML Page node
	Return Value	XML IP Result node
BlankPageValue	VB Declaration	Public Function BlankPageValue(BlankPage As IXMLDOMNode, Optional Default As Boolean = False) As Boolean
	Description	Returns status of the blank page value
	Parameters	XML Page node
	Return Value	True = Blank Page False = not a Blank Page
FirstIPResult	VB Declaration	Public Function FirstIPResult(Page As IXMLDOMNode, ResultType As String) As IXMLDOMNode
	Description	Returns the first XML IP Results node of type ResultType for the provided XML Page node
	Parameters	XML Page node, IP Node Name
	Return Value	XML IP Results node
NextIPResult	VB Declaration	Public Function NextIPResult(Page As IXMLDOMNode, CurrentResult As IXMLDOMNode) As IXMLDOMNode
	Description	Returns the next XML IP Results node after the provided XML Page node.
	Parameters	XML Page node, Current XML IP Results node
	Return Value	XML IP Results node
FirstOCRResult	VB Declaration	Public Function FirstOCRResult(Page As IXMLDOMNode) As IXMLDOMNode
	Description	Returns the first XML OCRResult node in the provided XML Page node
	Parameters	XML Page node
	Return Value	XML OCRResult node
NextOCRResult	VB Declaration	Public Function NextOCRResult(Page As IXMLDOMNode, CurrentOCRResult As IXMLDOMNode) As IXMLDOMNode
	Description	Returns the next XML OCRResult node in the provided XML Page node starting after the provided XML OCRResult node
	Parameters	Page As IXMLDOMNode, CurrentOCRResult As IXMLDOMNode
	Return Value	XML OCRResult node
OCRDocument	VB Declaration	Public Function OCRDocument(OCRResult As IXMLDOMNode) As String
	Description	Returns the full file name of the document where QuickScan put result of OCR processing of this page.

	Parameters	XML OCRResult node
	Return Value	The path to the document
OCRQuality	VB Declaration	Public Function OCRQuality(OCRResult As IXMLDOMNode) As Long
	Description	Returns the quality of recognition result. 0 = the worst quality 100 = the best quality
	Parameters	XML OCRResult node
	Return Value	OCR quality value

Troubleshooting

If the problem is…	Then try this…
I created an exporter OCX, but it does not appear in the QuickScan list of exporters.	The class, which Implements PixQSI.Exporter, must have the Instancing property set to **MultiUse**. Refer to “Adding a Class module” on page 5.