ZEBRA CoreScanner COM API for windows

v1.0.0.0

Jan 2023

Contents

CoreScanner COM API Introduction

Overview

The CoreScanner COM object provides an easy to use yet powerful and extendible set of API commands to interface with scanner devices- RFD40 and RFD90. The COM API can be used in C#/C++ applications. To build a C# COM API application use Microsoft .NET 4.8 and Visual Studio 2017 or above.

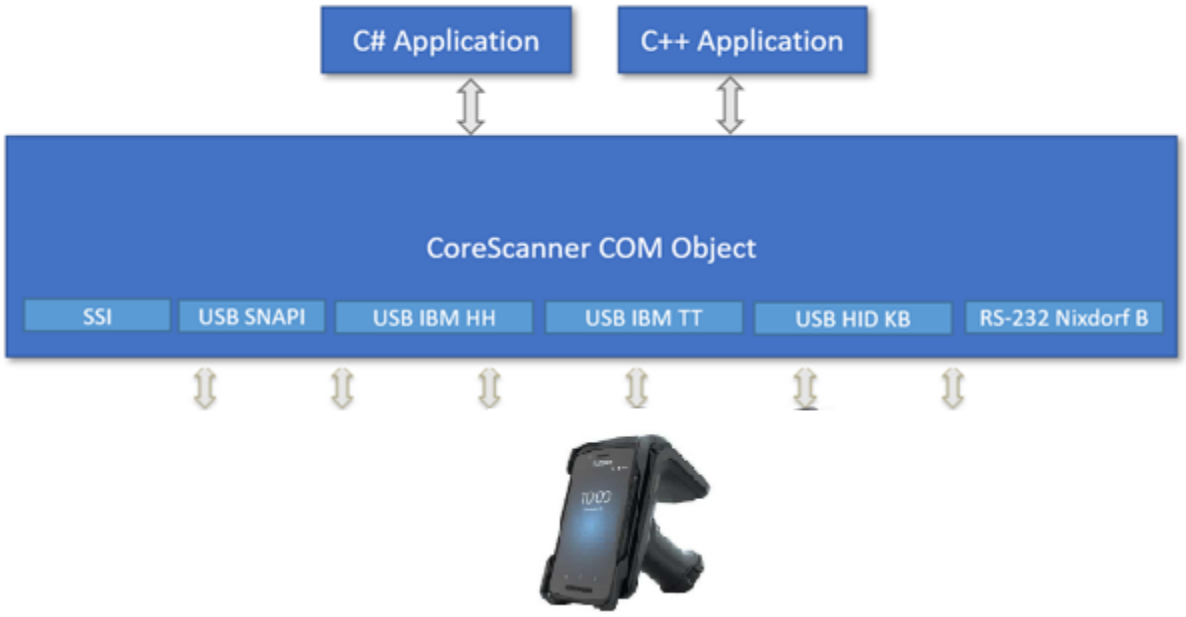

Figure 1-1 *CoreScanner COM API*

The COM API commands include:

- Open
- GetScanners
- ExecCommand
- ExecCommandAsync
- Close.

Once the COM object is Open and GetScanners command is invoked and the list of connected scanners is retrieved, all other methods execute through the ExecCommand and ExecCommandAsync commands.

In addition to the commands above, the COM object supports seven types of events:

- ImageEvent
- VideoEvent
- BarcodeEvent
- PNPEvent
- ScanRMDEvent
- CommandResponseEvent
- IOEvent
- BinaryDataEvent.

Sample Utilities Provided in the SDK

The Zebra Scanner SDK includes sample utilities that demonstrate the main functionalities of the SDK. You can gain an understanding of the Zebra Scanner SDK by using these test utilities.

In addition, this section describes how to use the utilities functionality.

Creation of COM Object And Registration for Events

```
using CoreScanner;
…
m_pCoreScanner = new CoreScanner.CCoreScannerClass(); //create COM object
…
```

Register for COM Events

```
/* Event registration for COM Service */
m_pCoreScanner.ImageEvent += new
CoreScanner._ICoreScannerEvents_ImageEventEventHandler(OnImageEvent);
m_pCoreScanner.VideoEvent += new
CoreScanner._ICoreScannerEvents_VideoEventEventHandler(OnVideoEvent);
m_pCoreScanner.BarcodeEvent += new
CoreScanner._ICoreScannerEvents_BarcodeEventEventHandler(OnBarcodeEvent);
m_pCoreScanner.PNPEvent += new
CoreScanner._ICoreScannerEvents_PNPEventEventHandler(OnPNPEvent);
m_pCoreScanner.ScanRMDEvent += new
CoreScanner._ICoreScannerEvents_ScanRMDEventEventHandler(OnScanRMDEvent);
m_pCoreScanner.CommandResponseEvent += new
CoreScanner._ICoreScannerEvents_CommandResponseEventEventHandler(OnCommandResponseEven
t);
m_pCoreScanner.IOEvent += new
CoreScanner._ICoreScannerEvents_IOEventEventHandler(OnIOEvent);
```

Calling Open Command

```
private void Connect()
{
if (m_bSuccessOpen)
{
return;
}
int appHandle = 0;
GetSelectedScannerTypes();
int status = STATUS_FALSE;
try
{
m_pCoreScanner.Open(appHandle, m_arScannerTypes, m_nNumberOfTypes, out
status);
…
}
```

Calling Close Command

```
private void Disconnect()
{
if (m_bSuccessOpen)
{
```

```
int appHandle = 0;
int status = STATUS_FALSE;
try
{
m_pCoreScanner.Close(appHandle, out status);
…
}
```

Calling GetScanners Command

```
private void ShowScanners()
{
lstvScanners.Items.Clear();
combSlcrScnr.Items.Clear();
m_arScanners.Initialize();
if (m_bSuccessOpen)
{
m_nTotalScanners = 0;
short numOfScanners = 0;
int nScannerCount = 0;
string outXML = "";
int status = STATUS_FALSE;
int[] scannerIdList = new int[MAX_NUM_DEVICES];
try
{
m_pCoreScanner.GetScanners(out numOfScanners, scannerIdList, out outXML,
out status);
…
}
```

Calling ExecCommand Command and ExecCommandAsync Command

```
private void ExecCmd(int opCode, ref string inXml, out string outXml, out int status)
{
outXml = "";
status = STATUS_FALSE;
if (m_bSuccessOpen)
{
try
{
if (!chkAsync.Checked)
{
m_pCoreScanner.ExecCommand(opCode, ref inXml, out outXml, out
status);
}
else
{
m_pCoreScanner.ExecCommandAsync(opCode, ref inXml, out status);
}
}
…
}
```

Creating a COM API Windows Demo Project

To create a C# Windows Project in Visual Studio 2017:

1. Start Visual Studio 2017->Select File > New > Project->Visual C#, create a new Windows Forms Application project and follow the on-screen steps in Visual Studio.

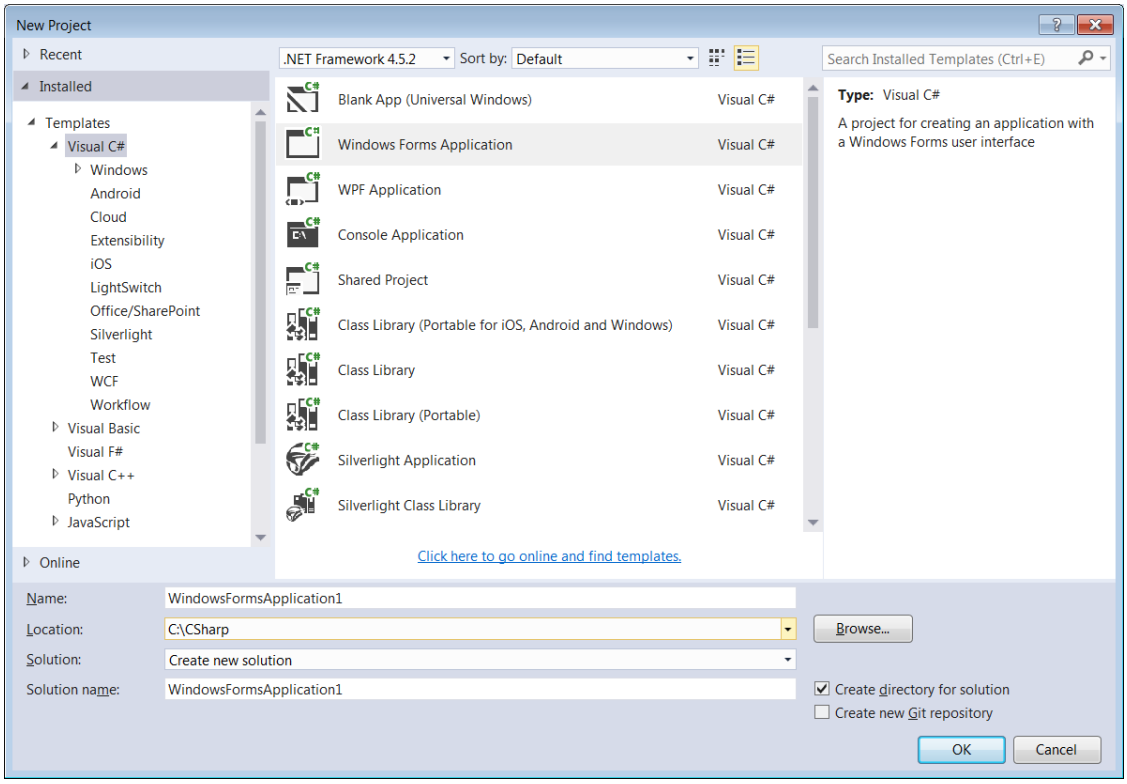

Figure 1-2 *Create New Project*

2. In Solution Explorer, right click ***References***, and select ***Add Reference***, and select COM Tab.

3. Select "**CoreScanner 1.0 Type Library**", and click ***OK*** to add a reference to the CoreScanner COM object.

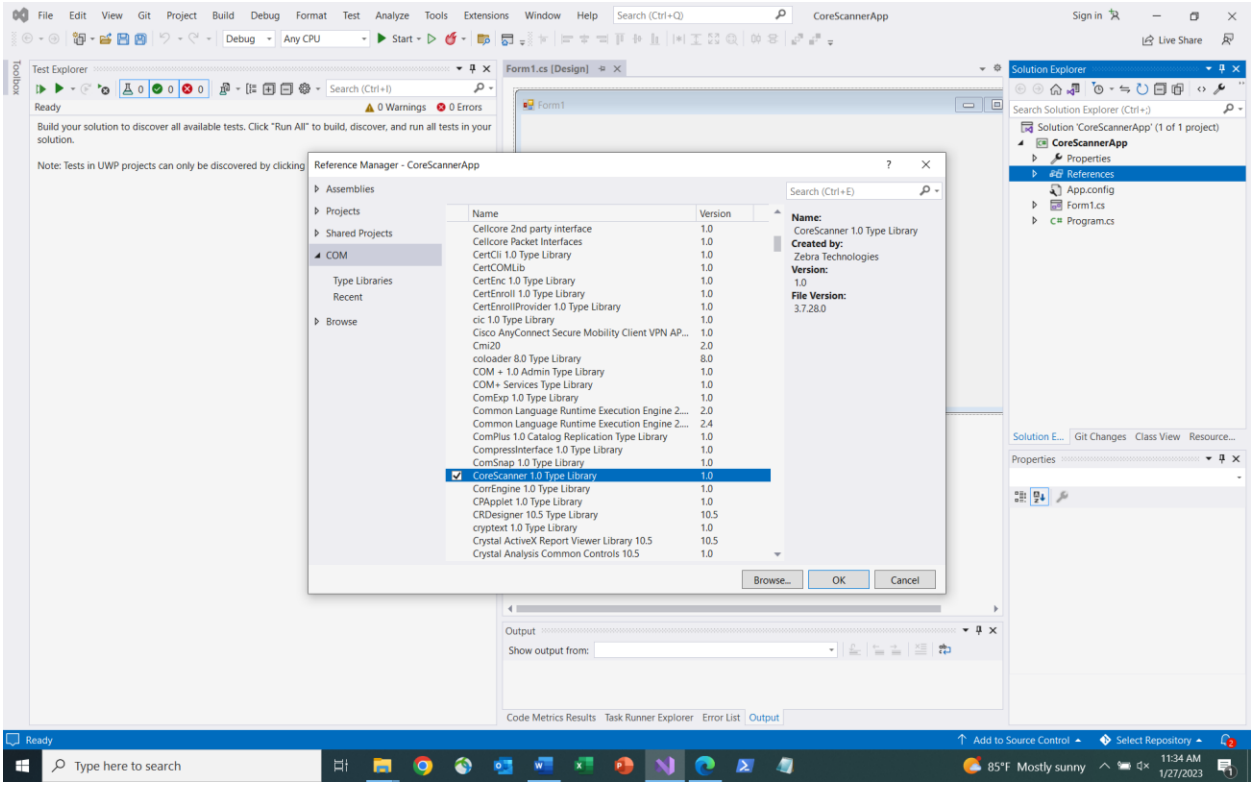

Figure 1-3 *CoreScanner COM object Example Application*

Use the following code snippet to discover scanners.

```
using CoreScanner;
using System;
using System.Diagnostics;
using System.Windows.Forms;

namespace CoreScannerApp
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            // Initialize CoreScanner COM object
            CoreScanner.CCoreScanner coreScannerObject = new CCoreScanner();
            const int StatusSuccess = 0;
            const int MaxNumDevices = 255;
            int appHandle = 0;
            const short NumberOfScannerTypes = 1;
            short[] scannerTypes = new short[NumberOfScannerTypes];
            scannerTypes[0] = (short) 1; //  All scanner types
            int status = -1;
                // Open CoreScanner COM Object
                coreScannerObject.Open(appHandle, // Application handle
                    scannerTypes, // Array of scanner types
```

```
                    NumberOfScannerTypes, // Length of scanner types array
                    out status); // Command execution success/failure return status

                if (status == StatusSuccess)
                {
                    Console.WriteLine("CoreScanner Open() - Success");
                    short numOfScanners = 0;
                    string outXml = "";
                    int[] scannerIdList = new int[MaxNumDevices];
                    // Get connected scanners
                    coreScannerObject.GetScanners(out numOfScanners,//total scanners discovered
                        scannerIdList, // Returns array of connected scanner ids
                        out outXml,    // Output xml containing discovered scanners information
                        out status);   // Command execution success/failure return status

                    if (status == StatusSuccess)
                    {
                        Log("CoreScanner GetScanners()- Success");
                        Log(" Total Scanners : " + numOfScanners);
                        string scannerIds = "";
                        Array.Resize(ref scannerIdList, numOfScanners);
                        scannerIds = String.Join(", ", scannerIdList);
                        Log(" Scanner IDs :" + scannerIds);
                        Log(" Out xml : " + Environment.NewLine + outXml);
                    }
                }
                // Close CoreScanner COM object
                coreScannerObject.Close(appHandle, out status);
            }

            private void Log(string msg)
            {
                txtStatus.Text += msg+Environment.NewLine;
            }
        }
    }
```

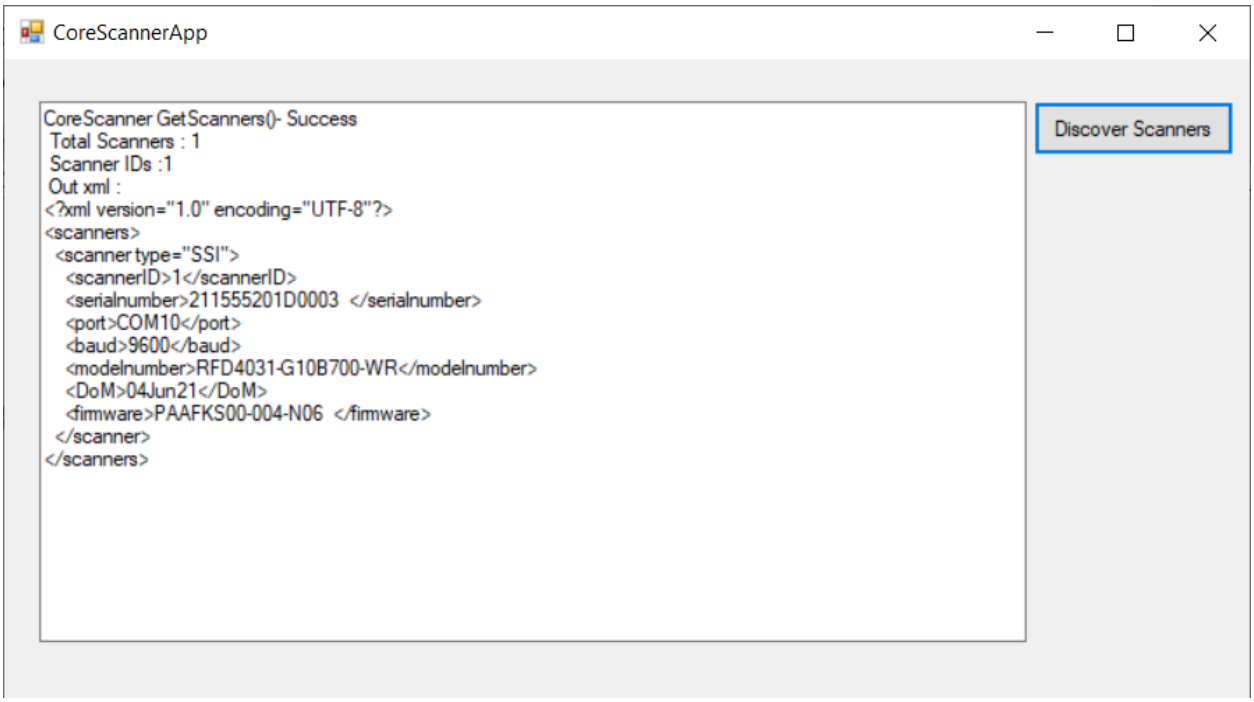

Figure 1-3 *Example Application output*