WebBroker on Raspberry Pi

Delphi & C++Builder on Raspberry Pi with Android

Jim McKeeth, Embarcadero Technologies
jim.mckeeth@embarcadero.com
Twitter: @JimMcKeeth | Blog: delphi.org/?p=3330

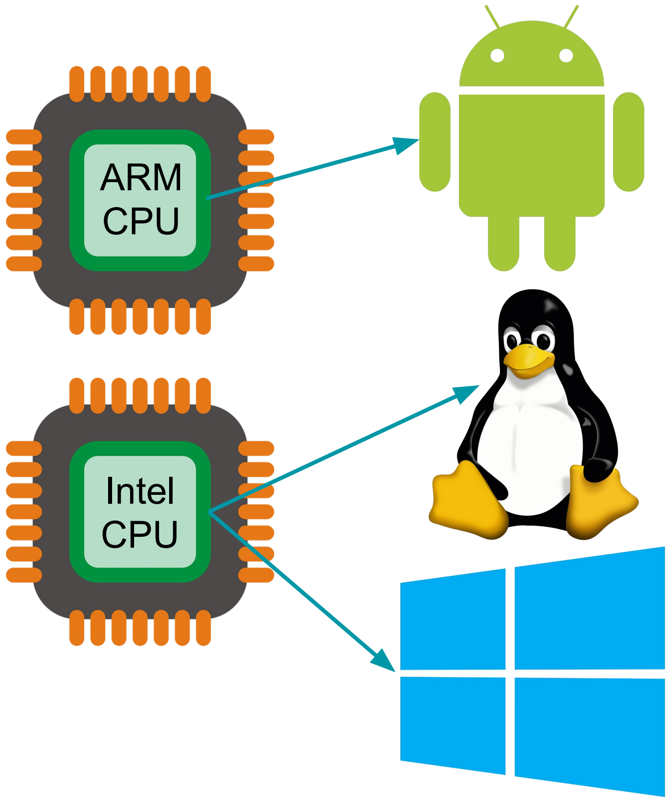
More info: delphi.org/?p=3330

Why WebBroker on Raspberry Pi 3?

- Raspberry Pi is a ubiquitous low cost computer costing $35 usd
- Small size and low cost makes it ideal to function as an edge node
  - Collecting information on site and sharing it online
- Raspberry Pi 3 has all these features (only one I've tested on RPi3)
  - Ethernet & Wi-Fi
  - Runs Android
  - HDMI output
  - Micro SD-Card storage
  - USB Host ports for additional peripherals (GPS, Bluetooth, Camera, Storage, etc.)
- Android is available via
  - Google Android Things (Tested, but uncertain future)
  - Emteria.os (Tested and promising future)
  - GeekTillItHurtz.com (Tested, but more DIY) geektillithertz.com/wordpress/?s=android+raspberry+pi+3

More info: delphi.org/?p=3330

Other Similar Platform Options



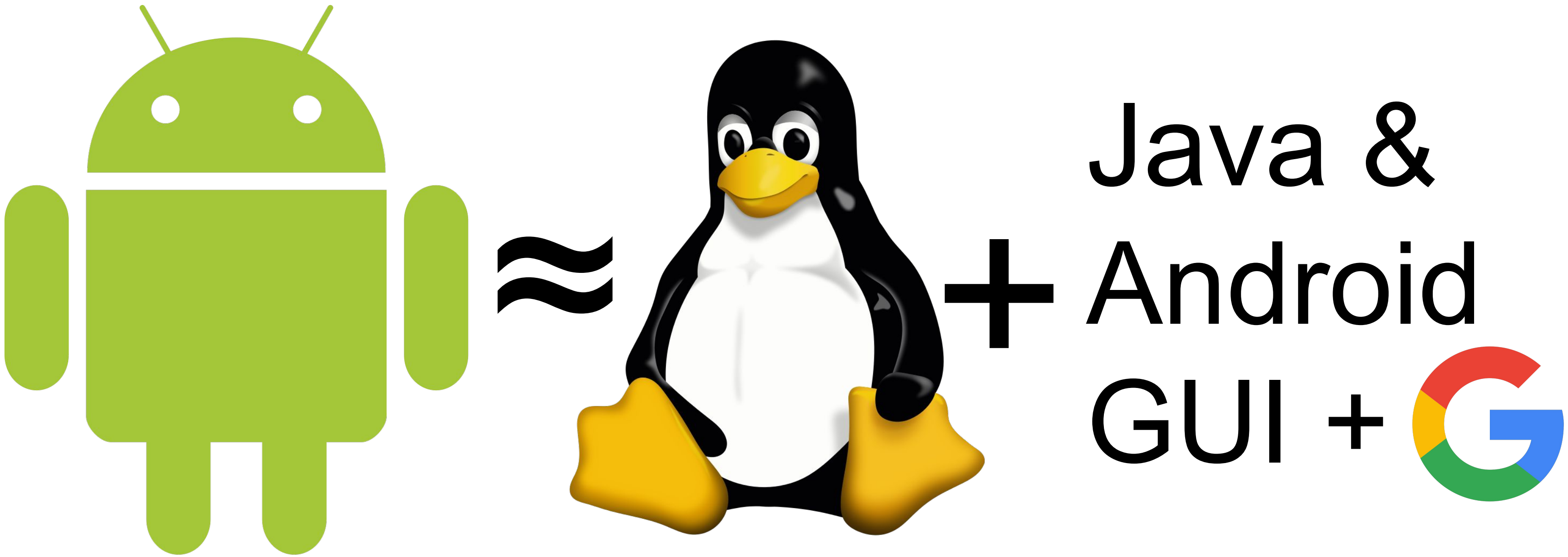
Android on ARM

- Raspberry Pi 3 Model B - raspberrypi.org
- Banana Pi - banana-pi.org
- BeagleBone Black - beagleboard.org/black
- Pine64.org
- ODROID from HardKernel.com
- Orange Pi - OrangePi.org

Linux or Windows on Intel

- LattePanda - lattepanda.com
- Intel Compute Stick - embt.co/IntelComputeStk
- Udoo x86 Ultra - udoo.org
- Up-Board - up-board.org
- MinnowBoard Turbot - minnowboard.org
- HummingBoard-Gate - solid-run.com
- Jaguar Board - jaguarboard.org

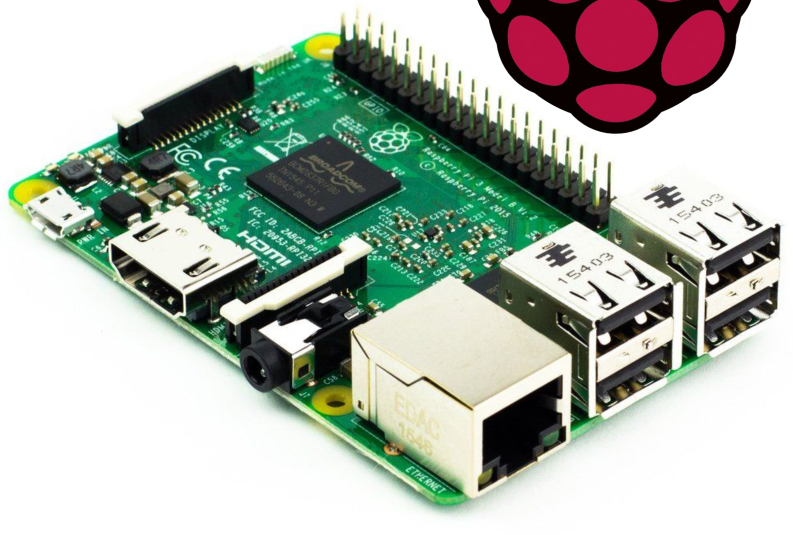
Android *is* Linux



Android is a layer on top of Linux with a Java based API

Raspberry Pi 3 Specifications

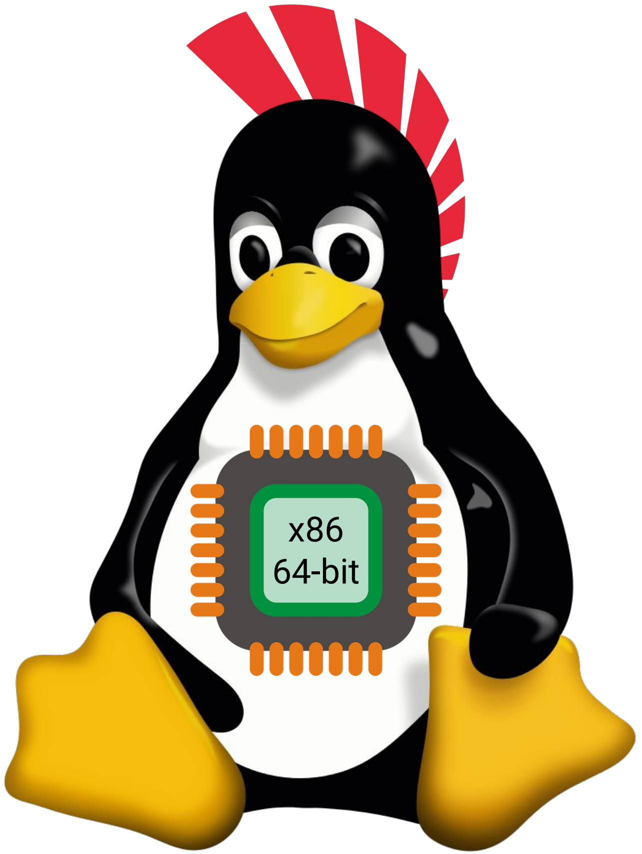
- Usually around $35 USD
- A 1.2GHz 64-bit quad-core ARMv8 CPU
- 802.11n Wireless LAN
- Bluetooth 4.1 & Bluetooth Low Energy (BLE)
- 1GB RAM & Micro SD card slot
- 4 USB ports, 40 GPIO pins, Camera interfa
- HDMI, Ethernet, audio jack & composite vid
- VideoCore IV 3D graphics core



raspberrypi.org/products/raspberry-pi-3-model-b/

Why not target Raspbian (Linux)?

- Delphi’s Linux support is for 64-bit x86
- Raspberry Pi 3 has an ARM CPU
- Delphi & C++Builder support Android on ARM
- Raspbian is a Linux based OS for RPi3
- If you want to target Linux then you need a 64-bit x86 based board



What is Android Things?

- An Android-based embedded operating system platform
- Announced at Google I/O 2015.
- Aimed for use with low-power and memory constrained IoT devices
- Currently on version 1.0.11 (April 2019)
  source.android.com/security/bulletin/2019-04-01
  developer.android.com/things/versions/releases
- Offers 2 developer platforms
  - NXP i.MX7D (More powerful & secure)
  - Raspberry Pi 3 Model B (More affordable)

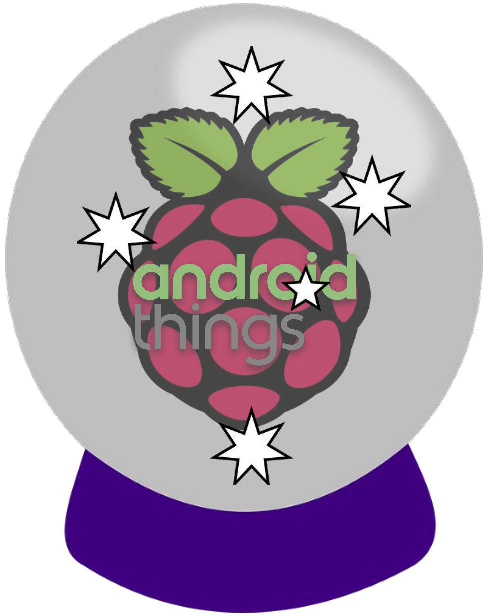
Using Android Things + RPi3 with RAD Studio

- Download the Android Things Setup Utility (you need a free Google account)
- Use it to create an SD Card (just use a generic image)
- Boot up the Raspberry Pi 3
- Connect a screen to see the IP address
  (or look on your router, etc.)
- Optionally connect keyboard & mouse to configure
- From Windows type `adb connect <ip address>`
  - You need **ADB** on your path
- Deploy to RPI3 via RAD Studio

developer.android.com/things/hardware/raspberrypi

Future of Android Things….

- Google announced Android Things refocusing on Feb 2019 android-developers.googleblog.com/2019/02/an-update-on-android-things.html
- There were two Android Things updates since then
- Not planning to support *production* releases of Android Things on NXP & RPi devices
- Still for “up to 100 devices for non-commercial use.”
- I’ve had really good luck with this, but since the future is uncertain I’m going to focus on **Emteria OS** here



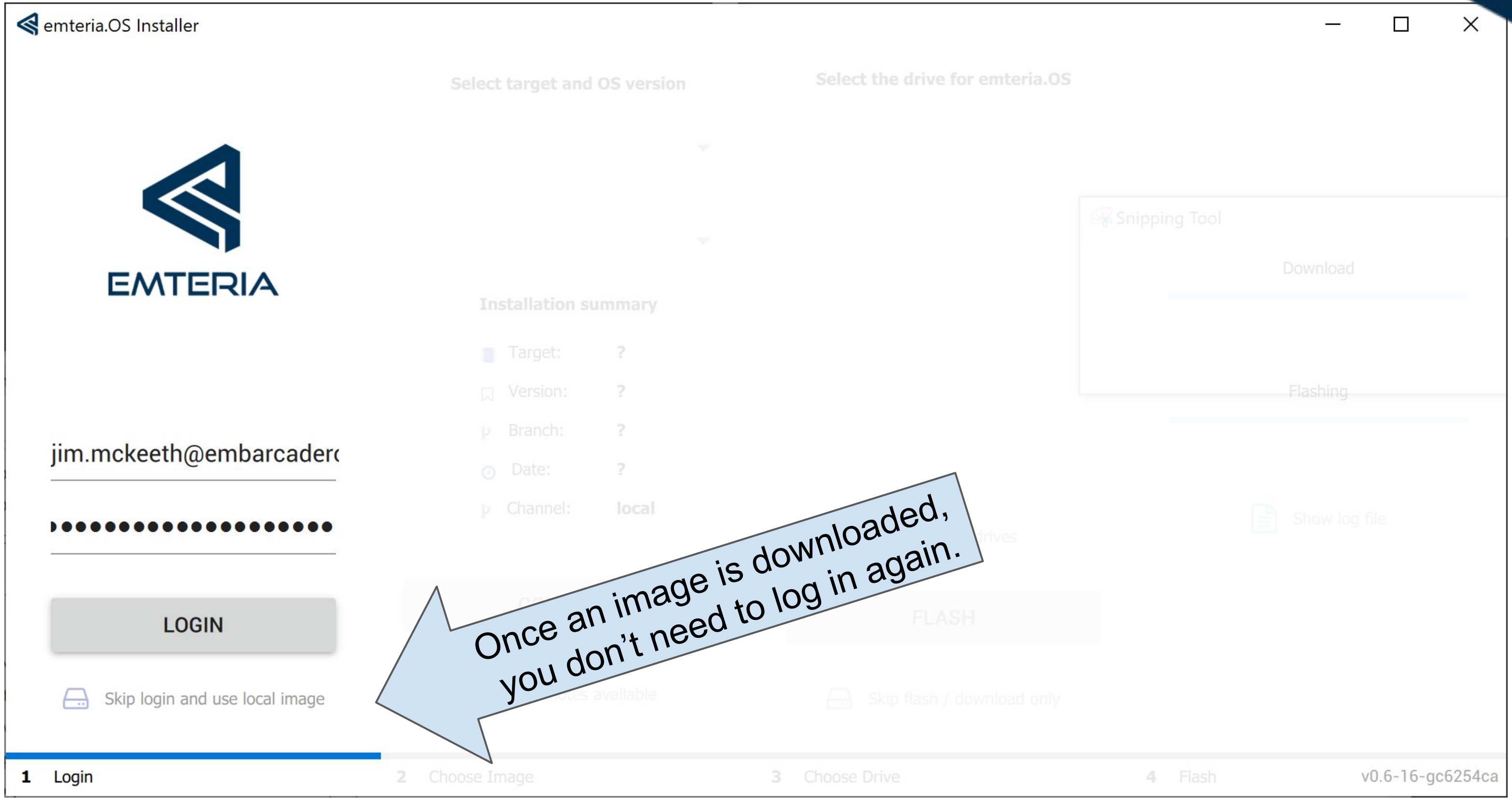
Emteria OS

- Formerly known as Real Time Android (RTAndroid)
- Three license options
  - Free trial (nag screen, corner watermark, and 8 hour reboot)
  - € 19 per device personal license
  - € 99 per device commercial license
- Based on 7.1.2 Nougat AOSP (Android Open Source Project)
- There is a known WiFi disconnect issue if running Bluetooth
- RPi3 Emteria documentation help.emteria.com/kb/716026
- Doesn't seem to support the latest OpenGL-ES, so no GUI support for now.

Installing Emteria OS

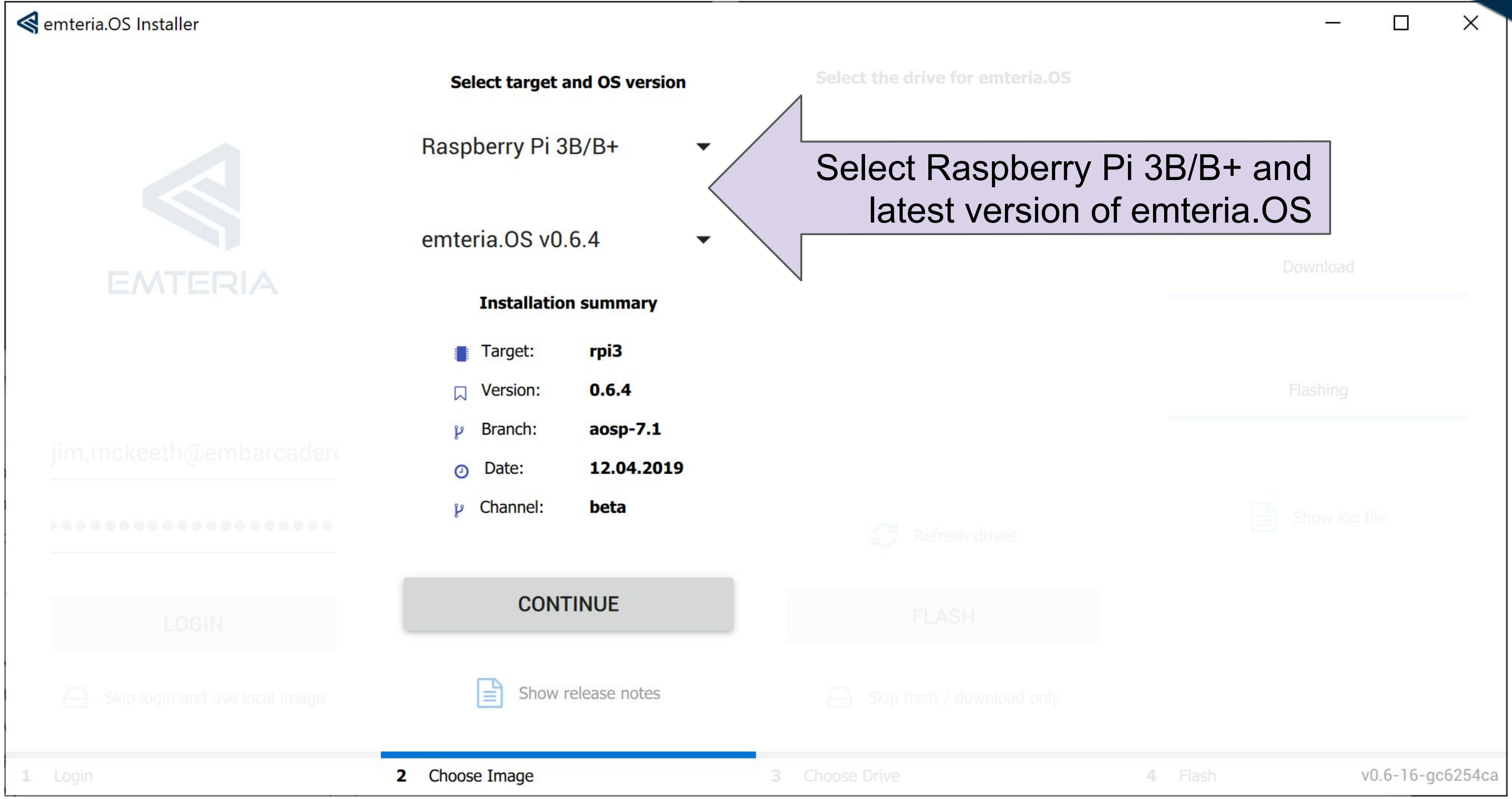
1. Create Emteria.OS account
   - emteria.com/
2. Download and install Emteria installer
   - emteria.com/CustomerArea/Downloads
3. Log into Emteria installer which will download the image and write to SD Card
4. Put SD Card in Raspberry Pi 3
5. Attach monitor, keyboard, & mouse to Raspberry Pi 3 to set it up
6. Boot Raspberry Pi 3, run through the setup & configuration
7. Connect to Raspberry Pi 3 with ADB via it's IP address
8. Deploy Delphi or C++Builder Android apps to Raspberry Pi 3
9. Set Raspberry Pi 3 to automatically launch your app
10. Optionally remove keyboard, mouse, & monitor

Emteria Installation….



Once an image is downloaded, you don't need to log in again.

Emteria Installation….


emteria.OS Installer

Select target and OS version

Raspberry Pi 3B/B+

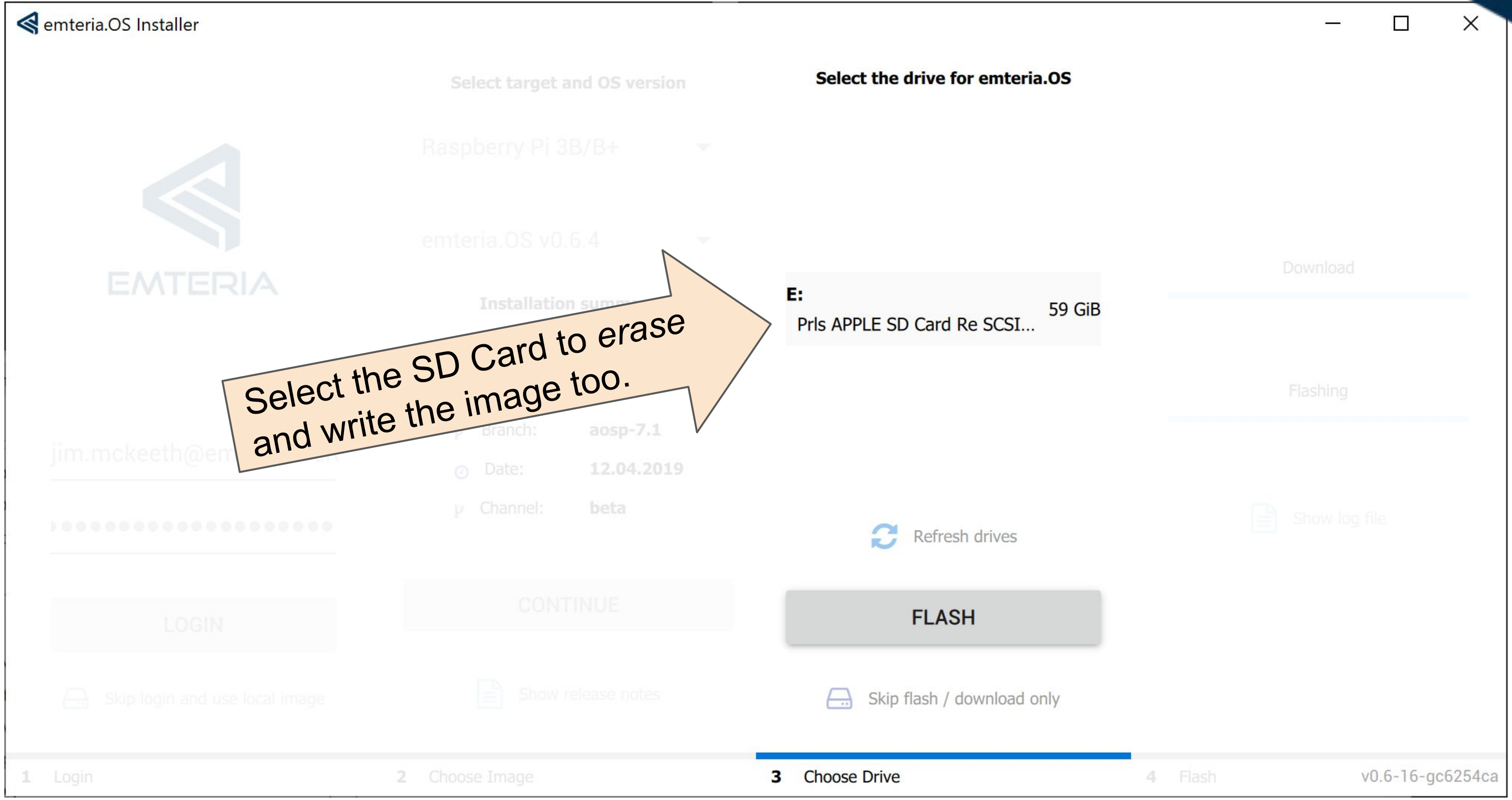
emteria.OS v0.6.4

Installation summary

Target:	**rpi3**
Version:	**0.6.4**
Branch:	**aosp-7.1**
Date:	**12.04.2019**
Channel:	**beta**

Select Raspberry Pi 3B/B+ and latest version of emteria.OS

Emteria Installation….



emteria.OS Installer

Select target and OS version

Raspberry Pi 3B/B+

emteria.OS v0.6.4

Installation summary

Branch: aosp-7.1

Date: 12.04.2019

Channel: beta

CONTINUE

Show release notes

jim.mckeeth@em

LOGIN

Skip login and use local image

Select the drive for emteria.OS

E:
Prls APPLE SD Card Re SCSI... 59 GiB

Refresh drives

FLASH

Skip flash / download only

Download

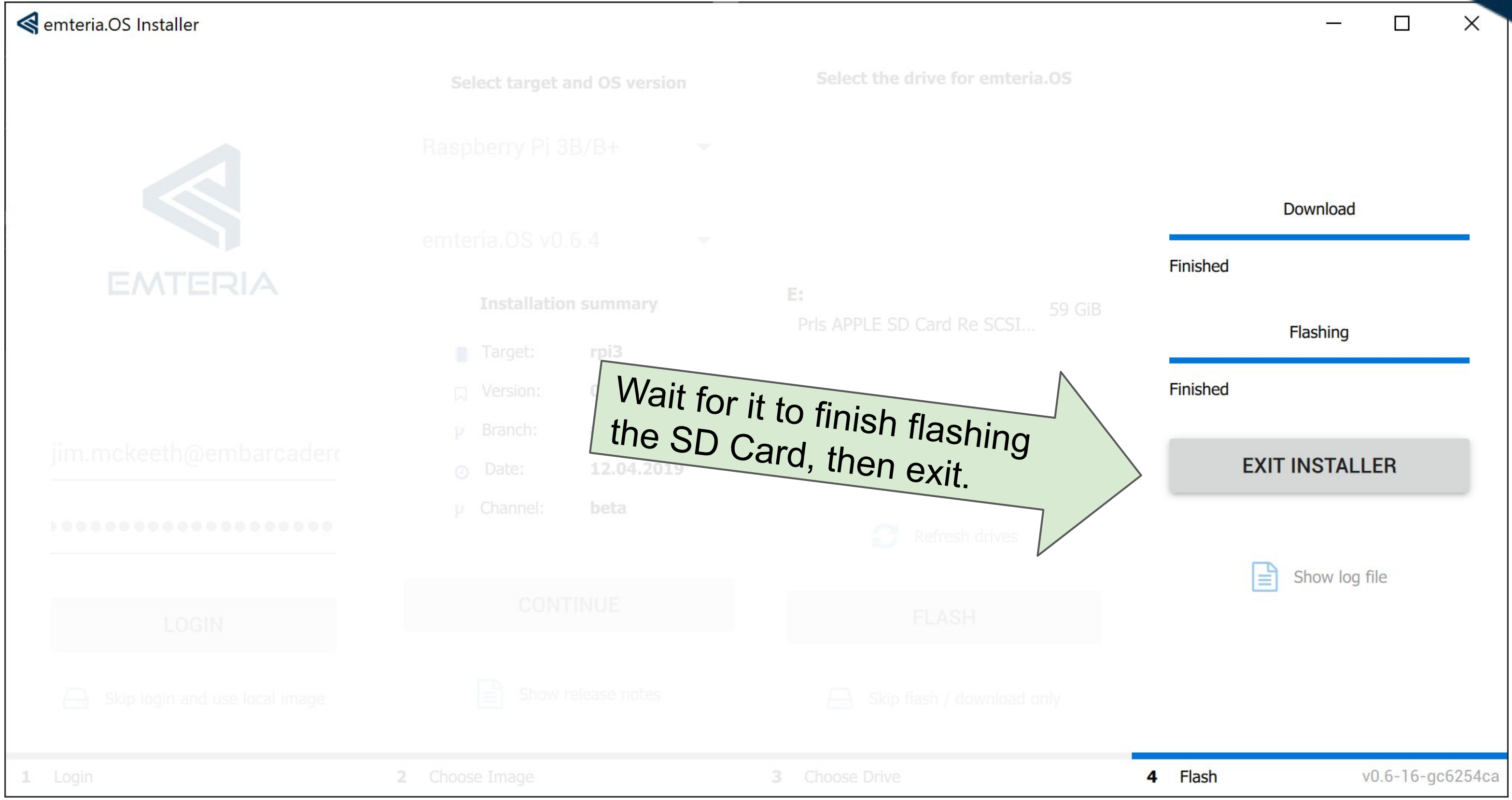
Flashing

Show log file

1 Login 2 Choose Image 3 Choose Drive 4 Flash v0.6-16-gc6254ca

Select the SD Card to erase and write the image too.

Emteria Installation….



emteria.OS Installer

Select target and OS version

Select the drive for emteria.OS

Raspberry Pi 3B/B+

emteria.OS v0.6.4

Installation summary

Target: rpi3

Version:

Branch:

Date: 12.04.2019

Channel: beta

E:

Prls APPLE SD Card Re SCSI… 59 GiB

jim.mckeeth@embarcader

Download

Finished

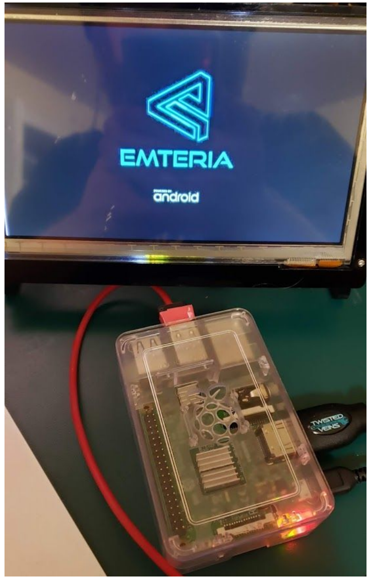
Flashing

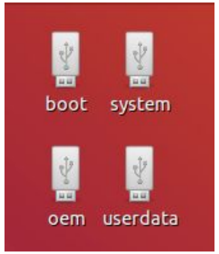
Finished

Wait for it to finish flashing the SD Card, then exit.

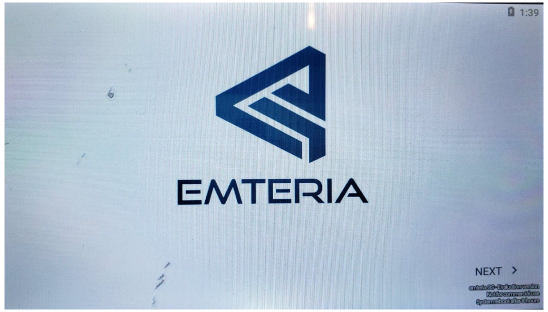
Emteria Configuration....

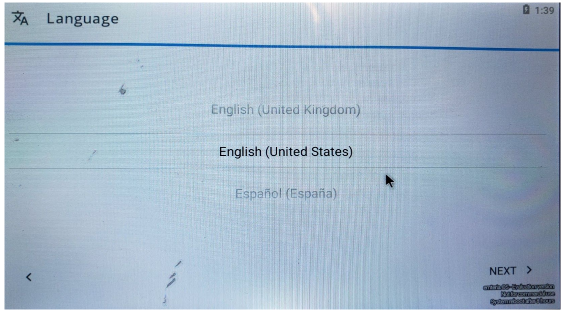
- Put SD Card in Raspberry Pi 3
- Attach monitor, keyboard, & mouse to Raspberry Pi 3
- Boot Raspberry Pi 3, run through the setup & configuration
- It supports a few monitors or you can configure it for different monitors
- *I found the mouse unreliable during the setup process. Had to click both left and right buttons to continue*
- In *theory* you could access the SD Card from Linux and pre-configure it. Then you don't need keyboard, mouse, and monitor.



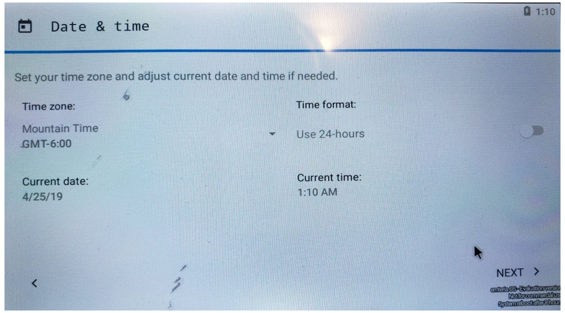


Emteria Setup….




Language

English (United Kingdom)
English (United States)
Español (España)


Date & time

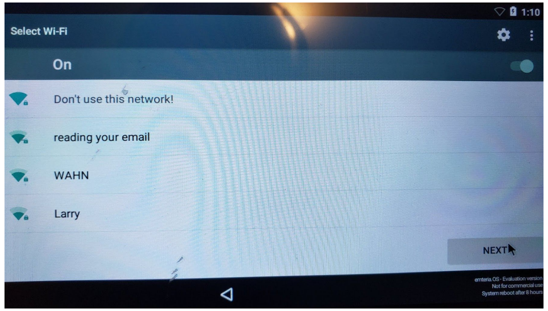
Set your time zone and adjust current date and time if needed.

Time zone: Mountain Time GMT-6:00

Time format: Use 24-hours

Current date: 4/25/19

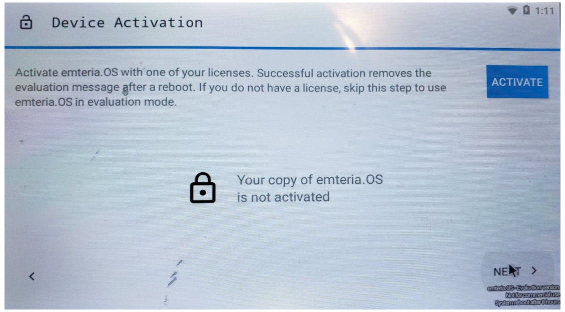
Current time: 1:10 AM


Select Wi-Fi

On

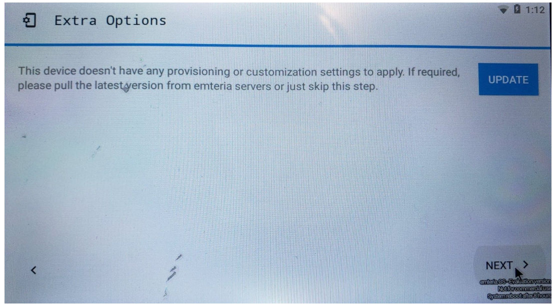
Don't use this network!
reading your email
WAHN
Larry

emteria.OS - Evaluation version
Not for commercial use
System reboot after 8 hours

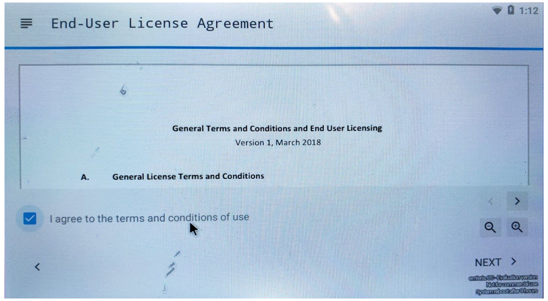

Device Activation

Activate emteria.OS with one of your licenses. Successful activation removes the evaluation message after a reboot. If you do not have a license, skip this step to use emteria.OS in evaluation mode.

Your copy of emteria.OS is not activated


Extra Options

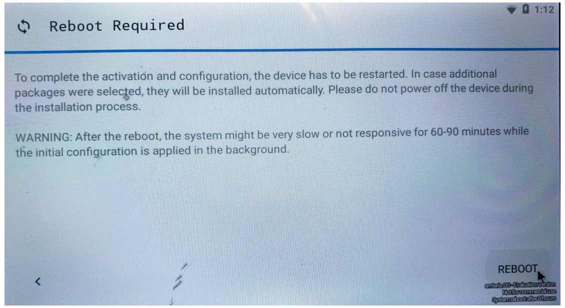
This device doesn't have any provisioning or customization settings to apply. If required, please pull the latest version from emteria servers or just skip this step.


End-User License Agreement

General Terms and Conditions and End User Licensing
Version 1, March 2018

A. General License Terms and Conditions

☑ I agree to the terms and conditions of use

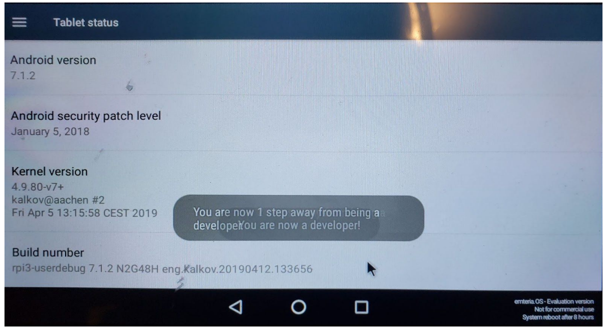

Reboot Required

To complete the activation and configuration, the device has to be restarted. In case additional packages were selected, they will be installed automatically. Please do not power off the device during the installation process.

WARNING: After the reboot, the system might be very slow or not responsive for 60-90 minutes while the initial configuration is applied in the background.

Emteria Configuration….

- Standard Android developer setup
- Turn on USB debug
- Go into Emteria platform settings
- Enable ADB over Ethernet
  (Also displays IP)


Tablet status

Android version
7.1.2

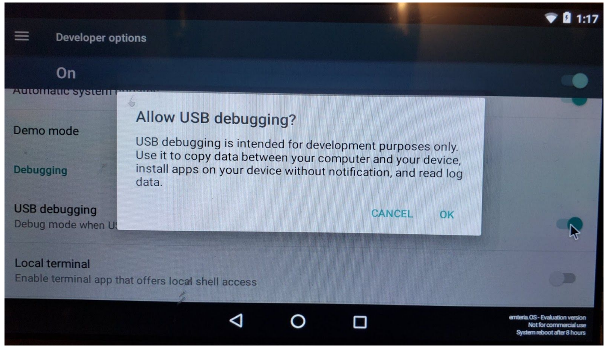
Android security patch level
January 5, 2018

Kernel version
4.9.80-v7+
kalkov@aachen #2
Fri Apr 5 13:15:58 CEST 2019

You are now 1 step away from being a developer!You are now a developer!

Build number
rpi3-userdebug 7.1.2 N2G48H eng.kalkov.20190412.133656

emteria.OS - Evaluation version
Not for commercial use
System reboot after 8 hours


1:17

Developer options

On

Automatic system

Demo mode

Debugging

USB debugging
Debug mode when U

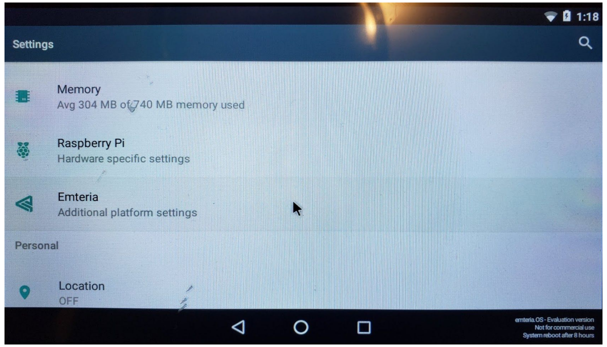
Allow USB debugging?

USB debugging is intended for development purposes only. Use it to copy data between your computer and your device, install apps on your device without notification, and read log data.

CANCEL OK

Local terminal
Enable terminal app that offers local shell access

emteria.OS - Evaluation version
Not for commercial use
System reboot after 8 hours


1:18

Settings

Memory
Avg 304 MB of 740 MB memory used

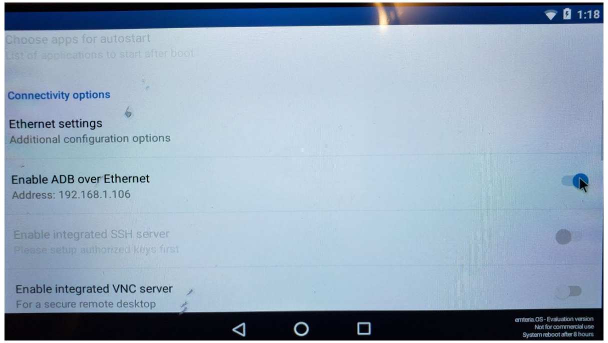
Raspberry Pi
Hardware specific settings

Emteria
Additional platform settings

Personal

Location
OFF

emteria.OS - Evaluation version
Not for commercial use
System reboot after 8 hours


1:18

Choose apps for autostart
List of applications to start after boot

Connectivity options

Ethernet settings
Additional configuration options

Enable ADB over Ethernet
Address: 192.168.1.106

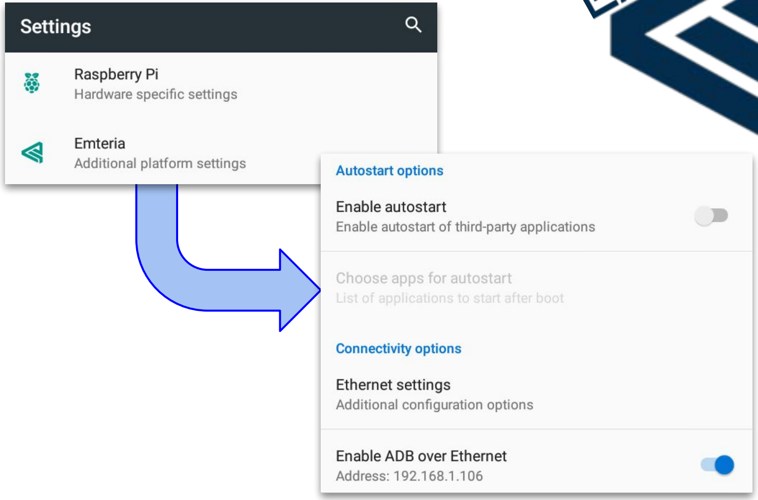
Enable integrated SSH server
Please setup authorized keys first

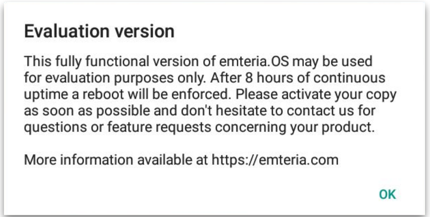
Enable integrated VNC server
For a secure remote desktop

emteria.OS - Evaluation version
Not for commercial use
System reboot after 8 hours

Emteria OS Notes

- Includes a few stock apps, including F-Droid, an alternative App Store
- The trial version shows a watermark, reboots after 8 hours, and occasionally displays “Evaluation version” message
- Includes additional Raspberry Pi and Emteria platform settings
- Emteria settings includes options to autostart an app of your choice
- RPi settings supports screen rotation

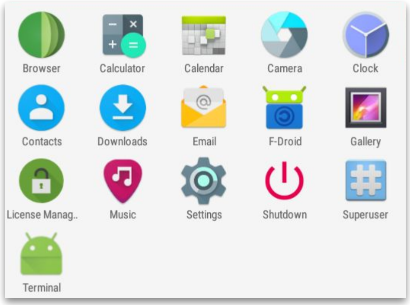

Settings
Raspberry Pi
Hardware specific settings
Emteria
Additional platform settings
Autostart options
Enable autostart
Enable autostart of third-party applications
Choose apps for autostart
List of applications to start after boot
Connectivity options
Ethernet settings
Additional configuration options
Enable ADB over Ethernet
Address: 192.168.1.106


Evaluation version

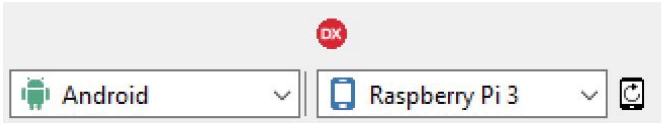
This fully functional version of emteria.OS may be used for evaluation purposes only. After 8 hours of continuous uptime a reboot will be enforced. Please activate your copy as soon as possible and don't hesitate to contact us for questions or feature requests concerning your product.

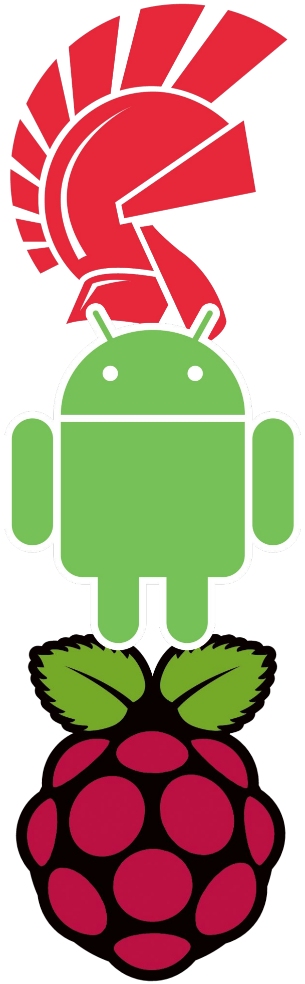
More information available at https://emteria.com

OK



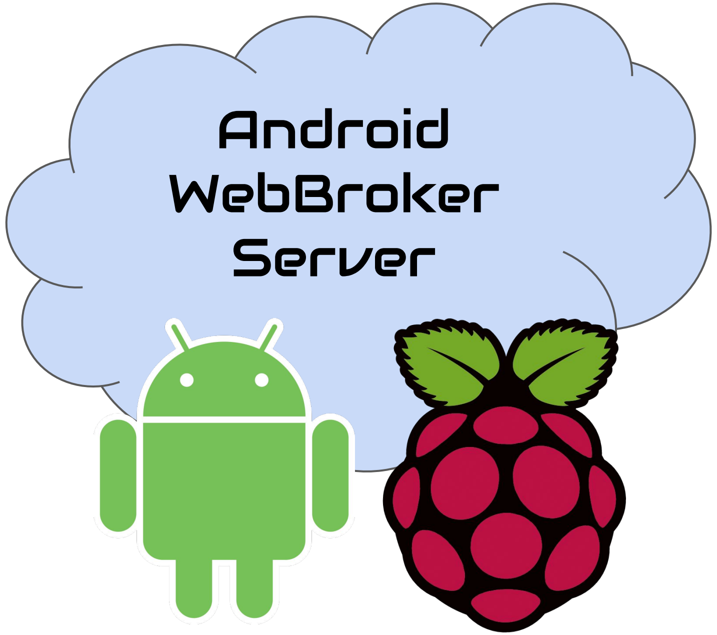
Connect to RPi Android via ADB

- On Raspberry Pi running Android
  - Enable developer mode
  - Settings → About → Status → *Get the IP Address*
- On development machine
  - Connect: `adb connect 192.168.1.106`
  - List: `adb devices`
- In your IDE Switch to Android and Refresh
  - 



WebBroker Server on Raspberry Pi Android

- There is no WebBroker project for Android
- But you can take a WebBroker project and *move* it to Android
- Requires a little manipulation, but it works
- Could also be an Android Service
- Works because WebBroker uses Indy which supports Linux and technically Android
- Could build other internet server projects with Indy too



Notice:
Raspberry Pi, Emteria OS, & Android Things are not "officially" tested or supported platforms.

“*Unofficial*” Android WebBroker Step-by-Step

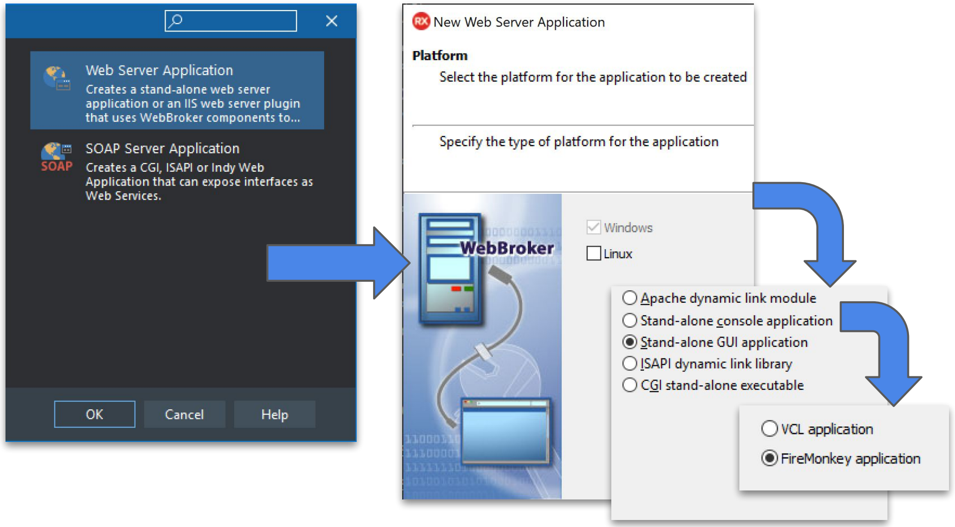
- In the same project group
    - New WebBroker ➜ Windows ➜ Stand Alone GUI ➜ FireMonkey
    - New Multi-Device Application ➜ Blank Application (remove mainform)
- Save them all in the same folder.
- Add the files from the WebBroker project to the Multi-Device project
- Copy the DPR code from the WebBroker DRP to the Multi-Device DPR
- You can delete the original WebBroker project (but not it’s files)
- Copy the following units from RTL to your project folder (others as needed)
    - `IdHTTPWebBrokerBridge.pas`
    - `Web.WebBroker.pas`
    - `Web.WebReq.pas`
    - `Web.WebConst.pas`
- Change target to Android
- Remove “`Uses WinApi.Windows, Winapi.ShellApi;`” and “`ShellExecute`”
- *This isn’t “officially” supported, & there is probably a better way!*

Create FMX GUI WebBroker Project

Works with C++Builder or Delphi

1. Web Server Application
2. Windows
3. Stand-alone GUI application
4. FireMonkey application

Android isn't an available target, but we will port the project across.



Web Server Application
Creates a stand-alone web server application or an IIS web server plugin that uses WebBroker components to...

SOAP Server Application
Creates a CGI, ISAPI or Indy Web Application that can expose interfaces as Web Services.

OK | Cancel | Help

New Web Server Application

Platform
Select the platform for the application to be created

Specify the type of platform for the application

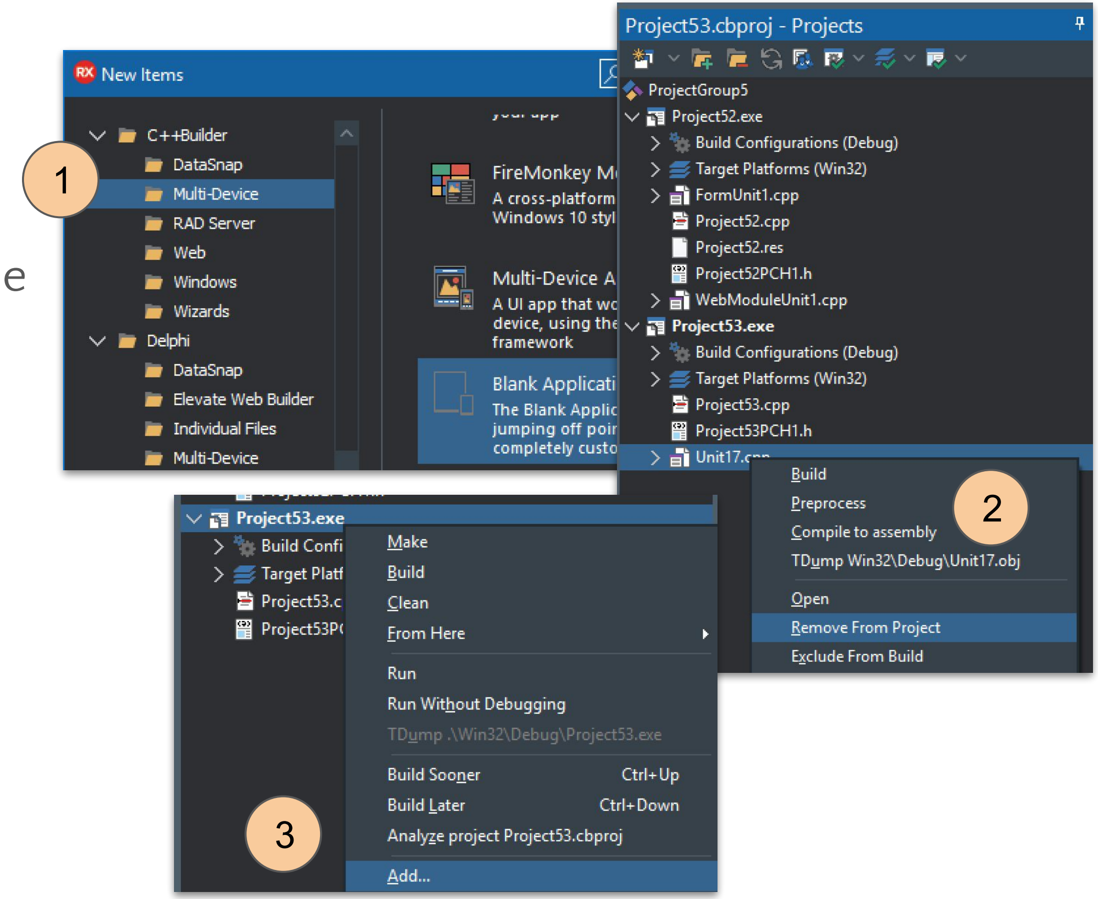
[✓ Windows]
[ ] Linux

( ) Apache dynamic link module
( ) Stand-alone console application
[✓ Stand-alone GUI application]
( ) ISAPI dynamic link library
( ) CGI stand-alone executable

( ) VCL application
[✓ FireMonkey application]

Create Blank Multi-Device Project

1. Use the same language
2. Remove the mainform from the new project
3. Save and add the WebModule to the new blank project
4. Move over the project code
5. Add a new TDataModule
6. Use the DataModule Constructor to start and initialize the server web module



Targeting Android

1. The WebBroker units are not in the Android packages, but they work there
2. Copy the necessary WebBroker files into your project folder (or a subfolder)
3. For C++ projects you also need the associated .hpp files (or regenerate them)
4. You may need to copy other WebBroker units later, as you add functionality
5. Point your project to the copied units
6. Change target to Android
7. *Reminder:* Don't redistribute the RTL source code!

Required units / files

- `Web.WebReq.pas`
- `Web.WebBroker.pas`
- `Web.WebConst.pas`
- `IdHTTPWebBrokerBridge.pas`
- `IdCompilerDefines.inc`

HPP files for C++

- `Web.WebReq.hpp`
- `Web.WebBroker.hpp`
- `Web.WebConst.hpp`
- `IdHTTPWebBrokerBridge.hpp`

C++ Project Source

```
//-----------------------------------------------
#include <fmx.h>
#pragma hdrstop
#include <Web.WebReq.hpp>
#ifdef USEPACKAGES
#pragma link "IndySystem.bpi"
#pragma link "IndyCore.bpi"
#pragma link "IndyProtocols.bpi"
#else
#pragma comment(lib, "IndySystem")
#pragma comment(lib, "IndyCore")
#pragma comment(lib, "IndyProtocols")
#endif
#include <IdException.hpp>
#pragma link "IdHTTPWebBrokerBridge"
#include <System.StartUpCopy.hpp>
//-----------------------------------------------
/* TWebModule: File Type */
USEFORM("WebModuleUnit1.cpp", WebModule1);
/* TDataModule: File Type */
USEFORM("MainDM.cpp", dmMain);
//-----------------------------------------------
```

```
//-----------------------------------------------------------
extern PACKAGE TComponentClass WebModuleClass;
extern "C" int FMXmain()
{
  try
  {
      if (WebRequestHandler() != NULL)
      {
        WebRequestHandler()->WebModuleClass = WebModuleClass;
      }
    Application->Initialize();
      Application->CreateForm(__classid(TdmMain), &dmMain);
            Application->Run();
  }
  catch (Exception &exception)
  {
      Sysutils::ShowException(&exception, System::ExceptAddr());
  }
  return 0;
}
//-----------------------------------------------------------
```

Full source: github.com/jimmckeeth/CppBuilderAndroidWebBroker

C++ DataModule “mainDM” Source

```
// Header
//---------------------------------------------------------------
#ifndef MainDMH
#define MainDMH
//---------------------------------------------------------------
#include <System.Classes.hpp>
#include <IdException.hpp>
#include <IdHTTPWebBrokerBridge.hpp>
#include <FMX.Types.hpp>
//---------------------------------------------------------------
class TdmMain : public TDataModule
{
__published:    // IDE-managed Components
        void __fastcall DataModuleCreate(TObject *Sender);
private:        // User declarations
        TIdHTTPWebBrokerBridge *FServer;
public:         // User declarations
        __fastcall TdmMain(TComponent* Owner);
};
//---------------------------------------------------------------
extern PACKAGE TdmMain *dmMain;
//---------------------------------------------------------------
#endif
```

```
// Code
//---------------------------------------------------------------
#pragma hdrstop
#include "MainDM.h"
//---------------------------------------------------------------
#pragma package(smart_init)
#pragma classgroup "FMX.Controls.TControl"
#pragma resource "*.dfm"
TdmMain *dmMain;
//---------------------------------------------------------------
__fastcall TdmMain::TdmMain(TComponent* Owner)
        : TDataModule(Owner)
{
}
//---------------------------------------------------------------
void __fastcall TdmMain::DataModuleCreate(TObject *Sender)
{
  FServer = new TIdHTTPWebBrokerBridge(this);
  if (!FServer->Active)
  {
        FServer->Bindings->Clear();
        FServer->DefaultPort = StrToInt(8000);
        FServer->Active = true;
  }
}
//---------------------------------------------------------------
```

Full source: github.com/jimmckeeth/CppBuilderAndroidWebBroker

Delphi DPR Project Source

```
program AndroidWebBroker;

uses
  System.StartUpCopy,
  Web.WebReq,
  IdHTTPWebBrokerBridge,
  FMX.Forms,
  WebModuleUnit1 in 'WebModuleUnit1.pas' {WebModule1: TWebModule},
  mainDM in 'mainDM.pas' {dmMain: TDataModule};

{$R *.res}

begin
  if WebRequestHandler <> nil then
    WebRequestHandler.WebModuleClass := WebModuleClass;
  Application.Initialize;
  Application.CreateForm(TdmMain, dmMain);
  Application.Run;
end.
```

Full source: github.com/jimmckeeth/DelphiAndroidWebBroker

Delphi DataModule “mainDM” Source

```
unit mainDM;

interface

uses
  System.SysUtils, System.Classes,
  IdHTTPWebBrokerBridge, Web.HTTPApp;

type
  TdmMain = class(TDataModule)
    procedure DataModuleCreate(Sender: TObject);
  private
    { Private declarations }
    FServer: TidHTTPWebBrokerBridge;
  public
    { Public declarations }
  end;

var
  dmMain: TdmMain;

{ continued ... }
```

```
implementation

{%CLASSGROUP 'FMX.Controls.TControl'}

{$R *.dfm}

procedure TdmMain.DataModuleCreate(Sender: TObject);
begin
  FServer := TIdHTTPWebBrokerBridge.Create(Self);
  if not FServer.Active then
  begin
    FServer.Bindings.Clear;
    FServer.DefaultPort := 8888;
    FServer.Active := True;
  end;
end;

end.
```

Full source: github.com/jimmckeeth/DelphiAndroidWebBroker

Final Notes

- Emteria OS doesn’t have the latest OpenGL-ES, so you can’t have a form
    - (There may be a work around for this, but we really don’t need a GUI)
- This means your app runs without a GUI, so it is just a black screen
- If you run it on Windows there is no UI, so you need to end task (or find a different way to tell it to terminate)
- On Emteria OS you can set it to autostart via the Emteria platform settings
- Android Things supports GUI, but it isn’t necessary so wastes processing

More Information

- Blog post: delphi.org/?p=3330 ← This will have updated slides & links too
- Previous coverage: embt.co/DelphiSBC
- Related Chrome OS / Chromebook: embt.co/DelphiChromeOS
- Delphi Code: github.com/jimmckeeth/DelphiAndroidWebBroker
- C++ Code: github.com/jimmckeeth/CppBuilderAndroidWebBroker
- Android Things: developer.android.com/things
- Emteria.OS: emteria.com
- DocWiki:
    - docwiki.embarcadero.com/RADStudio/en/Creating_WebBroker_Applications
    - docwiki.embarcadero.com/RADStudio/en/Using_Web_Broker_Index