Debugging Cordova Applications

Overview

Debugging is a process that consists of finding the cause of defects in applicative code and application user interface.

- Cordova applications consist of web-based resources such as HTML, JavaScript & CSS, and optional native code (written in Java, Objective-C, Swift, C#, ...).
- Native code can be debugged by using standard tools that are provided by the platform SDK, such as XCode, Android LogCat, or Microsoft Visual Studio.

This tutorial explores various approaches to debugging a Cordova application, whether running locally via an Emulator or Simulator, or while running in a physical mobile device.

> Learn more about Cordova debugging and testing in the Cordova website: Debugging applications (https://cordova.apache.org/docs/en/latest/guide/next/index.html#link-testing-on-a-simulator-vs-on-a-real-device).

Jump to:

Debugging with the IBM Mobile Browser Simulator

You can use IBM MobileFirst Platform Foundation's Mobile Browser Simulator (MBS) to preview and debug MobileFirst applications.
To use the MBS, open a **Command-line** window and run the command:

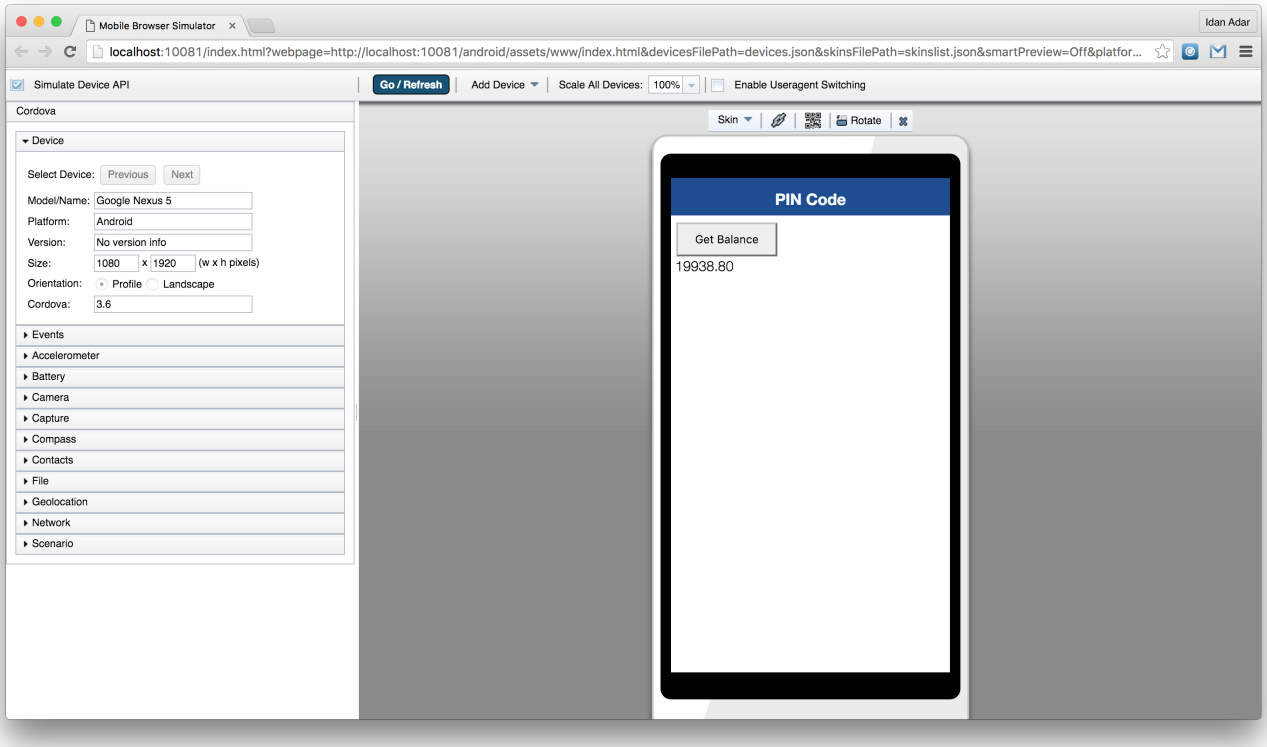
```
mfpdev app preview
```

If your application consists of more than one platform - specify the platform to preview:

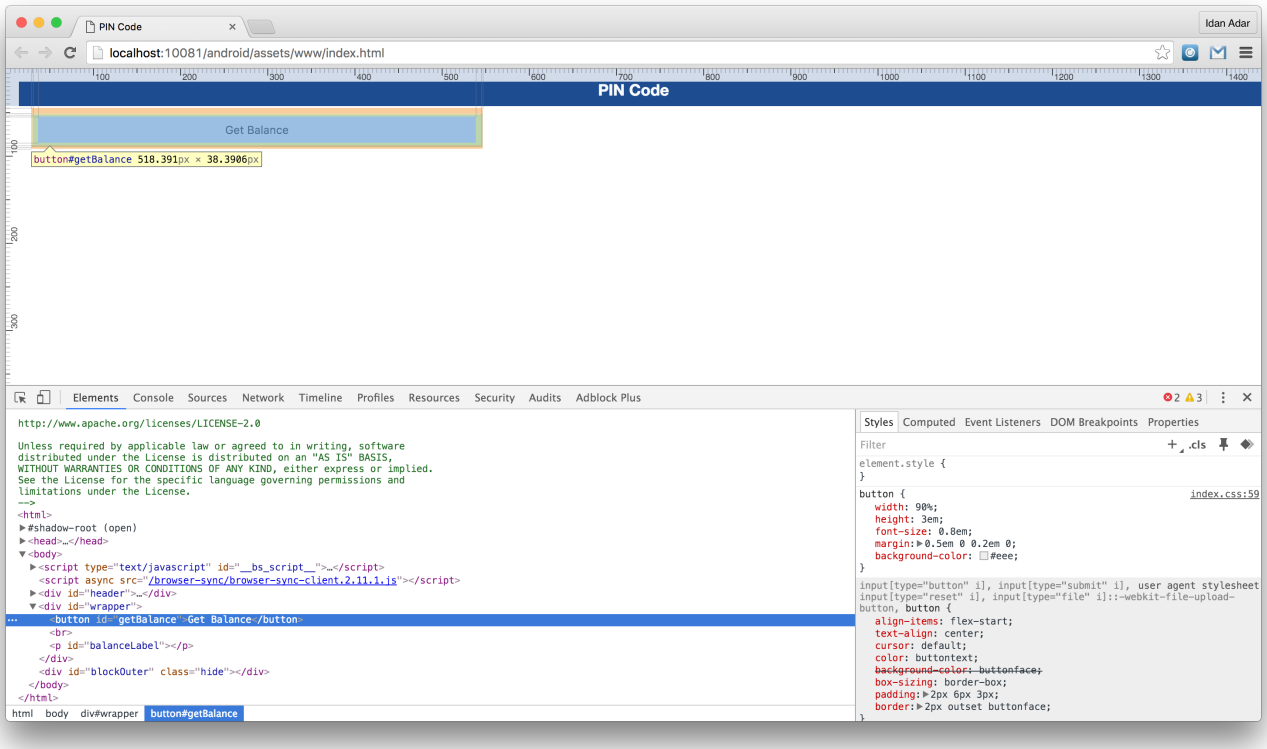
```
mfpdev app preview -p <platform>
```

> **Important:** The preview feature has several known limitations. Your application may not behave as expected in the preview. For example, it bypasses security features using a confidential client, so challenge handlers are not triggered.

Mobile Browser Simulator



Simple Preview



> Learn more about the MobileFirst CLI in the Using MobileFirst CLI to manage MobileFirst artifacts (../using-mobilefirst-cli-to-manage-mobilefirst-artifacts) tutorial.

Debugging with Ripple

Apache Ripple™ is a web based mobile environment simulator for debugging mobile web applications. It lets you run a Cordova application in your browser and fake various Cordova features. For example, it can fake the camera API by letting you select a picture locally from your computer.

Installing Ripple

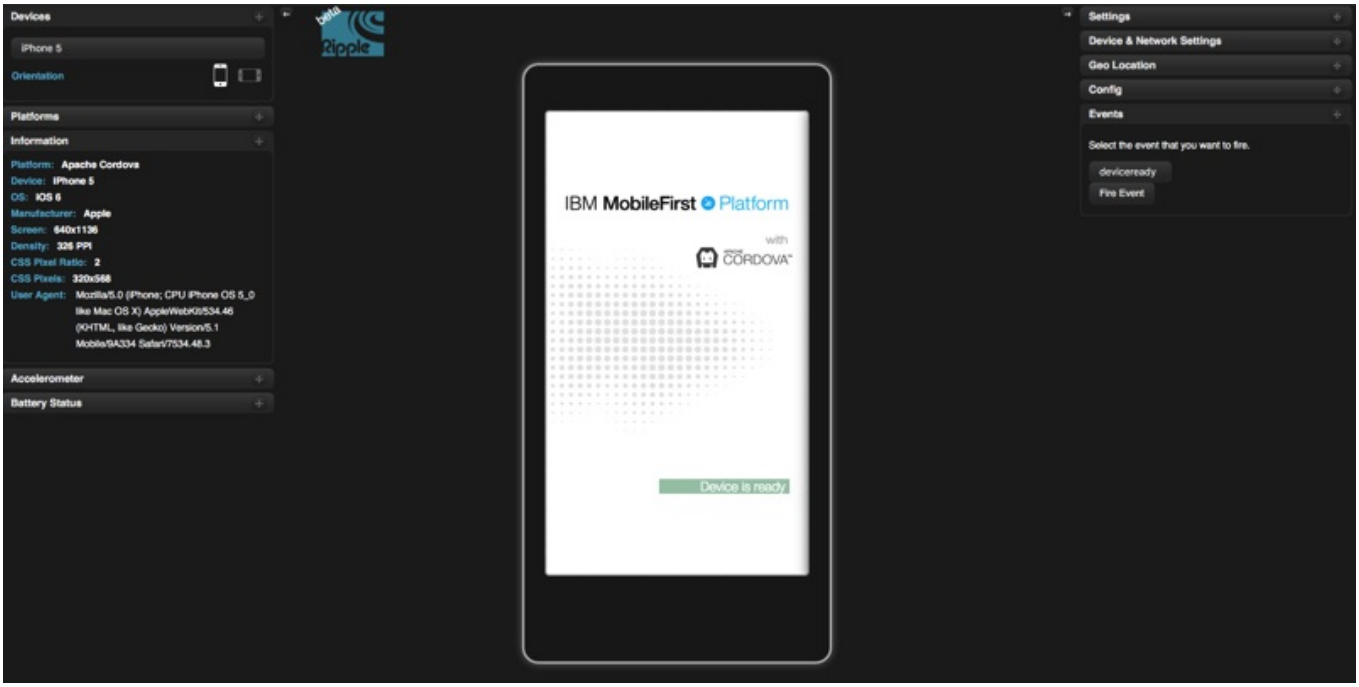
1. Download and install the latest version of Node.js (https://nodejs.org/en/). You can verify Node.js installation by typing `npm -v` in terminal.
2. Open terminal and type:

```
npm install -g ripple-emulator
```

Running application using Ripple

After Ripple is installed open terminal from your Cordova project location and type:

```
ripple emulate
```



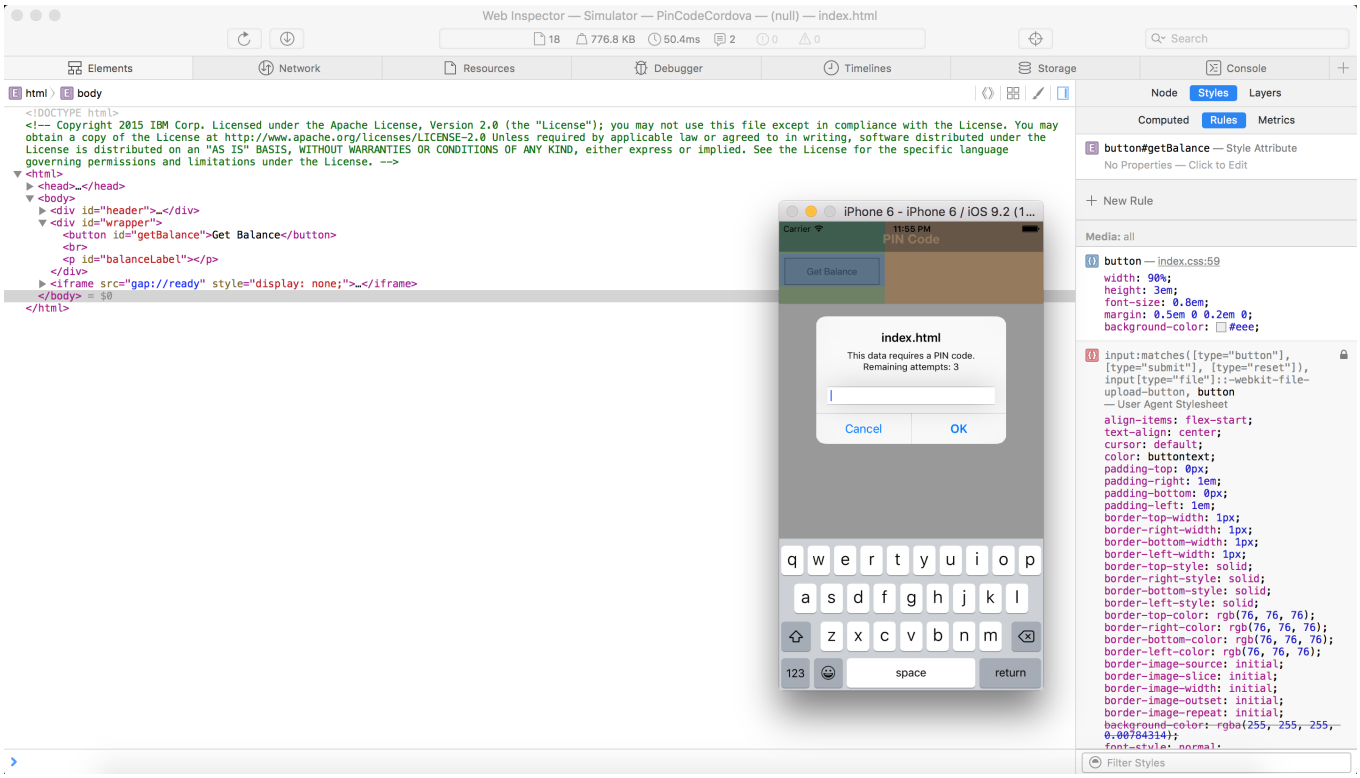
> More information about Apache Ripple™ can be found on the Apache Ripple page (http://ripple.incubator.apache.org/) or npm ripple-emulator page (https://www.npmjs.com/package/ripple-emulator).

Debugging with iOS Remote Web Inspector

Starting iOS 6, Apple introduced a remote Web Inspector (https://developer.apple.com/safari/tools/) for debugging web applications on iOS devices. To debug, make sure that the device (or iOS Simulator) has the **Private Browsing** option turned off.

1. To enable Web Inspector on the device, Tap **Settings > Safari > Advanced > Web Inspector**.
2. To start debugging, connect the iOS device to a Mac, or start the simulator.
3. In Safari, go to **Preferences > Advanced**, and select the **Show Develop menu in menu bar** checkbox.

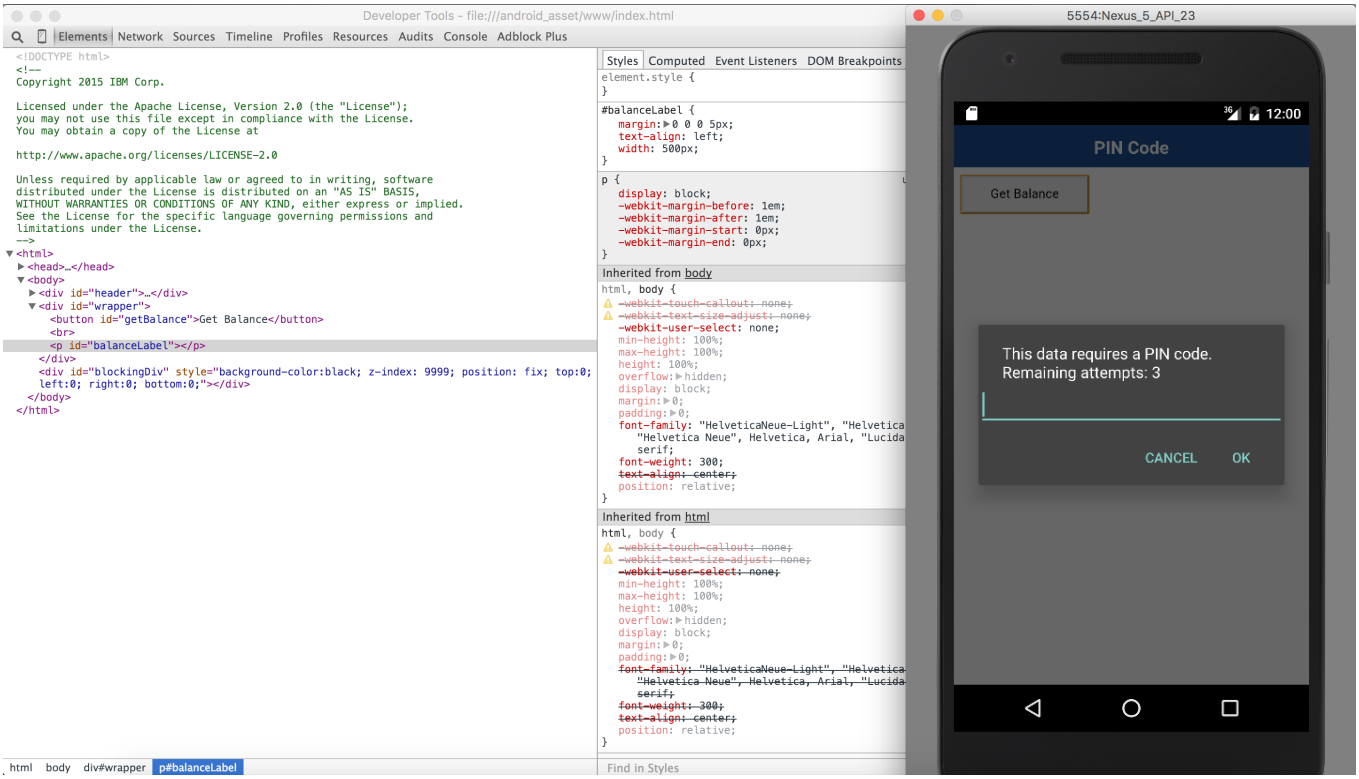
4. In Safari, select **Develop > [your device ID] > [your application HTML file]**.



Debugging with Chrome Remote Web Inspector

Using Google Chrome it is possible to remotely inspect web applications on Android devices or the Android Emulator.
This action requires Android 4.4 or later, Chrome 32 or later. Additionally, in the `AndroidManifest.xml` file, `targetSdkVersion = 19` or above is required. In the `project.properties` file, `target = 19` or above is required.

1. Start the application in the Android Emulator or a connected device.
2. In Chrome, enter the following URL in the address bar: `chrome://inspect`.
3. Press **Inspect** for the relevant application.



Debugging with IBM MobileFirst Logger

IBM MobileFirst Platform Foundation provides a `WL.Logger` object that can be used to print log messages.
`WL.Logger` contains several levels of logging: `WL.Logger.info`, `WL.Logger.debug`, `WL.Logger.error`.

> For more information, see the documentation for `WL.Logger` in the API reference part of the user documentation.

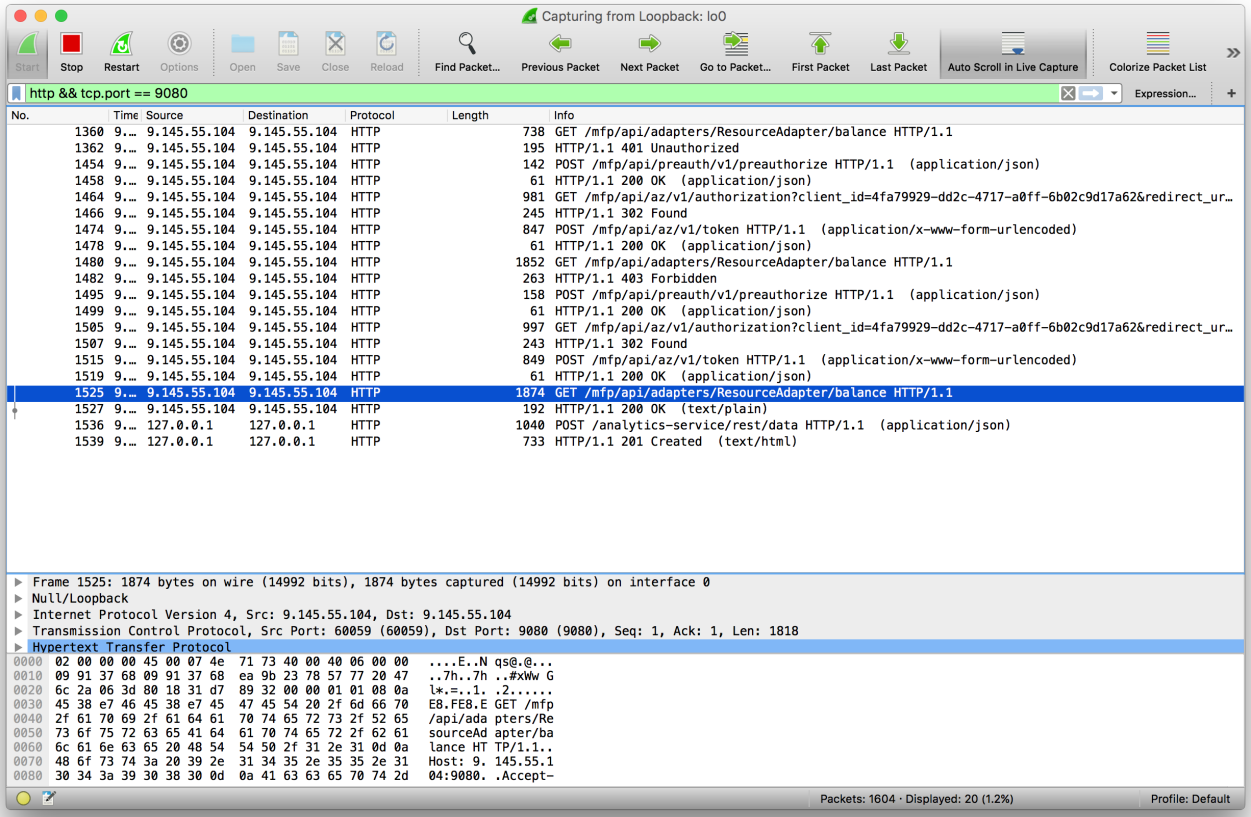
Inspecting the log:

- **Developer console** when previewing a platform using a Simulator or Emulator.
- **LogCat** when it is running on Android device
- **XCode Console** when it is running on an iOS device
- **Visual Studio Output** when it is running on a Windows devices.

Debugging with WireShark

Wireshark is a network protocol analyzer that can be used to see what happens in the network. You can use filters to follow only what is required.

> For more information, see the WireShark (http://www.wireshark.org) website.


Capturing from Loopback: lo0
Start
Stop
Restart
Options
Open
Save
Close
Reload
Find Packet...
Previous Packet
Next Packet
Go to Packet...
First Packet
Last Packet
Auto Scroll in Live Capture
Colorize Packet List
http && tcp.port == 9080
Expression...
No. Time Source Destination Protocol Length Info
1360 9... 9.145.55.104 9.145.55.104 HTTP 738 GET /mfp/api/adapters/ResourceAdapter/balance HTTP/1.1
1362 9... 9.145.55.104 9.145.55.104 HTTP 195 HTTP/1.1 401 Unauthorized
1454 9... 9.145.55.104 9.145.55.104 HTTP 142 POST /mfp/api/preauth/v1/preauthorize HTTP/1.1 (application/json)
1458 9... 9.145.55.104 9.145.55.104 HTTP 61 HTTP/1.1 200 OK (application/json)
1464 9... 9.145.55.104 9.145.55.104 HTTP 981 GET /mfp/api/az/v1/authorization?client_id=4fa79929-dd2c-4717-a0ff-6b02c9d17a62&redirect_ur...
1466 9... 9.145.55.104 9.145.55.104 HTTP 245 HTTP/1.1 302 Found
1474 9... 9.145.55.104 9.145.55.104 HTTP 847 POST /mfp/api/az/v1/token HTTP/1.1 (application/x-www-form-urlencoded)
1478 9... 9.145.55.104 9.145.55.104 HTTP 61 HTTP/1.1 200 OK (application/json)
1480 9... 9.145.55.104 9.145.55.104 HTTP 1852 GET /mfp/api/adapters/ResourceAdapter/balance HTTP/1.1
1482 9... 9.145.55.104 9.145.55.104 HTTP 263 HTTP/1.1 403 Forbidden
1495 9... 9.145.55.104 9.145.55.104 HTTP 158 POST /mfp/api/preauth/v1/preauthorize HTTP/1.1 (application/json)
1499 9... 9.145.55.104 9.145.55.104 HTTP 61 HTTP/1.1 200 OK (application/json)
1505 9... 9.145.55.104 9.145.55.104 HTTP 997 GET /mfp/api/az/v1/authorization?client_id=4fa79929-dd2c-4717-a0ff-6b02c9d17a62&redirect_ur...
1507 9... 9.145.55.104 9.145.55.104 HTTP 243 HTTP/1.1 302 Found
1515 9... 9.145.55.104 9.145.55.104 HTTP 849 POST /mfp/api/az/v1/token HTTP/1.1 (application/x-www-form-urlencoded)
1519 9... 9.145.55.104 9.145.55.104 HTTP 61 HTTP/1.1 200 OK (application/json)
1525 9... 9.145.55.104 9.145.55.104 HTTP 1874 GET /mfp/api/adapters/ResourceAdapter/balance HTTP/1.1
1527 9... 9.145.55.104 9.145.55.104 HTTP 192 HTTP/1.1 200 OK (text/plain)
1536 9... 127.0.0.1 127.0.0.1 HTTP 1040 POST /analytics-service/rest/data HTTP/1.1 (application/json)
1539 9... 127.0.0.1 127.0.0.1 HTTP 733 HTTP/1.1 201 Created (text/html)
Frame 1525: 1874 bytes on wire (14992 bits), 1874 bytes captured (14992 bits) on interface 0
Null/Loopback
Internet Protocol Version 4, Src: 9.145.55.104, Dst: 9.145.55.104
Transmission Control Protocol, Src Port: 60059 (60059), Dst Port: 9080 (9080), Seq: 1, Ack: 1, Len: 1818
Hypertext Transfer Protocol
0000 02 00 00 00 45 00 07 4e 71 73 40 00 40 06 00 00 ....E..N qs@.@...
0010 09 91 37 68 09 91 37 68 ea 9b 23 78 57 77 20 47 ..7h..7h ..#xWw G
0020 6c 2a 06 3d 80 18 31 d7 89 32 00 00 01 01 08 0a l*.=..1. .2......
0030 45 38 e7 46 45 38 e7 45 47 45 54 20 2f 6d 66 70 E8.FE8.E GET /mfp
0040 2f 61 70 69 2f 61 64 61 70 74 65 72 73 2f 52 65 /api/ada pters/Re
0050 73 6f 75 72 63 65 41 64 61 70 74 65 72 2f 62 61 sourceAd apter/ba
0060 6c 61 6e 63 65 20 48 54 54 50 2f 31 2e 31 0d 0a lance HT TP/1.1..
0070 48 6f 73 74 3a 20 39 2e 31 34 35 2e 35 35 2e 31 Host: 9. 145.55.1
0080 30 34 3a 39 30 38 30 0d 0a 41 63 63 65 70 74 2d 04:9080. .Accept-
Packets: 1604 · Displayed: 20 (1.2%)
Profile: Default