

Api Methods (v2.0.x)

Beep()

To add sound "beep" after successful scan please use the code below:

```
readerDevice.Beep()
```

It plays an audio signal on the MX device.

Connect()/Disconnect()

The process of connecting and disconnecting to a MX device is done via the Connect/Disconnect methods

Connect()

```
readerDevice.Connect(IONConnectionCompletedListener listenerObject)
```

will try to connect to the device. When the **Connect()** is executed we get the status of the action in the **OnConnectionCompleted** listener

```
OnConnectionCompleted(ReaderDevice reader, Throwable error){
    if(error != null){
        //do something with the error
        readerDisconnected();
    }
}
```

Disconnect()

```
readerDevice.Disconnect()
```

will disconnect from the MX device.

Examples:

```
readerDevice = ReaderDevice.GetMXDevice(mContext);
readerDevice.StartAvailabilityListening();
readerDevice.SetReaderDeviceListener(this);
readerDevice.EnableImage(true);
readerDevice.Connect(this);

...
```

```
public void OnAvailabilityChanged(ReaderDevice reader)
```

```
{
    if (reader.GetAvailability() == Availability.Available)
    {
        readerDevice.Connect(this);
    }
    ...
}
```

EnableImage()

The result from a successful scan can return an image. This is the last frame that resolved in a successful scan. To enable / disable this, we can use the API method:

```
void readerDevice.EnableImage(bool enable)
```

```
readerDevice = ReaderDevice.GetPhoneCameraDevice(mContext, CameraMode.NoAimer, PreviewOption.Defaults, rlMainContainer);
readerDevice.EnableImage(true);
readerDevice.Connect(this);
```

EnableImageGraphics()

The result from a successful scan can return an SVG image graphics. This is the last frame that resolved in a successful scan. To enable / disable this, we can use the API method:

```
void readerDevice.EnableImageGraphics(bool enable)
```

```
readerDevice = ReaderDevice.GetPhoneCameraDevice(mContext, CameraMode.NoAimer, PreviewOption.Defaults, rlMainContainer);
readerDevice.EnableImageGraphics(true);
readerDevice.Connect(this);
```

GetAvailability()

Before we can connect to an MX device, we need to know if there's one available for that task.

```
Availability reader.GetAvailability()
```

It return **Availability** object that can be **Available**, **Unavailable** or **Unknown**

An MX device is available when there is an USB connection to our smartphone.

Example

```
public void OnAvailabilityChanged(ReaderDevice reader)
{
    if (reader.GetAvailability() == Availability.Available)
    {
        readerDevice.Connect(this);
    }
}
```

```
}  
...
```

GetDeviceBatteryLevel()

If we want to check the battery level of the MX device we can use

```
void readerDevice.GetDeviceBatteryLevel(IOnDeviceBatteryLevelListener listenerObject)
```

Retrieves the current battery percentage level of the reader device as input parameter in **OnDeviceBatteryLevelReceived** listener method

Example

```
public void OnConnectionStateChanged(ReaderDevice reader)
{
    if (reader.ConnectionState == ConnectionState.Connected)
    {
        reader.GetDeviceBatteryLevel(this);
        .....
    }
}

public void OnDeviceBatteryLevelReceived(ReaderDevice p0, int p1, Throwable p2)
{
    int batteryLevel = p1;
}
```

IsLightsOn()/SetLightsOn()

If we want to check whether all lights of the MX device are turned on or off

```
void readerDevice.IsLightsOn(IOnLightsListener listenerObject)
```

Retrieves if lights of the reader device are turned on or off as input parameter in **OnLightsOnCompleted** listener method

To turned MX device light on or off we use

```
void reader.SetLightsOn(bool _enable, IOnLightsListener listenerObject)
```

Example

```
public void OnConnectionStateChanged(ReaderDevice reader)
{
    if (reader.ConnectionState == ConnectionState.Connected)
    {
        reader.IsLightsOn(this);
        .....
    }
}

public void OnLightsOnCompleted(ReaderDevice p0, Java.Lang.Boolean p1, Throwable p2)
{
}
```

```
bool lightsON = p1.BooleanValue();
}
```

IsSymbologyEnabled()/SetSymbologyEnabled()

If we want to check some symbology is enabled or disabled we can use

```
void readerDevice.IsSymbologyEnabled(Symbology _symbology, IOnSymbologyListener listenerObject)
```

Retrieves if symbology is enabled or disabled as input parameter in **OnSymbologyEnabled** listener method

To enable specific symbology we use

```
void readerDevice.SetSymbologyEnabled(Symbology _symbology, bool _enable, IOnSymbologyListener listenerObject)
```

Example

```
public void OnConnectionStateChanged(ReaderDevice reader)
{
    if (reader.ConnectionState == ConnectionState.Connected)
    {
        reader.IsSymbologyEnabled(Symbology.Azteccode, this);
        .....
    }
}
public void OnSymbologyEnabled(ReaderDevice p0, Symbology p1, Java.Lang.Boolean p2, Throwable p3)
{
}
```

ResetConfig()

To reset MX Device configuration settings to default use

```
void reader.ResetConfig(IOnResetConfigListener listenerObject)
```

In **OnResetConfigCompleted** listener we can check if resetting was successful by checking if Throwable input parameter error is null or not

```
public void OnResetConfigCompleted(ReaderDevice p0, Throwable p1)
{
    if (p1 != null)
        Console.WriteLine("Resetting was unsuccessful. Error: " + p1.Message);
}
```

SetReaderDeviceListener()

To register listener functions **OnAvailabilityChanged**, **OnConnectionStateChanged** and **OnReadResultReceived** we need to call this method before we try to connect to reader device

```
void readerDevice.SetReaderDeviceListener(IReaderDeviceListener listenerObject)
```

```
public void OnAvailabilityChanged(ReaderDevice reader)
{
    if (reader.GetAvailability() == Availability.Available)
    {
        readerDevice.Connect(this);
    }
}
```

```
public void OnConnectionStateChanged(ReaderDevice reader)
{
    if (reader.ConnectionState == ConnectionState.Connected)
    {
    }
    else if (reader.ConnectionState == ConnectionState.Disconnected)
    {
    }
}
```

```
public void OnReadResultReceived(ReaderDevice reader, ReadResults results)
{
    if (results.Count > 0)
    {
        ReadResult result = results.GetResultAt(0);

        if (result.IsGoodRead)
        {
            ...
        }
    }
}
```

Example

```
readerDevice = ReaderDevice.GetPhoneCameraDevice(mContext, CameraMode.NoAimer, PreviewOption.Defaults, rlMainContainer);
readerDevice.SetReaderDeviceListener(this);
readerDevice.Connect(this);
```

StartAvailabilityListening()/StopAvailabilityListening()

After we call **readerDevice.SetReaderDeviceListener(IReaderDeviceListener listenerObject)** method and register **OnAvailabilityChanged** listener function we can start/stop availability listening

StartAvailabilityListening()

```
void readerDevice.StartAvailabilityListening()
```

will start listening reader device availability and will trigger listener function every time when availability is changed

StopAvailabilityListening()

```
void readerDevice.StopAvailabilityListening()
```

will stop listening reader device availability

Examples:

```
readerDevice = ReaderDevice.GetMXDevice(mContext);
readerDevice.StartAvailabilityListening();
readerDevice.SetReaderDeviceListener(this);
readerDevice.Connect(this);
```

```
protected override void Dispose(bool disposing)
{
    if (readerDevice != null && readerDevice.ConnectionState == ConnectionState.Connected)
    {
        readerDevice.SetReaderDeviceListener(null);
        readerDevice.Disconnect();
    }

    if (readerDevice != null)
    try
    {
        readerDevice.StopAvailabilityListening();
    }
    catch (System.Exception e) { }

    base.Dispose(disposing);
}
```

StartScanning()/StopScanning()

After we connect and configure our reader device and set all settings that we need we can start scanning process.

To start scanning we use method

```
void readerDevice.StartScanning()
```

Scanning process will stop when our reader successfully scan some barcode or we can stop scanning manually with this method

```
void readerDevice.StopScanning()
```

When scanning is stopped, no matter with successful scan or with stopScanning() method **OnReadResultReceived** listener function will be called where we can check our scan result

```
public void OnReadResultReceived(ReaderDevice reader, ReadResults results)
{
    if (results.Count > 0)
    {
        ReadResult result = results.GetResultAt(0);

        if (result.IsGoodRead)
        {
            Symbology sym = result.Symbology;
            if (sym != null)
            {
                tvSymbology.Text = sym.Name;
            }
            else
            {
                tvSymbology.Text = "UNKNOWN SYMBOLOGY";
            }
            tvCode.Text = result.ReadString();
        }
        else
        {

```

```
        tvSymbology.Text = "NO READ";  
        tvCode.Text = "";  
    }  
    ivPreview.SetImageBitmap(result.Image);  
}  
}
```