```
'Class module
Sub Class_Globals
	Private mTarget As Object
	Type DBResult (Tag As Object, Columns As Map, Rows As List)
	Type DBCommand (Name As String, Parameters() As Object)
	Private link As String
	Private bc As ByteConverter
	Private T_NULL = 0, T_STRING = 1, T_SHORT = 2, T_INT = 3, T_LONG = 4, T_FLOAT = 5 _
		,T_DOUBLE = 6, T_BOOLEAN = 7, T_BLOB = 8 As Byte
	Private VERSION As Float = 0.9
	Private tempArray(1) As Object
End Sub

'Target - The module that handles JobDone (usually Me).
'ConnectorLink - URL of the Java server.
Public Sub Initialize (Target As Object, ConnectorLink As String)
	mTarget = Target
	link = ConnectorLink
End Sub

'Sends a query request.
'Command - Query name and parameters.
'Limit - Maximum rows to return or 0 for no limit.
'Tag - An object that will be returned in the result.
Public Sub ExecuteQuery(Command As DBCommand, Limit As Int, Tag As Object)
	Dim j As HttpJob
	Dim ms As OutputStream
	Dim out2 As OutputStream = StartJob(j,ms, Tag)

	WriteObject(Command.Name, out2)
	WriteInt(Limit, out2)
	WriteList(Command.Parameters, out2)
	out2.Close
	j.PostBytes(link & "?method=query", ms.ToBytesArray)
End Sub
Public Sub ExecuteBatch(ListOfCommands As List, Tag As Object)
	Dim j As HttpJob
	Dim ms As OutputStream
	Dim out2 As OutputStream = StartJob(j,ms, Tag)
	WriteInt(ListOfCommands.Size, out2)
	For Each Command As DBCommand In ListOfCommands
		WriteObject(Command.Name, out2)
		WriteList(Command.Parameters, out2)
	Next
	out2.Close
	j.PostBytes(link & "?method=batch", ms.ToBytesArray)
End Sub

'Similar to ExecuteBatch. Sends a single command.
Public Sub ExecuteCommand(Command As DBCommand, Tag As Object)
	ExecuteBatch(Array As DBCommand(Command), Tag)
End Sub

Private Sub StartJob(j As HttpJob, MemoryStream As OutputStream, Tag As Object) As OutputStream
	j.Initialize("DBRequest", mTarget)
	j.Tag = Tag
	MemoryStream.InitializeToBytesArray(0)
	Dim compress As CompressedStreams
	Dim out As OutputStream = compress.WrapOutputStream(MemoryStream, "gzip")
	WriteObject(VERSION, out)
	Return out
End Sub

Private Sub WriteList(Parameters As List, out As OutputStream)
	Dim data() As Byte
	If Parameters = Null OR Parameters.IsInitialized = False Then
		Dim Parameters As List
		Parameters.Initialize
	End If
	data = bc.IntsToBytes(Array As Int(Parameters.Size))
	out.WriteBytes(data, 0, data.Length)
	For Each o As Object In Parameters
		WriteObject(o, out)
	Next
End Sub

Private Sub WriteObject(o As Object, out As OutputStream)
	Dim data() As Byte
	tempArray(0) = o
	If tempArray(0) = Null Then
		out.WriteBytes(Array As Byte(T_NULL), 0, 1)
	Else If tempArray(0) Is Short Then
		out.WriteBytes(Array As Byte(T_SHORT), 0, 1)
		data = bc.ShortsToBytes(Array As Short(o))
	Else If tempArray(0) Is Int Then
```

```
                out.WriteBytes(Array As Byte(T_INT), 0, 1)
                data = bc.IntsToBytes(Array As Int(o))
        Else If tempArray(0) Is Float Then
                out.WriteBytes(Array As Byte(T_FLOAT), 0, 1)
                data = bc.FloatsToBytes(Array As Float(o))
        Else If tempArray(0) Is Double Then
                out.WriteBytes(Array As Byte(T_DOUBLE), 0, 1)
                data = bc.DoublesToBytes(Array As Double(o))
        Else If tempArray(0) Is Long Then
                out.WriteBytes(Array As Byte(T_LONG), 0, 1)
                data = bc.LongsToBytes(Array As Long(o))
        Else If tempArray(0) Is Boolean Then
                out.WriteBytes(Array As Byte(T_BOOLEAN), 0, 1)
                Dim b As Boolean = 0
                Dim data(1) As Byte
                If b Then data(0) = 1 Else data(0) = 0
        Else If GetType(tempArray(0)) = "[B" Then
                data = o
                out.WriteBytes(Array As Byte(T_BLOB), 0, 1)
                WriteInt(data.Length, out)
        Else 'If o Is String Then (treat all other values as string)
                out.WriteBytes(Array As Byte(T_STRING), 0, 1)
                data = bc.StringToBytes(o, "UTF8")
                WriteInt(data.Length, out)
        End If
        If data.Length > 0 Then out.WriteBytes(data, 0, data.Length)
End Sub

Private Sub ReadObject(In As InputStream) As Object
        Dim data(1) As Byte
        In.ReadBytes(data, 0, 1)
        Select data(0)
                Case T_NULL
                        Return Null
                Case T_SHORT
                        Dim data(2) As Byte
                        Return bc.ShortsFromBytes(ReadBytesFully(In, data, data.Length))(0)
                Case T_INT
                        Dim data(4) As Byte
                        Return bc.IntsFromBytes(ReadBytesFully(In, data, data.Length))(0)
                Case T_LONG
                        Dim data(8) As Byte
                        Return bc.LongsFromBytes(ReadBytesFully(In, data, data.Length))(0)
                Case T_FLOAT
                        Dim data(4) As Byte
                        Return bc.FloatsFromBytes(ReadBytesFully(In, data, data.Length))(0)
                Case T_DOUBLE
                        Dim data(8) As Byte
                        Return bc.DoublesFromBytes(ReadBytesFully(In, data, data.Length))(0)
                Case T_BOOLEAN
                        Dim b As Byte = ReadByte(In)
                        Return b = 1
                Case T_BLOB
                        Dim len As Int = ReadInt(In)
                        Dim data(len) As Byte
                        Return ReadBytesFully(In, data, data.Length)
                Case Else
                        Dim len As Int = ReadInt(In)
                        Dim data(len) As Byte
                        ReadBytesFully(In, data, data.Length)
                        Return BytesToString(data, 0, data.Length, "UTF8")
        End Select
End Sub

Private Sub ReadBytesFully(In As InputStream, Data() As Byte, Len As Int) As Byte()
        Dim count = 0, read As Int
        Do While count < Len AND read > -1
                read = In.ReadBytes(Data, count, Len - count)
                count = count + read
        Loop
        Return Data
End Sub

Private Sub WriteInt(i As Int, out As OutputStream)
        Dim data() As Byte
        data = bc.IntsToBytes(Array As Int(i))
        out.WriteBytes(data, 0, data.Length)
End Sub

Private Sub ReadInt(In As InputStream) As Int
        Dim data(4) As Byte
        Return bc.IntsFromBytes(ReadBytesFully(In, data, data.Length))(0)
End Sub

Private Sub ReadByte(In As InputStream) As Byte
```

```
        Dim data(1) As Byte
        In.ReadBytes(data, 0, 1)
        Return data(0)
End Sub

'Handles the Job result and returns a DBResult.
Public Sub HandleJob(Job As HttpJob) As DBResult
        Dim start As Long = DateTime.Now
        Dim In As InputStream = Job.GetInputStream
        Dim cs As CompressedStreams
        In = cs.WrapInputStream(In, "gzip")
        Dim serverVersion As Float = ReadObject(In) 'ignore
        Dim method As String = ReadObject(In)
        Dim table As DBResult
        table.Initialize
        table.Columns.Initialize
        table.rows.Initialize
        table.Tag = Job.Tag
        If method = "query" Then
                Dim numberOfColumns As Int = ReadInt(In)
                For i = 0 To numberOfColumns - 1
                        table.Columns.Put(ReadObject(In), i)
                Next
                Do While ReadByte(In) = 1
                        Dim rowObjects(numberOfColumns) As Object
                        table.rows.Add(rowObjects)
                        For col = 0 To numberOfColumns - 1
                                Dim o As Object = ReadObject(In)
                                rowObjects(col) = o
                        Next
                Loop
        Else If method = "batch" Then
                table.Columns.Put("AffectedRows", 0)
                Dim rows As Int = ReadInt(In)
                For i = 0 To rows - 1
                        table.rows.Add(Array As Object(ReadInt(In)))
                Next
        End If
        In.Close
        Log("HandleJob: " & (DateTime.Now - start))
        Return table
End Sub
'Reads a file and returns the file as a bytes array.
Public Sub FileToBytes(Dir As String, FileName As String) As Byte()
        Dim out As OutputStream
        out.InitializeToBytesArray(0)
        Dim In As InputStream = File.OpenInput(Dir, FileName)
        File.Copy2(In, out)
        out.Close
        Return out.ToBytesArray
End Sub
'Converts an image to a bytes array (for BLOB fields).
Public Sub ImageToBytes(Image As Bitmap) As Byte()
        Dim out As OutputStream
        out.InitializeToBytesArray(0)
        Image.WriteToStream(out, 100, "JPEG")
        out.Close
        Return out.ToBytesArray
End Sub
'Converts a bytes array to an image (for BLOB fields).
Public Sub BytesToImage(bytes() As Byte) As Bitmap
        Dim In As InputStream
        In.InitializeFromBytesArray(bytes, 0, bytes.Length)
        Dim bmp As Bitmap
        bmp.Initialize2(In)
        Return bmp
End Sub
'Prints the table to the logs.
Public Sub PrintTable(Table As DBResult)
        Log("Tag: " & Table.Tag & ", Columns: " & Table.Columns.Size & ", Rows: " & Table.Rows.Size)
        Dim sb As StringBuilder
        sb.Initialize
        For Each col In Table.Columns.Keys
                sb.Append(col).Append(TAB)
        Next
        Log(sb.ToString)
        For Each row() As Object In Table.Rows
                Dim sb As StringBuilder
                sb.Initialize
                For Each record As Object In row
                        sb.Append(record).Append(TAB)
                Next
                Log(sb.ToString)
        Next
End Sub
```