How to generate report from SQL server VB in Crystal Reports with ByteScout Barcode SDK

How to generate report from SQL server VB in Crystal Reports

Generate report from SQL server VB is easy to implement in Crystal Reports if you use these source codes below. ByteScout Barcode SDK is the robost library (Software Development Kit) that is designed for automatic generation of high-quality barcodes for printing, electronic documents and pdf. All popular barcode types are supported from Code 39 and Code 129 to QR Code, UPC, GS1, GS-128, Datamatrix, PDF417, Maxicode and many others. Provides support for full customization of fonts, colors, output and printing sizes. Special tools are included to verify output quality and printing quality. Can add generated barcode into new or existing documents, images and PDF. It can generate report from SQL server VB in Crystal Reports.

This rich sample source code in Crystal Reports for ByteScout Barcode SDK includes the number of functions and options you should do calling the API to generate report from SQL server VB. In order to implement the functionality, you should copy and paste this code for Crystal Reports below into your code editor with your app, compile and run your application. You can use these Crystal Reports sample examples in one or many applications.

ByteScout free trial version is available for download from our website. It includes all these programming tutorials along with source code samples.

Crystal Reports - CrystalReport1.vb

```
'------------------------------------------------------------------------------
'
'     This code was generated by a tool.
'     Runtime Version:4.0.30319.42000
'
'     Changes to this file may cause incorrect behavior and will be lost if
'     the code is regenerated.
'
'------------------------------------------------------------------------------

Option Strict Off
Option Explicit On

Imports CrystalDecisions.CrystalReports.Engine
Imports CrystalDecisions.ReportSource
Imports CrystalDecisions.Shared
Imports System
Imports System.ComponentModel
```

```
Public Class CrystalReport1
    Inherits ReportClass

    Public Sub New()
        MyBase.New
    End Sub

    Public Overrides Property ResourceName() As String
        Get
            Return "CrystalReport1.rpt"
        End Get
        Set
            'Do nothing
        End Set
    End Property

    Public Overrides Property NewGenerator() As Boolean
        Get
            Return true
        End Get
        Set
            'Do nothing
        End Set
    End Property

    Public Overrides Property FullResourceName() As String
        Get
            Return "ReportFromSqlServer.CrystalReport1.rpt"
        End Get
        Set
            'Do nothing
        End Set
    End Property


      _
    Public ReadOnly Property Section1() As
CrystalDecisions.CrystalReports.Engine.Section
        Get
            Return Me.ReportDefinition.Sections(0)
        End Get
    End Property


      _
    Public ReadOnly Property Section2() As
CrystalDecisions.CrystalReports.Engine.Section
        Get
            Return Me.ReportDefinition.Sections(1)
        End Get
    End Property


      _
    Public ReadOnly Property Section3() As
CrystalDecisions.CrystalReports.Engine.Section
        Get
            Return Me.ReportDefinition.Sections(2)
        End Get
    End Property


      _
    Public ReadOnly Property Section4() As
```

```
CrystalDecisions.CrystalReports.Engine.Section
        Get
            Return Me.ReportDefinition.Sections(3)
        End Get
    End Property


      _
    Public ReadOnly Property Section5() As
CrystalDecisions.CrystalReports.Engine.Section
        Get
            Return Me.ReportDefinition.Sections(4)
        End Get
    End Property
End Class


  _
Public Class CachedCrystalReport1
    Inherits Component
    Implements ICachedReport

    Public Sub New()
        MyBase.New
    End Sub


      _
    Public Overridable Property IsCacheable() As Boolean Implements
CrystalDecisions.ReportSource.ICachedReport.IsCacheable
        Get
            Return true
        End Get
        Set
            '
        End Set
    End Property


      _
    Public Overridable Property ShareDBLogonInfo() As Boolean Implements
CrystalDecisions.ReportSource.ICachedReport.ShareDBLogonInfo
        Get
            Return false
        End Get
        Set
            '
        End Set
    End Property


      _
    Public Overridable Property CacheTimeOut() As System.TimeSpan Implements
CrystalDecisions.ReportSource.ICachedReport.CacheTimeOut
        Get
            Return CachedReportConstants.DEFAULT_TIMEOUT
        End Get
        Set
            '
        End Set
    End Property

    Public Overridable Function CreateReport() As
CrystalDecisions.CrystalReports.Engine.ReportDocument Implements
CrystalDecisions.ReportSource.ICachedReport.CreateReport
```

```
        Dim rpt As CrystalReport1 = New CrystalReport1()
        rpt.Site = Me.Site
        Return rpt
    End Function

    Public Overridable Function GetCustomizedCacheKey(ByVal request As
RequestContext) As String Implements
CrystalDecisions.ReportSource.ICachedReport.GetCustomizedCacheKey
        Dim key As [String] = Nothing
        '// The following is the code used to generate the default
        '// cache key for caching report jobs in the ASP.NET Cache.
        '// Feel free to modify this code to suit your needs.
        '// Returning key == null causes the default cache key to
        '// be generated.
        '
        'key = RequestContext.BuildCompleteCacheKey(
        '    request,
        '    null,       // sReportFilename
        '    this.GetType(),
        '    this.ShareDBLogonInfo );
        Return key
    End Function
End Class
```

Crystal Reports - DataSet1.Designer.vb

```
'------------------------------------------------------------------------------
'
'     This code was generated by a tool.
'     Runtime Version:4.0.30319.42000
'
'     Changes to this file may cause incorrect behavior and will be lost if
'     the code is regenerated.
'
'------------------------------------------------------------------------------

Option Strict Off
Option Explicit On



'''

'''Represents a strongly typed in-memory cache of data.
'''


  _
Partial Public Class DataSet1
    Inherits Global.System.Data.DataSet
```

```
    Private tableProducts As ProductsDataTable

    Private _schemaSerializationMode As Global.System.Data.SchemaSerializationMode =
Global.System.Data.SchemaSerializationMode.IncludeSchema


      _
    Public Sub New()
        MyBase.New
        Me.BeginInit
        Me.InitClass
        Dim schemaChangedHandler As
Global.System.ComponentModel.CollectionChangeEventHandler = AddressOf
Me.SchemaChanged
        AddHandler MyBase.Tables.CollectionChanged, schemaChangedHandler
        AddHandler MyBase.Relations.CollectionChanged, schemaChangedHandler
        Me.EndInit
    End Sub


      _
    Protected Sub New(ByVal info As
Global.System.Runtime.Serialization.SerializationInfo, ByVal context As
Global.System.Runtime.Serialization.StreamingContext)
        MyBase.New(info, context, false)
        If (Me.IsBinarySerialized(info, context) = true) Then
            Me.InitVars(false)
            Dim schemaChangedHandler1 As
Global.System.ComponentModel.CollectionChangeEventHandler = AddressOf
Me.SchemaChanged
            AddHandler Me.Tables.CollectionChanged, schemaChangedHandler1
            AddHandler Me.Relations.CollectionChanged, schemaChangedHandler1
            Return
        End If
        Dim strSchema As String = CType(info.GetValue("XmlSchema",
GetType(String)),String)
        If (Me.DetermineSchemaSerializationMode(info, context) =
Global.System.Data.SchemaSerializationMode.IncludeSchema) Then
            Dim ds As Global.System.Data.DataSet = New Global.System.Data.DataSet()
            ds.ReadXmlSchema(New Global.System.Xml.XmlTextReader(New
Global.System.IO.StringReader(strSchema)))
            If (Not (ds.Tables("Products")) Is Nothing) Then
                MyBase.Tables.Add(New ProductsDataTable(ds.Tables("Products")))
            End If
            Me.DataSetName = ds.DataSetName
            Me.Prefix = ds.Prefix
            Me.Namespace = ds.Namespace
            Me.Locale = ds.Locale
            Me.CaseSensitive = ds.CaseSensitive
            Me.EnforceConstraints = ds.EnforceConstraints
            Me.Merge(ds, false, Global.System.Data.MissingSchemaAction.Add)
            Me.InitVars
        Else
            Me.ReadXmlSchema(New Global.System.Xml.XmlTextReader(New
Global.System.IO.StringReader(strSchema)))
        End If
        Me.GetSerializationData(info, context)
        Dim schemaChangedHandler As
Global.System.ComponentModel.CollectionChangeEventHandler = AddressOf
Me.SchemaChanged
        AddHandler MyBase.Tables.CollectionChanged, schemaChangedHandler
        AddHandler Me.Relations.CollectionChanged, schemaChangedHandler
```

```
    End Sub

      _
    Public ReadOnly Property Products() As ProductsDataTable
        Get
            Return Me.tableProducts
        End Get
    End Property

      _
    Public Overrides Property SchemaSerializationMode() As
Global.System.Data.SchemaSerializationMode
        Get
            Return Me._schemaSerializationMode
        End Get
        Set
            Me._schemaSerializationMode = value
        End Set
    End Property

      _
    Public Shadows ReadOnly Property Tables() As
Global.System.Data.DataTableCollection
        Get
            Return MyBase.Tables
        End Get
    End Property

      _
    Public Shadows ReadOnly Property Relations() As
Global.System.Data.DataRelationCollection
        Get
            Return MyBase.Relations
        End Get
    End Property

      _
    Protected Overrides Sub InitializeDerivedDataSet()
        Me.BeginInit
        Me.InitClass
        Me.EndInit
    End Sub

      _
    Public Overrides Function Clone() As Global.System.Data.DataSet
        Dim cln As DataSet1 = CType(MyBase.Clone,DataSet1)
        cln.InitVars
        cln.SchemaSerializationMode = Me.SchemaSerializationMode
        Return cln
    End Function

      _
    Protected Overrides Function ShouldSerializeTables() As Boolean
        Return false
    End Function

      _
    Protected Overrides Function ShouldSerializeRelations() As Boolean
        Return false
    End Function
```

```
      _
    Protected Overrides Sub ReadXmlSerializable(ByVal reader As
Global.System.Xml.XmlReader)
        If (Me.DetermineSchemaSerializationMode(reader) =
Global.System.Data.SchemaSerializationMode.IncludeSchema) Then
            Me.Reset
            Dim ds As Global.System.Data.DataSet = New Global.System.Data.DataSet()
            ds.ReadXml(reader)
            If (Not (ds.Tables("Products")) Is Nothing) Then
                MyBase.Tables.Add(New ProductsDataTable(ds.Tables("Products")))
            End If
            Me.DataSetName = ds.DataSetName
            Me.Prefix = ds.Prefix
            Me.Namespace = ds.Namespace
            Me.Locale = ds.Locale
            Me.CaseSensitive = ds.CaseSensitive
            Me.EnforceConstraints = ds.EnforceConstraints
            Me.Merge(ds, false, Global.System.Data.MissingSchemaAction.Add)
            Me.InitVars
        Else
            Me.ReadXml(reader)
            Me.InitVars
        End If
    End Sub

      _
    Protected Overrides Function GetSchemaSerializable() As
Global.System.Xml.Schema.XmlSchema
        Dim stream As Global.System.IO.MemoryStream = New
Global.System.IO.MemoryStream()
        Me.WriteXmlSchema(New Global.System.Xml.XmlTextWriter(stream, Nothing))
        stream.Position = 0
        Return Global.System.Xml.Schema.XmlSchema.Read(New
Global.System.Xml.XmlTextReader(stream), Nothing)
    End Function

      _
    Friend Overloads Sub InitVars()
        Me.InitVars(true)
    End Sub

      _
    Friend Overloads Sub InitVars(ByVal initTable As Boolean)
        Me.tableProducts = CType(MyBase.Tables("Products"),ProductsDataTable)
        If (initTable = true) Then
            If (Not (Me.tableProducts) Is Nothing) Then
                Me.tableProducts.InitVars
            End If
        End If
    End Sub

      _
    Private Sub InitClass()
        Me.DataSetName = "DataSet1"
        Me.Prefix = ""
        Me.Namespace = "http://tempuri.org/DataSet1.xsd"
        Me.EnforceConstraints = true
        Me.SchemaSerializationMode =
Global.System.Data.SchemaSerializationMode.IncludeSchema
```

```
        Me.tableProducts = New ProductsDataTable()
        MyBase.Tables.Add(Me.tableProducts)
    End Sub

      _
    Private Function ShouldSerializeProducts() As Boolean
        Return false
    End Function

      _
    Private Sub SchemaChanged(ByVal sender As Object, ByVal e As
Global.System.ComponentModel.CollectionChangeEventArgs)
        If (e.Action = Global.System.ComponentModel.CollectionChangeAction.Remove)
Then
            Me.InitVars
        End If
    End Sub

      _
    Public Shared Function GetTypedDataSetSchema(ByVal xs As
Global.System.Xml.Schema.XmlSchemaSet) As
Global.System.Xml.Schema.XmlSchemaComplexType
        Dim ds As DataSet1 = New DataSet1()
        Dim type As Global.System.Xml.Schema.XmlSchemaComplexType = New
Global.System.Xml.Schema.XmlSchemaComplexType()
        Dim sequence As Global.System.Xml.Schema.XmlSchemaSequence = New
Global.System.Xml.Schema.XmlSchemaSequence()
        Dim any As Global.System.Xml.Schema.XmlSchemaAny = New
Global.System.Xml.Schema.XmlSchemaAny()
        any.Namespace = ds.Namespace
        sequence.Items.Add(any)
        type.Particle = sequence
        Dim dsSchema As Global.System.Xml.Schema.XmlSchema = ds.GetSchemaSerializable
        If xs.Contains(dsSchema.TargetNamespace) Then
            Dim s1 As Global.System.IO.MemoryStream = New
Global.System.IO.MemoryStream()
            Dim s2 As Global.System.IO.MemoryStream = New
Global.System.IO.MemoryStream()
            Try
                Dim schema As Global.System.Xml.Schema.XmlSchema = Nothing
                dsSchema.Write(s1)
                Dim schemas As Global.System.Collections.IEnumerator =
xs.Schemas(dsSchema.TargetNamespace).GetEnumerator
                Do While schemas.MoveNext
                    schema =
CType(schemas.Current,Global.System.Xml.Schema.XmlSchema)
                    s2.SetLength(0)
                    schema.Write(s2)
                    If (s1.Length = s2.Length) Then
                        s1.Position = 0
                        s2.Position = 0

                        Do While ((s1.Position <> s1.Length)  _
                                    AndAlso (s1.ReadByte = s2.ReadByte))


                        Loop
                        If (s1.Position = s1.Length) Then
                            Return type
                        End If
```

```
                End If

            Loop
        Finally
            If (Not (s1) Is Nothing) Then
                s1.Close
            End If
            If (Not (s2) Is Nothing) Then
                s2.Close
            End If
        End Try
    End If
    xs.Add(dsSchema)
    Return type
End Function


  _
Public Delegate Sub ProductsRowChangeEventHandler(ByVal sender As Object, ByVal e
As ProductsRowChangeEvent)

'''

'''Represents the strongly named DataTable class.
'''


  _
Partial Public Class ProductsDataTable
    Inherits Global.System.Data.DataTable
    Implements Global.System.Collections.IEnumerable

    Private columnProduct_ID As Global.System.Data.DataColumn

    Private columnProduct_Name As Global.System.Data.DataColumn

    Private columnProduct_Description As Global.System.Data.DataColumn

    Private columnBarCode As Global.System.Data.DataColumn


      _
    Public Sub New()
        MyBase.New
        Me.TableName = "Products"
        Me.BeginInit
        Me.InitClass
        Me.EndInit
    End Sub


      _
    Friend Sub New(ByVal table As Global.System.Data.DataTable)
        MyBase.New
        Me.TableName = table.TableName
        If (table.CaseSensitive <> table.DataSet.CaseSensitive) Then
            Me.CaseSensitive = table.CaseSensitive
        End If
        If (table.Locale.ToString <> table.DataSet.Locale.ToString) Then
            Me.Locale = table.Locale
        End If
        If (table.Namespace <> table.DataSet.Namespace) Then
            Me.Namespace = table.Namespace
        End If
```

```
            Me.Prefix = table.Prefix
            Me.MinimumCapacity = table.MinimumCapacity
        End Sub

        _
        Protected Sub New(ByVal info As
Global.System.Runtime.Serialization.SerializationInfo, ByVal context As
Global.System.Runtime.Serialization.StreamingContext)
            MyBase.New(info, context)
            Me.InitVars
        End Sub

        _
        Public ReadOnly Property Product_IDColumn() As Global.System.Data.DataColumn
            Get
                Return Me.columnProduct_ID
            End Get
        End Property

        _
        Public ReadOnly Property Product_NameColumn() As
Global.System.Data.DataColumn
            Get
                Return Me.columnProduct_Name
            End Get
        End Property

        _
        Public ReadOnly Property Product_DescriptionColumn() As
Global.System.Data.DataColumn
            Get
                Return Me.columnProduct_Description
            End Get
        End Property

        _
        Public ReadOnly Property BarCodeColumn() As Global.System.Data.DataColumn
            Get
                Return Me.columnBarCode
            End Get
        End Property

        _
        Public ReadOnly Property Count() As Integer
            Get
                Return Me.Rows.Count
            End Get
        End Property

        _
        Public Default ReadOnly Property Item(ByVal index As Integer) As ProductsRow
            Get
                Return CType(Me.Rows(index),ProductsRow)
            End Get
        End Property

        _
        Public Event ProductsRowChanging As ProductsRowChangeEventHandler

          _
```

```
        Public Event ProductsRowChanged As ProductsRowChangeEventHandler

        _
        Public Event ProductsRowDeleting As ProductsRowChangeEventHandler

        _
        Public Event ProductsRowDeleted As ProductsRowChangeEventHandler

        _
        Public Overloads Sub AddProductsRow(ByVal row As ProductsRow)
            Me.Rows.Add(row)
        End Sub

        _
        Public Overloads Function AddProductsRow(ByVal Product_ID As Integer, ByVal
Product_Name As String, ByVal Product_Description As String, ByVal BarCode() As Byte)
As ProductsRow
            Dim rowProductsRow As ProductsRow = CType(Me.NewRow,ProductsRow)
            Dim columnValuesArray() As Object = New Object() {Product_ID,
Product_Name, Product_Description, BarCode}
            rowProductsRow.ItemArray = columnValuesArray
            Me.Rows.Add(rowProductsRow)
            Return rowProductsRow
        End Function

        _
        Public Overridable Function GetEnumerator() As
Global.System.Collections.IEnumerator Implements
Global.System.Collections.IEnumerable.GetEnumerator
            Return Me.Rows.GetEnumerator
        End Function

        _
        Public Overrides Function Clone() As Global.System.Data.DataTable
            Dim cln As ProductsDataTable = CType(MyBase.Clone,ProductsDataTable)
            cln.InitVars
            Return cln
        End Function

        _
        Protected Overrides Function CreateInstance() As Global.System.Data.DataTable
            Return New ProductsDataTable()
        End Function

        _
        Friend Sub InitVars()
            Me.columnProduct_ID = MyBase.Columns("Product ID")
            Me.columnProduct_Name = MyBase.Columns("Product Name")
            Me.columnProduct_Description = MyBase.Columns("Product Description")
            Me.columnBarCode = MyBase.Columns("BarCode")
        End Sub

        _
        Private Sub InitClass()
            Me.columnProduct_ID = New Global.System.Data.DataColumn("Product ID",
GetType(Integer), Nothing, Global.System.Data.MappingType.Element)
            MyBase.Columns.Add(Me.columnProduct_ID)
            Me.columnProduct_Name = New Global.System.Data.DataColumn("Product Name",
GetType(String), Nothing, Global.System.Data.MappingType.Element)
            MyBase.Columns.Add(Me.columnProduct_Name)
```

```
            Me.columnProduct_Description = New Global.System.Data.DataColumn("Product
Description", GetType(String), Nothing, Global.System.Data.MappingType.Element)
            MyBase.Columns.Add(Me.columnProduct_Description)
            Me.columnBarCode = New Global.System.Data.DataColumn("BarCode",
GetType(Byte()), Nothing, Global.System.Data.MappingType.Element)
            MyBase.Columns.Add(Me.columnBarCode)
            Me.columnProduct_Name.MaxLength = 100
            Me.columnProduct_Description.MaxLength = 255
        End Sub


        _
        Public Function NewProductsRow() As ProductsRow
            Return CType(Me.NewRow,ProductsRow)
        End Function


        _
        Protected Overrides Function NewRowFromBuilder(ByVal builder As
Global.System.Data.DataRowBuilder) As Global.System.Data.DataRow
            Return New ProductsRow(builder)
        End Function


        _
        Protected Overrides Function GetRowType() As Global.System.Type
            Return GetType(ProductsRow)
        End Function


        _
        Protected Overrides Sub OnRowChanged(ByVal e As
Global.System.Data.DataRowChangeEventArgs)
            MyBase.OnRowChanged(e)
            If (Not (Me.ProductsRowChangedEvent) Is Nothing) Then
                RaiseEvent ProductsRowChanged(Me, New
ProductsRowChangeEvent(CType(e.Row,ProductsRow), e.Action))
            End If
        End Sub


        _
        Protected Overrides Sub OnRowChanging(ByVal e As
Global.System.Data.DataRowChangeEventArgs)
            MyBase.OnRowChanging(e)
            If (Not (Me.ProductsRowChangingEvent) Is Nothing) Then
                RaiseEvent ProductsRowChanging(Me, New
ProductsRowChangeEvent(CType(e.Row,ProductsRow), e.Action))
            End If
        End Sub


        _
        Protected Overrides Sub OnRowDeleted(ByVal e As
Global.System.Data.DataRowChangeEventArgs)
            MyBase.OnRowDeleted(e)
            If (Not (Me.ProductsRowDeletedEvent) Is Nothing) Then
                RaiseEvent ProductsRowDeleted(Me, New
ProductsRowChangeEvent(CType(e.Row,ProductsRow), e.Action))
            End If
        End Sub


        _
        Protected Overrides Sub OnRowDeleting(ByVal e As
Global.System.Data.DataRowChangeEventArgs)
            MyBase.OnRowDeleting(e)
```

```
            If (Not (Me.ProductsRowDeletingEvent) Is Nothing) Then
                RaiseEvent ProductsRowDeleting(Me, New
ProductsRowChangeEvent(CType(e.Row,ProductsRow), e.Action))
            End If
        End Sub

          _
        Public Sub RemoveProductsRow(ByVal row As ProductsRow)
            Me.Rows.Remove(row)
        End Sub

          _
        Public Shared Function GetTypedTableSchema(ByVal xs As
Global.System.Xml.Schema.XmlSchemaSet) As
Global.System.Xml.Schema.XmlSchemaComplexType
            Dim type As Global.System.Xml.Schema.XmlSchemaComplexType = New
Global.System.Xml.Schema.XmlSchemaComplexType()
            Dim sequence As Global.System.Xml.Schema.XmlSchemaSequence = New
Global.System.Xml.Schema.XmlSchemaSequence()
            Dim ds As DataSet1 = New DataSet1()
            Dim any1 As Global.System.Xml.Schema.XmlSchemaAny = New
Global.System.Xml.Schema.XmlSchemaAny()
            any1.Namespace = "http://www.w3.org/2001/XMLSchema"
            any1.MinOccurs = New Decimal(0)
            any1.MaxOccurs = Decimal.MaxValue
            any1.ProcessContents =
Global.System.Xml.Schema.XmlSchemaContentProcessing.Lax
            sequence.Items.Add(any1)
            Dim any2 As Global.System.Xml.Schema.XmlSchemaAny = New
Global.System.Xml.Schema.XmlSchemaAny()
            any2.Namespace = "urn:schemas-microsoft-com:xml-diffgram-v1"
            any2.MinOccurs = New Decimal(1)
            any2.ProcessContents =
Global.System.Xml.Schema.XmlSchemaContentProcessing.Lax
            sequence.Items.Add(any2)
            Dim attribute1 As Global.System.Xml.Schema.XmlSchemaAttribute = New
Global.System.Xml.Schema.XmlSchemaAttribute()
            attribute1.Name = "namespace"
            attribute1.FixedValue = ds.Namespace
            type.Attributes.Add(attribute1)
            Dim attribute2 As Global.System.Xml.Schema.XmlSchemaAttribute = New
Global.System.Xml.Schema.XmlSchemaAttribute()
            attribute2.Name = "tableTypeName"
            attribute2.FixedValue = "ProductsDataTable"
            type.Attributes.Add(attribute2)
            type.Particle = sequence
            Dim dsSchema As Global.System.Xml.Schema.XmlSchema =
ds.GetSchemaSerializable
            If xs.Contains(dsSchema.TargetNamespace) Then
                Dim s1 As Global.System.IO.MemoryStream = New
Global.System.IO.MemoryStream()
                Dim s2 As Global.System.IO.MemoryStream = New
Global.System.IO.MemoryStream()
                Try
                    Dim schema As Global.System.Xml.Schema.XmlSchema = Nothing
                    dsSchema.Write(s1)
                    Dim schemas As Global.System.Collections.IEnumerator =
xs.Schemas(dsSchema.TargetNamespace).GetEnumerator
                    Do While schemas.MoveNext
                        schema =
```

```
CType(schemas.Current,Global.System.Xml.Schema.XmlSchema)
                        s2.SetLength(0)
                        schema.Write(s2)
                        If (s1.Length = s2.Length) Then
                            s1.Position = 0
                            s2.Position = 0

                            Do While ((s1.Position <> s1.Length)  _
                                        AndAlso (s1.ReadByte = s2.ReadByte))


                            Loop
                            If (s1.Position = s1.Length) Then
                                Return type
                            End If
                        End If

                    Loop
                Finally
                    If (Not (s1) Is Nothing) Then
                        s1.Close
                    End If
                    If (Not (s2) Is Nothing) Then
                        s2.Close
                    End If
                End Try
            End If
            xs.Add(dsSchema)
            Return type
        End Function
    End Class

    '''

    '''Represents strongly named DataRow class.
    '''

    Partial Public Class ProductsRow
        Inherits Global.System.Data.DataRow

        Private tableProducts As ProductsDataTable


          _
        Friend Sub New(ByVal rb As Global.System.Data.DataRowBuilder)
            MyBase.New(rb)
            Me.tableProducts = CType(Me.Table,ProductsDataTable)
        End Sub


          _
        Public Property Product_ID() As Integer
            Get
                Try
                    Return CType(Me(Me.tableProducts.Product_IDColumn),Integer)
                Catch e As Global.System.InvalidCastException
                    Throw New Global.System.Data.StrongTypingException("The value for
column 'Product ID' in table 'Products' is DBNull.", e)
                End Try
            End Get
            Set
                Me(Me.tableProducts.Product_IDColumn) = value
```

```
            End Set
        End Property

          _
        Public Property Product_Name() As String
            Get
                Try
                    Return CType(Me(Me.tableProducts.Product_NameColumn),String)
                Catch e As Global.System.InvalidCastException
                    Throw New Global.System.Data.StrongTypingException("The value for
column 'Product Name' in table 'Products' is DBNull.", e)
                End Try
            End Get
            Set
                Me(Me.tableProducts.Product_NameColumn) = value
            End Set
        End Property

          _
        Public Property Product_Description() As String
            Get
                Try
                    Return
CType(Me(Me.tableProducts.Product_DescriptionColumn),String)
                Catch e As Global.System.InvalidCastException
                    Throw New Global.System.Data.StrongTypingException("The value for
column 'Product Description' in table 'Products' is DBNull.", e)
                End Try
            End Get
            Set
                Me(Me.tableProducts.Product_DescriptionColumn) = value
            End Set
        End Property

          _
        Public Property BarCode() As Byte()
            Get
                Try
                    Return CType(Me(Me.tableProducts.BarCodeColumn),Byte())
                Catch e As Global.System.InvalidCastException
                    Throw New Global.System.Data.StrongTypingException("The value for
column 'BarCode' in table 'Products' is DBNull.", e)
                End Try
            End Get
            Set
                Me(Me.tableProducts.BarCodeColumn) = value
            End Set
        End Property

          _
        Public Function IsProduct_IDNull() As Boolean
            Return Me.IsNull(Me.tableProducts.Product_IDColumn)
        End Function

          _
        Public Sub SetProduct_IDNull()
            Me(Me.tableProducts.Product_IDColumn) = Global.System.Convert.DBNull
        End Sub

          _
```

```
        Public Function IsProduct_NameNull() As Boolean
            Return Me.IsNull(Me.tableProducts.Product_NameColumn)
        End Function

          _
        Public Sub SetProduct_NameNull()
            Me(Me.tableProducts.Product_NameColumn) = Global.System.Convert.DBNull
        End Sub

          _
        Public Function IsProduct_DescriptionNull() As Boolean
            Return Me.IsNull(Me.tableProducts.Product_DescriptionColumn)
        End Function

          _
        Public Sub SetProduct_DescriptionNull()
            Me(Me.tableProducts.Product_DescriptionColumn) =
Global.System.Convert.DBNull
        End Sub

          _
        Public Function IsBarCodeNull() As Boolean
            Return Me.IsNull(Me.tableProducts.BarCodeColumn)
        End Function

          _
        Public Sub SetBarCodeNull()
            Me(Me.tableProducts.BarCodeColumn) = Global.System.Convert.DBNull
        End Sub
    End Class

    '''

    '''Row event argument class
    '''


      _
    Public Class ProductsRowChangeEvent
        Inherits Global.System.EventArgs

        Private eventRow As ProductsRow

        Private eventAction As Global.System.Data.DataRowAction


          _
        Public Sub New(ByVal row As ProductsRow, ByVal action As
Global.System.Data.DataRowAction)
            MyBase.New
            Me.eventRow = row
            Me.eventAction = action
        End Sub


          _
        Public ReadOnly Property Row() As ProductsRow
            Get
                Return Me.eventRow
            End Get
        End Property


          _
```

```
        Public ReadOnly Property Action() As Global.System.Data.DataRowAction
            Get
                Return Me.eventAction
            End Get
        End Property
    End Class
End Class

Namespace DataSet1TableAdapters

    '''

    '''Represents the connection and commands used to retrieve and save data.
    '''


      _
    Partial Public Class ProductsTableAdapter
        Inherits Global.System.ComponentModel.Component

        Private WithEvents _adapter As Global.System.Data.SqlClient.SqlDataAdapter

        Private _connection As Global.System.Data.SqlClient.SqlConnection

        Private _commandCollection() As Global.System.Data.SqlClient.SqlCommand

        Private _clearBeforeFill As Boolean


          _
        Public Sub New()
            MyBase.New
            Me.ClearBeforeFill = true
        End Sub


          _
        Private ReadOnly Property Adapter() As
Global.System.Data.SqlClient.SqlDataAdapter
            Get
                If (Me._adapter Is Nothing) Then
                    Me.InitAdapter
                End If
                Return Me._adapter
            End Get
        End Property


          _
        Friend Property Connection() As Global.System.Data.SqlClient.SqlConnection
            Get
                If (Me._connection Is Nothing) Then
                    Me.InitConnection
                End If
                Return Me._connection
            End Get
            Set
                Me._connection = value
                If (Not (Me.Adapter.InsertCommand) Is Nothing) Then
                    Me.Adapter.InsertCommand.Connection = value
                End If
                If (Not (Me.Adapter.DeleteCommand) Is Nothing) Then
                    Me.Adapter.DeleteCommand.Connection = value
                End If
```

```
                If (Not (Me.Adapter.UpdateCommand) Is Nothing) Then
                    Me.Adapter.UpdateCommand.Connection = value
                End If
                Dim i As Integer = 0
                Do While (i < Me.CommandCollection.Length)
                    If (Not (Me.CommandCollection(i)) Is Nothing) Then

CType(Me.CommandCollection(i),Global.System.Data.SqlClient.SqlCommand).Connection =
value
                    End If
                    i = (i + 1)
                Loop
            End Set
        End Property


          _
        Protected ReadOnly Property CommandCollection() As
Global.System.Data.SqlClient.SqlCommand()
            Get
                If (Me._commandCollection Is Nothing) Then
                    Me.InitCommandCollection
                End If
                Return Me._commandCollection
            End Get
        End Property


          _
        Public Property ClearBeforeFill() As Boolean
            Get
                Return Me._clearBeforeFill
            End Get
            Set
                Me._clearBeforeFill = value
            End Set
        End Property


          _
        Private Sub InitAdapter()
            Me._adapter = New Global.System.Data.SqlClient.SqlDataAdapter()
            Dim tableMapping As Global.System.Data.Common.DataTableMapping = New
Global.System.Data.Common.DataTableMapping()
            tableMapping.SourceTable = "Table"
            tableMapping.DataSetTable = "Products"
            tableMapping.ColumnMappings.Add("Product ID", "Product ID")
            tableMapping.ColumnMappings.Add("Product Name", "Product Name")
            tableMapping.ColumnMappings.Add("Product Description", "Product
Description")
            Me._adapter.TableMappings.Add(tableMapping)
            Me._adapter.InsertCommand = New Global.System.Data.SqlClient.SqlCommand()
            Me._adapter.InsertCommand.Connection = Me.Connection
            Me._adapter.InsertCommand.CommandText = "INSERT INTO [dbo].[Products]
([Product ID], [Product Name], [Product Description]"& _
                ") VALUES (@Product_ID, @Product_Name, @Product_Description)"
            Me._adapter.InsertCommand.CommandType =
Global.System.Data.CommandType.Text
            Me._adapter.InsertCommand.Parameters.Add(New
Global.System.Data.SqlClient.SqlParameter("@Product_ID",
Global.System.Data.SqlDbType.Int, 0, Global.System.Data.ParameterDirection.Input, 0,
0, "Product ID", Global.System.Data.DataRowVersion.Current, false, Nothing, "", "",
""))
```

```
            Me._adapter.InsertCommand.Parameters.Add(New
Global.System.Data.SqlClient.SqlParameter("@Product_Name",
Global.System.Data.SqlDbType.NVarChar, 0,
Global.System.Data.ParameterDirection.Input, 0, 0, "Product Name",
Global.System.Data.DataRowVersion.Current, false, Nothing, "", "", ""))
            Me._adapter.InsertCommand.Parameters.Add(New
Global.System.Data.SqlClient.SqlParameter("@Product_Description",
Global.System.Data.SqlDbType.NVarChar, 0,
Global.System.Data.ParameterDirection.Input, 0, 0, "Product Description",
Global.System.Data.DataRowVersion.Current, false, Nothing, "", "", ""))
        End Sub


          _
        Private Sub InitConnection()
            Me._connection = New Global.System.Data.SqlClient.SqlConnection()
            Me._connection.ConnectionString =
Global.ReportFromSqlServer.Settings.Default.example_dbConnectionString
        End Sub


          _
        Private Sub InitCommandCollection()
            Me._commandCollection = New Global.System.Data.SqlClient.SqlCommand(0) {}
            Me._commandCollection(0) = New Global.System.Data.SqlClient.SqlCommand()
            Me._commandCollection(0).Connection = Me.Connection
            Me._commandCollection(0).CommandText = "SELECT [Product ID], [Product
Name], [Product Description] FROM dbo.Products"
            Me._commandCollection(0).CommandType =
Global.System.Data.CommandType.Text
        End Sub


          _
        Public Overloads Overridable Function Fill(ByVal dataTable As
DataSet1.ProductsDataTable) As Integer
            Me.Adapter.SelectCommand = Me.CommandCollection(0)
            If (Me.ClearBeforeFill = true) Then
                dataTable.Clear
            End If
            Dim returnValue As Integer = Me.Adapter.Fill(dataTable)
            Return returnValue
        End Function


          _
        Public Overloads Overridable Function GetData() As DataSet1.ProductsDataTable
            Me.Adapter.SelectCommand = Me.CommandCollection(0)
            Dim dataTable As DataSet1.ProductsDataTable = New
DataSet1.ProductsDataTable()
            Me.Adapter.Fill(dataTable)
            Return dataTable
        End Function


          _
        Public Overloads Overridable Function Update(ByVal dataTable As
DataSet1.ProductsDataTable) As Integer
            Return Me.Adapter.Update(dataTable)
        End Function


          _
        Public Overloads Overridable Function Update(ByVal dataSet As DataSet1) As
Integer
            Return Me.Adapter.Update(dataSet, "Products")
```

```
        End Function

          _
        Public Overloads Overridable Function Update(ByVal dataRow As
Global.System.Data.DataRow) As Integer
            Return Me.Adapter.Update(New Global.System.Data.DataRow() {dataRow})
        End Function

          _
        Public Overloads Overridable Function Update(ByVal dataRows() As
Global.System.Data.DataRow) As Integer
            Return Me.Adapter.Update(dataRows)
        End Function

          _
        Public Overloads Overridable Function Insert(ByVal Product_ID As
Global.System.Nullable(Of Integer), ByVal Product_Name As String, ByVal
Product_Description As String) As Integer
            If (Product_ID.HasValue = true) Then
                Me.Adapter.InsertCommand.Parameters(0).Value =
CType(Product_ID.Value,Integer)
            Else
                Me.Adapter.InsertCommand.Parameters(0).Value =
Global.System.DBNull.Value
            End If
            If (Product_Name Is Nothing) Then
                Me.Adapter.InsertCommand.Parameters(1).Value =
Global.System.DBNull.Value
            Else
                Me.Adapter.InsertCommand.Parameters(1).Value =
CType(Product_Name,String)
            End If
            If (Product_Description Is Nothing) Then
                Me.Adapter.InsertCommand.Parameters(2).Value =
Global.System.DBNull.Value
            Else
                Me.Adapter.InsertCommand.Parameters(2).Value =
CType(Product_Description,String)
            End If
            Dim previousConnectionState As Global.System.Data.ConnectionState =
Me.Adapter.InsertCommand.Connection.State
            If ((Me.Adapter.InsertCommand.Connection.State And
Global.System.Data.ConnectionState.Open)  _
                        <> Global.System.Data.ConnectionState.Open) Then
                Me.Adapter.InsertCommand.Connection.Open
            End If
            Try
                Dim returnValue As Integer = Me.Adapter.InsertCommand.ExecuteNonQuery
                Return returnValue
            Finally
                If (previousConnectionState =
Global.System.Data.ConnectionState.Closed) Then
                    Me.Adapter.InsertCommand.Connection.Close
                End If
            End Try
        End Function
    End Class
End Namespace
```

Crystal Reports - Form1.Designer.vb

```
Partial Class Form1
        Private components As System.ComponentModel.IContainer = Nothing

        Protected Overloads Overrides Sub Dispose(disposing As Boolean)
                If disposing AndAlso (components IsNot Nothing) Then
                        components.Dispose()
                End If
                MyBase.Dispose(disposing)
        End Sub

        #Region "Windows Form Designer generated code"

        Private Sub InitializeComponent()
                Me.crystalReportViewer1 = New
CrystalDecisions.Windows.Forms.CrystalReportViewer()
                Me.CrystalReport11 = New ReportFromSqlServer.CrystalReport1()
                Me.SuspendLayout()
                '
                ' crystalReportViewer1
                '
                Me.crystalReportViewer1.ActiveViewIndex = 0
                Me.crystalReportViewer1.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle
                Me.crystalReportViewer1.Dock = System.Windows.Forms.DockStyle.Fill
                Me.crystalReportViewer1.Location = New System.Drawing.Point(0, 0)
                Me.crystalReportViewer1.Name = "crystalReportViewer1"
                Me.crystalReportViewer1.ReportSource = Me.CrystalReport11
                Me.crystalReportViewer1.Size = New System.Drawing.Size(799, 566)
                Me.crystalReportViewer1.TabIndex = 0
                '
                ' Form1
                '
                Me.AutoScaleDimensions = New System.Drawing.SizeF(6F, 13F)
                Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
                Me.ClientSize = New System.Drawing.Size(799, 566)
                Me.Controls.Add(Me.crystalReportViewer1)
                Me.Name = "Form1"
                Me.Text = "Form1"
                Me.ResumeLayout(False)

        End Sub

        #End Region

        Private crystalReportViewer1 As
CrystalDecisions.Windows.Forms.CrystalReportViewer
        Private CrystalReport11 As CrystalReport1
```

```
End Class
```

Crystal Reports - Form1.vb

```
Imports System.Data
Imports System.Diagnostics
Imports System.Windows.Forms
Imports System.Data.SqlClient
Imports Bytescout.BarCode

Public Partial Class Form1
        Inherits Form
        Public Sub New()
                InitializeComponent()

                Try
                        ' MODIFY THE CONNECTION STRING WITH YOUR SERVER CONNECTION
INFO!!!
                        Const  connectionString As String = "Data
Source=localhost\SQLEXPRESS;Initial Catalog=master;Integrated Security=true;"

                        Using connection As New SqlConnection(connectionString)
                                connection.Open()

                                ' Create a database for demonstration purposes

'''////////////////////////////////////////////////////////////////////////////


                                Dim o As [Object] = ExecuteQueryScalar(connection,
"SELECT DB_ID('example_db')")

                                ' if 'example_db' does not exist, create it
                                If o Is Nothing OrElse TypeOf o Is DBNull Then
                                        ' Create empty database
                                        ExecuteQueryWithoutResult(connection, "CREATE
DATABASE example_db")
                                        ' Switch to created database
                                        ExecuteQueryWithoutResult(connection, "USE
example_db")
                                        ' Create a table
                                        ExecuteQueryWithoutResult(connection, "CREATE
TABLE Products ([Product ID] int, [Product Name] nvarchar(100), [Product Description]
nvarchar(255))")
                                        ' Fill the table with data
                                        ExecuteQueryWithoutResult(connection, "INSERT
Products VALUES(1, 'Spreadsheet Tools', 'Convert XLS, XLSX, CSV, ODS spreadsheet into
HTML, PDF, XLS, XLSX, CSV formats WITHOUT EXCEL installed')")
                                        ExecuteQueryWithoutResult(connection, "INSERT
```

```
Products VALUES(2, 'Watermarking PRO', 'Professional tool to protect images: multiple
watermarks, custom position for watermarks, image effects, EXIF and IPTC macros for
text and more')")
                                        ExecuteQueryWithoutResult(connection, "INSERT
Products VALUES(3, 'Watermarking', 'Protect copyrights for your images with
professional looking watermarks with this easy to use tool')")
                                        ExecuteQueryWithoutResult(connection, "INSERT
Products VALUES(4, 'PPT To Video Scout', 'converts PowerPoint presentations (PPT,
PPTX) into AVI,MPEG,WMV, FLV (flash video) video movies with sound')")
                                End If


                                ' Create a datataset from query.
                                ' Query result columns must conform to field names we
used in the report designer

                                Dim dataAdapter As New SqlDataAdapter("SELECT
[Product ID], [Product Name], [Product Description] FROM example_db.dbo.Products",
connection)

                                ' fill dataset
                                Dim dataSet As New DataSet()
                                dataAdapter.Fill(dataSet)

                                ' don't forget to close the connection
                                connection.Close()

                                ' add virtual column into the result table
                                dataSet.Tables(0).Columns.Add(New
DataColumn("BarCode", GetType(Byte())))

                                ' create barcode object
                                Dim bc As New Barcode(SymbologyType.Code39)
                                bc.DrawCaption = False

                                For Each row As DataRow In dataSet.Tables(0).Rows
                                        ' set barcode value
                                        bc.Value = (Convert.ToString(row("Product
ID")))

                                        ' retrieve generated image bytes
                                        Dim barcodeBytes As Byte() =
bc.GetImageBytesWMF()

                                        ' fill virtual column with generated image
bytes
                                        row("BarCode") = barcodeBytes
                                Next

                                ' set report datasource
                                CrystalReport11.SetDataSource(dataSet.Tables(0))
                        End Using
                Catch ex As Exception
                        Trace.WriteLine("Error: " & ex.Message)
                End Try
        End Sub

        Private Shared Sub ExecuteQueryWithoutResult(connection As SqlConnection,
query As String)
                Using command As New SqlCommand(query, connection)
```

```
                        command.ExecuteNonQuery()
                End Using
        End Sub

        Private Shared Function ExecuteQueryScalar(connection As SqlConnection, query
As String) As Object
                Using command As New SqlCommand(query, connection)
                        Return command.ExecuteScalar()
                End Using
        End Function
End Class
```

Crystal Reports - Program.vb

```
Imports System.Collections.Generic
Imports System.Windows.Forms

NotInheritable Class Program
        Private Sub New()
        End Sub
         _
        Friend Shared Sub Main()
                Application.EnableVisualStyles()
                Application.Run(New Form1())
        End Sub
End Class
```

Crystal Reports - packages.config

```
```

FOR MORE INFORMATION AND FREE TRIAL:

[Download Free Trial SDK (on-premise version)]

[Read more about ByteScout Barcode SDK]

[Explore documentation]

[Visit www.ByteScout.com]

or

[Get Your Free API Key for www.PDF.co Web API]