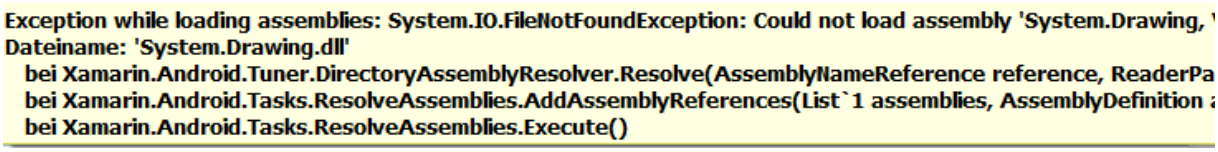
Fast-Report product page:
“It can be used with Windows, Linux, Mac OS X and any operating system that supports Xamarin Mono”

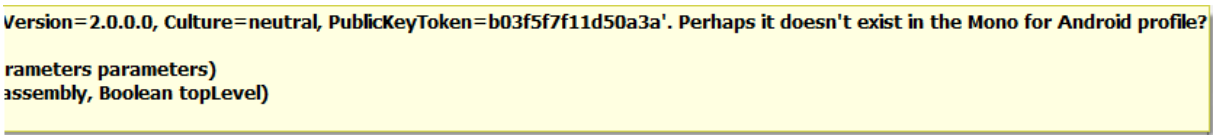
Step 1:

Create a sample Solution, add FastReport.Mono.dll and call the following Code:

```
Report test = new Report();
```

- ➔ MAC: Compiles, runs and works (show as dialog) on MAC (Our PoC platform)
- ➔ Android: Does not compile:





Conclusion:
System.Drawing.-Namespace is available, but Xamarin for Android cannot resolve the DLL, while it can for MAC.

This problem seems to be known, as there is a fast-report thread for it, but no answer.
http://www.fast-report.com/en/forum/index.php?showtopic=10048

Step 2:

As for some awkward reason Xamarin Android couldn’t resolve the required dll for Fastreport, and we could not find any options that could change this behavior we tried to **download and add the mono System.Drawing.dll manually.** (We tried every workaround step with assemblies from mono version 2.0, as well as 4.0)

Result:
The project compiles now, as the compiler could resolve System.Drawing.dll.
However the following call will still result in an exception.

```
Report test = new Report();
```

The exception said, that libgdiplus.dll could not be found.

Conclusion:
System.Drawing.dll from Mono is a wrapper for libgdiplus.dll?

Step 3:

We still need to get our X-Platform-Reports that we've spent months on to design run on Android aswell, so we investigated further on how to resolve the missing libgdiplus.dll error.

So we tried to get a libgdiplus.so implementation from a linux-system, embed it as AndroidNativeLibrary in the application in those folders :

/lib/armeabi/libgdiplus.so
/lib/armeabi-v71/libgdiplus.so
/lib/x86/libgdiplus.so

After configuring the Application to support all of these three ABI under Options -> Android Build -> Advanced.

(Source for this procedure: http://takeshich.hatenablog.com/entry/2013/12/04/000000)

Now we had to tell the System to use this .so for the libgdiplus implementation, by adding a System.Drawing.dll.config next to the dll itself, specifying:

```
<configuration>
  <dllmap dll="gdiplus.dll" target="libgdiplus.so"/>
</configuration>
```

We tried everything imaginable to locate the "target.so" and googled for hours with no success.

The only change that had an impact was putting __Internal as target

```
<configuration>
  <dllmap dll="gdiplus.dll" target="__Internal"/>
</configuration>
```

This seemed wrong in this case, but now at least we receive a new error:

System.TypeInitializationException: An exception was thrown by the type initializer for System.Drawing.GDIPlus → System.EntryPointNotFoundException: GdiplusStartup

Conclusion

The fact that we had a new errormessage, that was looking for something GDISpecific ("GdiplusStartup") instead of just "dll / file not found" felt like we were on the right track and maybe just used an incompatible gdiplus-file… but at this point after a long way we ran out of further ideas.