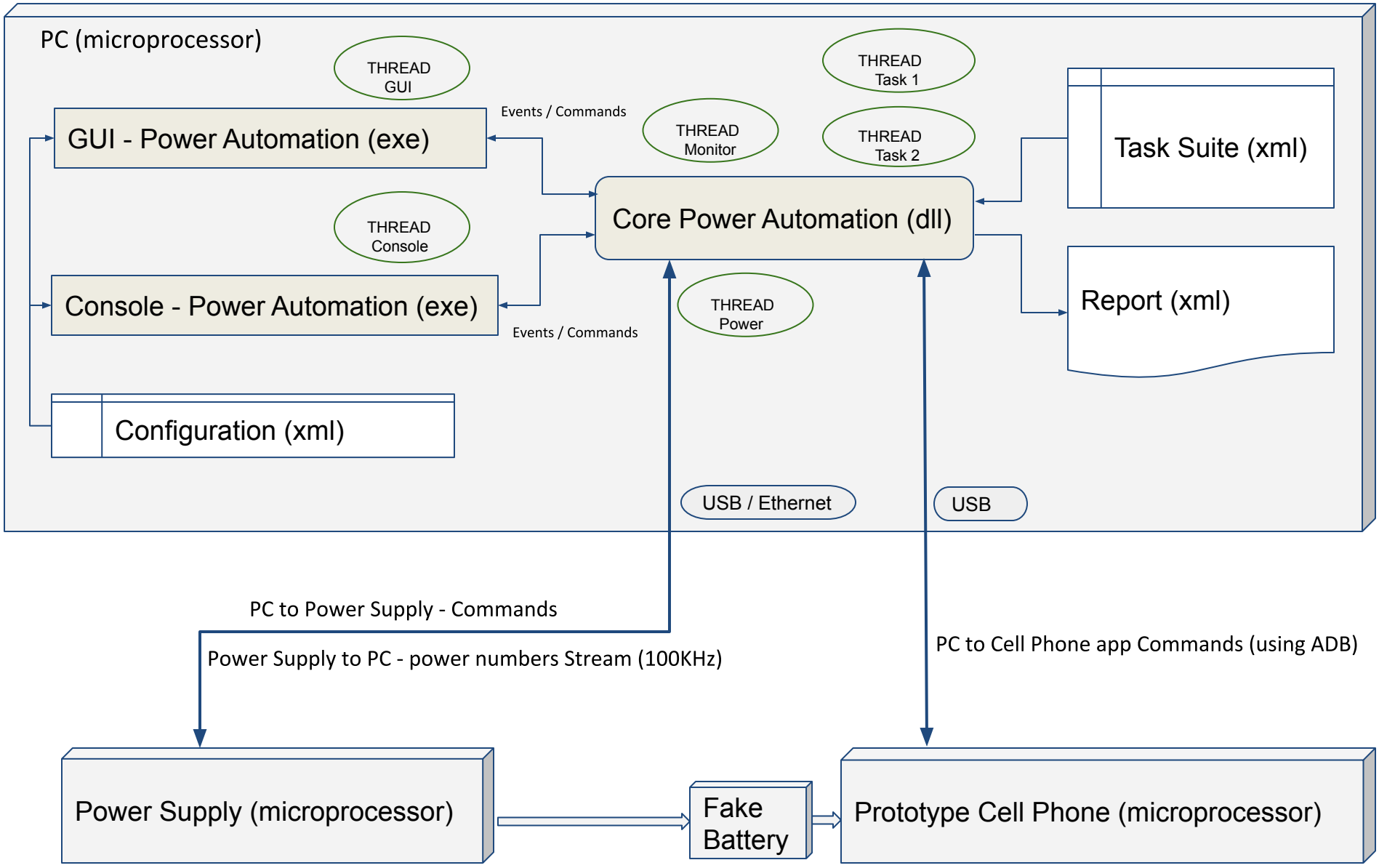
Qualcomm Power Automation Toolkit

Documentation Design

Objective

- Create a simple and efficient software platform to measure power consumption for mobile devices
- Integrate and control power measurement devices
- Possibility to extend the software platform with custom modules for new test cases
- The execution tasks have to be highly automated and provide expert analyses based on configuration files and generated reports

BLOCK DIAGRAM



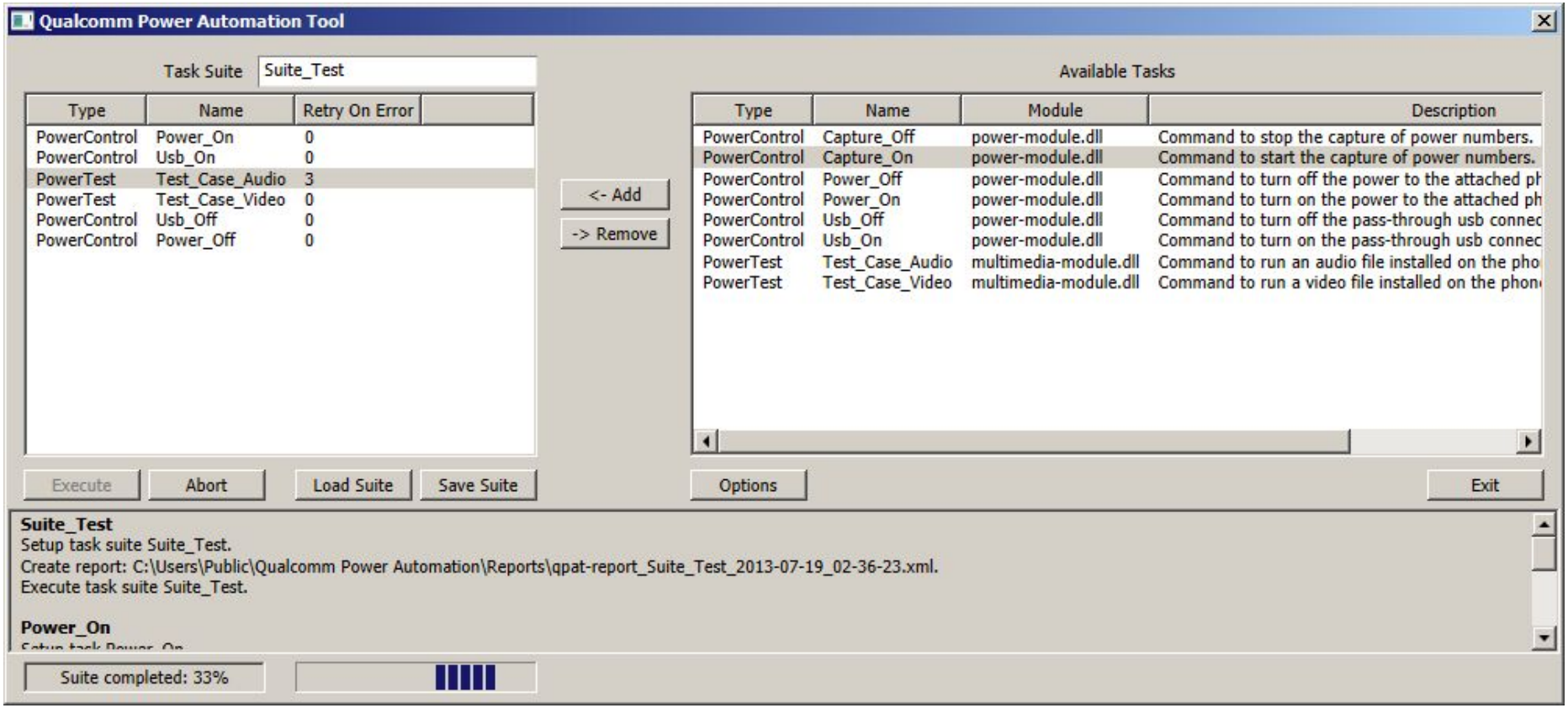
FUNCTIONAL LOGIC

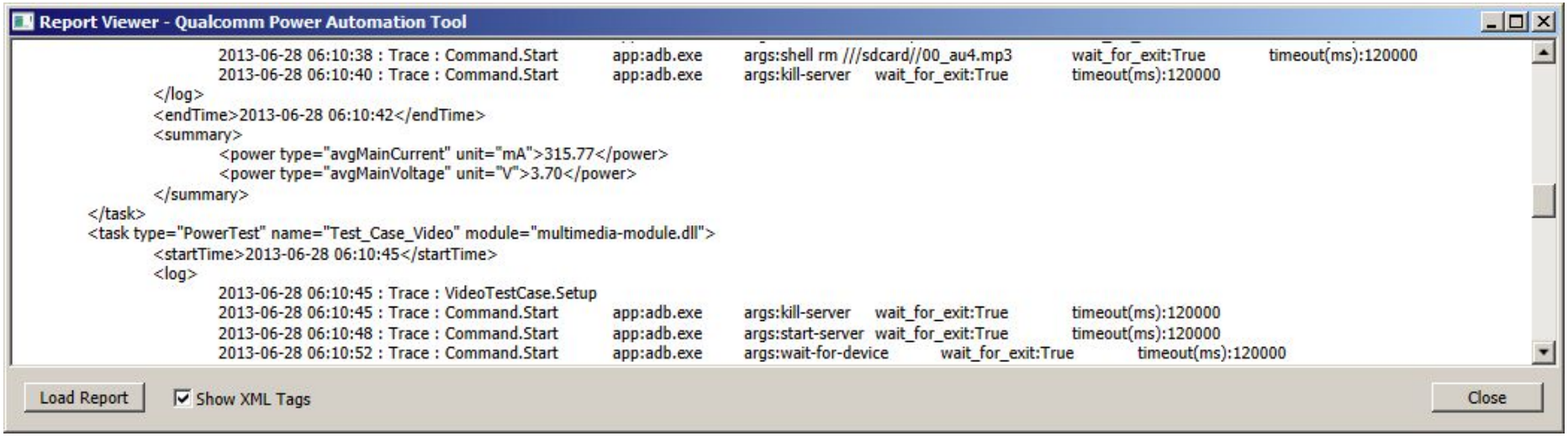
- Core Automation Tool runs together with its GUI app and console app on a PC (notebook). Input configuration, task suites (test units) and output reports are in XML file format residing on the PC.
- GUI app is WPF based and it is event driven (implements a thread message loop). Console app has its own main thread for the user interaction at the terminal prompt, used to be integrated in scripts.
- Core Automation dynamic link library has a Monitor thread used to asynchronously launch phone tasks, send the power supply commands and interface with the GUI app and console app through events.
- Task units for on the cell phone run on their own threads on the PC and interface with the cell phones using ADB (Android Debug Bridge). The task thread captures the standard input and output of ADB external executable and asynchronously sends commands to the phone and receives results from the phone.
- The power supply metrics are captured from the output stream by the Power thread and computed (average current and voltage) and saved in a report file.

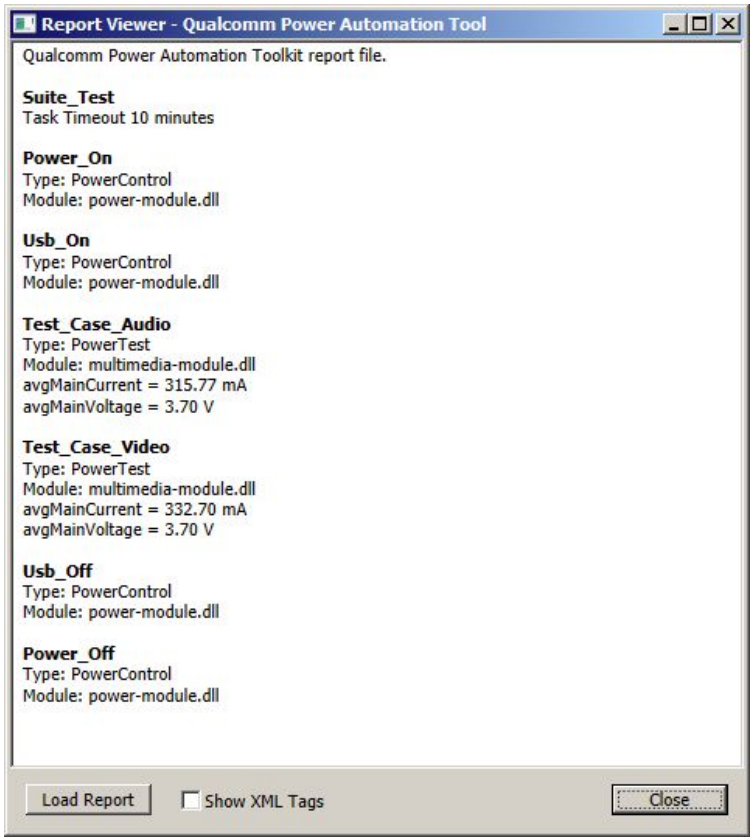
GUI – Qualcomm Power Automation Toolkit

- Provides a simple user interface that loads an XML configuration file for presetting options as paths for file resources, chipset types and OS types. The configuration settings can be modified and saved using UI.
- Loads at run time plug-in modules which embed test cases (in dll format).
- Loads or creates suite test cases which can be saved in XML files and executed.
- Generates an XML report that is automatically shown in a viewer application. The viewer application is simple and can be used without automation tool as an executable independent viewer, closing Automation Tool will not close the current instance of viewer application.

GUI – Qualcomm Power Automation Toolkit




Report Viewer - Qualcomm Power Automation Tool
2013-06-28 06:10:38 : Trace : Command.Start app:adb.exe args:shell rm ///sdcard//00_au4.mp3 wait_for_exit:True timeout(ms):120000
2013-06-28 06:10:40 : Trace : Command.Start app:adb.exe args:kill-server wait_for_exit:True timeout(ms):120000
</log>
<endTime>2013-06-28 06:10:42</endTime>
<summary>
<power type="avgMainCurrent" unit="mA">315.77</power>
<power type="avgMainVoltage" unit="V">3.70</power>
</summary>
</task>
<task type="PowerTest" name="Test_Case_Video" module="multimedia-module.dll">
<startTime>2013-06-28 06:10:45</startTime>
<log>
2013-06-28 06:10:45 : Trace : VideoTestCase.Setup
2013-06-28 06:10:45 : Trace : Command.Start app:adb.exe args:kill-server wait_for_exit:True timeout(ms):120000
2013-06-28 06:10:48 : Trace : Command.Start app:adb.exe args:start-server wait_for_exit:True timeout(ms):120000
2013-06-28 06:10:52 : Trace : Command.Start app:adb.exe args:wait-for-device wait_for_exit:True timeout(ms):120000
Load Report
Show XML Tags
Close

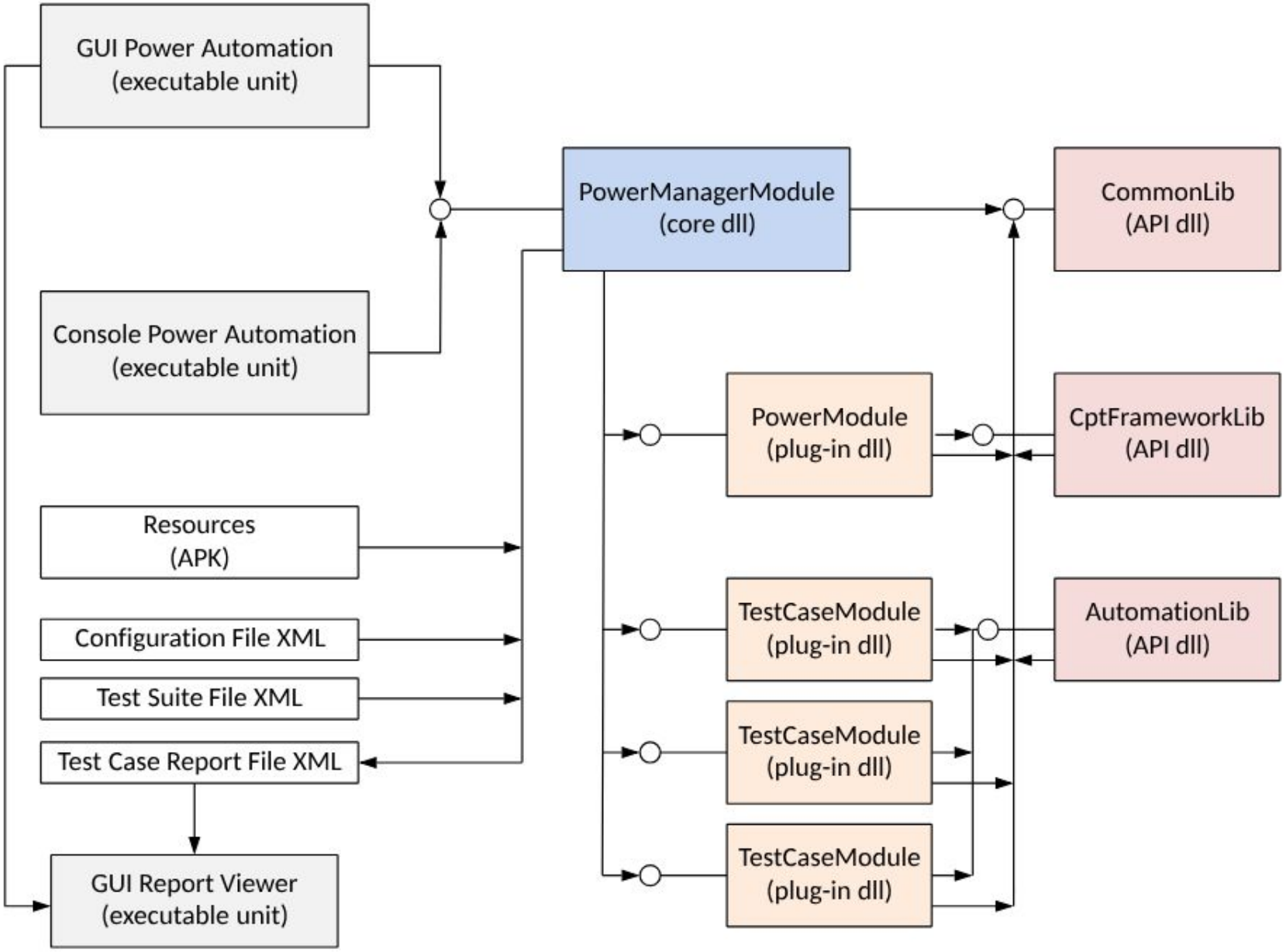

Report Viewer - Qualcomm Power Automation Tool
Qualcomm Power Automation Toolkit report file.
Suite_Test
Task Timeout 10 minutes
Power_On
Type: PowerControl
Module: power-module.dll
Usb_On
Type: PowerControl
Module: power-module.dll
Test_Case_Audio
Type: PowerTest
Module: multimedia-module.dll
avgMainCurrent = 315.77 mA
avgMainVoltage = 3.70 V
Test_Case_Video
Type: PowerTest
Module: multimedia-module.dll
avgMainCurrent = 332.70 mA
avgMainVoltage = 3.70 V
Usb_Off
Type: PowerControl
Module: power-module.dll
Power_Off
Type: PowerControl
Module: power-module.dll
Load Report
Show XML Tags
Close

MODULES – Qualcomm Power Automation Toolkit

The software modules have to keep minimum dependencies and be specialized:

1. Executable units as GUI Power Automation Tool, Console Power Automation Tool and Report View Power Automation Tool
2. Core modules which implement the application logic as Power Manager Module
3. Application programming libraries as Common Lib, Cpt Framework Lib and Automation Lib
4. Plug-in modules as Power Test Case Modules.

MODULES – Qualcomm Power Automation Toolkit

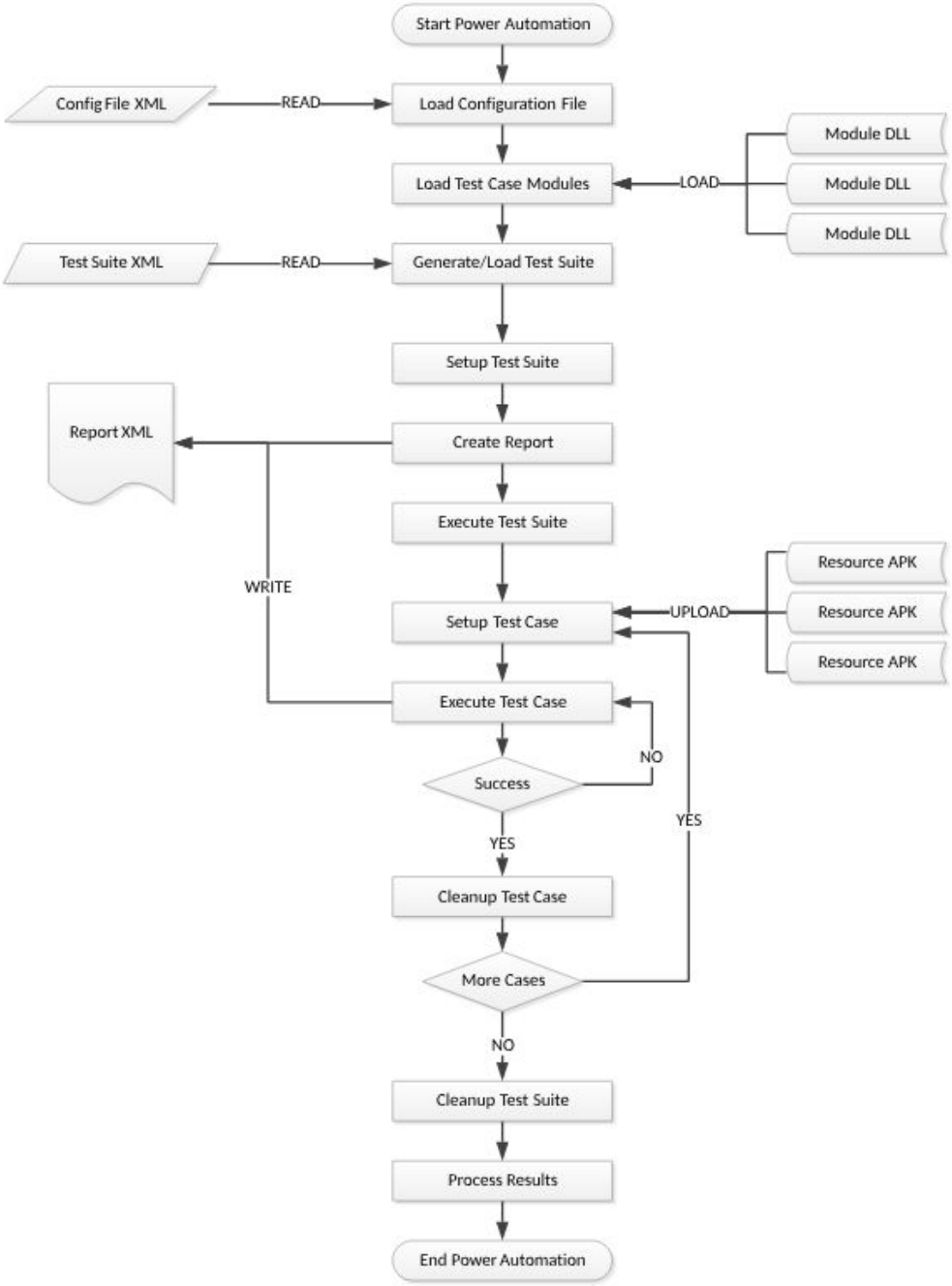


FLOW CHART – Qualcomm Power Automation Toolkit

The Flow Execution Control is implemented by Power Manager Module:

1. Loads the configuration file
2. Loads the plug-in modules
3. Creates (using GUI) or loads the test suite file
4. Executes the test suite that is translated in execution of each test case one or more times (in case of errors), in a chosen sequence.
5. Generates the report
6. Processes the test case results captured in report using a validation and analyses logic.

FLOW CHART – Qualcomm Power Automation Toolkit



CONFIGURATION FILES – Qualcomm Power Automation Toolkit

- The configuration files are kept in XML format.
- The report files are also kept in XML format.

XML format is self explicit and provides a flexibility in manipulation using many available parsers, many available readers and is it an industry standard platform independent.

The configuration files and reports can be extended with new features using new tags and still keeping the compatibility both backward and forward: not all versions of Automation Tools have to understand and use all XML blocks, so a processing tool will only make use of what it understands from XML file.

`qpat-config.xml`

```
<qpatCfg>
	<pathReports>C:\Users\Username\Reports</pathReports>
	<pathSuites>C:\Users\Username\Suites</pathSuites>
	<pathModules>C:\Program Files\Power Measurement\Modules</pathModules>
	<pathResources>C:\Program Files\Power Measurement\Resources</pathResources>
	<typeChipset>8974</typeChipset>
	<typeOS>Android 4.x</typeOS>
	<logLevel>All</logLevel>
</qpatCfg>
```

`qpat-suite.xml`

```
<taskSuite name="Suite_A">
	<task name="Power_On" module="power-module.dll" />
	<task name="Test_Case_X" module="x-y-z-module.dll" retryOnError=3 />
	<task name="Test_Case_Z" module="x-y-z-module.dll" />
	<task name="Test_Case_A" module="a-module.dll" />
	<task name="Power_Off" module="power-module.dll" />
</taskSuite>
```

`qpat-report_suitename_datetime.xml`

```
<taskSuite name="Suite_A">
	<task name="Power_On" module="power-module.dll">
		<startTime>2012-01-01 10:10:00</startTime>
		<log>
			2012-01-01 10:10:05 : Message : Text
		</log>
		<endTime>2012-01-01 10:10:10</endTime>
	</task>
	<task name="Test_Case_X" module="x-y-z-module.dll">
		<startTime>2012-01-01 10:20:00</startTime>
		<log>
			2012-01-01 10:20:40 : Message : Text
			2012-01-01 10:20:45 : Error : Text
			2012-01-01 10:20:50 : Warning : Text
			2012-01-01 10:20:55 : Trace : Text
		</log>
		<endTime>2012-01-01 10:30:00</endTime>
	</task>
	<task name="Power_Off" module="power-module.dll">
		<startTime>2012-01-01 10:40:00</startTime>
		<log>
			2012-01-01 10:40:05 : Message : Text
		</log>
		<endTime>2012-01-01 10:40:10</endTime>
	</task>
</taskSuite>
```

Console Power Automation Tool

It will load the configuration files then execute the automation task.

```
Terminal-Prompt > qpat.exe -c "path\qpat-config.xml" -s "path\qpat-suite.xml"

Terminal-Prompt > qpat.exe {-h|-help|/h|/help}
```

GUI Power Automation Tool

It will load the configuration files and show up the GUI application; the automation task will not execute, the user has to interact with the GUI.

```
Terminal-Prompt > qpatwin.exe [-c "path\qpat-config.xml" [-s
"path\qpat-suite.xml"]]

Terminal-Prompt > qpatwin.exe {-h|-help|/h|/help}
```

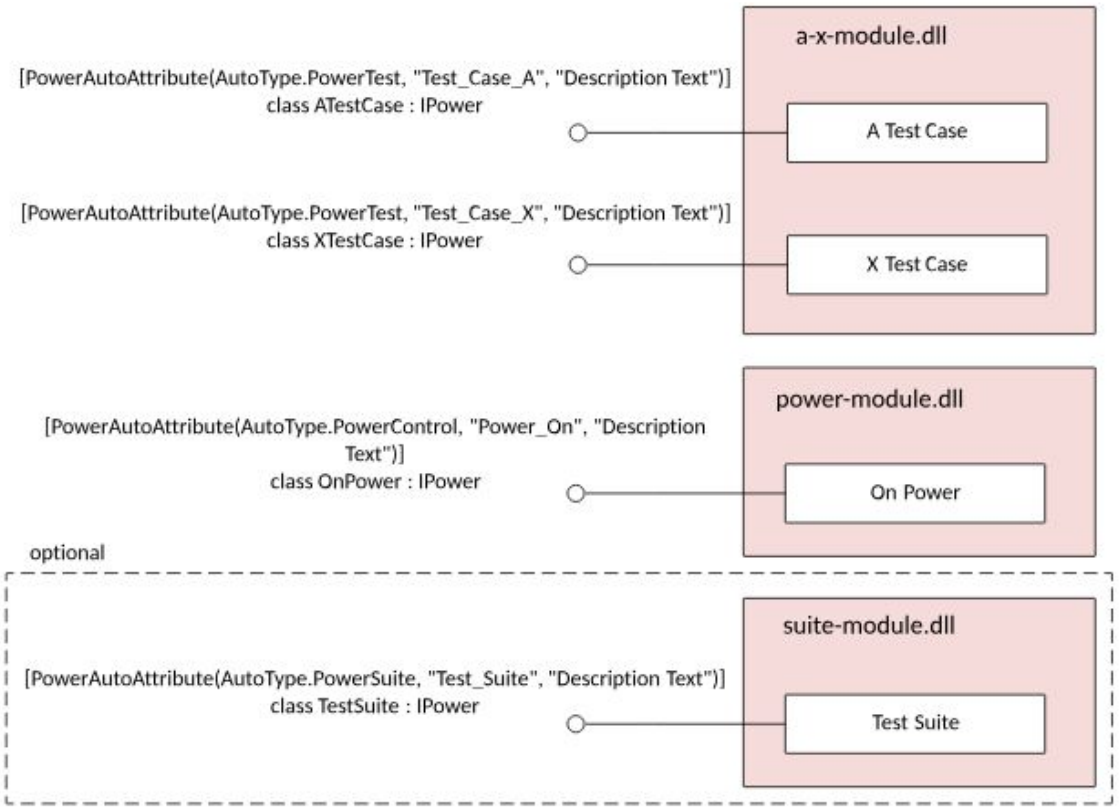
GUI Power Automation Report Viewer

It will load the report file and present it in the format conforming with the flag for XML tags.

```
Terminal-Prompt > qpatview.exe [[-showxmltags {yes|no}] -r
"path\qpat-report.xml"]

Terminal-Prompt > qpatview.exe {-h|-help|/h|/help}
```

```
qpat.exe returns an error level 0 on success and 1 or more on failure.

Such return values can be tested in shell batch files with "if ERRORLEVEL n" statement. It can
also be used %ERRORLEVEL% environment variable when command extensions are enabled and
%ERRORLEVEL% environment variable was not explicitly set.


Dictionary:
[ ] – optional argument
{ | } – one argument from list manadatory
```

Plug-in Module




```
public interface IPower
{
      StatusResult Setup();
      StatusResult Execute();
      StatusResult Abort();
      StatusResult Cleanup();
       event EventHandler<PowerEvetnArgs> PreMeasure;
       event EventHandler<PowerEventArgs> PostMeasure;
}

public class PowerEventArgs
{
      public StatusResult StatusResult { get; set; }
}

public enum AutoType { PowerControl, PowerTest, PowerSuite }

public class PowerAutoAttribute : Attribute
{
      public AutoType type;
      public string name;
      public string description;
}
```