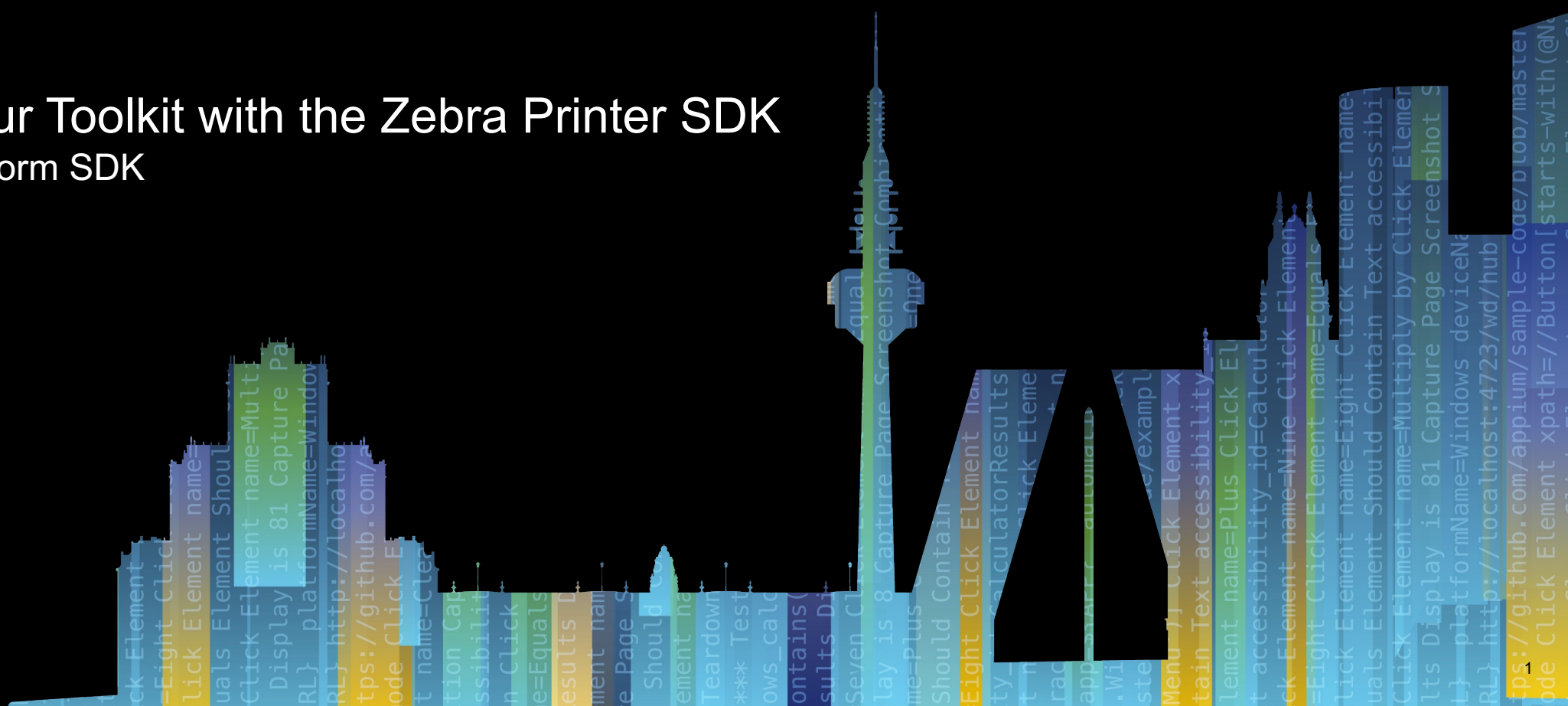


Power Up Your Toolkit with the Zebra Printer SDK Link-OS Multiplatform SDK

Steven Si

Sr. Software Engineer



How do users benefit from unmatched value



BETTER PERFORMANCE

Enable your printers to perform at their peak for simple, effortless printing.

SUPERIOR MANAGEABILITY

Simplified yet powerful tools for easy remote management of any size printer deployment.

EASY INTEGRATION

Our printers integrate into your existing technology architecture — painlessly.



How do users benefit from unmatched value



DEVELOPMENT TOOLS



Multiplatform
SDK



Browser
Print



PrintConnect



Cloud
Connect

VISIBILITY TOOLS



MDM/EMM
Connectors



Visibility
Services

MANAGEMENT TOOLS



Printer Profile
Manager Enterprise



Bluetooth® Printer
Management



Printer Setup
Utility

PRODUCTIVITY TOOLS



ZebraDesigner



Virtual
Devices



PDF
Direct



Enterprise
Connectors



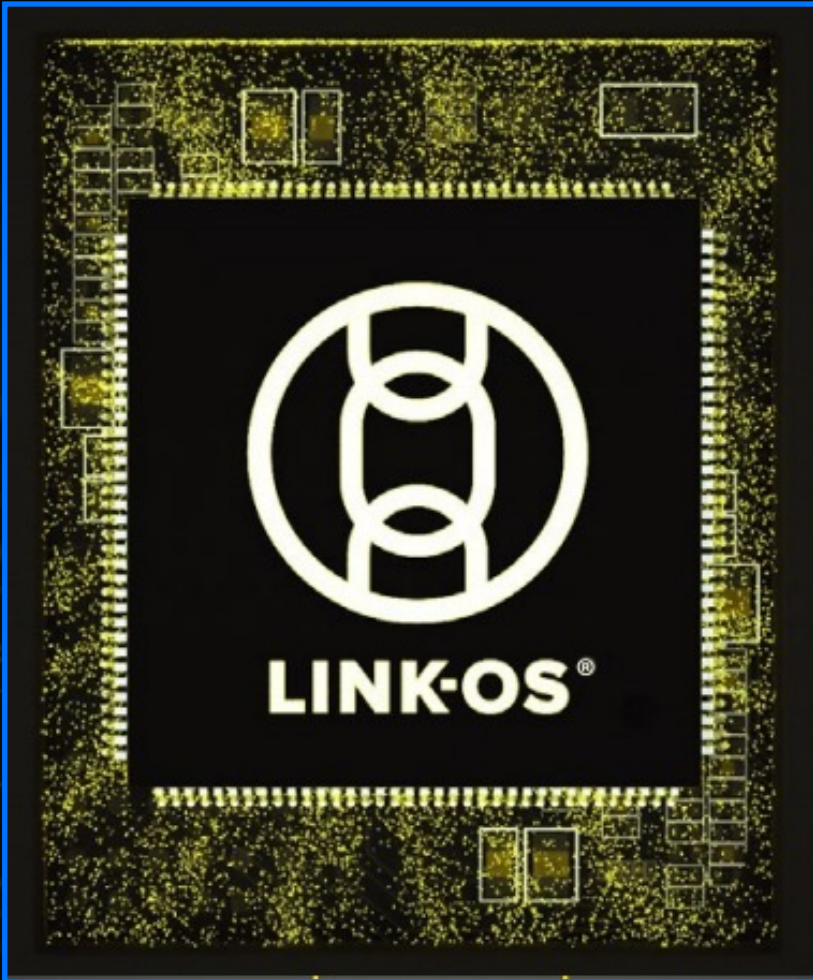
Print
Station



Pairing
Solutions



The most intelligent printer operating system in existence



Zebra's one-of-a-kind enterprise printer operating system that powers Print DNA capabilities:

The ZPL standard

Consistency across all models

Cloud connectivity

Secure low-latency connection

Battle tested wireless

Over 20 years of experience

Unicode

Ready for global solutions

Extensible

Link-OS has evolved to meet modern needs



The Link-OS Printer family



ZD500R



ZD510-HC



ZD421 Series



ZD411 Series



ZD611 & ZD621 Series



ZT200 Series

ZT400 Series

ZT500 Series

ZT600 Series



ZQ300 Plus Series



ZQ511 & ZQ521 Series



ZQ600 Plus Series



Development Tools



Multiplatform SDK



Browser Print



PrintConnect



Cloud Connect

INTEGRATE

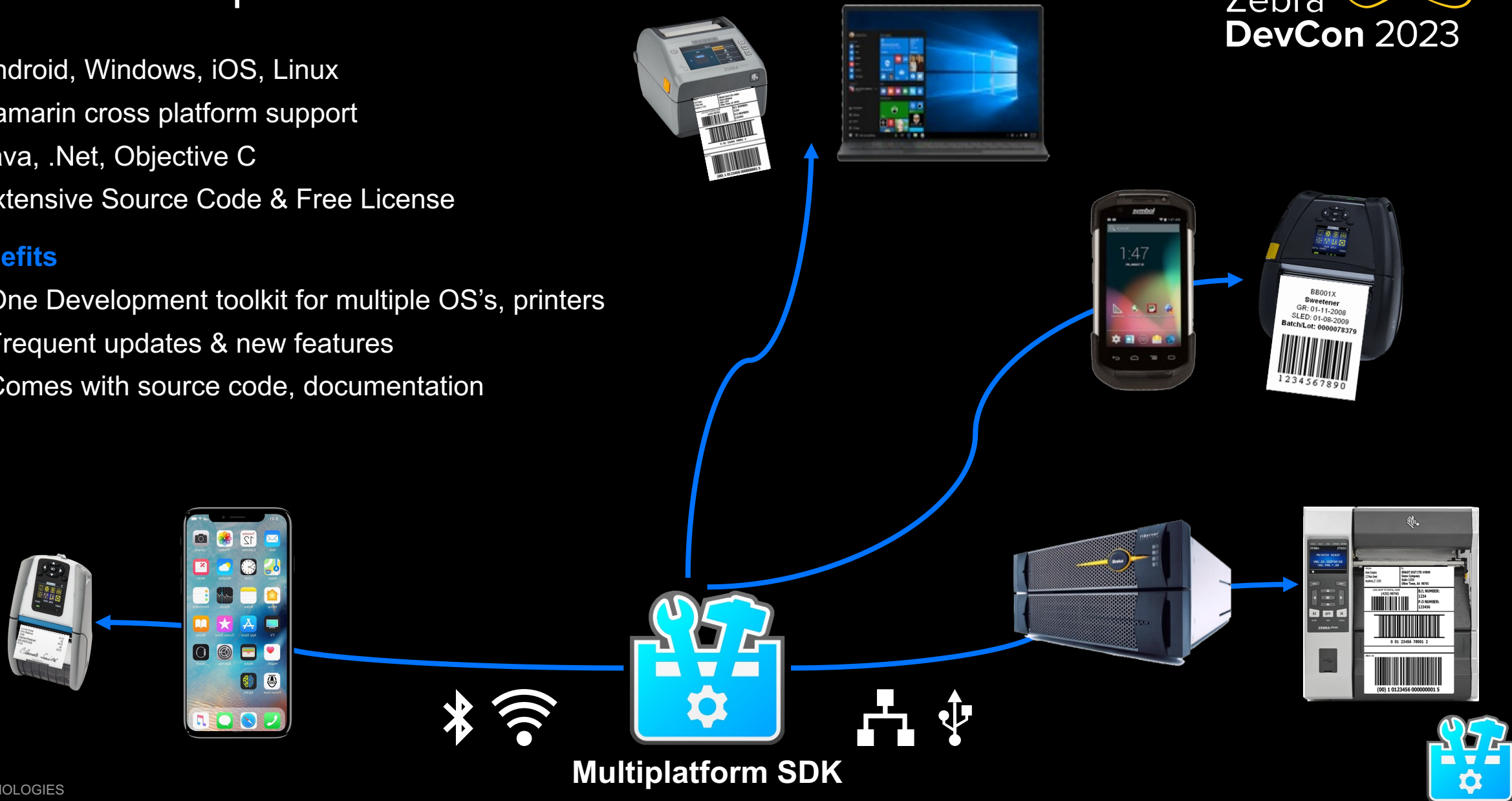


Link-OS Multiplatform SDK

- Android, Windows, iOS, Linux
- Xamarin cross platform support
- Java, .Net, Objective C
- Extensive Source Code & Free License

Benefits

1. One Development toolkit for multiple OS's, printers
2. Frequent updates & new features
3. Comes with source code, documentation



Link-OS Multiplatform SDK

Rich functionalities

Printer Discovery

USB, Bluetooth, BTLE
or Network

Printer Connectivity

USB, Bluetooth, BTLE
Network, WebSocket

Printer Status Checking

Errors, Warning,
Alerts

Printer Conversion

True Type to ZPL
Font

Graphics Conversion

PNG & BMP to ZPL
Graphics

Template Filling

Fill ZPL templates
with variable data

Printer Management

Create/send
profiles/Printer OS's

Command Mode

Scriptable command
line

Simplified Pairing

Using the Print
Touch feature

Developer Demos, Sample Code & Documentation

Both Source Code and Compiled demo code for commonly used features
Sample code for all major functions
Extensive API documentation that embeds within IDE platforms

Where to get it SDK

- [Link-OS™ Multiplatform SDK](#)
 - Android, iOS, Xamarin
 - PC (Java, .NET, .NET/Xamarin)
 - WebServices

Ways to find it:

- <https://www.zebra.com/us/en/software/printer-software/multiplatform-sdk.html>
- Launchpad

Online Documentation

- <http://techdocs.zebra.com/>



Link-OS SDK

The Link-OS SDK makes creating powerful printer apps simple and straightforward.

[Android](#) [Android BTLE](#) [Xamarin](#) [iOS](#) [PC](#) [Web Services](#) [Samples](#)

Printer Discovery

USB, Bluetooth®, BTLE, Network

- USB

- ZDesigner Driver USB Connection

```
UsbDiscoverer.getZebraDriverPrinters(discoveryandler);
```

- Direct USB Connection

```
UsbDiscoverer.getZebraUsbPrinters(discoveryandler);
```

- Bluetooth

```
BluetoothDiscoverer.findPrinters(discoveryandler);
```

- Bluetooth Low Energy

```
BluetoothLeDiscoverer.findPrinters(discoveryandler);
```

- Network

```
NetworkDiscoverer.findPrinters(discoveryandler);
```

```
1 package test.zebra.sdk.discovery.examples;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 import com.zebra.sdk.printer.discovery.DiscoveredPrinter;
7 import com.zebra.sdk.printer.discovery.DiscoveryException;
8 import com.zebra.sdk.printer.discovery.DiscoveryHandler;
9 import com.zebra.sdk.printer.discovery.NetworkDiscoverer;
10
11 public class NetworkDiscovererExample {
12     public static void main(String[] args) {
13         DiscoveryHandler discoveryHandler = new DiscoveryHandler() {
14             List<DiscoveredPrinter> printers = new ArrayList<DiscoveredPrinter>();
15
16             public void foundPrinter(DiscoveredPrinter printer) {
17                 printers.add(printer);
18             }
19
20             public void discoveryFinished() {
21                 for (DiscoveredPrinter printer : printers) {
22                     System.out.println(printer);
23                 }
24                 System.out.println("Discovered " + printers.size() + " printers.");
25             }
26
27             public void discoveryError(String message) {
28                 System.out.println("An error occurred during discovery : " + message);
29             }
30         };
31         try {
32             System.out.println("Starting printer discovery.");
33             NetworkDiscoverer.findPrinters(discoveryHandler);
34         } catch (DiscoveryException e) {
35             e.printStackTrace();
36         }
37     }
38 }
39
```

Printer Connection – Printing

USB, Bluetooth®, BTLE, Network

- USB

- ZDesigner Driver USB Connection

```
Connection conn =  
    new DriverPrinterConnection(printerName);
```

- Direct USB Connection

```
Connection conn =  
    new UsbConnection(usbDirectAddress);
```

- Bluetooth

```
Connection conn =  
    new BluetoothConnection(btMacAddress);
```

- Bluetooth Low Energy

```
Connection conn =  
    new BluetoothLeConnection(btMacAddress);
```

- Network

```
Connection conn =  
    new TcpConnection(ipAddr, portNum);
```

```
1 package test.zebra.sdk.comm.examples;  
2  
3 import com.zebra.sdk.comm.Connection;  
4 import com.zebra.sdk.comm.ConnectionException;  
5 import com.zebra.sdk.comm.TcpConnection;  
6 import com.zebra.sdk.printer.ZebraPrinter;  
7 import com.zebra.sdk.printer.ZebraPrinterFactory;  
8 import com.zebra.sdk.printer.ZebraPrinterLanguageUnknownException;  
9  
10 public class TcpConnectionExample {  
11  
12     public static void main(String[] args) throws Exception {  
13         new TcpConnectionExample().sendZplOverTcp("1.2.3.4");  
14         new TcpConnectionExample().sendCpclOverTcp("1.2.3.4");  
15         new TcpConnectionExample().printConfigLabelUsingDnsName("PrinterName");  
16     }  
17  
18     private void sendZplOverTcp(String theIpAddress) throws ConnectionException {  
19         // Instantiate connection for ZPL TCP port at given address  
20         Connection thePrinterConn = new TcpConnection(theIpAddress, TcpConnection.DEFAULT_ZPL_TCP_PORT);  
21  
22         try {  
23             // Open the connection - physical connection is established here.  
24             thePrinterConn.open();  
25  
26             // This example prints "This is a ZPL test." near the top of the label.  
27             String zplData = "\xA0F020,20^A0N,25,25^FDThis is a ZPL test.^FS^XZ";  
28  
29             // Send the data to printer as a byte array.  
30             thePrinterConn.write(zplData.getBytes());  
31         } catch (ConnectionException e) {  
32             // Handle communications error here.  
33             e.printStackTrace();  
34         } finally {  
35             // Close the connection to release resources.  
36             thePrinterConn.close();  
37         }  
38     }  
39 }
```

Printer Status Connection – Not for Printing

Bluetooth®, BTLE, Network

- Bluetooth

```
Connection statusConn =  
    new BluetoothStatusConnection(btMacAddress);
```

- Bluetooth Low Energy

```
Connection statusConn =  
    new BluetoothLeStatusConnection(btMacAddress);
```

- Network

```
Connection statusConn =  
    new TcpStatusConnection(ipAddr,  
        TcpStatusConnection.DEFAULT_STATUS_TCP_PORT);
```

Connection for Status & SGD (JSON) Only

```
1  import com.zebra.sdk.bluetooth.BluetoothLeStatusConnection;  
2  import com.zebra.sdk.comm.Connection;  
3  import com.zebra.sdk.comm.ConnectionException;  
4  import com.zebra.sdk.printer.SGD;  
5  
6  import android.app.Activity;  
7  import android.content.Context;  
8  import android.os.Bundle;  
9  
10 public class BluetoothLeStatusConnectionExample extends Activity {  
11  
12     protected void onCreate(Bundle savedInstanceState) {  
13         super.onCreate(savedInstanceState);  
14  
15         String theBtMacAddress = "00:11:BB:DD:55:FF";  
16         Context context = getApplicationContext();  
17         sendJSONOverStatusChannel(theBtMacAddress, context);  
18     }  
19  
20     private void sendJSONOverStatusChannel(final String theBtMacAddress, final Context context) {  
21         new Thread(new Runnable() {  
22             public void run() {  
23                 Connection thePrinterConn = null;  
24                 try {  
25                     // Instantiate a status only connection for given Bluetooth&reg; MAC Address.  
26                     thePrinterConn = new BluetoothLeStatusConnection(theBtMacAddress);  
27  
28                     // Open the connection – physical connection is established here.  
29                     thePrinterConn.open();  
30  
31                     // This sends down JSON to the status channel to retrieve the 'appl.name' setting  
32                     String firmwareVersion = SGD.GET("appl.name", thePrinterConn);  
33  
34                     System.out.println("The firmware version is : " + firmwareVersion);  
35                 } catch (Exception e) {  
36                     // Handle communications error here.  
37                     e.printStackTrace();  
38                 } finally {  
39                     // Close the connection to release resources.  
40                     if (null != thePrinterConn) {  
41                         try {  
42                             thePrinterConn.close();  
43                         } catch (ConnectionException e) {  
44                             e.printStackTrace();  
45                         }  
46                     }  
47                 }  
48             }  
49         }).start();  
50     }  
51 }  
52 }
```

Multichannel Connection

Status Channel and Printing Channel

- Bluetooth

 - Classic

```
MultichannelConnection mConn = new  
    MultichannelBluetoothConnection(btMacAddress);
```

 - BTLE

```
MultichannelConnection mConn = new  
    MultichannelBluetoothLeConnection(btMacAddress);
```

- Network (TCP)

```
MultichannelConnection mConn = new  
    MultichannelTcpConnection(ipAddr);
```

- Network (WebSocket)

```
MultichannelConnection mConn = new  
    MultichannelRemoteConnection(uniqueId);
```

```
1 private void runMultichannelDemo() {  
2     hasPrintJobFinished = false;  
3  
4     final MultichannelBluetoothLeConnection multichannelConnection = new MultichannelBluetoothLeConnection(btMacAddress);  
5  
6     try {  
7         multichannelConnection.open();  
8  
9         new Thread(new Runnable() {  
10            public void run() {  
11                int statusQueryCount = 1;  
12                List<String> odometerSettings = Arrays.asList("odometer.total_label_count", "odometer.total_print_length");  
13                PrinterLanguage pl = null;  
14                LinkOsInformation linkOsVersion = null;  
15                try {  
16                    pl = PrinterLanguage.getLanguage(SGD.GET("device.languages", multichannelConnection));  
17                    linkOsVersion = new LinkOsInformation(SGD.GET("appl.link_os_version", multichannelConnection));  
18                } catch (Exception e1) {  
19                }  
20                if (pl != null && linkOsVersion != null) {  
21                    try {  
22                        while (multichannelConnection.isConnected() && !hasPrintJobFinished) {  
23  
24                            long startTime = System.currentTimeMillis();  
25                            final Map<String, String> odometerValues =  
26                                new SettingsValues().getValues(odometerSettings, multichannelConnection.getStatusChannel(), pl, linkOsVersion);  
27                            final long totalTime = System.currentTimeMillis() - startTime;  
28                            final int count = statusQueryCount++;  
29                            runOnUiThread(new Runnable() {  
30                                public void run() {  
31                                    updateGui(odometerValues, null, totalTime, count);  
32                                }  
33                            });  
34                        }  
35                    } catch (ZebraIllegalArgumentEception e) {  
36                        e.printStackTrace();  
37                    } catch (ConnectionEception e) {  
38                        e.printStackTrace();  
39                    }  
40                }  
41            }  
42        }).start();  
43  
44        new Thread(new Runnable() {  
45            public void run() {  
46  
47                try {  
48                    // Send the "^XA" to open the channel and sleep to hold the channel open while querying the  
49                    // status.  
50                    // Sleep for 2 seconds after sending the end of the label to let the user see the print job  
51                    // finish while querying the status.  
52                    multichannelConnection.getPrintingChannel().write(beginningOfLabel.getBytes());  
53                    DemoSleeper.sleep(5000);  
54                    multichannelConnection.getPrintingChannel().write(endOfLabel.getBytes());  
55                    DemoSleeper.sleep(2000);  
56                    hasPrintJobFinished = true;  
57                } catch (ConnectionEception e) {  
58                    e.printStackTrace();  
59                    hasPrintJobFinished = true;  
60                }  
61            }  
62        })  
63    }  
64  
65    }).start();  
66  
67    } catch (ConnectionEception e) {  
68        helper.showErrorDialogOnGuiThread(e.getMessage());  
69    } catch (Exception e) {  
70        helper.showErrorDialogOnGuiThread(e.getMessage());  
71    } finally {  
72        while (!hasPrintJobFinished) {  
73            DemoSleeper.sleep(100);  
74        }  
75        try {  
76            if (multichannelConnection != null) {  
77                multichannelConnection.close();  
78            }  
79        } catch (ConnectionEception e) {  
80            helper.showErrorDialogOnGuiThread(e.getMessage());  
81        }  
82    }  
83 }
```

Multichannel Connection – Cont.

Status Channel and Printing Channel

- SDK automatically uses the proper channel for certain API calls
- The status channel is automatically used for
 - `getCurrentStatus()`
 - `SGD.SET`, `SGD.GET` & `SGD.DO`

```
private void nonBlockingStatusReportingOverMultichannel(String theIpAddress) throws Exception {
    try {
        // Instantiate Multichannel connection for simultaneous printing and status reporting at given address
        Connection mChannelPrinterConn = ConnectionBuilder.build("TCP_MULTI:" + theIpAddress + ":9100:9200");

        // Opens the connection – physical connection is established here.
        mChannelPrinterConn.open();

        // Creates a Link-OS printing with the given connection
        ZebraPrinterLinkOs linkOsPrinter = ZebraPrinterFactory.getLinkOsPrinter(mChannelPrinterConn);

        // This is sent over the printing channel (9100 by default) and will block the printing channel until the
        // label format is completely sent.
        String labelFormatStartCommand = "^XA";
        linkOsPrinter.sendCommand(labelFormatStartCommand);

        String labelBody = "^F050,50^ADN,36,20^FDHello World!^FS";
        linkOsPrinter.sendCommand(labelBody);

        // This is sent over the status channel (9200 by default) and will return immediately even though the
        // printing channel is in use.
        // If a TcpConnection were used instead of a MultichannelTcpConnection, this would not be possible.
        PrinterStatus status = linkOsPrinter.getCurrentStatus();

        System.out.println("The printer PAUSED state is : " + status.isPaused);

        Thread.sleep(5000);
        // Send the end of label command to finish and print the label.
        String labelFormatEndCommand = "^XZ";
        linkOsPrinter.sendCommand(labelFormatEndCommand);

        // Close the connection to release resources.
        mChannelPrinterConn.close();
    } catch (ConnectionException e) {
        // Handle communications error here.
        e.printStackTrace();
    }
}
```

Printer Status

getCurrentStatus()

- Printing Channel

```
ZebraPrinter printer =  
    ZebraPrinterFactory.getLinkOsPrinter(printingConn);  
PrinterStatus status = printer.getCurrentStatus();
```

Note: Calling `getCurrentStatus()` on the printing channel might be blocked by an ongoing ZPL printing.

- Status Channel

```
ZebraPrinter printer =  
    ZebraPrinterFactory.getLinkOsPrinter(statusConn);  
PrinterStatus status = printer.getCurrentStatus();
```

- Multichannel

```
ZebraPrinter printer =  
    ZebraPrinterFactory.getLinkOsPrinter(multiConn);  
Or  
ZebraPrinter printer =  
    ZebraPrinterFactory.getLinkOsPrinter(multiConn.getStatusChannel());  
PrinterStatus status = printer.getCurrentStatus();
```

```
1  new Thread(new Runnable() {  
2      public void run() {  
3          int statusQueryCount = 1;  
4          List<String> odometerSettings = Arrays.asList("odometer.total_label_count", "odometer.total_print_length");  
5          try {  
6              ZebraPrinter printer = ZebraPrinterFactory.getLinkOsPrinter(multichannelConnection.getStatusChannel());  
7              while (multichannelConnection.getStatusChannel().isConnected() && !hasPrintJobFinished) {  
8                    
9                  long startTime = System.currentTimeMillis();  
10                 LinkOsInformation linkOsVersion = new LinkOsInformation(SGD.GET("appl.link_os_version", multichannelConnection));  
11                 final Map<String, String> odometerValues = new SettingsValues().getValues(odometerSettings,  
12                     multichannelConnection.getStatusChannel(), printer.getPrinterControlLanguage(), linkOsVersion);  
13                 final PrinterStatus myPrinterStatus = printer.getCurrentStatus();  
14                 final long totalTime = System.currentTimeMillis() - startTime;  
15                 final int count = statusQueryCount++;  
16                 runOnUiThread(new Runnable() {  
17                     public void run() {  
18                         updateGui(odometerValues, myPrinterStatus, totalTime, count);  
19                     }  
20                 });  
21             }  
22         } catch (ZebraIllegalArgumentException e) {  
23             e.printStackTrace();  
24         } catch (ConnectionException e) {  
25             e.printStackTrace();  
26         }  
27     }  
28 }  
29 }).start();
```

Printer Configuration

SGD (SET-GET-DO) or a File

- SGD.SET

```
// Set the device.languages on the printer to ZPL
SGD.SET("device.languages", "zpl", conn);
```

```
// Set the media.type on the printer to label
SGD.SET("media.type", "label", conn);
```

- SGD.GET

```
// Get the link_os version from the printer
String osVersion = SGD.GET("appl.link_os_version", conn);
```

```
// Get the firmware version from the printer
String fwVersion = SGD.GET("appl.name", conn);
```

- SGD.DO

```
// Reset the printer
SGD.DO("device.reset", "", conn);
```

- Configure with a file

- Put SGD configuration commands in a file.
- Send the file with the `SendFileContents()` API over the Printing Channel

```
Connection conn = null;
try {
    conn = discoveredPrinterUsb.getConnection();
    conn.open();
    ZebraPrinter printer = ZebraPrinterFactory.getLinkOsPrinter(conn);
    PrinterLanguage pl = printer.getPrinterControlLanguage();

    // Send the config file
    printer.sendFileContents(getConfigFilePath().getAbsolutePath());
} catch (ConnectionException e) {
    setStatus(e.getMessage() + e.getLocalizedMessage(), Color.RED);
} catch (Exception e) {
    // Do nothing
} finally {
    if (conn != null) {
        try {
            conn.close();
        } catch (ConnectionException e) {
            e.printStackTrace();
        }
    }
}
}
```

Printing Label

Recommended Common Workflow

1. Discover the Printer
2. Open the Connection
3. Check/Set the Language to ZPL
4. Check the Printer Status
5. Send the ZPL Label (or File) to the Printer
6. Check the Printer Status Again
7. Close the Connection

```
public void sendZplOverTcp(String theIpAddress) throws ConnectionException, InterruptedException {
    // Instantiate connection for ZPL TCP port at given address
    Connection conn = new TcpConnection(theIpAddress, TcpConnection.DEFAULT_ZPL_TCP_PORT);

    ZebraPrinter printer = null;

    try {
        // Open the connection – physical connection is established here.
        conn.open();

        // Set the language to ZPL
        SGD.SET("device.languages", "zpl", conn);

        // Get the printer instance
        printer = ZebraPrinterFactory.getLink0sPrinter(conn);

        // Check the status
        if (printer.getCurrentStatus().isReadyToPrint) {
            // Print a Hello World label.
            String zplHelloWorld = "^XA^F050,50^ADN,36,20^FDHello World!^FS^XZ";

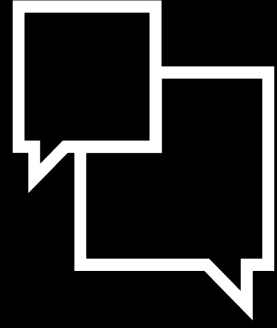
            // Send the data to printer as a byte array.
            conn.write(zplHelloWorld.getBytes());
        }
    } catch (ConnectionException e) {
        // Handle communications error here.
        e.printStackTrace();
    } finally {
        // Check the printer status again
        if (conn != null) {
            PrinterStatus status = printer.getCurrentStatus();
            while (status.isReceiveBufferFull ||
                status.isPartialFormatInProgress ||
                status.numberOfFormatsInReceiveBuffer != 0) {

                Thread.sleep(200);
                status = printer.getCurrentStatus();
            }

            // Close the connection to release resources.
            conn.close();
        }
    }
}
```


Resources

- Link-OS Multiplatform SDK
 - <https://www.zebra.com/us/en/support-downloads/printer-software/link-os-multiplatform-sdk.html>
- Online Documentation
 - <https://techdocs.zebra.com/link-os/>
- Samples on GitHub
 - Android Samples: <https://github.com/ZebraDevs/LinkOS-Android-Samples>
 - iOS Samples: <https://github.com/ZebraDevs/LinkOS-iOS-Samples>
 - Other Samples: <https://github.com/ZebraDevs/Zebra-Printer-Samples>



Questions

Zebra DevCon 2023



Thank You

ZEBRA and the stylized Zebra head are trademarks of Zebra Technologies Corp., registered in many jurisdictions worldwide. All other trademarks are the property of their respective owners.
©2023 Zebra Technologies Corp. and/or its affiliates. All rights reserved.

