# Zebra Web Printing Solutions

# Overview

To more effectively work across multiple operating systems and use the cloud, developers are increasingly building web-based or hybrid (web / native OS) apps.

Unfortunately, many operating systems limit the connections between a browser and peripheral devices, such as a printer. This presents a significant challenge to developers.

This Zebra Application Note document outlines the following options that enable you to create apps with bi-directional communications between your app and the printer.

- TCP/IP Back – End
- Cloud Connect
- Browser Print
- Enterprise Browser
- URL Schema

Each section contains:

- An overview of the solution
- Process Flow
- Use cases
- Reason to Use
- If available, reference information and sample code

# TCP/IP Back – End

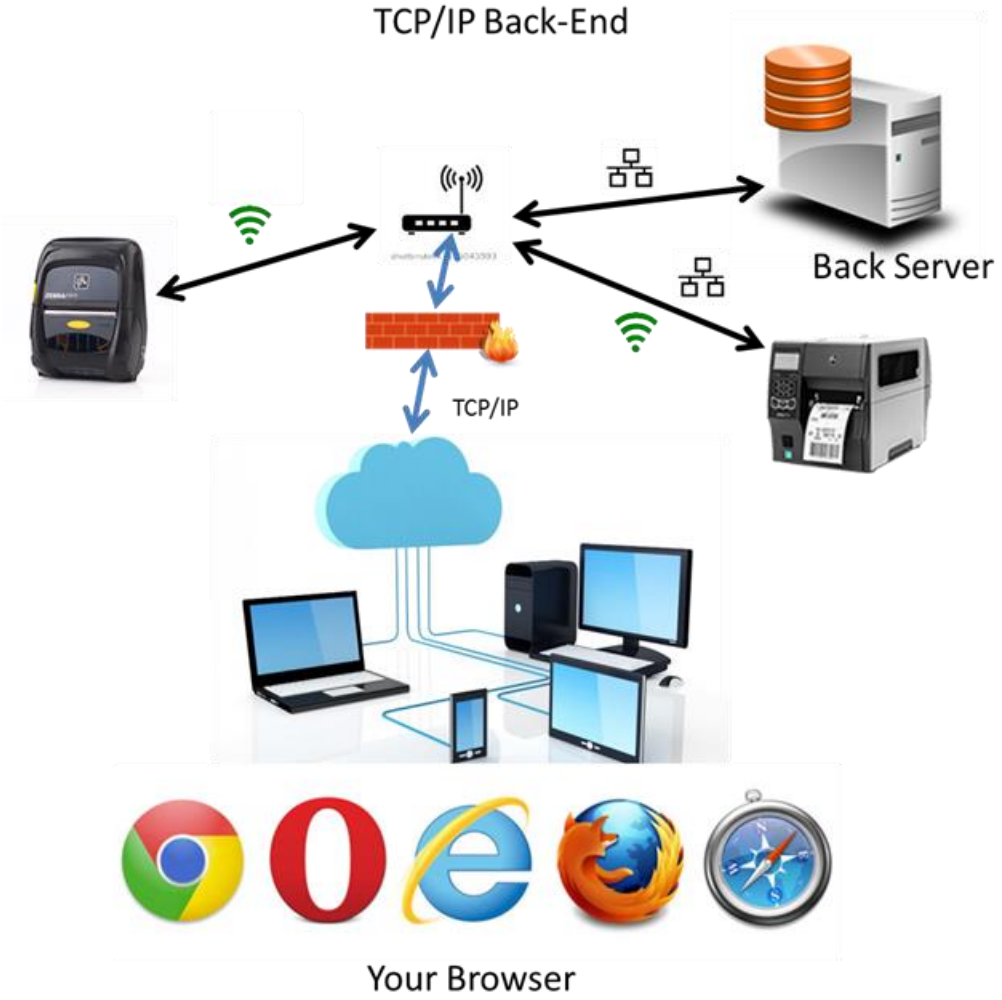
This web printing solution is one of the most used methods to enable printing applications with multiples printers connected to a network server.

The client makes the TCP/IP request through a web-browser app, but the action is executed at the back-end server. It is very important to highlight that the printers connect to the back-end server, rather than the front-end client-side.

An internal web-browser app prints over the internal network. The server manages the IP address connections to the printer.

Link-OS™ Multiplatform SDK provides a set of libraries that allow connecting Legacy and Link-OS™ printers by using this protocol with the Java libraries created for Zebra printers.

While recommended, it is not necessary to use the Link-OS Multiplatform SDK for back-end development. For Zebra examples using Java and C#, refer to the Sample Code section of this solution.




**Use Cases**

This solution assumes the following scenarios:

- The web-app can print from any Device (Mobile, PC, MAC, and Linux).
- The web-app will be printing from any web-browser that requires a back-end server.
- Printers are only accessible on an internal network.

**Reason to Use**

Recommended scenario for this solution:

- Internal intranet web apps where all the printers are networked.
- When you only want to connect to local networked printers from your website.

**Sample Code**

1. Java Example for Network Printing
2. **Java Network Connection CPCL**

```
import java.io.*;
import java.net.*;
class TCPClient
{
publicstaticvoid main (String argv[]) throws Exception
{
// The line below illustrates the default port 6101 for mobile printers 9100 is the default port number
// for desktop and tabletop printers
Socket clientSocket=new Socket("10.17.50.105",6101);

DataOutputStream outToServer = new DataOutputStream(clientSocket.getOutputStream() );
//The data being sent in the lines below illustrate CPCL

outToServer.writeBytes("! 0 200 200 203 1" + '\n' + "CENTER" + '\n');
outToServer.writeBytes("TEXT 0 3 10 50 JAVA TEST" + '\n' + "PRINT" + '\n');

clientSocket.close();
}
}
```

3. **TCP/IP Bi-directional Programming Example Using Sockets - C#**

```
/*********************************************
 * CONFIDENTIAL AND PROPRIETARY
 *
 * The source code and other information contained herein is the confidential and the exclusive property of
 * ZIH Corp. and is subject to the terms and conditions in your end user license agreement.
 * This source code, and any other information contained herein, shall not be copied, reproduced, published,
 * displayed or distributed, in whole or in part, in any medium, by any means, for any purpose except as
 * expressly permitted under such license agreement.
 *
 * Copyright ZIH Corp. 2010
 *
 * ALL RIGHTS RESERVED
 *********************************************/

//C# example illustrating bidirectional tcp/ip communications using System.Net.Sockets:
// Zebra Technical Support
using System;
using System.Data;
using System.Drawing;
using System.Windows.Forms;
using System.Net.Sockets;
using System.Net;
using System.IO;

namespace Socket_example
{
   public partial class Form1 : Form
   {
      public Form1()
```

```
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {

            System.Net.Sockets.TcpClient Zebraclient = new TcpClient();
            try
            {
              Zebraclient.SendTimeout = 1500;
              Zebraclient.ReceiveTimeout = 1500;
              //defining ip address and port number
              Zebraclient.Connect("10.17.50.202", 6101);
               //Zebraclient.Connect("10.17.50.202",9100);
            }
            catch
            {
               MessageBox.Show("Not connected, verify connection");
            }
            if (Zebraclient.Connected == true)
            {
              //send and receive illustrated below
               MessageBox.Show("Connected!!");
               NetworkStream mynetworkstream;
               StreamReader mystreamreader;
              StreamWriter mystreamwriter;
              mynetworkstream = Zebraclient.GetStream();
               mystreamreader = new StreamReader(mynetworkstream);
               mystreamwriter = new StreamWriter(mynetworkstream);
               string commandtosend = "! U1 getvar ""comm.baud""";
               //string commandtosend = "~hs";
               mystreamwriter.WriteLine(commandtosend);
               mystreamwriter.Flush();
               char[] mk = null;
               mk = new char[20];
               mystreamreader.Read(mk,0,mk.Length);
               string data1 = new string(mk);
               textBox1.Text = data1;
               Zebraclient.Close();
            }
        }
    }
}
```

## Cloud Connect

Cloud Connect is a set of libraries in Java that were created to be used in partner with WebLink features of the new Link-OS™ Printers.

The main feature of this technology is that it allows connecting Link-OS Zebra printers directly to the Internet through web-sockets.

The Zebra Link-OS Multiplatform SDK is built into the tool so that developers have access to many of the printing features and functionality. It allows you to create powerful cloud tools.

This solution allows you to connect to printers directly from a cloud server, addressing the gap with the TCP/IP back-end solution.

Cloud Connect Solution

### Use Cases

This solution assumes the following scenarios:

- The web-app can print from any Device (Mobile, PC, MAC, and Linux).
- Link-OS Printers can be connected through the Internet to cloud-hosted or on-premise servers.
- Link-OS Printers can be remote or local.
- Developers desire control of printing and device status via bi-directional communication.
- Users and developers desire an always ON connection.
- Users want to use Link-OS printers.
- Most printers are networked

### Reason to Use

Recommended scenario for this solution:

- Large multi-networked or externally hosted websites where full control of printers is needed at the server level.
- Device management, IoT, or full featured print job management system scenarios.

### Reference Information

Cloud Connect Overview

Creating Your Own WebLink Endpoint

Cloud Connectivity

### Sample Code

Refer to the Zebra Link-OS Multiplatform SDK WebLink section developer demos.

# Browser Print

This is an app that you can install on a PC to connect through a JavaScript library to Zebra printers in a bi-directional way.

The power of this solution is that it connects to Legacy and Link-OS printers. It can connect to multiple local ports simultaneously via USB and your web app will be portable to multiple OS's.

Download the client application – JavaScript library and sample code.


### Use Cases

This solution assumes the following scenarios:

- The web-app will be printing from PC.
- The web-app will connect the printers to the front-end client computer.
- Developers only need to make minimal modifications to their web-app to enable printing capabilities.
- Developers desire control of printing and device status via bi-directional communication.
- Developers need to connect Legacy and Link-OS printers to local machine from a web-browser app.

### Reason to Use

Recommended scenario for this solution:

- Websites with limited printing where the you do not know what devices, printers, communication paths, or browsers will be used by your customers.
- Web apps that want to print to USB printers connected to the end user's browser.

### Reference Information

Download – Includes Client App, JavaScript library, User Guide, API docs, and Sample code.

# Enterprise Browser

Enterprise Browser is a HTML5 web browser, like Google Chrome, that is available for many Zebra Mobile Computers.

It gives you access to many of the hardware features like the scanner, camera, and alarms that most browsers block. The Zebra Link-OS Multiplatform SDK is also built into it, so print features are also accessible via JavaScript API's, very similar to RhoMobile API's.


**Use Cases**

This solution assumes the following scenarios:

- The web-app prints from a Zebra Android or Windows device.
- You are willing to make a few modifications to your web-app to enable printing capabilities.
- You desire control of the printing and device status.
- The development will be in JavaScript.

**Reason to Use**

Recommended scenario for this solution:

- Web apps that intend to take full use of Zebra capabilities including scanning, printing, security, and/or other device features.

**Reference Information**

Enterprise Browser 1.8 Overview

Enterprise Browser API Reference

**Sample Code**

Printing Tutorial and Sample Code

# URL Schema

This is a solution that enables application developers to print within websites via native mobile applications.

It is a way to create an intent from within a webpage that calls into a printing app. The expectation is that there is an app available to take the intent and process it as a print job request.

Zebra has several partners who have these types of apps available. For details, refer to Zebra Validation Program .

The native app can use the full set of communication types from Bluetooth to Wi-Fi or NFC, to communicate with the printer.


### Use Cases

This solution assumes the following scenarios:

- The web-app primarily prints from an Android or iOS device.
- You desire a simple and easy integration to your web-app.
- Your customers are willing to have an additional printing app installed on their devices.
- You are willing to use 3rd party apps in conjunction with your web-app, or create your own native app.

### Reason to Use

Recommended scenario for this solution:

- Mobile focused web apps where the website developer does not desire complete control of the printer.

### Reference Information

URL schemes for iOS and Android (1/2)

Zebra Validated apps that you can use with URL schemas in your webpage:

- MOBI PRINT - Mobile Printing Made Easy
- centsoftware - Mobile printing tools and applications

## See Also

- For any further information, sample code and solutions or to request further content, visit the Zebra Developer Portal.

## Document Control

| Version | Date | Description |
|---|---|---|
| 1 | February, 2016 | Initial release |
| 2 | August, 2016 | Changed the Web Print Driver name to Browser Print. Added the “Reason to Use” topic to each section. |
| 3 | September, 2016 | Clarified Browser Print section to show current capabilities. |
| 4 | 2nd October 2017 | Updated URLs |

## Disclaimer

All links and information provided within this document are correct at time of writing.

Created for Zebra Global ISV Program by Zebra Development Services.