**Notes**

[How to connect streaming data and display in Power BI](https://stackoverflow.com/questions/63963951/how-to-connect-streaming-data-and-display-in-power-bi)

For your use case you will not be able to do it all in Power BI, you'll need to push the streaming data to some sort of database, Azure SQL Server/Azure Cosmos DB/Azure Databricks using Event Hubs and maybe Streaming Analytics as well. So for example your stream sends the data to the event hub endpoint, then sends it to a database, then Power BI reads it.

You can then link a Power BI report to that database and set it up in Direct Query mode, and then set the page refresh on the report. Depending on the type of service your period of refresh for none Embedded/Premium workspaces will be 30mins or more, and not 'realtime'

# ZKTeco Biometric Device Getting Started

Important note:awalys delete the : IZKEM in the ZkemClient.cs

* [**Download BioMatrix source code - 551.8 KB**](https://www.codeproject.com/KB/cs/1104538/BioMatrix.zip)
* [**Download Programmers\_Guide\_Manual - 289.4 KB**](https://www.codeproject.com/KB/cs/1104538/Biometric_Device_SDK-_Programmers_Guide.zip)
* [**Download the SDK - 1.4 MB**](https://www.codeproject.com/KB/cs/1104538/TheSDK.zip)

## **Introduction**

There are requirements where you have to integrate a biometric device into your own application or some part of it like fetching the attendance records, user information, backing up the fingerprint templates into your system database or export it somewhere remotely, etc.

For that very purpose, this article is written as a getting started guide with a few pieces of source code such that it helps C# developers to save some valuable time with all the troublesome work.

**Note**: Though the devices will be of various models, with a different set of sdk, the implementation procedure will be somewhat the same.

The following article contains some useful old pieces of code, if you ever get your hands on a biometric device (fingerprint reader for Time & Attendance) and want to extract gold(data) out of it.

### **Implemented Device Specification**

Copy Code

Device Name : K14

Firmware Ver : ver. 4.0.1 ( build 39 )

Vendor : ZKTeco Inc.

FP : 500

Records : 50000

**NOTE:**

If you are working with a different model of the device like **ZK4500, ZK9500, SLK20M or SLK20R**. [Here is a good starting](https://www.debuggershub.com/c-zkteco-fingerprint-scanner-implementation-zk4500-slk20m-slk20r-zk9500/) point for those device models.

If you are working with a different brand like the Realtime Biometric

Prior to going through the following article, it would be beneficial if you have some good understanding of the following things:

* How a biometric device works, its features and functionalities
* How to Register/ Unregister DLLs using the Regsvr32 utility
* Knowledge about how the ref and out parameters work

The article and the source code provided will help you with the following things:

* Registering the SDK components
* Ping the device
* Connecting to the device
* Extracting some random data out of the device (Log Data, User Info, etc.)
* Source code to upload data to the device
* Restarting the device
* How to register to device specific events [Not Implemented/ You will have to implement it yourself per your need]

### **Background**

Biometric Devices are being used extensively in many corporations through out the world these days. Companies like [ZKTeco](https://www.zkteco.com/) manufacture **biometric access control and Time and Attendance** in various shapes and sizes with different set of features per model. They provide a better way of user authentication and security in organizations who implement it. It is a growing demand in today's world. Hence, developers are likely to come across its implementation at some point.

### **Using the Code**

**Note**: The follow procedures are applicable for the **K14 model** devices. Other devices might have a different set of SDK, hence a slightly different set of procedures.

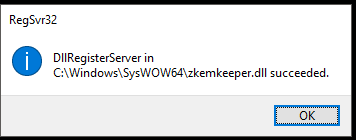
In order to implement any biometric device, you will have to register the related components in the system, which can be easily done by using the Regsvr32 utility. If you don't know how it works, please go through some blog posts in the related topic. We already have a batch file ready for that very purpose, Hence, we are cool here.

You can download the SDK from [here](http://www.zkteco.com/support/SDK/39_55.html). I have already attached an SDK sample along with this article. You can go through the source code inside the Register\_SDK.batto see the registration mechanism.

### **Registering the Components**

1. Fire up a Command Prompt with Administrator Privilege
2. Navigate to the SDK location
3. Execute the Register\_SDK batch file

### **On Successful Registration**



To implement the biometric device, you will have to use the zkemkeeper.dll.

### **Implementing the SDK**

1. Add Reference to the zkemkeeper.dll in a project. The DLL gives access to the following things:
2. Create a class file and implement the IZKEM interface.
3. Use the CZKEMclass file to perform device related operations.

Let's say your Biometric device is connected in the local area network. To make a successful communication with the device, you need to know the **IP address**and the **Port No.**

Copy Code

IP address : Assign a static IP in the device itself

Default Port : 4370

Machine No : 1

Now we have a valid IP and Port, let's go through the source code used in a demo project entitled BioMetrix which I have attached along with this article. The application helps by showing some of the API implementations.

The ZKenClient.cs file implements the IZKEM interface which contains the following code.

### **Connecting to the Device**

The following code helps connect to the device at the given IPAddress and Port and on successful connection, it registers some events.

(The events are not implemented though, you will have to implement it yourself as per your need.)

C++

Copy Code

public bool Connect\_Net(string IPAdd, int Port)

{

if (objCZKEM.Connect\_Net(IPAdd, Port))

{

if (objCZKEM.RegEvent(1, 65535))

{

objCZKEM.OnConnected += ObjCZKEM\_OnConnected;

objCZKEM.OnDisConnected += objCZKEM\_OnDisConnected;

objCZKEM.OnEnrollFinger += ObjCZKEM\_OnEnrollFinger;

objCZKEM.OnFinger += ObjCZKEM\_OnFinger;

objCZKEM.OnAttTransactionEx +=

new \_IZKEMEvents\_OnAttTransactionExEventHandler(zkemClient\_OnAttTransactionEx);

}

return true;

}

return false;

}

### **Registering Events**

To fire any event when something occurs in the device, we need to register the event:

C++

Copy Code

bool RegEvent(int dwMachineNumber, int EventMask);

*// Entering 65535 for the EventMast value registers for all events*

The EventMask requires an integer value whose value varies based on the type of event. The complete list of EventMast values are given below:

|  |  |  |
| --- | --- | --- |
| 1 |  | OnAttTransaction, OnAttTransactionEx |
| 2 (1<<1) | OnFinger |  |
| 4 (1<<2) | OnNewUser |  |
| 8 (1<<3) | OnEnrollFinger |  |
| 16 (1<<4) | OnKeyPress |  |
| 256 (1<<7) | OnVerify |  |
| 512 (1<<8) | OnFingerFeature |  |
| 1024 (1<<9) | OnDoor, OnAlarm |  |
| 2048 (1<<10) | OnHIDNum |  |
| 4096 (1<<11) | OnWriteCard |  |
| 8192 (1<<12) | OnEmptyCard |  |
| 16384 (1<<13) | OnDeleteTemplate |  |

To register for multiple-events, we can perform XOR operations between binary codes of related events.

To register all real-time events, the value of Eventmast can be set as 65535 which I have implemented in the above code.

### **Fetching User Info**

The user information can be fetched by providing the machine number. The API implements the outparameter to give back the output. I had stored some random user data in the fingerprint device and used the following code to retrieve it back and display it. The TmpData is the users (fingerprint) template data.

C++

Shrink ▲   Copy Code

public ICollection<UserInfo> GetAllUserInfo(ZkemClient objZkeeper, int machineNumber)

{

string sdwEnrollNumber = string.Empty, sName = string.Empty,

sPassword = string.Empty, sTmpData = string.Empty;

int iPrivilege = 0, iTmpLength = 0, iFlag = 0, idwFingerIndex;

bool bEnabled = false;

ICollection<UserInfo> lstFPTemplates = new List<UserInfo>();

objZkeeper.ReadAllUserID(machineNumber);

objZkeeper.ReadAllTemplate(machineNumber);

while (objZkeeper.SSR\_GetAllUserInfo(machineNumber, out sdwEnrollNumber,

out sName, out sPassword, out iPrivilege, out bEnabled))

{

for (idwFingerIndex = 0; idwFingerIndex < 10; idwFingerIndex++)

{

if (objZkeeper.GetUserTmpExStr(machineNumber, sdwEnrollNumber,

idwFingerIndex, out iFlag, out sTmpData, out iTmpLength))

{

UserInfo fpInfo = new UserInfo();

fpInfo.MachineNumber = machineNumber;

fpInfo.EnrollNumber = sdwEnrollNumber;

fpInfo.Name = sName;

fpInfo.FingerIndex = idwFingerIndex;

fpInfo.TmpData = sTmpData;

fpInfo.Privelage = iPrivilege;

fpInfo.Password = sPassword;

fpInfo.Enabled = bEnabled;

fpInfo.iFlag = iFlag.ToString();

lstFPTemplates.Add(fpInfo);

}

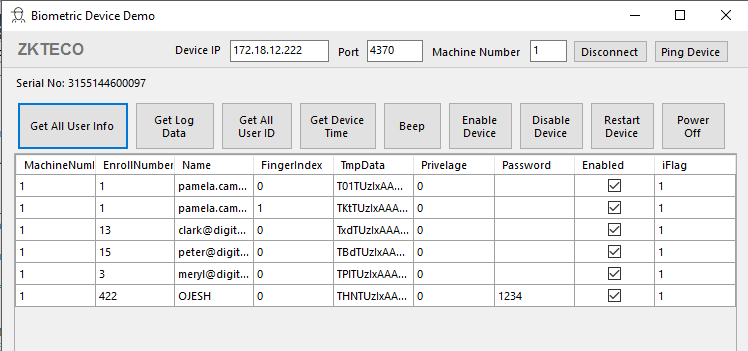
}

}

return lstFPTemplates;

}

### **Result**



### **Conclusion**

Well, that gives you the starting point for your project. With a biometric device in hand, you can now experiment with the source code yourself and play along.

There are numerous procedures and events to be implemented which you can use per you need. Also, you might want to take help from the SDK that I have attached along with this article. There are other SDK/manuals which you can find online for more detailed implementation.

Feel free for any queries and feedbacks.

Peace !!

[Unable to find manifest signing certificate in the certificate store” - even when add new key](https://stackoverflow.com/questions/11957295/unable-to-find-manifest-signing-certificate-in-the-certificate-store-even-wh)

**Within your Solution Explorer:**

1. right click on your project
2. click on properties
3. usually on the left-hand side, select the "Signing" tab
4. check off the Sign the ClickOnce manifests
5. Make sure you save!