

Índice:

CAPÍTULO 2. PROGRAMAÇÃO EM VISUAL BASIC	2
2.1 MÓDULOS DE INICIALIZAÇÃO E FECHAMENTO DO SENSOR.....	2
2.1.1 Inicializando o módulo	2
Método 1.....	2
Método 2.....	2
2.1.2 Declarando um objeto filho	2
Declarando um device object.....	3
Declarando um extraction object.....	3
Declarando um matching object.....	3
Declarando um FPData object.....	3
Declarando um FPIImage object.....	3
Declarando um Search object.....	4
2.1.3 Fechar o módulo após o uso	4
2.2 RELACIONANDO OS SENSORES NA PROGRAMAÇÃO	4
2.2.1 Listar dispositivos	4
2.2.2 Inicializando o dispositivo	5
2.2.3 Fechando o dispositivo	6
2.3 FINGERPRINT ENROLLMENT.....	6
2.4 FINGERPRINT VERIFICATION.....	7
2.5 AMBIENTE CLIENT/SERVER	7
2.5.1 Fingerprint Enrollment	7
2.5.2 Fingerprint Verification.....	8
2.6 USANDO PAYLOAD	9
2.6.1 Inserindo um payload dentro de um FIR.....	9
2.6.2 Extraíndo um payload do FIR	10
2.7 MUDANDO A INTERFACE DE USUÁRIO DO NBioAPI.....	11
2.8 FINGERPRINT IDENTIFICATION	11
2.9 ATIVAR O AUTO-ON.....	12

Capítulo 2. Programação em Visual Basic

Este capítulo veremos programação em Visual Basic no qual usa o módulo NBioBSP COM. O módulo NBioBSP COM é designado a facilitar a integração do NBioBSP para desenvolvedores que utilizam ferramentas RAD e desenvolvimento Web.

O módulo NBioBSP COM não suporta todas as funções do módulo NBioBSP. Quando o desenvolvimento for para Web, os dados da impressão digital devem ser controlados no formato texto. Estes dados podem ser controlados no formato texto ou binário, para desenvolvimento em Visual Basic ou em Delphi.

*É aconselhável a leitura do arquivo: Introdução - Guia de desenvolvimento eNBSP SDK (Português) para melhor entendimento deste material. Para mais informações veja o guia completo: Guia completo de desenvolvimento eNBSP SDK (Inglês).

2.1 Módulos de inicialização e fechamento do sensor

2.1.1 Inicializando o módulo

Existe duas maneiras para inicializar o módulo NBioBSP COM, declarando um novo objeto ou usando a função **CreateObject()**. Usando qualquer um dos métodos terá o mesmo resultado.

Método 1

```
Dim objNBioBSP As NBioBSPCOMLib.NBioBSP
...
Set objNBioBSP = New NBioBSPCOMLib.NBioBSP
```

* Para usar esse método, você deve adicionar “NITGEN NBioBSP SDK Add-on Pack” como Referência do Menu do projeto.

Método 2

```
Dim objNBioBSP As NBioBSPCOMLib.NBioBSP
...
Set objNBioBSP = CreateObject("NBioBSPCOM.NBioBSP")
```

2.1.2 Declarando um objeto filho

Existem seis objetos filhos usados no módulo NBioBSP COM demonstrados abaixo:

Declarando um device object

Este objeto é usado para abrir, fechar e atribuir configurações no dispositivo.

```
Dim objDevice As IDevice `Declarando device object
...
Set objDevice = objNBioBSP.Device
```

Declarando um extraction object

Este objeto é usado para capturar e registrar os dados de uma impressão digital.

```
Dim objExtraction As IExtraction `Declarando Extraction Object
...
Set objExtraction = objNBioBSP.Extraction
```

Declarando um matching object

Este objeto é usado para comparação (1:1).

```
Dim objMatching As IMatching `Declarando Matching Object
...
Set objMatching = objNBioBSP.Matching
```

Declarando um FPData object

Este objeto é usado para converter e recriar um dado de impressão digital.

```
Dim objFPData As IFPData `Declarando FPData Object
...
Set objFPData = objNBioBSP.FPData
```

Declarando um FPImage object

Este objeto é usado para extrair e salvar as imagens da impressão digital.

```
Dim objFPImage As IFPImage `Declarando FPImage Object
...
Set objFPImage = objNBioBSP.FImage
```

Declarando um Search object

Este objeto é usado para executar as funções do IndexSearch Engine.

```
Dim objIndexSearch As IIndexSearch `Declaration Search Object
```

```
...
```

```
Set objIndexSearch = objNBioBSP.IndexSearch
```

2.1.3 Fechar o módulo após o uso

Declare o objeto como livre ao sair da aplicação. Quando a aplicação é encerrada, essa operação é feita automaticamente no Visual Basic.

```
Set objNBioBSP = nothing ' Liberando NBioBSPCOM object
```

2.2 Relacionando os sensores na programação

O dispositivo deve ser aberto antes para que possa utilizá-lo. Use o método **Enumerate** para determinar qual dispositivo está ligado ao sistema.

2.2.1 Listar dispositivos

Antes de abrir o dispositivo, use o método **Enumerate** para determinar o número e o tipo dos dispositivos listados no computador. Uma vez ativado, o número de dispositivos listados no computador irá aparecer na propriedade do **EnumCount** e o ID de cada dispositivo irá aparecer na propriedade do **EnumDeviceID**. **EnumDeviceID** é um array do tipo LONG. **EnumDeviceID** é composto dos nomes de dispositivo e os números de exemplo deles.

DeviceID = Instance Number + Device Name

Se existe apenas um dispositivo para cada sistema, o número de exemplo será "0". Neste caso, o nome do dispositivo terá o mesmo valor que o ID do dispositivo. Para mais informação, consulte o **NBioBSP SDK Programmer's Manual**.

Device Name	Value	Notes
NBioBSP_DEVICE_NAME_FDP02	1(0x01)	FDP02 device
NBioBSP_DEVICE_NAME_FDU01	2(0x02)	FDU01 device

[Predefined Device names]

Device ID	Value	Notes
NBioBSP_DEVICE_ID_NONE	0(0x0000)	No devices
NBioBSP_DEVICE_ID_FDP02_0	1(0x0001)	The first instance of FDP02
NBioBSP_DEVICE_ID_FDU01_0	2(0x0002)	The first instance of FDU01
NBioBSP_DEVICE_ID_AUTO_DETECT	255(0x00FF)	Detect device automatically

[Predefined Device IDs]

Veja o exemplo de como usar o método **Enumerate**.

```
Set objDevice = objNBioBSP.Device ' Set Device object
...
Call objDevice.Enumerate ' enumerate devices
...
comboDevice.AddItem "Auto_Detect"
...
For DeviceNumber = 0 To objNBioBSP.EnumCount
Select Case objNBioBSP.EnumDeviceID(DeviceNumber)
Case NBioBSP_DEVICE_ID_FDU04_0
comboDevice.AddItem "FDU04"
Case NBioBSP_DEVICE_ID_FDU14_0
comboDevice.AddItem "FDU14"
End Select
Next i
...
```

O **EnumDeviceID (DeviceNumber)** pode ser demonstrado como o número de dispositivo para o **DeviceNumber** nas propriedades do EnumDeviceID (DeviceNumber). Como por exemplo, EnumDeviceID (0) irá mostrar o ID do primeiro dispositivo.

2.2.2 Inicializando o dispositivo

O método **Open** é usado para iniciar o dispositivo para o **NBioBSP.COM**. A inicialização do dispositivo deve ser feita antes dele exercer suas funções, como registro, captura e verificação. Para essas funções funcionarem corretamente deve – se executar primeiramente o método **Open**.

Se você não tem certeza de qual dispositivo está instalado, utilize o método **Enumerate** para determinar qual dispositivo está previamente instalado.

```
DeviceID = NBioBSP_DEVICE_ID_FDP02_0 'first instance of FDP02
objNBioBSP.OpenDevice(DeviceID)
If objNBioBSP.ErrorCode = NBioBSPERROR_NONE Then
' Dispositivo aberto com sucesso ...
Else
'Falha ao abrir dispositivo ...
End If
```

O dispositivo pode ser detectado automaticamente usando o **NBioBSP_DEVICE_ID_AUTO_DETECT**. *Recomendável.

```
objNBioBSP.OpenDevice(NBioBSP_DEVICE_ID_AUTO_DETECT)
```

NBioBSP_DEVICE_ID_AUTO_DETECT usa o último dispositivo detectado no computador.

2.2.3 Fechando o dispositivo

O método `close` deve ser usado para fechar o dispositivo. O mesmo **DeviceID**, usado para chamar o método **Open**, deve ser usado novamente para chamar o método **Close**.

```
DeviceID = NBioBSP_DEVICE_ID_FDP02_0
objNBioBSP.OpenDevice (DeviceID)
...
objNBioBSP.CloseDevice(DeviceID)
If objNBioBSP.ErrorCode = NBioBSPERROR_NONE Then
    'Dispositivo fechado com sucesso ...
Else
    ' Falha ao fechar o dispositivo ...
End If
```

O atual dispositivo deve ser fechado ao abrir outro dispositivo.

2.3 Fingerprint Enrollment

O método **Enroll** é usado para registrar as impressões digitais. Este método deve ser usado após declarar o *extraction object*. Todos os dados das impressões digitais são usados no formato binário ou um texto codificado encontrado no módulo **NBioBSP**. Os dados da impressão digital serão armazenados no **FIR** ou no **TextEncoedFIR**, para então registrado-los. O **TextEncoedFIR** tem o valor do tipo **String**.

```
Dim szFIRTextData As String
Dim szPayload As String
...
Set objExtraction = objNBioBSP.Extraction
Call objExtracion.Enroll(szPayload, null)
If objNBioBSP.ErrorCode = NBioBSPERROR_NONE Then
    ' Registro com sucesso...
szTextEncodedFIR = objExtraction.TextEncodedFIR
    ' Gravar FIR em arquivo ou DB
Else
    ' Falha no registro ...
End If
```

Os dados das impressões digitais são armazenado como **TextEncoedFIR** quando utilizar um Banco de dados. Os dados das impressões digitais também podem ser salvos com o formato binário como segue o exemplo:

```
Dim biFIR() As Byte
...
ReDim biFIR(objExtraction.FIRLength) As Byte
BiFIR = Space(objExtraction.FIRLength)
```

BiFIR = objExtraction.FIR

2.4 Fingerprint verification

O método **Verify** executa uma validação de impressão digital, usando os dados de uma impressão digital previamente cadastrada, através da comparação. Este método deve ser usado após declarar o *Matching object*. O resultado é salvo como um valor nas propriedades do **MatchingResult** no qual retorna o valor “1” para sucesso e “0” para falha na verificação.

```
Dim storedFIRTextData As String
Dim szPayload As String
...
' Ler o FIRText armazenado a partir de um arquivo ou DB.
...
Set objMatching = objNBioBSP.Matching
Call objNBioBSP.Verify(storedFIRTextData) ' TextEncodedFIR
If objMatching.MatchingResult = NBioBSP_TRUE then
' Verificado com sucesso
If objMatching.ExistPayload = NBioAPI_TRUE then
`Exist
szPayload = objMatching.TextEncodedPayload
Else
...
End if
Else
' Falha ao verificar
End if
```

Define	Value
NBioBSP_TRUE	1
NBioBSP_FALSE	0

[Values used in IsMatched]

2.5 Ambiente Client/Server

Em ambientes Client/Server, o enrollment das impressões digitais e os matchings se encontram em diferentes locais. As impressões digitais são geralmente registradas pelo cliente e depois encaminhadas para o Server.

O método **Enroll** registra as impressões digitais enquanto método **Capture** verifica as digitais.

2.5.1 Fingerprint Enrollment

Use o método Enroll para registrar as impressões digitais no Client.

```

Dim szTextEncodedFIR As String
Dim szPayload As String
...
Set objExtraction = objNBioBSP.Extraction
Call objExtraction.Enroll(szPayload, null)
If objExtraction.ErrorCode = NBioBSPERROR_NONE Then
    ' Registro com sucesso ...
szTextEncodedFIR = objExtraction.TextEncodedFIR
    ' Gravar FIR em arquivo ou DB
Else
    ' Falha no registro ...
End If

```

2.5.2 Fingerprint Verification

Use o método **Capture** para registrar a impressão digital no Client. Enquanto o método **Enroll** permite armazenar diversas digitais em um único **FIR** o método **Capture** registra apenas uma digital. O método **Capture** deve ser usado após declarar o *Extraction Object* . Insira qual o propósito da captura no parâmetro. O valor para o propósito, definido no *header* é variável, mas esse método permite somente NBioAPI_FIR_PURPOSE_VERIFY como valor.

```

Dim szTextEncodedFIR As String
...
Set objExtraction = objNBioBSP.Extraction
Call objExtraction.Capture(NBioAPI_FIR_PURPOSE_VERIFY)
If objExtraction.ErrorCode = NBioBSPERROR_NONE Then
    ' Captura com sucesso ...
szTextEncodedFIR = objExtraction.TextEncodedFIR
    ' Gravar FIR em arquivo ou DB
Else
    ' Falha na captura ...
End If

```

Define	Description
NBioAPI_FIR_PURPOSE_VERIFY	for Verification
NBioAPI_FIR_PURPOSE_IDENTIFY	for Identify (currently not used)
<i>NBioAPI_FIR_PURPOSE_ENROLL</i>	for Registration
<i>NBioAPI_FIR_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY</i>	for Verification(Only)
<i>NBioAPI_FIR_PURPOSE_ENROLL_FOR_IDENTIFICATION_ONLY</i>	for Identification(Only)
NBioAPI_FIR_PURPOSE_AUDIT	for Audit (currently not used)
NBioAPI_FIR_PURPOSE_UPDATE	for Update (currently not used)
[Values used in Capture]	

O método **VerifyMatch** utiliza dois FIRs, um determinado FIR transmitido pela rede e um outro FIR previamente registrado. Confira a propriedade do **MatchingResult** para checar o resultado; "1"

para realizado com sucesso e “0” para verificação incorreta. Após a verificação, o método retorna o payload.

```
Dim storedFIRTextData As String
Dim processedFIRTextData As String
Dim szPayload As String
...
`Compara FIR gerado a partir do cliente com FIR armazenado no DB
Set objMatching = objNBioBSP.Matching
Call objMatching.VerifyMatch(processedFIRTextData, storedFIRTextData)
If objMatching.MatchingResult = NBioAPI_TRUE then
` Comparação com sucesso
If objMatching.ExistPayload = NBioAPI_TRUE then
`Exist
szPayload = objMatching.TextEncodedPayload
Else
...
End if
Else
` Falha na comparação
End if
```

2.6 Usando Payload

Outras informações dentro dos dados fingerprint são chamados de **payload**. Somente dados do tipo texto codificado podem ser usados no módulo **NBioBsP** como **payload**.

2.6.1 Inserindo um payload dentro de um FIR

Quando for registrar uma impressão digital, use o método **Enroll** para incluir um **payload** nos dados da impressão digital. O método **CreateTemplate** pode ser usado para inserir um **payload** dentro de um FIR já existente. O método **Enroll** irá usar um dado da impressão digital para prover um parâmetro de comparação.

```
Dim szTextEncodedFIR As String
Dim szPayload As String
...
szPayload = "Your Payload Data"
...
Set objExtraction = objNBioBSP.Extraction
Call objExtraction.Enroll(szPayload, null)
If objExtraction.ErrorCode = NBioBSPERROR_NONE Then
` Registro com sucesso ...
szTextEncodedFIR = objExtraction.TextEncodedFIR
` Gravar FIR em arquivo ou DB
Else
` Falha no registro ...
End If
```

Use o método **CreateTemplate** para inserir um **payload** em um dado de impressão digital já existente. O método **CreateTemplate** pode também adicionar um novo dado de impressão digital

dentro de um dado de impressão digital já existente. Da mesma maneira que o método **Enroll**, o novo dado da impressão digital será inserido dentro das propriedades do **TextEncodedFIR**. Esse método deve ser executado antes de declarar um **FPData object**.

```
Dim storedFIRTextData As String
Dim newFIRTextData As String
Dim szPayload As String
...
szPayload = "Your Payload Data"
...
Set objFPData = objNBioBSP.FPData
Call objFPData.CreateTemplate(storedFIRTextData, null, szPayload)
If objFPData.ErrorCode = NBioBSPERROR_NONE Then
' Template criado com sucesso ...
newFIRTextData = objFPData.TextEncodedFIR
' Gravar FIR em um arquivo ou DB
Else
' Falha no registro ...
End If
```

2.6.2 Extraindo um payload do FIR

O **payload** nos registros de identificação das impressões digitais (dados registrados) somente será extraído se a partir do método **Verify** ou o método **VerifyMatch** retornar *true*.

Checar a propriedade **IsPayload** depois de incluir, para verificar se existe um **payload** no FIR. Se o **ExistPayload** é *true*, o **payload** será mostrado nas propriedades do **TextEncodedPayload**.

```
Dim storedFIRTextData As String
Dim szPayload As String
...
' Ler FIRText a partir de um arquivo ou DB.
...
Set objMatching = objNBioBSP.Matching
objMatching.Verify(storedFIRTextData)
If objMatching.MatchingResult = NBioAPI_TRUE Then
' Verificado com sucesso
If objNBioBSP.ExistPayload = NBioAPI_TRUE Then
' Payload existente
szPayload = objMatching.TextEncodedPayload
End If
Else
' Falha na verificação
End if
```

Extraindo os **payloads** usando o método **VerifyMatch** é a mesma operação usando o método **Verify**. Ao executar **VerifyMatch**, como primeiro parâmetro, use os dados de modo comparativo e como segundo parâmetro, use os dados armazenados (Template registrado).

Os dados do **payload** podem somente ser extraídos de um dado FIR no segundo parâmetro (enrolledFIRTextData). Embora o dado FIR no primeiro parâmetro inclua o **payload**, o mesmo não é retornado.

```
...
```

```

Set objMatching = objNBioBSP.Matching
Call objMatching.VerifyMatch(capturedFIRTextData, enrolledFIRTextData)
if objMatching.MatchingResult = NBioAPI_TRUE then
' Verificado com sucesso
if objMatching.ExistPayload = NBioAPI_TRUE then
' Get payload
szPayload = objMatching.TextEncodedPayload
end if
end if

```

2.7 Mudando a interface de usuário do NBioAPI

O módulo **NBioBSP.COM** oferece recursos para customização da UI (user interface) básica. Use o método **SetSkinResource** para carregar os recursos da UI em diversas línguas.

```

Dim szSkinPath
...
CommonDialog.ShowOpen
If CommonDialog.CancelError = False then
szSkinPath = CommonDialog.FileName
objNBioBSP.SetSkinResource(szSkinPath)
End if

```

Para que o mesmo Skin (Pop-UP) seja utilizado, inclua somente a linha abaixo no código especificando o local da DLL.

```

// Set skin resource
objNBioBSP.SetSkinResource("C:\Arquivos de programas\NITGEN eNBSP\SDK\Skins\
NBSP2Por.dll")

```

2.8 FingerPrint Identification

Use o método IndexSearch para armazenar e identificar os templates. No objeto IndexSearch criado, devem ser adicionados o template (impressão digital) e o id dos usuários. Se estes dados estiverem armazenados em um DB, então devem ser carregado no IndexSearch através de uma rotina de repetição, até que todos os templates e IDs do banco forem adicionados.

O processo de identificação ocorrerá diretamente com os dados armazenados neste objeto (na memória) e não no DB, é este processo que torna a busca instantânea. O resultado da identificação será o ID do template identificado.

Adicionando no IndexSearch :

```

Dim nUserID As Long 'ID do User.
Dim szFir As String 'Template do User
...
SQL = "SELECT ID, Template FROM TbUser;"

```

```

If objconn.ExecutarSQLRS(sql, ors) Then
  While Not ors.EOF
    nUserID = CDb(ors!ID)
    szFir = ors!digital
    Call objIndexSearch.AddFIR(szFir, nUserID)
    ors.MoveNext
  Wend
End If

```

Identificando:

```

Call objDevice.Open(NBioAPI_DEVICE_ID_AUTO_DETECT)
Call objExtraction.Capture(NBioAPI_FIR_PURPOSE_VERIFY)
Call objDevice.Close(NBioAPI_DEVICE_ID_AUTO_DETECT)

szTextEncodeFIR = objExtraction.TextEncodeFIR
Call objIndexSearch.IdentifyUser(szTextEncodeFIR, 6) 'Faz a identificação do usuário. 1º
Parametro: String capturada a identificar. 2º Parametro: Nível de segurança (Varia de 1 à 9).
If objIndexSearch.ErrorCode = NBioAPIERROR_NONE Then
  'User identificado com sucesso
  User_id = objIndexSearch.UserID 'objIndexSearch.UserID irá retornar o ID do user
  identificado. Com este valor deve-se fazer a busca no Database dos demais dados do usuário.
Else
  'User não identificado
End If

```

2.9 Ativar o Auto-On

Use o método CheckFinger para ativar o auto-on ou auto-captura, ou seja, a captura é efetuada automaticamente pelo sensor biométrico quando o dedo é posicionado pelo mesmo. Esta característica é existente apenas no modelo do Hamster II.

O método CheckFinger retorna o valor 0 ou 1, sendo 1 para indicar uma impressão digital presente. Este método deve ser obrigatoriamente utilizado em uma estrutura de repetição.

```

Private Sub Timer()

Timer.Interval = 300
Timer.Enabled = True
Call objDevice.Open(NBioAPI_DEVICE_ID_AUTO_DETECT)
  //Captura sem Pop-up
Call objExtraction.WindowStyle = NBioBSPTType.WINDOW_STYLE.INVISIBLE

  //Verifica se existe um dedo posicionado no sensor
If objDevice.CheckFinger > 0 Then
  //Tempo de captura
Call objExtraction.DefaultTimeout = 3000
Call objExtraction.Capture(NBioAPI_FIR_PURPOSE_VERIFY)
End If

End Sub

```