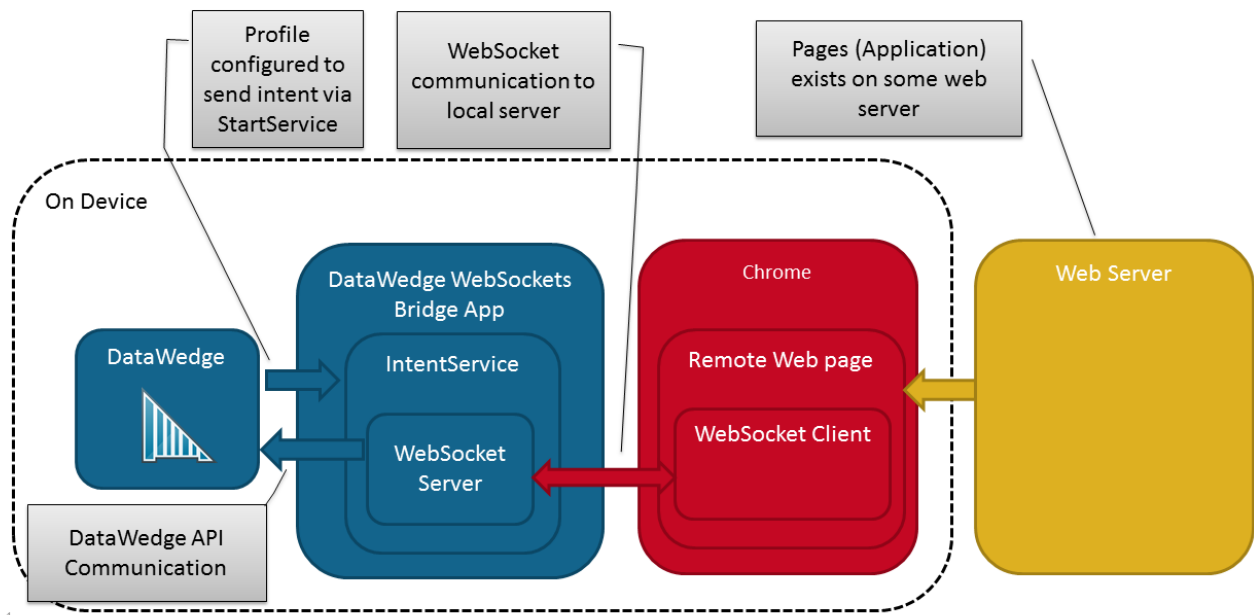


DARRYN CAMPBELL

Mobile computing and enterprise software development



DataWedge to WebSockets Bridge

11th June 2017 4 By DARRYN CAMPBELL

This blog aims to describe a very specific scenario, to interact with Zebra's [DataWedge](#) product with JavaScript through a browser on the device.

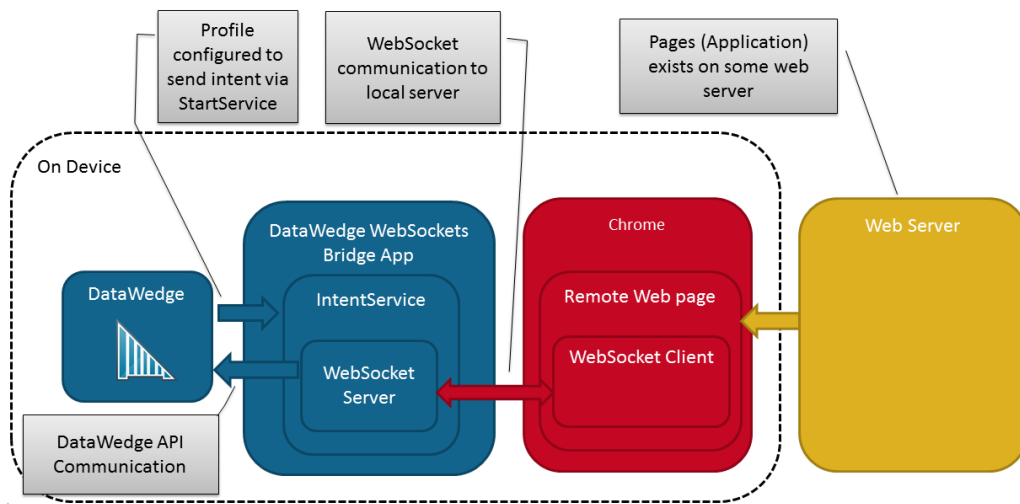
Why would you want to do this?

On Zebra devices, there are a number of options to integrate with the device hardware:

- [Enterprise Browser](#) is the primary means to control data capture on Zebra mobile computers via JavaScript but it only runs on Zebra devices and is a paid product.
- A JavaScript framework like Cordova [could be used](#) but JavaScript frameworks *can* add bloat to an application and complicate the build process making maintenance more difficult.
- The application could be written to run entirely within the web browser and interact with the device scanner through DataWedge [keystroke output](#). This is a great cross-platform solution which should work universally but occasionally differences in handling key presses across the different browser vendors and OS versions might make this approach unworkable. This interaction is also one way and cannot make use of the [DataWedge API](#).

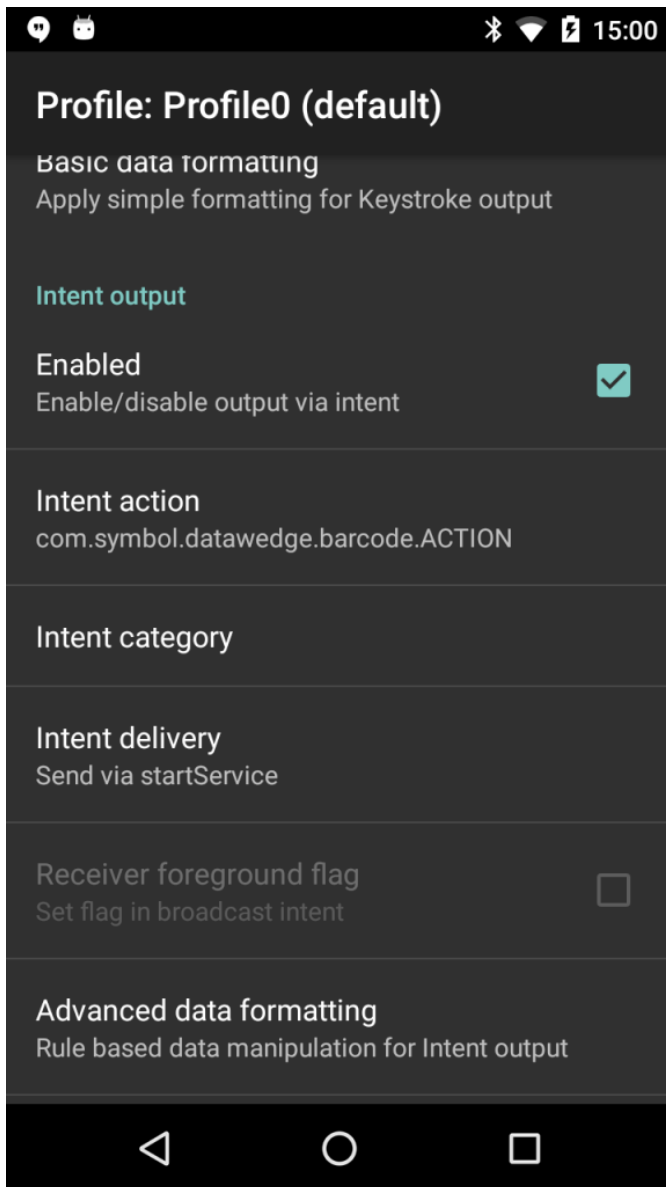
So, enter an additional way to interact with DataWedge through JavaScript, utilising WebSockets.

The architecture is as follows (Note, the browser can be Chrome or any WebSocket capable browser):



DataWedge

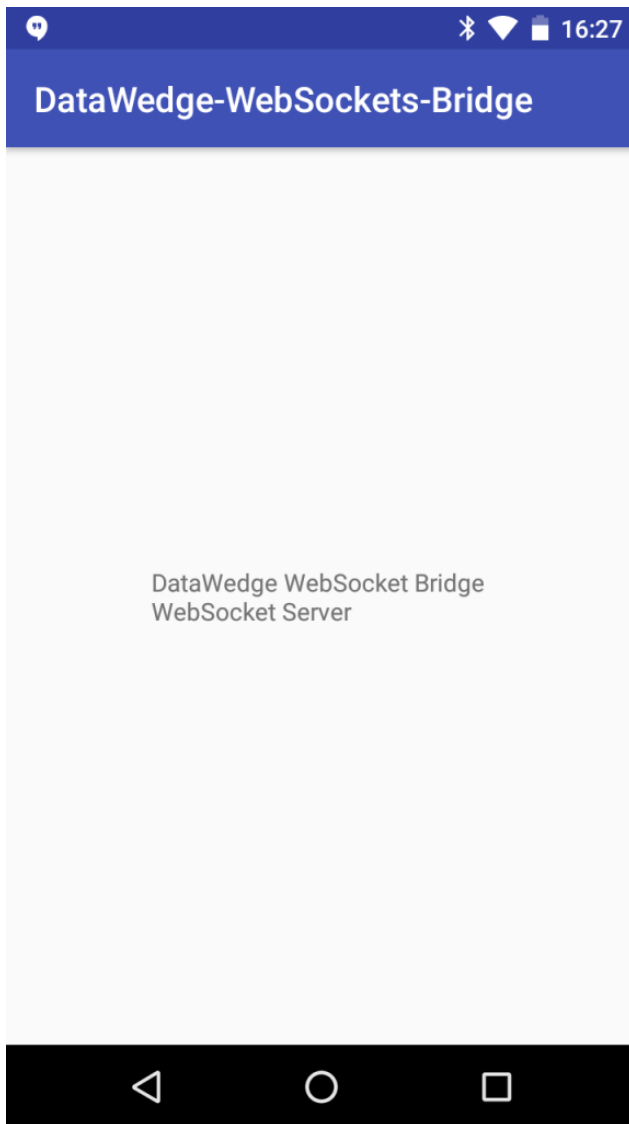
DataWedge is responsible for all data acquisition and a **profile** is configured to send scans via an **Intent**. Note that in my example I send the DataWedge intent via `startService` for simplicity since the recipient is using an `IntentService` which can be reused.



The above screenshot shows the Intent output portion of the DataWedge profile. Note we are sending Intents with the action 'com.symbol.datawedge.barcode.ACTION' and the default category.

Bridge application

The data capture intents sent in response to barcode scans are received by what I have termed the DataWedge WebSockets bridge application, this application is available on github under MIT license here: <https://github.com/darryncampbell/datawedge-websockets-bridge>. The application hosts a WebSocket server bound to the localhost and acts as a hub to route traffic between DataWedge and the application: barcode scans are received at the IntentService and forwarded over WebSocket to a connected WebSocket client. Requests to control DataWedge via the [DW API](#) are received over WebSocket from connected clients and forwarded to DataWedge via a broadcast intent.



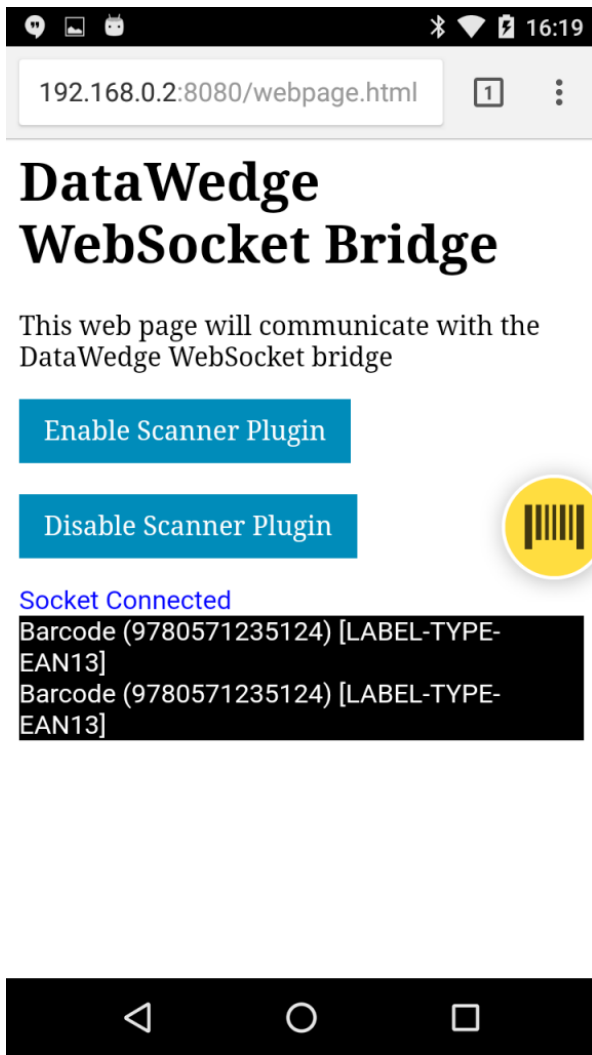
The bridge application has a very simple UI which is not designed to be used, the entirety of the application logic is contained within its service which is started when the application is started, when the device is booted or when a scan is received.

Web application

The end user application runs entirely within a web page and whilst shown as using Chrome in the diagram, it will work with any web browser capable of opening and maintaining a WebSocket connection to localhost:

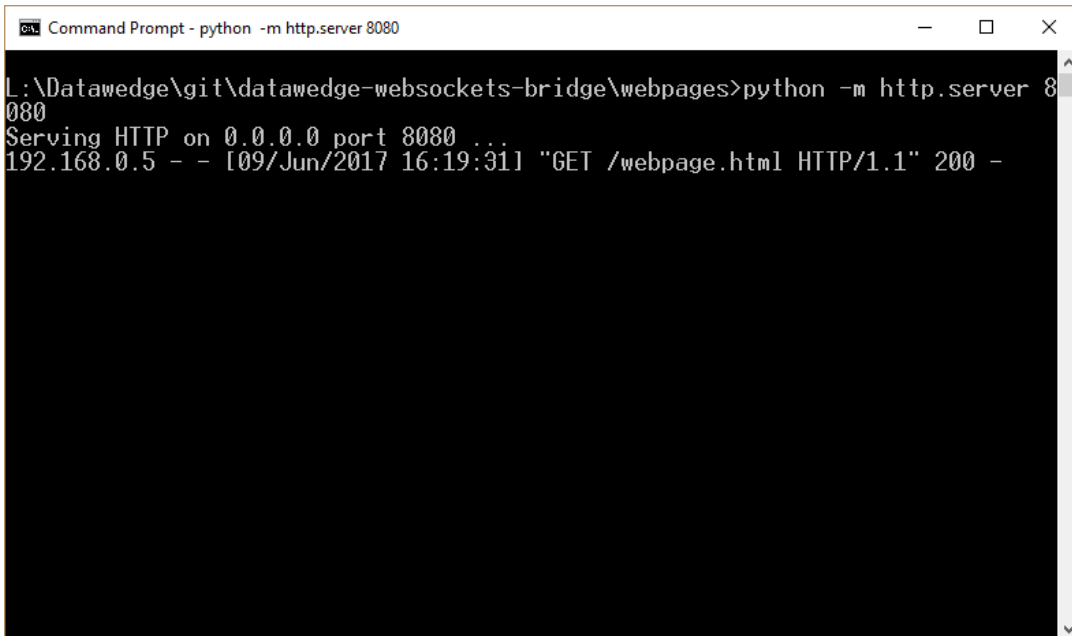
```
var address = "127.0.0.1";  
var port = "12345";  
ws = new WebSocket("ws://" + address + ":" + port + "/");
```

WebSockets are used to both receive data scanned as well as send commands to DataWedge. A simple sample page is hosted in the same repository as the bridge application, <https://github.com/darryncampbell/datawedge-websockets-bridge/blob/master/webpages/webpage.html> and shows how to receive and send at a basic level. The WebSocket connection would expect to be re-established on each new page load.



The simple sample page is shown above: two buttons are exposed to [enable or disable the scanner laser](#) through the DataWedge API. Note that I have very deliberately used the version of this API compatible with DataWedge 6.0 and higher, ignoring the new 6.3 API which is compatible with fewer devices at the time of writing. Received barcode scans are shown with a black background; if you do not see any barcode data being received then please ensure DataWedge is correctly configured on your device as explained earlier.

The page is being hosted from a local server which in my case a simple python server but obviously in a real deployment this could be any external host.



```
Command Prompt - python -m http.server 8080
L:\Datawedge\git\datawedge-sockets-bridge\webpages>python -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 ...
192.168.0.5 - - [09/Jun/2017 16:19:31] "GET /webpage.html HTTP/1.1" 200 -
```

Conclusion

Whilst the described principle for interacting with DataWedge via WebSockets will definitely work (at least it did when I tested it on my TC51 GMS Marshmallow device) some may be put off by running a WebSocket server internally on the device for reasons of security or performance. You could consider a more complex architecture where both the bridge application and web application act as WebSocket clients and the WebSocket server is some NodeJS app resident on a server somewhere, but I have not tried that.

There are also easier techniques e.g. Enterprise Browser or DataWedge [keystroke output](#) as I described at the top of this blog and **these should be used in preference to the WebSocket technique where possible**, but the easier techniques may not suit everybody's use case.

Steps to use the demo:

1. Configure a DataWedge profile as described earlier and ensure the profile is active when you launch the browser
2. Install the bridge application and either launch it or scan a barcode to start the WebSocket server
3. Start a web server and host the demo page
4. Navigate to the demo webpage on the device using Chrome or a similar browser capable of WebSockets
5. Scan a barcode, if everything is set up correctly you should see scans appear on the page. If you do not see scans, check the DataWedge setup and whether the WebSockets are connected, the status is shown in blue on the page
6. Disable and re-enable the laser using the buttons on the page, again if everything is set up correctly the buttons should affect the laser.

Share this:



Related

[AppForum 2019: EMEA \(Warsaw\)](#)
11th April 2019
In "Speaking"

[AppForums 2019: NALA \(Las Vegas\)](#)
27th August 2019
In "Speaking"

[Controlling DataWedge via Key Presses](#)
7th July 2017
In "Zebra Technologies"

Category [Zebra Technologies](#)

Tags [DataWedge](#)

4 Comments



Gabriele Tassoni says:

22nd October 2019 at 1:27 pm

Hi,

is this bridge working also with the RFID part of the DataWedge? Also for the RFID read, the datawedge still uses an intent based approach, so it looks like it could be compatible, I just expect a slightly different object coming through the websocket on the javascript part. anyway, in few days I'm checking it.

Reply



darryncampbell says:

28th October 2019 at 8:54 am

Hi, sorry, I missed this comment! I have never tried the bridge with the RFID plugin (the bridge actually predates the plugin) but it should work in principle, though as you say you would need to make changes to which APIs are called and how the return data is processed. The only DataWedge APIs related to RFID seem to be <https://techdocs.zebra.com/datawedge/7-4/guide/api/softfidtrigger/> and <https://techdocs.zebra.com/datawedge/7-4/guide/api/setconfig/> so you don't have quite the same level of control as you do over the barcode scanner but for most developers it should be sufficient.

Reply



Nil Micola says:

28th October 2019 at 4:08 pm

Hi Darryn,

First of all thank you for your work.

I was using this bridge on TC56 terminals but this does not work after updating the devices to Android 8.1 Oreo (DataWedge 7.3) Do you know if we have to do some change on the DataWedge WebSocket Bridge?

Thank you in advanced.

[Reply](#)**darryncampbell** says:*29th October 2019 at 7:42 am*

@Nil I merged a pull request a few months back that reported to work around the background service limitations – <https://github.com/darryncampbell/datawedge-websockets-bridge/pull/3> so I recommend using the master branch. Unfortunately I am no longer actively maintaining this project

[Reply](#)

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name ***Email *****Website** **Notify me of follow-up comments by email.** **Notify me of new posts by email.**

SEARCH