BioLink Solutions

BioLink SDK

Developer's Guide

Version 6.x

BioLink Solutions
box # 404, c/o IPS,
511 Avenue of the Americas,
PMB 572, New York,
NY 10011, USA

Web site: http://www.biolinksolutions.com/
Technical support e-mail: support@biolinksolutions.com
Contact e-mail: sales@biolinksolutions.com

Contents

Table of Figures

Chapter 1

Introduction

Thank you for choosing the **BioLink Software Development Kit (BSDK)**.The BioLink SDK provides for quick and seamless integration of biometric identification features into various solutions and systems. To use BSDK, the developer can have little or even no knowledge of biometric technologies.

BSDK offers you a smart way for building security and access control subsystems in any kind of business software with personal identification or authentication.

With **BSDK** you can increase biometric systems performance by using multi-processor systems. BSDK supports identification utilizing all CPU cores simultaneously.

BSDK is compliant with other BioLink's solutions, including all fingerprint scanners of the BioLink U-Match family, automated multi-biometric information system BioLink AMIS and biometric identification servers BioLink Authenteon.

In This Chapter

BSDK purpose

BioLink SDK is a a full-featured set of mathematical algorithms providing any scale developers the ability to smoothly integrate unique biometric technologies into new or existent various access control systems, client or server hardware, software and so forth. BioLink SDK saves the developer the trouble of the deep knowledge of biometrics and allows to concentrate solely on application development and design.

BioLink SDK provides a set of functions that allow developers to perform the following operations:

- Read fingerprint images from various fingerprint scanning devices;
- Check fingerprint images quality to ensure stronger recognition;
- Convert fingerprint images to biometric templates;
- Perform matching by comparing a fingerprint template with another fingerprint template.
- Speed-up matching by using all CPU cores simultaneously when operating in large databases.
- Compress and decompress images using the brand new BioLink WSQ Gray-scale Fingerprint Image Compression Algorithm.

BioLink SDK includes:

- C, JAVA, .NET libraries.
- A number of samples, which demonstrate all SDK functions and capabilities in different environments and conditions, providing better comprehension and implementation of the SDK possibilities;
- Drivers for supported fingerprint scanner devices from various vendors;
- Source code examples (C/C++, C#, Java, JScript, Delphi).

BSDK advantages

The **major advantages of BioLink SDK** are:

- 1-to-N matching can utilize all CPU cores providing increased performance when operating large databases.
- Wide range of supported programming languages and platforms.
- Support for numerous fingerprint scanners with USB and Ethernet interfaces, in both embedded and standalone versions.
- Compliance with international and industry standards.
- Biometric server provider (included into the BSDK package).
- Graphic interface to obtain and process fingerprint images available in compiled version and source codes.
- Universal low-level API (Application Programming Interface) for consistent operation with various fingerprint scanners.
- Ability to operate both with fingerprints obtained from biometric scanners and their images.
- Support for fault-tolerant high-performance BioLink IDenium service and automated multi-biometric information system BioLink AMIS (for large-scale identification projects).

BSDK supports the following **international standards**:

- WSQ (Wavelet Scalar Quantization).
- ANSI INCITS 378-2004.

System requirements

The workstation, on which you are going to install BioLink SDK, must meet the following requirements:

- Pentium IV 1500 MHz processor or better;
- Windows 2000/XP/2003/Vista;
- .NET framework 2.0;
- Microsoft Visual Studio .Net 2005.

Requirements for the RAM available depend on the application you are developing. For further details refer to your development environment documentation.

The target platform for the application developed with BSDK must at least meet the following requirements:

- Windows 2000/XP/2003/Vista;
- USB port for License Key (or Ethernet for network License);
- .Net Framework 2.0 (required for BioLink.Biometrics2 API);
- Pentium IV 1500 MHz processors are strongly recommended.

Multi-core CPU is an advantage when operating large databases.

Note. More limitation may be applied by the scanner device drivers.

Supported devices

BioLink SDK enables applications to work with the following fingerprint identification devices:

- Windows
  - BioLink U-Match 3.5
  - Cross Match L SCAN 100
  - Eikon Fingerprint Reader (TCRE)
  - BioLink U-Match 5.0 with integrated smart card reader
- Linux
  - BioLink U-Match 3.5

Version history

Version 6.0

- BSDK API has been completely rewritten providing unified support for different mathematical algorithms.
- 1-to-N matching can utilize all CPU cores providing increased performance when operating large databases.
- Support for Linux and WinCE has been added.
- Support for JAVA programming environment has been added.
- JScript samples has been introduced.
- Documentation has been redesigned, significantly updated to include updated API; new updated colorized source code samples introduced.
- JAVA API reference introduced including complete JAVA API documentation
- C API Reference has been completely rewritten
- VS 2008 MSDN style has been applied for the .NET API reference.

Installation Notes

To develop applications using BSDK algorithms and functions you should install BSDK on your computer. To do it launch BSDK Installation Wizard and follow the instructions on the screen.

Figure 1: BSDK Autorun program

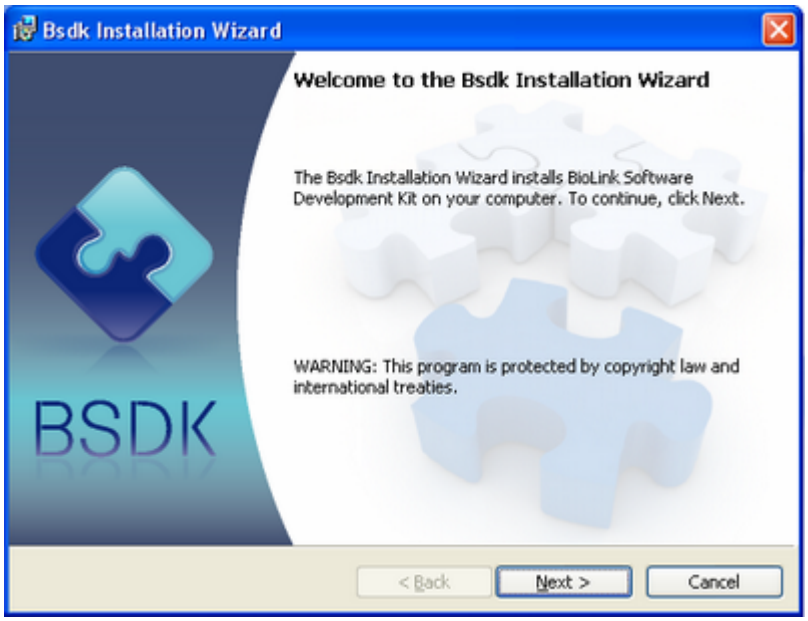

Figure 2: BSDK Installation Wizard

Please note that complete BSDK installation is required only for developers creating programs with this software.
You shouldn't install BSDK on each end-user computer, on which your BSDK application will operate. You should only include necessary BSDK files in your application's installation package in order to have this files copied.

For list of all files required for BSDK redistribution, see the ***Redistributing BioLink SDK*** (on p. 14) topic.

Documentation

BioLink SDK documentation comprises the following:

- *BioLink SDK Developer's Guide* (this document) - gives a brief introduction to BioLink SDK, describes general BSDK purpose, features and advantages, explains BioLink SDK licensing and protection mechanisms and presents a number of basic instructions on how to start developing your applications with BioLink SDK.
- *BioLink SDK C reference* - provides complete explanation to the BioLink SDK C API;
- *BioLink SDK .NET reference* - delves into BioLink SDK .NET API targeted at Microsoft .Net Framework developers;
- *BioLink SDK Java reference* - explains classes and functions comprising the BioLink SDK Java API enabling BioLink biometrics support in Java applications.

Package contents

The **BioLink SDK** package should include the following items:

1) CD with SDK libraries:
   - Fingerprint scanners drivers;
   - Native C API, .NET, Java libraries;
   - ActiveX User Interface;
   - Sample applications and source code examples.

2) 1 USB key or BioLink U-Match 3.5 scanner with embedded license.

3) Documentation (electronic) .

Verify the inventory before the installation of BioLink SDK against the above list. If any inventoried item(s) are missing, please contact the distributor of your BioLink SDK.

Note. The developer of this software reserves the right to slightly modify the package contents as it considers necessary.

Errata

BioLink Solutions make every effort to ensure that there are no errors or misprints in the text of all documents supplied with BSDK. However, no one is perfect, and mistakes do occur. If you find an error in one of our documents, like a spelling mistake or faulty piece of code, we would be very grateful for your feedback. By sending in errata you may save another reader hours of frustration, and at the same time you will be helping us provide even higher quality of our documentation.

Chapter 2

Developing with BSDK

New BioLink SDK was designed taking into consideration the experience of using the previous BSDK versions. The known issues, pitfalls, and API inconvenience has been considered to be corrected and improved in the new version of BSDK. The BSDK architecture implements advanced object-oriented concepts and ideas. Therefore it will be easy to begin working with BSDK both for rookie and guru developers.

The great feature of BSDK is that you do not need to be a guru of biometrics to start developing biometric applications. BSDK save you much time of learning biometric algorithms, techniques and best practices presenting you a universal solution allowing you to seamlessly integrate advanced biometric technologies into your existing access control systems, visitors' registration solutions, time and attendance utilities. The list of applications where you can implement BSDK could be continued endlessly.

Linux systems are also supported, thus expanding the horizons of biometric implementation. The leading software vendors have already evaluated Linux systems and have found them innovative, perspective and capable of performing the most complex tasks. Nowadays Linux is used in many hardware appliances, so implementing biometrics in Linux software project can elevate this project to an unmatched level of security and reliability never achieved on the other platform. And the point is, that you do nor require any material and/or human resources to make it a reality.

In This Chapter

BSDK architecture

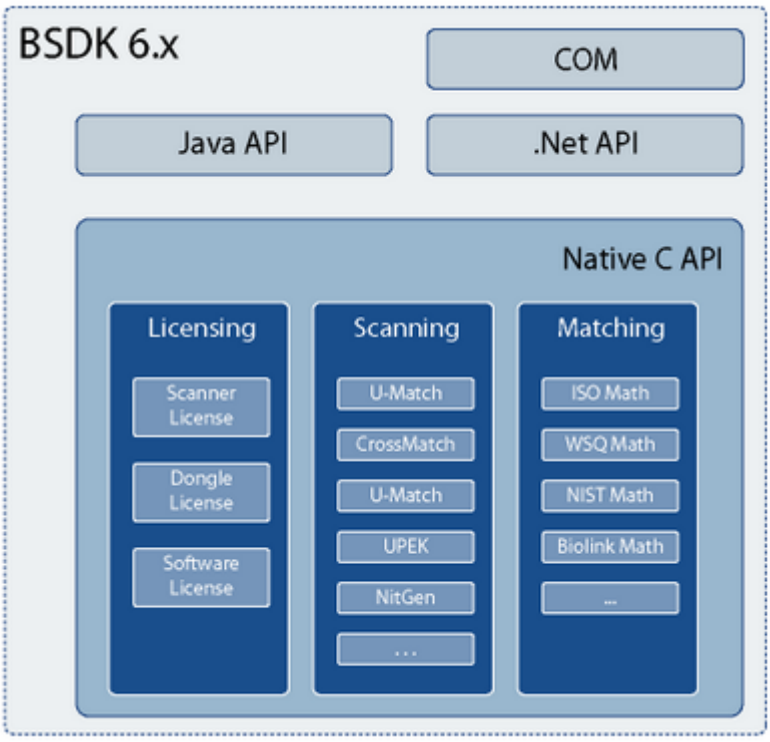

Figure 3: BSDK Architecture

The core of BioLink SDK is built upon the **Native C API**. Other APIs (Java and .NET) can be accessed through the corresponding wrappers. The Native C API uses interfaces to provide unified access to all core objects and functions.

BSDK C API comprises the following three main parts:

1) Compiled dynamic libraries;

2) Documentation;

3) A number of samples demonstrating general usage scenarios and best practices

Windows samples are compiled in Visual Studio 2008 IDE and contain project files for this VS edition. Linux samples have no complied modules and should be recompiled in your programming environment.

General guidelines for developers

BSDK C API is the core API for all platforms and is intended to be used mainly by C developers. If you are a JAVA or NET programmer move toward the corresponding SDKs. They will significantly save your time when learning BSDK core algorithms.

If your preferred programming language is not directly supported (there no compiled libraries for it) you will have to write wrappers over core C libraries on your own.

BSDK C API Architecture

All BSDK operation performed by BSDK objects such as **Bsdk_Image**, **Bsdk_Matcher** and so on. All functions named according to theirs parent object. For example for **Bsdk_Image**, all its functions have Bsdk_Image_ prefix.

There are two main object types:

- objects that should be created and then explicitly destroyed (for example, **Bsdk_Image**, **Bsdk_Template**, etc.);
- objects which life time is equal to their owner object (for example **Bsdk_Finger**, **Bsdk_Impression**, etc.); they should not be manually destroyed

All explicitly created BSDK objects should be destroyed by **Bsdk_Destroy_Object** functions to release their resources.

Thus, for the first type of the objects a common usage scenario is the following:

1) **Creating object** - explicitly (for example **Bsdk_Image_Create** returns created **Bsdk_Image**) or implicitly (for example **Bsdk_Scanner_AcquireImage** returns created Bsdk_Image).
2) **Using created object**.
3) **Destroying explicitly created object** by Bsdk_Destroy_Object.

For the second type of the object steps 1 and 3 are not required.

Common BSDK development scenarios

All **common BSDK development scenarios** should have at least three of the following main steps:

1) Create BSDK license object.
2) Create and work with BSDK object to perform necessary actions.
3) Destroy every created object to release resources.
4) Release license.

The code below demonstrates the acquiring image from the scanner (C++):

```
// 1st step: create license object to work with protected methods.
Bsdk_License_Create( BSDK_LICENSE_TYPE_ALL );

// 2nd step: create suitable BSDK objects to perform necessary
actions (in this scenario -  to acquire image)
Bsdk_DeviceListPtr deviceList;
Bsdk_DeviceList_Create( &deviceList, Bsdk_Device_Type_Any );

Bsdk_DeviceDescriptorPtr deviceDescriptor =
Bsdk_DeviceList_GetDeviceDescriptor( deviceList, 0 );

Bsdk_ScannerPtr scanner;
Bsdk_Scanner_Create( &scanner, deviceDescriptor );

Bsdk_ImagePtr image;
Bsdk_Scanner_AcquireImage( scanner, &image );
```

```
// 3rd step: destroy objects in use to release resources allocated
by them
Bsdk_Destroy_Object( deviceList );
Bsdk_Destroy_Object( scanner );
Bsdk_Destroy_Object( image );

// Last step: release resources allocated by the License
Bsdk_License_Close();
```

Redistributing BioLink SDK

General Principles

Redistribution is the process by which you distribute a finished application or component to be installed on other computers. The BSDK redistribution implies the redistribution of DLL (dynamic-link library) components, which are used by the application.

In C\C++ case it is enough simply to copy required DLLs to `system32` folder. While in case of .NET application some additional registration steps have to be performed.

C/C++ Software redistribution

In case of redistribution of BSDK with application written on C/C++ the following components must be provided:

- **C/C++ Win32**
  - General
- bsdk6x.dll
  - Scanners support
- ftrScanAPI.dll (U-Match 3.5, U-Match 5.0)
- iaapi.dll (Eikon Fingerprint Reader (TCRE))
- CaptureDLL.dll (Cross Match L SCAN 100)
- LS_COM.dll (Cross Match L SCAN 100)
- LS_GRB.dll (Cross Match L SCAN 100)
- LS_ILM.dll (Cross Match L SCAN 100)
- LS_IMG.dll (Cross Match L SCAN 100)
- LS_ITKND.dll (Cross Match L SCAN 100)
- LS_VIS.dll (Cross Match L SCAN 100)
- LSCAN100.dll (Cross Match L SCAN 100)
- **C/C++ Linux**
  - General

- libbsdk2.so
- libbsdk2.a
  - Scanners support
- libftrScanAPI.so (U-Match 3.5)

.NET (C#) software redistribution

In case of redistribution of BSDK with application based on .NET library (written on C#) the following components must be provided:

- **.NET Win32**
  - General
- Biolink.Biometrics2.dll
- bsdk6x.dll
  - Scanners support
- ftrScanAPI.dll (U-Match 3.5, U-Match 5.0)
- iaapi.dll (Eikon Fingerprint Reader (TCRE))
- CaptureDLL.dll (Cross Match L SCAN 100)
- LS_COM.dll (Cross Match L SCAN 100)
- LS_GRB.dll (Cross Match L SCAN 100)
- LS_ILM.dll (Cross Match L SCAN 100)
- LS_IMG.dll (Cross Match L SCAN 100)
- LS_ITKND.dll (Cross Match L SCAN 100)
- LS_VIS.dll (Cross Match L SCAN 100)
- LSCAN100.dll (Cross Match L SCAN 100)

If the COM interface is used (by VB6 or JS for example) the following registration required:

```
RegAsm.exe /tlb Biolink.Biometrics2.dll
```

Java software redistribution

You can build BSDK Java programs targeted at both Windows and Linux platforms. So the list of all components which should be redistributed with your software will be the following:

- **Java Win32**:
  - General

- bsdk6x.dll
- bsdk6x_jni.dll
- bsdk6x.jar
- swt.jar (windows)
  - Scanners support
- ftrScanAPI.dll (U-Match 3.5)

Java Linux:

- General
- libbsdk2.so
- libbsdk6x_jni.so
- bsdk6x.jar
- swt.jar (linux)
  - Scanners support
- libftrScanAPI.so (U-Match 3.5)

Chapter 3

How To

The **How To** section contains the topics, describing basic operations to be performed using BSDK software. The sample source code is also included.

The source code is given in the following languages (in the order of appearance):

- C/C++;
- C#;
- Java;
- JScript;
- Delphi.

In This Chapter

How to use License object

The BioLink SDK is protected by a license.

The **BioLink SDK License** can be of the following types (see the ***BSDK licensing*** (on p. 53) topic for more information about BSDK license types):

- embedded in Rainbow Sentinel USB dongle (only for Windows operating systems);
- embedded in BioLink U-Match 3.5 scanner;
- software.

To start work with BSDK the **License** object must be created. Any BSDK protected method call (compare, scan, identify) will fail with "*No License*" error if there is no successfully created and still existent License object.

The following operations are protected:

- template comparison (**Matcher** object);
- scanning (**Scanner** object);
- enumeration of devices (**DeviceList** object);
- template creation (**ImageProcessor** object).

During **License** object creation the license is not checked i.e. the **License** object constructor will return *Success* regardless of license presence.

The following code fragments illustrates how to create a **License** object.

C++

```
#include <Bsdk2.h>

int main( int argc, char* argv[] )
{
 //creating license
 Bsdk_License_Create( BSDK_LICENSE_TYPE_ALL );

 //... The program

 //destoying license
 Bsdk_License_Close();

 return 0;
}
```

C#

```
using System;
using Biolink.Biometrics2;

namespace Sample
{
      class Class1
      {
            static void Main(string[] args)
            {
                  try
                  {
```

```
                        // Create License object. The object must exist
                        // all the time during using BSDK
                        using (License license = new License())
                        {
                              //... The program
                        }

                  }catch(Exception ex)
                  {
                        Console.WriteLine("Error: {0}", ex.Message);
                  }
            }
      }
}
```

Java

```
import org.eclipse.swt.*;
import bsdk6x.*;

public class Class1
{
 public static void main(String[] args)
 {
        License license = null;
        try
        {
              // Create License object. The object must exist
            // all the time during using BSDK
              license = new License();

              //... The program
        }
        catch (Exception ex)
        {
              System.out.println(ex.getMessage());
        }
        finally
        {
              if (license != null)
                    license.dispose();
        }
 }
}
```

JScript

```
try
{
    var factory = new ActiveXObject("Biolink.Biometrics2.Factory");

    // Create License object. The object must exist
    // all the time during using BSDK
    var license = factory.CreateLicense();

    //... The program
}
catch(e)
{
    WScript.Echo(e.description);
}
finally
{
    if (license != null)
        license.Dispose();
    delete factory;
}
```

Delphi

```
unit BsdkDll;

interface

const
  BSDK_RESULT_BASE           = 0;
  BSDK_ERROR_INVALID_LICENSE = BSDK_RESULT_BASE - 12;

const
  BSDK_LICENSE_TYPE_FUTRONIC      = $00000001;
  BSDK_LICENSE_TYPE_SENTINEL      = $00000002;
  BSDK_LICENSE_TYPE_HARDWARE      = $00000004;
  BSDK_LICENSE_TYPE_SENTINEL_SHK  = $00000008;
  BSDK_LICENSE_TYPE_ALL           = $000000FF;

  function Bsdk_License_Create(tLicenseTypes : integer): integer; stdcall;
external 'bsdk6x.dll'
  function Bsdk_License_Close(): Integer; stdcall; external 'bsdk6x.dll'

  procedure CheckResult(result : integer);

implementation

uses SysUtils;

  procedure CheckResult(result : Integer);
  begin
      if (result <> BSDK_RESULT_BASE) then begin
          if  (result =  BSDK_ERROR_INVALID_LICENSE) then
              raise Exception.Create('No BSDK License');

          raise Exception.Create('BSDK Exception! Error: ' +
intToStr(result));
      end;
  end;

end.



program Samples;

{$APPTYPE CONSOLE}

uses
  SysUtils,
  BsdkDll;

var result : integer;

begin
    try
        //creating License object
        result := Bsdk_License_Create(BSDK_LICENSE_TYPE_ALL);
        CheckResult(result);

        Bsdk_License_Close();
    except
        On E : Exception do
            Writeln(E.Message);
    end;
end.
```

How to scan fingerprint images

First you need to create a **License** object which should exist all the time you use BSDK. Then you should obtain a list of all devices connected to a computer (using **DeviceList** object). The sample assumes that you choose the first found device from the list of the connected devices passing 0 to the **deviceList.DeviceDescriptor** method. The returned reference to the connected device is then used to create a **Scanner** object. The fingerprint image is then acquired from the scanner using **scanner.AcquireImage** method.

The following code fragments illustrates how to scan fingerprint image.

C++

```
#include <Bsdk2.h>
#include <Bscan.h>

int main( int argc, char* argv[] )
{
 //creating license
 Bsdk_License_Create( BSDK_LICENSE_TYPE_ALL );

 //creating the list of devices
 Bsdk_DeviceListPtr deviceList;
 Bsdk_DeviceList_Create( &deviceList, Bsdk_Device_Type_Any );

 //selecting first found device
 Bsdk_DeviceDescriptorPtr deviceDescriptor =
Bsdk_DeviceList_GetDeviceDescriptor( deviceList, 0 );

 //creating scanner object
 Bsdk_ScannerPtr scanner;
 Bsdk_Scanner_Create( &scanner, deviceDescriptor );

 //acquiring image from the scanner
 Bsdk_ImagePtr image;
 Bsdk_Scanner_AcquireImage( scanner, &image );

 //destroying objects
 Bsdk_Destroy_Object( deviceList );
 Bsdk_Destroy_Object( scanner );
 Bsdk_Destroy_Object( image );

 //destoying license
 Bsdk_License_Close();

 return 0;
}
```

C#

```
using System;
using Biolink.Biometrics2;

namespace Sample
{
      class Class1
      {
            static void Main(string[] args)
            {
                  try
                  {
                        // Create License object. The object must exist
                        // all the time during using BSDK
```

```
                        using (License license = new License())
                        {
                            //creating the list of devices
                            using (DeviceList deviceList = new DeviceList())
                            {
                                //selecting first found device
                                using (DeviceDescriptor deviceDescriptor =
deviceList.DeviceDescriptor(0))
                                {
                                    //creating Scanner object
                                    using (Scanner scanner = new
Scanner(deviceDescriptor))
                                    {
                                        //acquiring image from the scanner
                                        Image image = scanner.AcquireImage();
                                    }
                                }
                            }
                        }

                }catch(Exception ex)
                {
                      Console.WriteLine("Error: {0}", ex.Message);
                }
            }
      }
}
```

Java

```
package Sample;

import org.eclipse.swt.*;
import bsdk6x.*;

public class Class1
{
 public static void main(String[] args)
 {
        License license = null;
        DeviceList deviceList = null;
        DeviceDescriptor deviceDescriptor = null;
        Scanner scanner = null;
        Image image = null;
        try
        {
               // Create License object. The object must exist
              // all the time during using BSDK
               license = new License();

               //creating the list of devices
               deviceList = new DeviceList();
               //selecting first found device
               deviceDescriptor = deviceList.getDeviceDescriptor(0);
               //creating Scanner object
               scanner = new Scanner(deviceDescriptor);

               //acquiring image from the scanner
               image = scanner.acquireImage();

        }
        catch (Exception ex)
        {
               System.out.println(ex.getMessage());
        }
        finally
        {
               if (image != null)
```

```
                    image.dispose();
                if (scanner != null)
                    scanner.dispose();
                if (deviceDescriptor != null)
                    deviceDescriptor.dispose();
                if (deviceList != null)
                    deviceList.dispose();
                if (license != null)
                    license.dispose();
        }
 }
}
```

JScript

```
try
{
    var factory = new ActiveXObject("Biolink.Biometrics2.Factory");

    // Create License object. The object must exist
    // all the time during using BSDK
    var license = factory.CreateLicense();

    //creating the list of devices
    var deviceList = factory.CreateDeviceList();
    //selecting first found device
    var deviceDescriptor = deviceList.DeviceDescriptor(0);
    //creating Scanner object
    var scanner = factory.CreateScanner(deviceDescriptor);

    //acquiring image from the scanner
    var image = scanner.AcquireImage();
}
catch(e)
{
    WScript.Echo(e.description);
}
finally
{
    if (image != null)
        image.Dispose();;
    if (scanner != null)
        scanner.Dispose();
    if (deviceDescriptor != null)
        deviceDescriptor.Dispose();
    if (deviceList != null)
        deviceList.Dispose();
    if (license != null)
        license.Dispose();
    delete factory;
}
```

Delphi

```
unit BsdkDll;

interface

const
  BSDK_RESULT_BASE           = 0;
  BSDK_ERROR_INVALID_LICENSE = BSDK_RESULT_BASE - 12;

const
  BSDK_LICENSE_TYPE_FUTRONIC      = $00000001;
  BSDK_LICENSE_TYPE_SENTINEL      = $00000002;
  BSDK_LICENSE_TYPE_HARDWARE      = $00000004;
  BSDK_LICENSE_TYPE_SENTINEL_SHK  = $00000008;
  BSDK_LICENSE_TYPE_ALL           = $000000FF;
```

```
const
  Bsdk_Device_Type_Any                = $0000;
  Bsdk_Device_Type_Scanner_Any        = $0100;
  Bsdk_Device_Type_Scanner_Umatch     = $0101;
  Bsdk_Device_Type_Scanner_Upek       = $0102;
  Bsdk_Device_Type_Scanner_CrossMatch = $0103;

  function Bsdk_Destroy_Object(pObject : pointer): integer; stdcall; external
'bsdk6x.dll'

  function Bsdk_License_Create(tLicenseTypes : integer): integer; stdcall;
external 'bsdk6x.dll'
  function Bsdk_License_Close(): Integer; stdcall; external 'bsdk6x.dll'

  function Bsdk_DeviceList_Create(var pDeviceList : pointer; tDeviceType:
integer): integer; stdcall; external 'bsdk6x.dll'
  function Bsdk_DeviceList_GetDeviceDescriptor(pDeviceList : pointer;
iDeviceListIndex: integer): pointer; stdcall; external 'bsdk6x.dll'
  function Bsdk_Scanner_Create(var pScanner : pointer; pDeviceDescriptorPtr:
pointer): integer; stdcall; external 'bsdk6x.dll'
  function Bsdk_Scanner_AcquireImage(pScanner : pointer; var pImage: pointer):
integer; stdcall; external 'bsdk6x.dll'

  procedure CheckResult(result : integer);

implementation

uses SysUtils;

  procedure CheckResult(result : Integer);
  begin
      if (result <> BSDK_RESULT_BASE) then begin
          if  (result =  BSDK_ERROR_INVALID_LICENSE) then
              raise Exception.Create('No BSDK License');

          raise Exception.Create('BSDK Exception! Error: ' +
intToStr(result));
      end;
  end;

end.



program Samples;

{$APPTYPE CONSOLE}

uses
  SysUtils,
  BsdkDll;

var result : integer;
    pDeviceList : pointer = nil;
    pDeviceDescriptorPtr : pointer = nil;
    pScanner : pointer = nil;
    pImage : pointer = nil;

begin
    try
       try
            //creating License object
            result := Bsdk_License_Create(BSDK_LICENSE_TYPE_ALL);
            CheckResult(result);

            //creating the list of devices
```

```
            result := Bsdk_DeviceList_Create(pDeviceList,
Bsdk_Device_Type_Scanner_Any);
            CheckResult(result);

            //selecting first found device
            pDeviceDescriptorPtr :=
Bsdk_DeviceList_GetDeviceDescriptor(pDeviceList, 0);
            //creating Scanner object
            result := Bsdk_Scanner_Create(pScanner, pDeviceDescriptorPtr);
            CheckResult(result);

            //acquiring image from the scanner
            result := Bsdk_Scanner_AcquireImage(pScanner, pImage);
            CheckResult(result);

            Bsdk_License_Close();
        except
            On E : Exception do
                Writeln(E.Message);
        end;
    finally
        if (pImage <> nil) then
            Bsdk_Destroy_Object(pImage);
        if (pScanner <> nil) then
            Bsdk_Destroy_Object(pScanner);
        if (pDeviceDescriptorPtr <> nil) then
            Bsdk_Destroy_Object(pDeviceDescriptorPtr);
        if (pDeviceList <> nil) then
            Bsdk_Destroy_Object(pDeviceList);
    end;
end.
```

How to create, save and load template

The **ImageProcessor** object is used for building fingerprint templates from the images. To create a template you should pass **ImageSet** object (containing images acquired using **Scanner** object and added to the **ImageSet** with specific **FingerCode**) as a parameter of **ImageProcessor.CreateTemplate** method.

To save a template to a byte array you invoke **Save** method of the **Template** object. To load a template from a buffer array you use **Load** method specifying destination as a parameter.

The following code fragments illustrates how to create, save and load fingerprint template.

C++

```
#include <Bsdk2.h>
#include <Bscan.h>

int main( int argc, char* argv[] )
{
 //creating license
 Bsdk_License_Create( BSDK_LICENSE_TYPE_ALL );

 //creating the list of devices
 Bsdk_DeviceListPtr deviceList;
 Bsdk_DeviceList_Create( &deviceList, Bsdk_Device_Type_Any );

 //selecting first found device
```

```
 Bsdk_DeviceDescriptorPtr deviceDescriptor =
Bsdk_DeviceList_GetDeviceDescriptor( deviceList, 0 );

 //creating Scanner object
 Bsdk_ScannerPtr scanner;
 Bsdk_Scanner_Create( &scanner, deviceDescriptor );

 //acquiring image from the scanner
 Bsdk_ImagePtr image;
 Bsdk_Scanner_AcquireImage( scanner, &image );

 //creating ImageSet object to hold Image objects
 Bsdk_ImageSetPtr imageSet;
 Bsdk_ImageSet_Create( &imageSet );

 //adding Image object to image set with some finger index
 Bsdk_ImageSet_AddImage( imageSet, image, Bsdk_Finger_Index_UNKNOWN_FINGER );

 //creating ImageProcessor object to create template
 /* You can specify math type when creating ImageProcessor, Template or
Matcher objects.
  * The math type should be the same for all these objects.
  */
 Bsdk_ImageProcessorPtr imageProcessor;
 Bsdk_ImageProcessor_Create( &imageProcessor, Bsdk_Math_Type_Biolink );

 //creating template
 Bsdk_TemplatePtr template1;
 Bsdk_ImageProcessor_CreateTemplateFromImageSet( imageProcessor, imageSet,
&template1 );

 //saving template to buffer
 size_t size = Bsdk_Template_GetSize( template1 );
 unsigned char* buffer = new unsigned char[ size ];
 Bsdk_Template_Save( template1, buffer, size, 0 );

 //creating template
 Bsdk_TemplatePtr template2;
 Bsdk_Template_Create( &template2, Bsdk_Math_Type_Biolink );

 //loading template from buffer
 Bsdk_Template_Load( template2, buffer, size, 0 );

 delete[] buffer;

 //destroying objects
 Bsdk_Destroy_Object( deviceList );
 Bsdk_Destroy_Object( scanner );
 Bsdk_Destroy_Object( image );
 Bsdk_Destroy_Object( imageSet );
 Bsdk_Destroy_Object( imageProcessor );
 Bsdk_Destroy_Object( template1 );
 Bsdk_Destroy_Object( template2 );

 //destoying license
 Bsdk_License_Close();

 return 0;
}
```

C#

```
using System;
using Biolink.Biometrics2;

namespace Sample
{
    class Class1
```

```
    {
        static void Main(string[] args)
        {
            try
            {
                // Create License object. The object must exist
                // all the time during using BSDK
                using (License license = new License())
                {
                    //creating the list of devices
                    using (DeviceList deviceList = new DeviceList())
                    {
                        //selecting first found device
                        using (DeviceDescriptor deviceDescriptor =
deviceList.DeviceDescriptor(0))
                        {
                            //creating Scanner object
                            using (Scanner scanner = new
Scanner(deviceDescriptor))
                            {
                                //acquiring image from the scanner
                                Image image = scanner.AcquireImage();

                                byte[] bufferTemplate;

                                //creating ImageSet object to hold Image
objects
                                using (ImageSet imageSet = new ImageSet())
                                {
                                    //adding Image objects to ImageSet with
some FingerCode
                                    imageSet.AddImage(image, 0);

                                    //creating ImageProcessor object to create
template
                                    /* You can specify math type when creating
ImageProcessor, Template, TemplateSet or Matcher objects.
                                     * The math type should be the same for
all these objects.
                                     */
                                    using (ImageProcessor imgPrc = new
ImageProcessor())
                                    {
                                        //creating template
                                        Template templ =
imgPrc.CreateTemplate(imageSet);

                                        //saving template to bufferTemplate
byte array
                                        bufferTemplate = templ.ToArray();
                                    }
                                }

                                //loading template from buffer
                                Template template = new Template();
                                template.Load(bufferTemplate);
                            }
                        }
                    }
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine("Error: {0}", ex.Message);
            }
        }
    }
}
```

Java

```
package Sample;

import org.eclipse.swt.*;
import bsdk6x.*;

public class Class1
{
 public static void main(String[] args)
 {
        License license = null;
        DeviceList deviceList = null;
        DeviceDescriptor deviceDescriptor = null;
        Scanner scanner = null;
        Image image = null;
        ImageSet imageSet = null;
        ImageProcessor imgPrc = null;
        Template templ = null;
        Template template = null;
        try
        {
               // Create License object. The object must exist
               // all the time during using BSDK
               license = new License();

               //creating the list of devices
               deviceList = new DeviceList();
               //selecting first found device
               deviceDescriptor = deviceList.getDeviceDescriptor(0);
               //creating Scanner object
               scanner = new Scanner(deviceDescriptor);

               //acquiring image from the scanner
               image = scanner.acquireImage();

               //creating ImageSet object to hold Image objects
               imageSet = new ImageSet();
               //adding Image objects to ImageSet with some FingerCode
            imageSet.addImage(image, FingerCode.Unknown);

            //creating ImageProcessor object to create template
            /* You can specify math type when creating ImageProcessor,
Template, TemplateSet or Matcher objects.
             * The math type should be the same for all these objects.
             */
            imgPrc = new ImageProcessor();
            //creating template
            templ = imgPrc.createTemplate(imageSet);
            //saving template to bufferTemplate byte array
            byte[] bufferTemplate = templ.toArray();

            template = new Template();
            //loading template from buffer
            template.load(bufferTemplate);
        }
        catch (Exception ex)
        {
               System.out.println(ex.getMessage());
        }
        finally
        {
               if (template != null)
                      template.dispose();
               if (templ != null)
                      templ.dispose();
               if (imgPrc != null)
```

```
                imgPrc.dispose();
            if (imageSet != null)
                imageSet.dispose();
            if (image != null)
                image.dispose();
            if (scanner != null)
                scanner.dispose();
            if (deviceDescriptor != null)
                deviceDescriptor.dispose();
            if (deviceList != null)
                deviceList.dispose();
            if (license != null)
                license.dispose();
        }
 }
}
```

JScript

```
try
{
    var factory = new ActiveXObject("Biolink.Biometrics2.Factory");

    // Create License object. The object must exist
    // all the time during using BSDK
    var license = factory.CreateLicense();

    //creating the list of devices
    var deviceList = factory.CreateDeviceList();
    //selecting first found device
    var deviceDescriptor = deviceList.DeviceDescriptor(0);
    //creating Scanner object
    var scanner = factory.CreateScanner(deviceDescriptor);

    //acquiring image from the scanner
    var image = scanner.AcquireImage();

    //creating ImageSet object to hold Image objects
    var imageSet = factory.CreateImageSet();
    //adding Image objects to ImageSet with some FingerCode
    imageSet.AddImage(image, 0);

    //creating ImageProcessor object to create template
    //You can specify math type when creating ImageProcessor, Template,
TemplateSet or Matcher objects.
    //The math type should be the same for all these objects.
    var imgPrc = factory.CreateImageProcessor();
    //creating template
    var templ = imgPrc.CreateTemplate(imageSet);
    //saving template to bufferTemplate byte array
    var bufferTemplate = templ.ToArray();

    //loading template from buffer
    var template = factory.CreateTemplate();
    template.Load(bufferTemplate);
}
catch(e)
{
    WScript.Echo(e.description);
}
finally
{
    if (template != null)
        template.Dispose();
    if (templ != null)
        templ.Dispose();
    if (imgPrc != null)
        imgPrc.Dispose();
```

```
    if (imageSet != null)
        imageSet.Dispose();
    if (image != null)
        image.Dispose();;
    if (scanner != null)
        scanner.Dispose();
    if (deviceDescriptor != null)
        deviceDescriptor.Dispose();
    if (deviceList != null)
        deviceList.Dispose();
    if (license != null)
        license.Dispose();
    delete factory;
```

Delphi

```
unit BsdkDll;

interface

const
  BSDK_RESULT_BASE           = 0;
  BSDK_ERROR_INVALID_LICENSE = BSDK_RESULT_BASE - 12;

const
  BSDK_LICENSE_TYPE_FUTRONIC      = $00000001;
  BSDK_LICENSE_TYPE_SENTINEL      = $00000002;
  BSDK_LICENSE_TYPE_HARDWARE      = $00000004;
  BSDK_LICENSE_TYPE_SENTINEL_SHK  = $00000008;
  BSDK_LICENSE_TYPE_ALL           = $000000FF;

const
  Bsdk_Device_Type_Any                = $0000;
  Bsdk_Device_Type_Scanner_Any        = $0100;
  Bsdk_Device_Type_Scanner_Umatch     = $0101;
  Bsdk_Device_Type_Scanner_Upek       = $0102;
  Bsdk_Device_Type_Scanner_CrossMatch = $0103;

  function Bsdk_Destroy_Object(pObject : pointer): integer; stdcall; external
'bsdk6x.dll'

  function Bsdk_License_Create(tLicenseTypes : integer): integer; stdcall;
external 'bsdk6x.dll'
  function Bsdk_License_Close(): Integer; stdcall; external 'bsdk6x.dll'

  function Bsdk_DeviceList_Create(var pDeviceList : pointer; tDeviceType:
integer): integer; stdcall; external 'bsdk6x.dll'
  function Bsdk_DeviceList_GetDeviceDescriptor(pDeviceList : pointer;
iDeviceListIndex: integer): pointer; stdcall; external 'bsdk6x.dll'
  function Bsdk_Scanner_Create(var pScanner : pointer; pDeviceDescriptorPtr:
pointer): integer; stdcall; external 'bsdk6x.dll'
  function Bsdk_Scanner_AcquireImage(pScanner : pointer; var pImage: pointer):
integer; stdcall; external 'bsdk6x.dll'

  function Bsdk_ImageSet_Create(var pImageSet : pointer): Integer; stdcall;
external 'bsdk6x.dll'
  function Bsdk_ImageSet_AddImage(pImageSet : pointer; pImage : pointer;
iFinger : integer): integer; stdcall; external 'bsdk6x.dll'

  function Bsdk_ImageProcessor_Create(var pProcessor : pointer; mathType:
integer): integer; stdcall; external 'bsdk6x.dll'
  function Bsdk_ImageProcessor_CreateTemplateFromImageSet(pProcessor :
pointer; pImageSet: pointer; var pTemplate : pointer): integer; stdcall;
external 'bsdk6x.dll'

  function Bsdk_Template_Create(var pTemplate : pointer; mathType: integer):
integer; stdcall; external 'bsdk6x.dll'
```

```
  function Bsdk_Template_Load(pTemplate : pointer; pStreamPos : pointer;
tStreamSize : integer; var pTransfered : integer): integer; stdcall; external
'bsdk6x.dll'
  function Bsdk_Template_GetSize(pTemplate : pointer) : integer; stdcall;
external 'bsdk6x.dll'
  function Bsdk_Template_Save(pTemplate : pointer; pStreamPos : pointer;
tStreamSize : integer; var pTransfered : integer): integer; stdcall; external
'bsdk6x.dll'

  procedure CheckResult(result : integer);

implementation

uses SysUtils;

  procedure CheckResult(result : Integer);
  begin
      if (result <> BSDK_RESULT_BASE) then begin
          if  (result =  BSDK_ERROR_INVALID_LICENSE) then
              raise Exception.Create('No BSDK License');

          raise Exception.Create('BSDK Exception! Error: ' +
intToStr(result));
      end;
  end;

end.



program Samples;

{$APPTYPE CONSOLE}

uses
  SysUtils,
  BsdkDll;

var result : integer;
    pDeviceList : pointer = nil;
    pDeviceDescriptorPtr : pointer = nil;
    pScanner : pointer = nil;
    pImage : pointer = nil;
    pImageSet : pointer = nil;
    pImageProcessor : pointer = nil;
    bufferTemplate : PChar;
    templateLength, pTransfered : integer;
    pTempl : pointer = nil;
    pTemplate : pointer = nil;

begin
    try
       try
            //creating License object
            result := Bsdk_License_Create(BSDK_LICENSE_TYPE_ALL);
            CheckResult(result);

            //creating the list of devices
            result := Bsdk_DeviceList_Create(pDeviceList,
Bsdk_Device_Type_Scanner_Any);
            CheckResult(result);

            //selecting first found device
            pDeviceDescriptorPtr :=
Bsdk_DeviceList_GetDeviceDescriptor(pDeviceList, 0);
            //creating Scanner object
            result := Bsdk_Scanner_Create(pScanner, pDeviceDescriptorPtr);
            CheckResult(result);
```

```
            //acquiring image from the scanner
            result := Bsdk_Scanner_AcquireImage(pScanner, pImage);
            CheckResult(result);

            //creating ImageSet object to hold Image objects
            result := Bsdk_ImageSet_Create(pImageSet);
            CheckResult(result);

            //adding Image objects to ImageSet with some FingerCode
            result := Bsdk_ImageSet_AddImage(pImageSet, pImage, 0);
            CheckResult(result);

            //creating ImageProcessor object to create template
            //You can specify math type when creating ImageProcessor,
Template, TemplateSet or Matcher objects.
            //The math type should be the same for all these objects.
            result := Bsdk_ImageProcessor_Create(pImageProcessor, 0);
            CheckResult(result);

            //creating template
            result :=
Bsdk_ImageProcessor_CreateTemplateFromImageSet(pImageProcessor, pImageSet,
pTempl);
            CheckResult(result);

            //saving template to bufferTemplate byte array
            templateLength := Bsdk_Template_GetSize(pTempl);
            GetMem(bufferTemplate, templateLength);
            result := Bsdk_Template_Save(pTempl, bufferTemplate,
templateLength, pTransfered);
            CheckResult(result);

            //loading template from buffer
            result := Bsdk_Template_Create(pTemplate, 0);
            CheckResult(result);
            result := Bsdk_Template_Load(pTemplate, bufferTemplate,
templateLength, pTransfered);
            CheckResult(result);

            Bsdk_License_Close();
        except
            On E : Exception do
                Writeln(E.Message);
        end;
    finally
        if (pTemplate <> nil) then
            Bsdk_Destroy_Object(pTemplate);
        if (pTempl <> nil) then
            Bsdk_Destroy_Object(pTempl);
        FreeMem(bufferTemplate);
        if (pImageProcessor <> nil) then
            Bsdk_Destroy_Object(pImageProcessor);
        if (pImageSet <> nil) then
            Bsdk_Destroy_Object(pImageSet);
        if (pImage <> nil) then
            Bsdk_Destroy_Object(pImage);
        if (pScanner <> nil) then
            Bsdk_Destroy_Object(pScanner);
        if (pDeviceDescriptorPtr <> nil) then
            Bsdk_Destroy_Object(pDeviceDescriptorPtr);
        if (pDeviceList <> nil) then
            Bsdk_Destroy_Object(pDeviceList);
    end;
end.
```

How to compare two templates

To compare two templates you should create an instance of a **Matcher** object (for a refresher of how to create a fingerprint template, see the ***How to create, save and load template*** topic on p. 25). The **Matcher.Compare** method returns integer value representing a *threshold* between two templates.

However it is recommended to use **Matcher.Identify** method to compare templates when operating in large databases for it utilizes all available processors cores to speed-up compare process. For more information see the ***How to use TemplateSet*** topic on p. 41.

The following code fragments illustrates how to compare two templates.

C++

```
#include <Bsdk2.h>
#include <Bscan.h>

int main( int argc, char* argv[] )
{
 //creating license
 Bsdk_License_Create( BSDK_LICENSE_TYPE_ALL );

 //creating the list of devices
 Bsdk_DeviceListPtr deviceList;
 Bsdk_DeviceList_Create( &deviceList, Bsdk_Device_Type_Any );

 //selecting first found device
 Bsdk_DeviceDescriptorPtr deviceDescriptor =
Bsdk_DeviceList_GetDeviceDescriptor( deviceList, 0 );

 //creating Scanner object
 Bsdk_ScannerPtr scanner;
 Bsdk_Scanner_Create( &scanner, deviceDescriptor );

 //acquiring images from the scanner
 Bsdk_ImagePtr image1;
 Bsdk_Scanner_AcquireImage( scanner, &image1 );

 Bsdk_ImagePtr image2;
 Bsdk_Scanner_AcquireImage( scanner, &image2 );

 //creating ImageSet objects to hold Image objects
 Bsdk_ImageSetPtr imageSet1;
 Bsdk_ImageSet_Create( &imageSet1 );

 Bsdk_ImageSetPtr imageSet2;
 Bsdk_ImageSet_Create( &imageSet2 );

 //adding Image objects to ImageSet objects with some finger index
 Bsdk_ImageSet_AddImage( imageSet1, image1, Bsdk_Finger_Index_UNKNOWN_FINGER
);
 Bsdk_ImageSet_AddImage( imageSet2, image2, Bsdk_Finger_Index_UNKNOWN_FINGER
);

 //creating ImageProcessor object to create template
 /* You can specify math type when creating ImageProcessor, Template or
Matcher objects.
  * The math type should be the same for all these objects.
  */
 Bsdk_ImageProcessorPtr imageProcessor;
 Bsdk_ImageProcessor_Create( &imageProcessor, Bsdk_Math_Type_Biolink );
```

```
 //creating template
 Bsdk_TemplatePtr template1;
 Bsdk_ImageProcessor_CreateTemplateFromImageSet( imageProcessor, imageSet1,
&template1 );

 Bsdk_TemplatePtr template2;
 Bsdk_ImageProcessor_CreateTemplateFromImageSet( imageProcessor, imageSet2,
&template2 );

 //creating matcher
 Bsdk_MatcherPtr matcher;
 Bsdk_Matcher_Create( &matcher, Bsdk_Math_Type_Biolink, 0 );

 //matching template1 with template2
 int score;
 Bsdk_Matcher_Compare( matcher, template1, template2, &score );

 //destroying objects
 Bsdk_Destroy_Object( deviceList );
 Bsdk_Destroy_Object( scanner );
 Bsdk_Destroy_Object( image1 );
 Bsdk_Destroy_Object( image2 );
 Bsdk_Destroy_Object( imageSet1 );
 Bsdk_Destroy_Object( imageSet2 );
 Bsdk_Destroy_Object( imageProcessor );
 Bsdk_Destroy_Object( template1 );
 Bsdk_Destroy_Object( template2 );
 Bsdk_Destroy_Object( matcher );

 //destoying license
 Bsdk_License_Close();

 return 0;
```

}

C#

```
using System;
using Biolink.Biometrics2;

namespace Sample
{
    class Class1
    {
        static void Main(string[] args)
        {
            try
            {
                // Create License object. The object must exist
                // all the time during using BSDK
                using (License license = new License())
                {
                    //creating the list of devices
                    using (DeviceList deviceList = new DeviceList())
                    {
                        //selecting first found device
                        using (DeviceDescriptor deviceDescriptor =
deviceList.DeviceDescriptor(0))
                        {
                            //creating Scanner object
                            using (Scanner scanner = new
Scanner(deviceDescriptor))
                            {
                                //acquiring image from the scanner
                                Image image1 = scanner.AcquireImage();
```

```
                                Image image2 = scanner.AcquireImage();

                                //creating ImageSet object to hold Image
objects
                                using (ImageSet imageSet1 = new ImageSet())
                                {
                                    //adding Image objects to ImageSet with
some FingerCode
                                    imageSet1.AddImage(image1, 0);

                                    using (ImageSet imageSet2 = new
ImageSet())
                                    {
                                        imageSet2.AddImage(image2, 0);

                                        //creating ImageProcessor object to
create template
                                        /* You can specify math type when
creating ImageProcessor, Template, TemplateSet or Matcher objects.
                                         * The math type should be the same
for all these objects.
                                         */
                                        using (ImageProcessor imgPrc = new
ImageProcessor())
                                        {
                                            //creating templates
                                            Template template1 =
imgPrc.CreateTemplate(imageSet1);
                                            Template template2 =
imgPrc.CreateTemplate(imageSet2);

                                            //matching template1 with
template2
                                            using (Matcher matcher = new
Matcher())
                                            {
                                                int score =
matcher.Compare(template1, template2);
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine("Error: {0}", ex.Message);
            }
        }
    }
}
```

Java

```
package Sample;

import org.eclipse.swt.*;
import bsdk6x.*;

public class Class1
{
 public static void main(String[] args)
 {
        License license = null;
        DeviceList deviceList = null;
```

```
        DeviceDescriptor deviceDescriptor = null;
        Scanner scanner = null;
        Image image1 = null;
        Image image2 = null;
        ImageSet imageSet1 = null;
        ImageSet imageSet2 = null;
        ImageProcessor imgPrc = null;
            Template template1 = null;
            Template template2 = null;
            Matcher matcher = null;
        try
        {
               // Create License object. The object must exist
                  // all the time during BSDK usage
               license = new License();

               //creating the list of devices
               deviceList = new DeviceList();
               //selecting first found device
               deviceDescriptor = deviceList.getDeviceDescriptor(0);
               //creating Scanner object
               scanner = new Scanner(deviceDescriptor);

               //acquiring image from the scanner
               image1 = scanner.acquireImage();
               image2 = scanner.acquireImage();

               //creating ImageSet object to hold Image objects
               imageSet1 = new ImageSet();
               imageSet2 = new ImageSet();
               //adding Image objects to ImageSet with some FingerCode
            imageSet1.addImage(image1, FingerCode.Unknown);
            imageSet2.addImage(image2, FingerCode.Unknown);

            //creating ImageProcessor object to create template
            /* You can specify math type when creating ImageProcessor,
Template, TemplateSet or Matcher objects.
             * The math type should be the same for all these objects.
             */
            imgPrc = new ImageProcessor();
            //creating templates
            template1 = imgPrc.createTemplate(imageSet1);
            template2 = imgPrc.createTemplate(imageSet2);

            //matching template1 with template2
               matcher = new Matcher();
               int score = matcher.compare(template1, template2);

        }
        catch (Exception ex)
        {
               System.out.println(ex.getMessage());
        }
        finally
        {
               if (matcher != null)
                      matcher.dispose();
               if (template2 != null)
                      template2.dispose();
               if (template1 != null)
                      template1.dispose();
               if (imgPrc != null)
                      imgPrc.dispose();
               if (imageSet2 != null)
                      imageSet2.dispose();
               if (imageSet1 != null)
                      imageSet1.dispose();
```

```
                if (image2 != null)
                        image2.dispose();
                if (image1 != null)
                        image1.dispose();
                if (scanner != null)
                        scanner.dispose();
                if (deviceDescriptor != null)
                        deviceDescriptor.dispose();
                if (deviceList != null)
                        deviceList.dispose();
                if (license != null)
                        license.dispose();
        }
 }
}
```

JScript

```
try
{
    var factory = new ActiveXObject("Biolink.Biometrics2.Factory");

    // Create License object. The object must exist
    // all the time during using BSDK
    var license = factory.CreateLicense();

    //creating the list of devices
    var deviceList = factory.CreateDeviceList();
    //selecting first found device
    var deviceDescriptor = deviceList.DeviceDescriptor(0);
    //creating Scanner object
    var scanner = factory.CreateScanner(deviceDescriptor);

    //acquiring image from the scanner
    var image1 = scanner.AcquireImage();
    var image2 = scanner.AcquireImage();

    //creating ImageSet object to hold Image objects
    var imageSet1 = factory.CreateImageSet();
    var imageSet2 = factory.CreateImageSet();
    //adding Image objects to ImageSet with some FingerCode
    imageSet1.AddImage(image1, 0);
    imageSet2.AddImage(image2, 0);

    //creating ImageProcessor object to create template
    //You can specify math type when creating ImageProcessor, Template,
TemplateSet or Matcher objects.
    //The math type should be the same for all these objects.
    var imgPrc = factory.CreateImageProcessor();
    //creating templates
    var template1 = imgPrc.CreateTemplate(imageSet1);
    var template2 = imgPrc.CreateTemplate(imageSet2);

    //matching template1 with template2
    var matcher = factory.CreateMatcher();
    var score = matcher.Compare(template1, template2);
}
catch(e)
{
    WScript.Echo(e.description);
}
finally
{
    if (matcher != null)
        matcher.Dispose();
    if (template2 != null)
        template2.Dispose();
    if (template1 != null)
```

```
        template1.Dispose();
    if (imgPrc != null)
        imgPrc.Dispose();
    if (imageSet1 != null)
        imageSet1.Dispose();
    if (imageSet2 != null)
        iimageSet2.Dispose();
    if (image2 != null)
        image2.Dispose();
    if (image1 != null)
        image1.Dispose();
    if (scanner != null)
        scanner.Dispose();
    if (deviceDescriptor != null)
        deviceDescriptor.Dispose();
    if (deviceList != null)
        deviceList.Dispose();
    if (license != null)
        license.Dispose();
    delete factory;
}
```

Delphi

```
unit BsdkDll;

interface

const
  BSDK_RESULT_BASE           = 0;
  BSDK_ERROR_INVALID_LICENSE = BSDK_RESULT_BASE - 12;

const
  BSDK_LICENSE_TYPE_FUTRONIC      = $00000001;
  BSDK_LICENSE_TYPE_SENTINEL      = $00000002;
  BSDK_LICENSE_TYPE_HARDWARE      = $00000004;
  BSDK_LICENSE_TYPE_SENTINEL_SHK  = $00000008;
  BSDK_LICENSE_TYPE_ALL           = $000000FF;

const
  Bsdk_Device_Type_Any                = $0000;
  Bsdk_Device_Type_Scanner_Any        = $0100;
  Bsdk_Device_Type_Scanner_Umatch     = $0101;
  Bsdk_Device_Type_Scanner_Upek       = $0102;
  Bsdk_Device_Type_Scanner_CrossMatch = $0103;

  function Bsdk_Destroy_Object(pObject : pointer): integer; stdcall; external
'bsdk6x.dll'

  function Bsdk_License_Create(tLicenseTypes : integer): integer; stdcall;
external 'bsdk6x.dll'
  function Bsdk_License_Close(): Integer; stdcall; external 'bsdk6x.dll'

  function Bsdk_DeviceList_Create(var pDeviceList : pointer; tDeviceType:
integer): integer; stdcall; external 'bsdk6x.dll'
  function Bsdk_DeviceList_GetDeviceDescriptor(pDeviceList : pointer;
iDeviceListIndex: integer): pointer; stdcall; external 'bsdk6x.dll'
  function Bsdk_Scanner_Create(var pScanner : pointer; pDeviceDescriptorPtr:
pointer): integer; stdcall; external 'bsdk6x.dll'
  function Bsdk_Scanner_AcquireImage(pScanner : pointer; var pImage: pointer):
integer; stdcall; external 'bsdk6x.dll'

  function Bsdk_ImageSet_Create(var pImageSet : pointer): Integer; stdcall;
external 'bsdk6x.dll'
  function Bsdk_ImageSet_AddImage(pImageSet : pointer; pImage : pointer;
iFinger : integer): integer; stdcall; external 'bsdk6x.dll'
  procedure Bsdk_ImageSet_Clear(pImageSet : pointer); stdcall; external
'bsdk6x.dll'
```

```
  function Bsdk_ImageProcessor_Create(var pProcessor : pointer; mathType:
integer): integer; stdcall; external 'bsdk6x.dll'
  function Bsdk_ImageProcessor_CreateTemplateFromImageSet(pProcessor :
pointer; pImageSet: pointer; var pTemplate : pointer): integer; stdcall;
external 'bsdk6x.dll'

  function Bsdk_Matcher_Create(var pMatcher : pointer; mathType: integer;
threadCount : integer): integer; stdcall; external 'bsdk6x.dll'
  function Bsdk_Matcher_Compare(pMatcher : pointer; pTemplate1: pointer;
pTemplate2 : pointer; var score : integer): integer; stdcall; external
'bsdk6x.dll'

  procedure CheckResult(result : integer);

implementation

uses SysUtils;

  procedure CheckResult(result : Integer);
  begin
      if (result <> BSDK_RESULT_BASE) then begin
          if  (result =  BSDK_ERROR_INVALID_LICENSE) then
              raise Exception.Create('No BSDK License');

          raise Exception.Create('BSDK Exception! Error: ' +
intToStr(result));
      end;
  end;

end.



program Samples;

{$APPTYPE CONSOLE}

uses
  SysUtils,
  BsdkDll;

var result,
    score : integer;
    pDeviceList : pointer = nil;
    pDeviceDescriptorPtr : pointer = nil;
    pScanner : pointer = nil;
    pImage1 : pointer = nil;
    pImage2 : pointer = nil;
    pImageSet : pointer = nil;
    pImageProcessor : pointer = nil;
    pTemplate1 : pointer = nil;
    pTemplate2 : pointer = nil;
    pMatcher : pointer = nil;

begin
    try
       try
            //creating License object
            result := Bsdk_License_Create(BSDK_LICENSE_TYPE_ALL);
            CheckResult(result);

            //creating the list of devices
            result := Bsdk_DeviceList_Create(pDeviceList,
Bsdk_Device_Type_Scanner_Any);
            CheckResult(result);

            //selecting first found device
```

```
            pDeviceDescriptorPtr :=
Bsdk_DeviceList_GetDeviceDescriptor(pDeviceList, 0);
            //creating Scanner object
            result := Bsdk_Scanner_Create(pScanner, pDeviceDescriptorPtr);
            CheckResult(result);

            //acquiring image from the scanner
            result := Bsdk_Scanner_AcquireImage(pScanner, pImage1);
            CheckResult(result);
            result := Bsdk_Scanner_AcquireImage(pScanner, pImage2);
            CheckResult(result);

            //creating ImageSet object to hold Image objects
            result := Bsdk_ImageSet_Create(pImageSet);
            CheckResult(result);
            //adding Image objects to ImageSet with some FingerCode
            result := Bsdk_ImageSet_AddImage(pImageSet, pImage1, 0);
            CheckResult(result);

            //creating ImageProcessor object to create template
            //You can specify math type when creating ImageProcessor,
Template, TemplateSet or Matcher objects.
            //The math type should be the same for all these objects.
            result := Bsdk_ImageProcessor_Create(pImageProcessor, 0);
            CheckResult(result);

            //creating template
            result :=
Bsdk_ImageProcessor_CreateTemplateFromImageSet(pImageProcessor, pImageSet,
pTemplate1);
            CheckResult(result);

            Bsdk_ImageSet_Clear(pImageSet);
            result := Bsdk_ImageSet_AddImage(pImageSet, pImage2, 0);
            CheckResult(result);
            result :=
Bsdk_ImageProcessor_CreateTemplateFromImageSet(pImageProcessor, pImageSet,
pTemplate2);
            CheckResult(result);

            //matching template1 with template2
            result := Bsdk_Matcher_Create(pMatcher, 0, 0);
            CheckResult(result);
            result := Bsdk_Matcher_Compare(pMatcher, pTemplate1, pTemplate2,
score);
            CheckResult(result);

            Bsdk_License_Close();
        except
            On E : Exception do
                Writeln(E.Message);
        end;
    finally
        if (pMatcher <> nil) then
            Bsdk_Destroy_Object(pMatcher);
        if (pTemplate2 <> nil) then
            Bsdk_Destroy_Object(pTemplate2);
        if (pTemplate1 <> nil) then
            Bsdk_Destroy_Object(pTemplate1);
        if (pImageProcessor <> nil) then
            Bsdk_Destroy_Object(pImageProcessor);
        if (pImageSet <> nil) then
            Bsdk_Destroy_Object(pImageSet);
        if (pImage2 <> nil) then
            Bsdk_Destroy_Object(pImage2);
        if (pImage1 <> nil) then
            Bsdk_Destroy_Object(pImage1);
```

```
        if (pScanner <> nil) then
            Bsdk_Destroy_Object(pScanner);
        if (pDeviceDescriptorPtr <> nil) then
            Bsdk_Destroy_Object(pDeviceDescriptorPtr);
        if (pDeviceList <> nil) then
            Bsdk_Destroy_Object(pDeviceList);
    end;
end.
```

How to use TemplateSet

A convenient way of performing identification (1-to-N matching) is provided by BioLink SDK's **TemplateSet** object and **Matcher.Identify** method.

TemplateSet object represents a set of enrolled **Templates** among which identification will be performed. Building a set of **Templates** consists of creating **TemplateSet** object and adding **Templates** by invoking the **TemplateSet.Add** method. Parameters passed to **Add** are the **Template** itself , an identification number, used to uniquely identify a Template in a set and to associate it with an entry in the external storage like users' database and a string value, which can not be unique.

Matcher.Identify method searches the **TemplateSet** for a match, actually, a set of matches. **Identify** takes the following parameters as input: the fingerprint template to search for, **TemplateSet** where to search, match score threshold, and the maximum number of matches to return (topN) (if the last parameter is omitted all matches, which score is higher than the provided threshold, will be returned). Once called, **Identify** will compare the searched template with each template in a set forming a list of template identification numbers and match scores. This list is then sorted in descending order, match scores less than threshold are dropped, and topN records of the list are returned as an **IdentifyInfoSet** object.

IdentifyInfoSet object contains a collection of **IdentifyInfo** objects. A particular **IdentifyInfo** object within **IdentifyInfoSet** can be accessed by **IndentifyInfoSet.GetItem** method passing it's index as a parameter. Each **IdentifyInfo** object returns information about a fingerprint template in a **TemplateSet** and the result of comparing it with a fingerprint template being identified.

The main advantage of the **Matcher.Identify** method lies in its ability to employ all available CPU cores to perform 1-to-N matching in large databases. That is why it is not recommended to use **Matcher.Compare** method because it utilizes only one core and can be a bottleneck to overall system performance.

The following code fragments illustrates how to use TemplateSet.

C++

```
#include <Bsdk2.h>
#include <Bscan.h>

int main( int argc, char* argv[] )
{
 //creating license
 Bsdk_License_Create( BSDK_LICENSE_TYPE_ALL );
```

```
 //creating the list of devices
 Bsdk_DeviceListPtr deviceList;
 Bsdk_DeviceList_Create( &deviceList, Bsdk_Device_Type_Any );

 //selecting first found device
 Bsdk_DeviceDescriptorPtr deviceDescriptor =
Bsdk_DeviceList_GetDeviceDescriptor( deviceList, 0 );

 //creating Scanner object
 Bsdk_ScannerPtr scanner;
 Bsdk_Scanner_Create( &scanner, deviceDescriptor );

 //acquiring images from the scanner
 Bsdk_ImagePtr image1;
 Bsdk_Scanner_AcquireImage( scanner, &image1 );

 Bsdk_ImagePtr image2;
 Bsdk_Scanner_AcquireImage( scanner, &image2 );

 //creating ImageSet objects to hold Image objects
 Bsdk_ImageSetPtr imageSet1;
 Bsdk_ImageSet_Create( &imageSet1 );

 Bsdk_ImageSetPtr imageSet2;
 Bsdk_ImageSet_Create( &imageSet2 );

 //adding Image objects to ImageSet objects with some finger index
 Bsdk_ImageSet_AddImage( imageSet1, image1, Bsdk_Finger_Index_UNKNOWN_FINGER
);
 Bsdk_ImageSet_AddImage( imageSet2, image2, Bsdk_Finger_Index_UNKNOWN_FINGER
);

 //creating ImageProcessor object to create template
 /* You can specify math type when creating ImageProcessor, Template or
Matcher objects.
  * The math type should be the same for all these objects.
  */
 Bsdk_ImageProcessorPtr imageProcessor;
 Bsdk_ImageProcessor_Create( &imageProcessor, Bsdk_Math_Type_Biolink );

 //creating template
 Bsdk_TemplatePtr template1;
 Bsdk_ImageProcessor_CreateTemplateFromImageSet( imageProcessor, imageSet1,
&template1 );

 Bsdk_TemplatePtr template2;
 Bsdk_ImageProcessor_CreateTemplateFromImageSet( imageProcessor, imageSet2,
&template2 );

 //creating TemplateSet object to hold all templates for identification
 Bsdk_TemplateSetPtr templateSet;
 Bsdk_TemplateSet_Create( &templateSet );

 //adding templates to the templateSet
 Bsdk_TemplateSet_Add( templateSet, template1, 1, "template1" );
 Bsdk_TemplateSet_Add( templateSet, template2, 2, "template2" );

 //creating matcher
 Bsdk_MatcherPtr matcher;
 Bsdk_Matcher_Create( &matcher, Bsdk_Math_Type_Biolink, 0 );

 //matching template2 with templates in templateSet with threshold equal to
600
 //creating IdentifyInfoSet class holding a set of IdentifyInfo objects
returned by Identify method
 //IdentifyInfo objects are sorted by score in descending order
 Bsdk_IdentifyInfoSetPtr infoSet;
```

```
 Bsdk_Matcher_Identify( matcher, template2, templateSet, 600, &infoSet );

 int count = Bsdk_IdentifyInfoSet_GetSize( infoSet );

 for( int i = 0; i < count; i++ )
 {
       //IdentifyInfo object used to view matching score...
       Bsdk_IdentifyInfoPtr info = Bsdk_IdentifyInfoSet_GetItem( infoSet, i
);

       int id = Bsdk_IdentifyInfo_GetId( info );
       char* number = Bsdk_IdentifyInfo_GetNumber( info );
       int score = Bsdk_IdentifyInfo_GetScore( info );
 }

 //destroying objects
 Bsdk_Destroy_Object( deviceList );
 Bsdk_Destroy_Object( scanner );
 Bsdk_Destroy_Object( image1 );
 Bsdk_Destroy_Object( image2 );
 Bsdk_Destroy_Object( imageSet1 );
 Bsdk_Destroy_Object( imageSet2 );
 Bsdk_Destroy_Object( imageProcessor );
 Bsdk_Destroy_Object( template1 );
 Bsdk_Destroy_Object( template2 );
 Bsdk_Destroy_Object( templateSet );
 Bsdk_Destroy_Object( matcher );
 Bsdk_Destroy_Object( infoSet );

 //destoying license
 Bsdk_License_Close();

 return 0;
}
```

C#

```
using System;
using Biolink.Biometrics2;

namespace Sample
{
    class Class1
    {
        static void Main(string[] args)
        {
            try
            {
                // Create License object. The object must exist
                // all the time during using BSDK
                using (License license = new License())
                {
                    //creating the list of devices
                    using (DeviceList deviceList = new DeviceList())
                    {
                        //selecting first found device
                        using (DeviceDescriptor deviceDescriptor =
deviceList.DeviceDescriptor(0))
                        {
                            //creating Scanner object
                            using (Scanner scanner = new
Scanner(deviceDescriptor))
                            {
                                //acquiring image from the scanner
                                Image image1 = scanner.AcquireImage();
                                Image image2 = scanner.AcquireImage();

                                Template template1;
```

```
                                Template template2;

                                //creating ImageSet object to hold Image
objects
                                using (ImageSet imageSet1 = new ImageSet())
                                {
                                    //adding Image objects to ImageSet with
some FingerCode
                                    imageSet1.AddImage(image1, 0);

                                    using (ImageSet imageSet2 = new
ImageSet())
                                    {
                                        imageSet2.AddImage(image2, 0);

                                        //creating ImageProcessor object to
create template
                                        /* You can specify math type when
creating ImageProcessor, Template, TemplateSet or Matcher objects.
                                         * The math type should be the same
for all these objects.
                                         */
                                        using (ImageProcessor imgPrc = new
ImageProcessor())
                                        {
                                            //creating templates
                                            template1 =
imgPrc.CreateTemplate(imageSet1);
                                            template2 =
imgPrc.CreateTemplate(imageSet2);
                                        }
                                    }
                                }

                                //creating TemplateSet object to hold all
templates for identification
                                using (TemplateSet templateSet = new
TemplateSet())
                                {
                                    //adding templates to the templateSet
                                    templateSet.AddTemplate(template1, 1,
"template1");
                                    templateSet.AddTemplate(template2, 2,
"template2");

                                    //matching template2 with templates in
templateSet with threshold equal to 600
                                    using (Matcher matcher = new Matcher())
                                    {
                                        //creating IdentifyInfoSet class
holding a set of IdentifyInfo objects returned by Identify method
                                        //IdentifyInfo objects are sorted by
score in descending order
                                        using (IdentifyInfoSet infoSet =
matcher.Identify(template2, templateSet, 600))
                                        {
                                            for (int i = 0; i < infoSet.Size;
i++)
                                            {
                                                //IdentifyInfo object used to
view matching score...
                                                using (IdentifyInfo
identifyInfo = infoSet.GetItem(i))
                                                {

Console.WriteLine("Template id {0}, number {1}, score {2}",
```

```
                                                                    identifyInfo.Id,
identifyInfo.Number, identifyInfo.Score);
                                                        }
                                                    }
                                                }
                                            }
                                        }
                                    }
                                }
                            }
                        }
                        catch (Exception ex)
                        {
                            Console.WriteLine("Error: {0}", ex.Message);
                        }
                }
            }
}
```

Java

```
package Sample;

import org.eclipse.swt.*;
import bsdk6x.*;

public class Class1
{
 public static void main(String[] args)
 {
        License license = null;
        DeviceList deviceList = null;
        DeviceDescriptor deviceDescriptor = null;
        Scanner scanner = null;
        Image image1 = null;
        Image image2 = null;
        ImageSet imageSet1 = null;
        ImageSet imageSet2 = null;
        ImageProcessor imgPrc = null;
            Template template1 = null;
            Template template2 = null;
            TemplateSet templateSet = null;
        Matcher matcher = null;
        IdentifyInfoSet infoSet = null;
        IdentifyInfo identifyInfo = null;
        try
        {
               // Create License object. The object must exist
                   // all the time during BSDK usage
               license = new License();

               //creating the list of devices
               deviceList = new DeviceList();
               //selecting first found device
               deviceDescriptor = deviceList.getDeviceDescriptor(0);
               //creating Scanner object
               scanner = new Scanner(deviceDescriptor);

               //acquiring image from the scanner
               image1 = scanner.acquireImage();
               image2 = scanner.acquireImage();

               //creating ImageSet object to hold Image objects
               imageSet1 = new ImageSet();
               imageSet2 = new ImageSet();
```

```
              //adding Image objects to ImageSet with some FingerCode
            imageSet1.addImage(image1, FingerCode.Unknown);
            imageSet2.addImage(image2, FingerCode.Unknown);

            //creating ImageProcessor object to create template
            /* You can specify math type when creating ImageProcessor,
Template, TemplateSet or Matcher objects.
             * The math type should be the same for all these objects.
             */
            imgPrc = new ImageProcessor();
            //creating templates
            template1 = imgPrc.createTemplate(imageSet1);
            template2 = imgPrc.createTemplate(imageSet2);

            //creating TemplateSet object to hold all templates for
identification
            templateSet = new TemplateSet();
            //adding templates to the templateSet
            templateSet.addTemplate(template1, 1, "template1");
            templateSet.addTemplate(template2, 2, "template2");

            //matching template2 with templates in templateSet with threshold
equal to 600
            matcher = new Matcher();
            //creating IdentifyInfoSet class holding a set of IdentifyInfo
objects returned by Identify method
            //IdentifyInfo objects are sorted by score in descending order
            infoSet = matcher.identify(template2, templateSet, 600);
            for (int i = 0; i < infoSet.getSize(); i++)
            {
                     if (identifyInfo != null)
                            identifyInfo.dispose();

                     //IdentifyInfo object used to view matching score...
               identifyInfo = infoSet.getItem(i);

               System.out.println("Template id " + identifyInfo.getId() +
                                                  ", number " +
identifyInfo.getNumber() +
                                                  ", score " +
identifyInfo.getScore());
             }
        }
        catch (Exception ex)
        {
               System.out.println(ex.getMessage());
        }
        finally
        {
               if (identifyInfo != null)
                      identifyInfo.dispose();
               if (infoSet != null)
                      infoSet.dispose();
               if (matcher != null)
                      matcher.dispose();
               if (templateSet != null)
                      templateSet.dispose();
               if (template2 != null)
                      template2.dispose();
               if (template1 != null)
                      template1.dispose();
               if (imgPrc != null)
                      imgPrc.dispose();
               if (imageSet2 != null)
                      imageSet2.dispose();
               if (imageSet1 != null)
                      imageSet1.dispose();
```

```
                if (image2 != null)
                        image2.dispose();
                if (image1 != null)
                        image1.dispose();
                if (scanner != null)
                        scanner.dispose();
                if (deviceDescriptor != null)
                        deviceDescriptor.dispose();
                if (deviceList != null)
                        deviceList.dispose();
                if (license != null)
                        license.dispose();
        }
 }
}
```

JScript

```
try
{
    var factory = new ActiveXObject("Biolink.Biometrics2.Factory");

    // Create License object. The object must exist
    // all the time during BSDK usage
    var license = factory.CreateLicense();

    //creating the list of devices
    var deviceList = factory.CreateDeviceList();
    //selecting first found device
    var deviceDescriptor = deviceList.DeviceDescriptor(0);
    //creating Scanner object
    var scanner = factory.CreateScanner(deviceDescriptor);

    //acquiring image from the scanner
    var image1 = scanner.AcquireImage();
    var image2 = scanner.AcquireImage();

    //creating ImageSet object to hold Image objects
    var imageSet1 = factory.CreateImageSet();
    var imageSet2 = factory.CreateImageSet();
    //adding Image objects to ImageSet with some FingerCode
    imageSet1.AddImage(image1, 0);
    imageSet2.AddImage(image2, 0);

    //creating ImageProcessor object to create template
    //You can specify math type when creating ImageProcessor, Template,
TemplateSet or Matcher objects.
    //The math type should be the same for all these objects.
    var imgPrc = factory.CreateImageProcessor();
    //creating templates
    var template1 = imgPrc.CreateTemplate(imageSet1);
    var template2 = imgPrc.CreateTemplate(imageSet2);

    //creating TemplateSet object to hold all templates for identification
    var templateSet = factory.CreateTemplateSet();
    //adding templates to the templateSet
    templateSet.AddTemplate(template1, 1, "template1");
    templateSet.AddTemplate(template2, 2, "template2");

    //matching template2 with templates in templateSet with threshold equal to
600
    var matcher = factory.CreateMatcher();
    //creating IdentifyInfoSet class holding a set of IdentifyInfo objects
returned by Identify method
    //IdentifyInfo objects are sorted by score in descending order
    var infoSet = matcher.Identify(template2, templateSet, 600);

    for (i = 0; i < infoSet.Size; i++)
```

```
    {
        if (identifyInfo != null)
            identifyInfo.Dispose();

        //IdentifyInfo object used to view matching score...
        var identifyInfo = infoSet.GetItem(i);
        WScript.Echo("Template id " + identifyInfo.Id + ", number " +
identifyInfo.Number + ", score " + identifyInfo.Score);
    }
}
catch(e)
{
    WScript.Echo(e.description);
}
finally
{
    if (identifyInfo != null)
        identifyInfo.Dispose();
    if (infoSet != null)
        infoSet.Dispose();
    if (matcher != null)
        matcher.Dispose();
    if (template2 != null)
        template2.Dispose();
    if (template1 != null)
        template1.Dispose();
    if (imgPrc != null)
        imgPrc.Dispose();
    if (imageSet1 != null)
        imageSet1.Dispose();
    if (imageSet2 != null)
        iimageSet2.Dispose();
    if (image2 != null)
        image2.Dispose();
    if (image1 != null)
        image1.Dispose();
    if (scanner != null)
        scanner.Dispose();
    if (deviceDescriptor != null)
        deviceDescriptor.Dispose();
    if (deviceList != null)
        deviceList.Dispose();
    if (license != null)
        license.Dispose();
    delete factory;
}
```

Delphi

```
unit BsdkDll;

interface

const
  BSDK_RESULT_BASE           = 0;
  BSDK_ERROR_INVALID_LICENSE = BSDK_RESULT_BASE - 12;

const
  BSDK_LICENSE_TYPE_FUTRONIC      = $00000001;
  BSDK_LICENSE_TYPE_SENTINEL      = $00000002;
  BSDK_LICENSE_TYPE_HARDWARE      = $00000004;
  BSDK_LICENSE_TYPE_SENTINEL_SHK  = $00000008;
  BSDK_LICENSE_TYPE_ALL           = $000000FF;

const
  Bsdk_Device_Type_Any                = $0000;
  Bsdk_Device_Type_Scanner_Any        = $0100;
  Bsdk_Device_Type_Scanner_Umatch     = $0101;
```

```
  Bsdk_Device_Type_Scanner_Upek       = $0102;
  Bsdk_Device_Type_Scanner_CrossMatch = $0103;

  function Bsdk_Destroy_Object(pObject : pointer): integer; stdcall; external
'bsdk6x.dll'

  function Bsdk_License_Create(tLicenseTypes : integer): integer; stdcall;
external 'bsdk6x.dll'
  function Bsdk_License_Close(): Integer; stdcall; external 'bsdk6x.dll'

  function Bsdk_DeviceList_Create(var pDeviceList : pointer; tDeviceType:
integer): integer; stdcall; external 'bsdk6x.dll'
  function Bsdk_DeviceList_GetDeviceDescriptor(pDeviceList : pointer;
iDeviceListIndex: integer): pointer; stdcall; external 'bsdk6x.dll'
  function Bsdk_Scanner_Create(var pScanner : pointer; pDeviceDescriptorPtr:
pointer): integer; stdcall; external 'bsdk6x.dll'
  function Bsdk_Scanner_AcquireImage(pScanner : pointer; var pImage: pointer):
integer; stdcall; external 'bsdk6x.dll'

  function Bsdk_ImageSet_Create(var pImageSet : pointer): Integer; stdcall;
external 'bsdk6x.dll'
  function Bsdk_ImageSet_AddImage(pImageSet : pointer; pImage : pointer;
iFinger : integer): integer; stdcall; external 'bsdk6x.dll'
  procedure Bsdk_ImageSet_Clear(pImageSet : pointer); stdcall; external
'bsdk6x.dll'

  function Bsdk_ImageProcessor_Create(var pProcessor : pointer; mathType:
integer): integer; stdcall; external 'bsdk6x.dll'
  function Bsdk_ImageProcessor_CreateTemplateFromImageSet(pProcessor :
pointer; pImageSet: pointer; var pTemplate : pointer): integer; stdcall;
external 'bsdk6x.dll'

  function Bsdk_TemplateSet_Create(var pTemplateSet : pointer; mathType:
integer): integer; stdcall; external 'bsdk6x.dll'
  function Bsdk_TemplateSet_Add(pTemplateSet : pointer; templ: pointer; id :
integer; number : string): integer; stdcall; external 'bsdk6x.dll'

  function Bsdk_IdentifyInfo_GetScore(pInfo : pointer): integer; stdcall;
external 'bsdk6x.dll'
  function Bsdk_IdentifyInfo_GetId(pInfo : pointer): integer; stdcall;
external 'bsdk6x.dll'
  function Bsdk_IdentifyInfo_GetNumber(pInfo : pointer): pchar; stdcall;
external 'bsdk6x.dll'

  function Bsdk_IdentifyInfoSet_GetSize(pInfoSet : pointer): integer; stdcall;
external 'bsdk6x.dll'
  function Bsdk_IdentifyInfoSet_GetItem(pInfoSet : pointer; index : integer):
pointer; stdcall; external 'bsdk6x.dll'

  function Bsdk_Matcher_Create(var pMatcher : pointer; mathType: integer;
threadCount : integer): integer; stdcall; external 'bsdk6x.dll'
  function Bsdk_Matcher_Identify(pMatcher : pointer; pTemplate : pointer;
pTemplateSet : pointer; minScore : integer; var pInfoSet : pointer): integer;
stdcall; external 'bsdk6x.dll'

  procedure CheckResult(result : integer);

implementation

uses SysUtils;

  procedure CheckResult(result : Integer);
  begin
      if (result <> BSDK_RESULT_BASE) then begin
          if  (result =  BSDK_ERROR_INVALID_LICENSE) then
              raise Exception.Create('No BSDK License');
```

```
          raise Exception.Create('BSDK Exception! Error: ' +
intToStr(result));
      end;
  end;

end.



program Samples;

{$APPTYPE CONSOLE}

uses
  SysUtils,
  BsdkDll;

var result : integer;
    pDeviceList : pointer = nil;
    pDeviceDescriptorPtr : pointer = nil;
    pScanner : pointer = nil;
    pImage1 : pointer = nil;
    pImage2 : pointer = nil;
    pImageSet : pointer = nil;
    pImageProcessor : pointer = nil;
    pTemplate1 : pointer = nil;
    pTemplate2 : pointer = nil;
    pTemplateSet : pointer = nil;
    pMatcher : pointer = nil;
    pInfoSet : pointer = nil;
    infoSetSize, i,
    id, score : integer;
    pIdentifyInfo : pointer;
    number : string;

begin
    try
       try
            //creating License object
            result := Bsdk_License_Create(BSDK_LICENSE_TYPE_ALL);
            CheckResult(result);

            //creating the list of devices
            result := Bsdk_DeviceList_Create(pDeviceList,
Bsdk_Device_Type_Scanner_Any);
            CheckResult(result);

            //selecting first found device
            pDeviceDescriptorPtr :=
Bsdk_DeviceList_GetDeviceDescriptor(pDeviceList, 0);
            //creating Scanner object
            result := Bsdk_Scanner_Create(pScanner, pDeviceDescriptorPtr);
            CheckResult(result);

            //acquiring image from the scanner
            result := Bsdk_Scanner_AcquireImage(pScanner, pImage1);
            CheckResult(result);
            result := Bsdk_Scanner_AcquireImage(pScanner, pImage2);
            CheckResult(result);

            //creating ImageSet object to hold Image objects
            result := Bsdk_ImageSet_Create(pImageSet);
            CheckResult(result);
            //adding Image objects to ImageSet with some FingerCode
            result := Bsdk_ImageSet_AddImage(pImageSet, pImage1, 0);
            CheckResult(result);
```

```
            //creating ImageProcessor object to create template
            //You can specify math type when creating ImageProcessor,
Template, TemplateSet or Matcher objects.
            //The math type should be the same for all these objects.
            result := Bsdk_ImageProcessor_Create(pImageProcessor, 0);
            CheckResult(result);

            //creating template
            result :=
Bsdk_ImageProcessor_CreateTemplateFromImageSet(pImageProcessor, pImageSet,
pTemplate1);
            CheckResult(result);

            Bsdk_ImageSet_Clear(pImageSet);
            result := Bsdk_ImageSet_AddImage(pImageSet, pImage2, 0);
            CheckResult(result);
            result :=
Bsdk_ImageProcessor_CreateTemplateFromImageSet(pImageProcessor, pImageSet,
pTemplate2);
            CheckResult(result);

            //creating TemplateSet object to hold all templates for
identification
            result := Bsdk_TemplateSet_Create(pTemplateSet, 0);
            CheckResult(result);

            //adding templates to the templateSet
            result := Bsdk_TemplateSet_Add(pTemplateSet, pTemplate1, 1,
'template1');
            CheckResult(result);
            result := Bsdk_TemplateSet_Add(pTemplateSet, pTemplate2, 2,
'template2');
            CheckResult(result);

            //matching template2 with templates in templateSet with threshold
equal to 600
            //matching template1 with template2
            result := Bsdk_Matcher_Create(pMatcher, 0, 0);
            CheckResult(result);
            //creating IdentifyInfoSet class holding a set of IdentifyInfo
objects returned by Identify method
            //IdentifyInfo objects are sorted by score in descending order
            result := Bsdk_Matcher_Identify(pMatcher, pTemplate2,
pTemplateSet, 600, pInfoSet);
            CheckResult(result);

            //IdentifyInfo object used to view matching score...
            infoSetSize := Bsdk_IdentifyInfoSet_GetSize(pInfoSet);
            //IdentifyInfo object used to view matching score...
            pIdentifyInfo := nil;
            for i := 0 to (infoSetSize - 1) do begin
                if (pIdentifyInfo <> nil) then
                    Bsdk_Destroy_Object(pIdentifyInfo);

                pIdentifyInfo := Bsdk_IdentifyInfoSet_GetItem(pInfoSet, i);

                id := Bsdk_IdentifyInfo_GetId(pIdentifyInfo);
                number := Bsdk_IdentifyInfo_GetNumber(pIdentifyInfo);
                score := Bsdk_IdentifyInfo_GetScore(pIdentifyInfo);
                WriteLn('Template id ', id, ' number ', number, ' score ',
score);
            end;

            Bsdk_License_Close();
        except
            On E : Exception do
                Writeln(E.Message);
        end;
```

```
    finally
        if (pIdentifyInfo <> nil) then
            Bsdk_Destroy_Object(pIdentifyInfo);
        if (pInfoSet <> nil) then
            Bsdk_Destroy_Object(pInfoSet);
        if (pMatcher <> nil) then
            Bsdk_Destroy_Object(pMatcher);
        if (pTemplateSet <> nil) then
            Bsdk_Destroy_Object(pTemplateSet);
        if (pTemplate2 <> nil) then
            Bsdk_Destroy_Object(pTemplate2);
        if (pTemplate1 <> nil) then
            Bsdk_Destroy_Object(pTemplate1);
        if (pImageProcessor <> nil) then
            Bsdk_Destroy_Object(pImageProcessor);
        if (pImageSet <> nil) then
            Bsdk_Destroy_Object(pImageSet);
        if (pImage2 <> nil) then
            Bsdk_Destroy_Object(pImage2);
        if (pImage1 <> nil) then
            Bsdk_Destroy_Object(pImage1);
        if (pScanner <> nil) then
            Bsdk_Destroy_Object(pScanner);
        if (pDeviceDescriptorPtr <> nil) then
            Bsdk_Destroy_Object(pDeviceDescriptorPtr);
        if (pDeviceList <> nil) then
            Bsdk_Destroy_Object(pDeviceList);
    end;
end.
```

Chapter 4

BSDK licensing

The core of BioLink SDK is a set of BioLink's proprietary mathematical algorithms, patented and protected by international copyright laws, nevertheless, technical protection to prevent unauthorized use has been implemented.

BioLink SDK license in Windows (see "BSDK licensing in Windows" on p. 53) operating environment can be of the following types:

- embedded in Rainbow Sentinel USB dongle;
- embedded in BioLink U-Match 3.5 scanner;
- software.

BioLink SDK license in Linux (see "BSDK licensing in Linux" on p. 57) environment can be of the following types:

- embedded in BioLink U-Match 3.5 scanner;
- software.

While U-Match 3.5 embedded license only provides the license for the computer it is attached to, Rainbow Sentinel USB dongle is also capable of providing licenses to other computers in the same network. Software license also provides the license for the computer, which hardware identifier was used to generate this license.

In This Chapter

BSDK licensing in Windows

BioLink U-Match 3.5 embedded license

The license embedded in BioLink U-Match v. 3.5 scanner allows running BSDK on a single computer, to which the scanner is attached. Pay attention to the fact, that all required drivers for BioLink U-Match v. 3.5 scanner should be installed.

The U-Match drivers for various operating systems (2K/XP/2003/Vista) can be found here (default installation):

```
Program Files\Biolink\BSDK 6.x\DRIVERS\U-Match3.5
```

Rainbow Sentinel USB dongle license

The license **embedded in Rainbow Sentinel USB dongle** allows you to provide license not only for the computer, which this dongle is attached to, but to other computers in the same network. This means that you do not have to buy as many dongles as required to provide license for all computers, on which you intend to utilize BSDK algorithms.

The Rainbow Sentinel USB dongle embedded license can be of two types:

- ***Local*** (see "Local license" on p. 54)
- ***Network*** (see "Network license" on p. 54)

You will find detailed information about these license types further in the two topics that follow.

Local license

The **Local USB license** allows running BSDK on a single computer. To run BSDK the Sentinel drivers must be installed and USB key must be plugged in USB port.

The Sentinel Drivers can be found here (default installation):

```
Program Files\Biolink\BSDK 6.x\drivers\Sentinel\SETUP\Sentinel
System Driver 5.42.0 (32-bit).msi
```

Network license

Network license allows to run multiple BSDK installations, which are protected by a single USB key. All BSDK installations work with USB license over the network using TCP/IP protocol. To use this license you should select a workstation and install the Sentinel Server software on it.

Sentinel Server Installation

To enable network license on client workstations, the Server must be properly configured. To configure the server first you have to install Sentinel System Drivers by running setup:

```
Program Files\BioLink\BSDK 6.x\drivers\Sentinel\SETUP\setup.exe
```

After successful installation you should run SuperSro Server configuration utility:

```
Program Files\BioLink\BSDK
6.x\drivers\Sentinel\Server\loadserv.exe
```

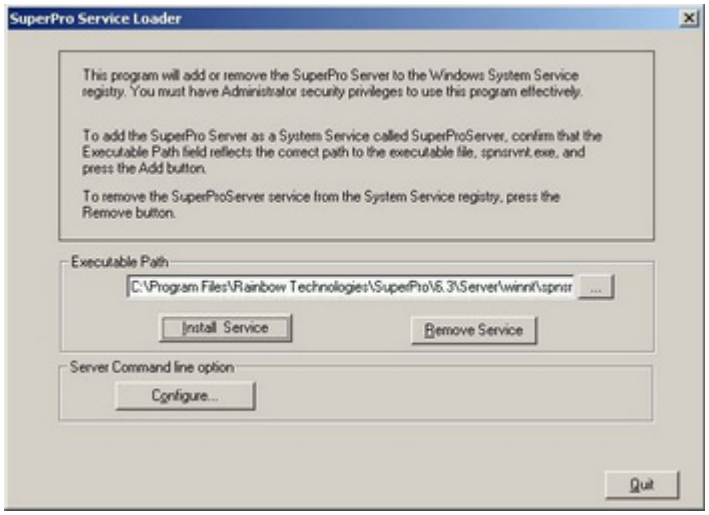

Figure 4: SuperPro Service Loader

Press **Install Service** button and quit the form. Now you can plug USB key into the server PC and it will be automatically available for all network clients.

Sentinel SuperSro Server uses network port 6001, therefore, if there is a firewall installed it should be configured to allow connections on port 6001.

Client Workstation Configuration

In order for workstation with BSDK to find the Sentinel license server the Sentinel System Driver should be installed on this workstation.
The workstation will attempt to contact the license server by sending the broadcast request. If broadcasts are prohibited in the network the IP address of the Sentinel SuperSro Server should be specified explicitly in the workstation's system registry as follows:

```
HKEY_LOCAL_MACHINE\SOFTWARE\BioLink\BSDK\5.0\lic_server="license.s
erver.IP.address"
```

Software license

Software BSDK license is generated for the computer on which BSDK applications are launched. This license is valid only for the computer which hardware identifier was used to generate this particular license. For other computers new licenses (using their unique hardware ids) should be generated.

To generate BioLink BSDK software license use **BioLink BSDK Activation** utility `(bsdk_activator.exe)` found in `X:\Program Files\BioLink\BSDK 6.x\Utils` directory.

On start this utility calculates the PC Hardware ID upon which the **Activation Code** is then will be computed.

To obtain activation code, complete the following:

1) In the **BioLink BSDK Activation** dialog box press either **Get Activation Code On-Line** or **Get Activation Code Off-Line** button.
The first button will transfer you to the BioLink Activation web-site, where you should submit all the required information in order to receive the **Activation Code**.
The second button allows you to fill in the special **Activation Request** form (simular to the one on the BioLink Activation web-site) and send it to BioLink Solutions via any means you prefer. BioLink (or one of its vendors) will send you the **Activation Code** in return. Use the second button only in case you do not have internet connection.

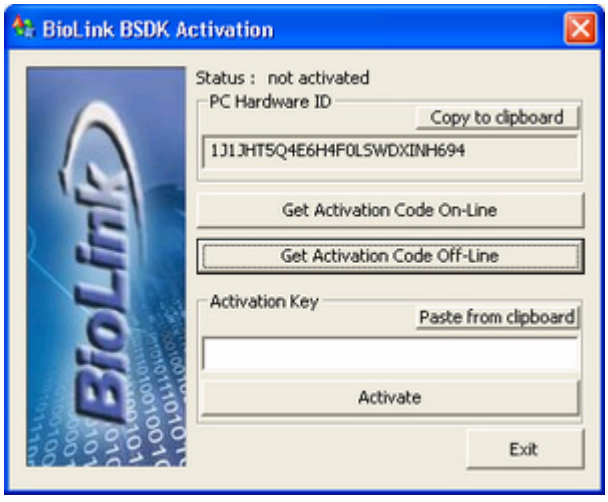

Figure 5: BioLink BSDK Activation utility

2) In the **BioLink BSDK Activation** dialog box, type the received **Activation Code** into the **Activation Key** box. Then press **Activate** button. BSDK software license for your computer will be generated, and BSDK features will be unlocked. **Activation Code** will be stored in the registry (`HKLM/SOFTWARE/BioLink/BSDK/6.x/ActivationCodes`).
Please note that **Status** label will change flagging that BSDK software license has been generated. You can click on the **Status** label to find out what components and features have become available.

You can also use the **BioLink BSDK Activation** utility for:

- extending functionality of the current BSDK version;
- enabling new features, released in a new version (or update) in BSDK.

BSDK licensing in Linux

There are two BSDK license types in Linux:

- embedded in BioLink U-Match 3.5 scanner;
- software.

However there are no applications allowing to modify and/or install license embedded in the BioLink U-Match 3.5 scanner.

As in Windows to obtain software license you should send your hardware identifier to BioLink Solutions or to a partner who sold you BSDK and receive the activation code in response.

The BioLink SDK software license file will look like the following: `bsdk6x_xxx.lic`. This file should be placed in the same folder with `libbsdk2.so`.

To generate BioLink SDK hardware license in Linux complete the following:

1) Run `activator.out` included in the distribution kit.

2) The **hardware id** will be displayed on the screen. Send this id to BioLink Solutions or to a partner who sold you BSDK.

3) When you receive activation code paste it the `activator.out` command prompt.

4) Press **Enter**.

The software BSDK license file will be created for the computer, which hardware id you specified.

To use BSDK on other computers you should repeat the aforementioned process for all computers, on which you develop and/or use BSDK applications.

Extending BSDK license

You should extend BSDK license if you want to:

- extend functionality to the current BSDK version;
- enable new features, released in a new version (or update) in BSDK.

Warning! You can extend BSDK license only on Windows operating systems.

Extending U-Match 3.5 license

A license embedded in U-Match v 3.5 scanner allows you running BSDK on the computer, to which this scanner is attached. Pay your attention to the fact, that there is a possibility that the scanner would have no license. In this case you can manually embed the license using specially provided **BioLink U-Match v.3.5 License Utility**.

Loading BioLink U-Match v.3.5 License Utility

To start BioLink U-Match v.3.5 License Utility, go to the `Program Files/BSDK 6.x/Utils` (default installation) directory and launch `matchbook35license.exe` file.
The application will be loaded and you will be able to embed and/or modify your license.

Extending BioLink U-Match license

To add/modify/extend license embedded in U-Match v.3.5 scanner, complete the following:

- acquire the list of available devices;
- save a file containing all the available devices' IDs and send this file to BioLink Solutions;
- receive the updated file from BioLink Solutions and install the license into your scanners.

All the above steps are covered in the topics below.

Acquiring the list of available devices

To begin extending license embedded in BioLink U-Match scanners, connect a device to the USB port. If you have more than one device you can plug all these devices at once. You can also plug devices one after another, if you have insufficient number of available USB ports.

Then complete the following:

1) Launch **BioLink U-Match v.3.5 License Utility** from the location specified in the *Extending U-Match 3.5 license* topic on p. 58.

2) On the **Get Id List** tab, click **Scan Devices** button to search for all attached and installed devices. The devices' IDs will be displayed in the field to the left.

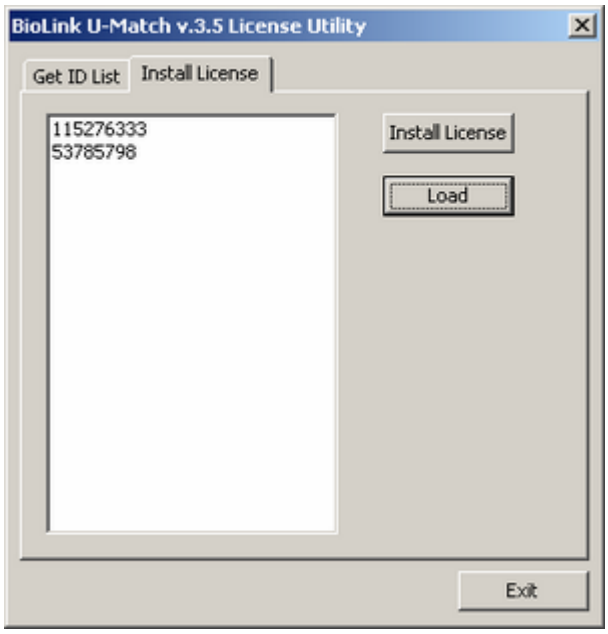

Figure 6: BioLink U-Match License Utility

3) If you need to connect other devices and there are no free USB ports, unplug already scanned devices and attach new ones. Then press **Scan Devices** button again.

4) When you are finished with scanning devices (all devices has been scanned and their IDs are present on the form), click **Save** button.

5) Windows® **Save As** dialog appears. Specify the name and location for the file and click **Save** button.

After you have completed the above steps, you should e-mail the generated file containing devices' IDs to BioLink Solutions. Afterwards you should wait until you receive the updated file from BioLink Solutions.

Installing updated license

When you receive the updated file from BioLink Solutions, complete the following:

1) Launch **BioLink U-Match v.3.5 License Utility** from the location specified in the ***Extending U-Match 3.5 license*** topic on p. 58.

2) Press **Load** button and specify location, where you have saved the updated file received from BioLink Solutions.

3) The IDs of devices will be displayed in the field to the left.

   The figure illustrating the interface of the BioLink U-Match 3.5 license utility can be found in the ***Acquiring the list of available devices*** topic on p. 58.

4) Press **Install License** button. Pay attention to the fact, that you cannot update the license in the device, which have different ID, than the ID sent to BioLink Solutions. In other words, before pressing **Install License** button you should connect the same devices, which you scanned on the **Get ID list** tab.

5) The next dialog box shows the status and availability of BSDK components, as stated in your BSDK license. Please check if all required BSDK features are enabled/disabled correctly. In case of errors or misconceptions, contact BioLink Support Team.

Figure 7: Confirm License Update dialog box

6) Press **Yes** if you want ot install the updated license and **No** if you choose not to make changes to the existing license.

Repeat the last two steps for all devices. If you do not have the sufficient number of USB ports to connect all devices you need to update, unplug already updated devices, connect new ones and press **Install License** button again.

Extending Rainbow Sentinel USB license

The license embedded in Rainbow Sentinel USB dongle can be extended by using the provided **Field Exchange Utility.**

Field Exchange Utility is a simple program that updates your hardware key for extending, adding or maintaining your current licenses.

The application can be found in `\Utils` folder, in the BioLink SDK installation directory.

Warning! You do not need to update your hardware key right after the purchase since you already have completely efficient key.You only need to update your key if a new version of BSDK has been released, or a new functionality has been added to the current release.

Warning! You do not need to update your hardware key right after the purchase since you already have completely efficient key.

You only need to update your key if a new version of BSDK has been released, or a new functionality has been added to the current release.

General Description

To update the hardware key used to run BioLink SDK, you must provide information about the key to BioLink Solutions. The **Field Exchange Utility** displays this information in the form of a locking code, similar to the following:

```
EHBFGYARDIJABRFLEBDH
```

You must communicate the locking code to BioLink Solutions via e-mail. After checking code validation BioLink Solutions will then give you a corresponding license code to enter into the Field Exchange Utility.

The following picture demonstrates Field Exchange Utility window:

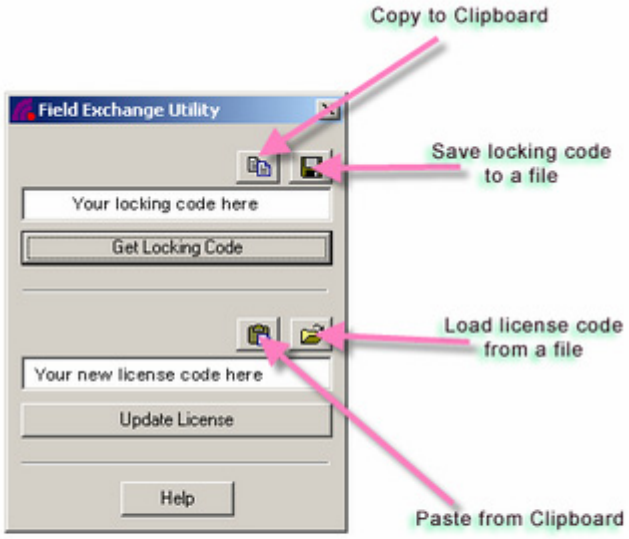

Figure 8: Field Exchange Utility dialogue box

Obtaining a locking code

To obtain a locking code, complete the following:

1) Verify the correct BioLink hardware key is attached to the appropriate port (USB by default) on your computer.

2) Click **Get Locking Code**. The locking code appears in the top field.

Note. If the message "Error" appears in the top field, make sure the key is firmly and correctly attached to the port and try again.

3) Do one of the following:

- Click the **Copy** button  to place the locking code on the clipboard.
- Click the **Save** button  to save the locking code to a file. Define a location and file name for the file, then click **Save**.
- Write down the code exactly as it appears in the field.

4) Send the locking code to BioLink Solutions.

5) Wait until you have received the license code from BioLink Solutions.

Warning! You can leave the Field Exchange Utility open, or you may close it. Either way, the license code you receive from BioLink Solutions will update your key correctly.

Entering a license code

To enter a license code, complete the following:

1) Verify the correct BioLink hardware key is attached to your computer. Only one key should be attached to your computer during the update process.

2) In the box above the **Update License** button, enter new BioLink license code. Do one of the following:

- Click the **Paste** button  to paste the code from the clipboard, if it was placed there.
- Click the **Open** button  to load the code from a file. Browse to locate the file (.LIC), then click **Open**.
- Type the code in the box. Be sure to enter it exactly as it was provided.

3) Click **Update License**. The key update process begins.

Note. It may take up to two or three minutes to complete the update process.

4) When *"Update Successful"* appears, close the **Field Exchange Utility**.

New BioLink SDK features are now unlocked, and you should have full access to them.

Remarks

If the update process is not successful, verify the hardware key is securely attached to your computer and try again. If the process is still unsuccessful, or you do not have access to the upgrades or licenses you purchased when the process is complete, contact BioLink Solutions for support assistance.

Index