ARM

PAL

PAL Porting Guide

PAL

Last Updated: 26 September 2016

Confidential Proprietary Notice

This document is CONFIDENTIAL and any use by you is subject to the terms of the agreement between you and ARM® or the terms of the agreement between you and the party authorized by ARM® to disclose this document to you.

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of ARM. ***No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.***

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information: (i) for the purposes of determining whether implementations infringe any third party patents; (ii) for developing technology or products which avoid any of ARM's intellectual property; or (iii) as a reference for modifying existing patents or patent applications or creating any continuation, continuation in part, or extension of existing patents or patent applications; or (iv) for generating data for publication or disclosure to third parties, which compares the performance or functionality of the ARM® technology described in this document with any other products created by you or a third party, without obtaining ARM's prior written consent.

THIS DOCUMENT IS PROVIDED "AS IS". ARM® PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, ARM® makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, third party patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM® BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM® HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word "partner" in reference to ARM's customers is not intended to create or refer to any partnership relationship with any other company. ARM® may make changes to this document at any time and without notice.

If any of the provisions contained in these terms conflict with any of the provisions of any click through or signed written agreement covering this document with ARM, then the click through or signed written agreement prevails over and supersedes the conflicting provisions of these terms. This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

Words and logos marked with ® or ™ are registered trademarks or trademarks of ARM® Limited or its affiliates in the EU and/or elsewhere. All rights reserved. Other brands and names mentioned in this *document may be the trademarks of their respective owners. Please follow ARM's trademark usage guidelines at http://www.arm.com/about/trademark-usage-guidelines.php.*

Copyright © 2015, ARM® Limited or its affiliates. All rights reserved.

ARM® Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

Internal Revision History

Rev	Date	Author	Description
1.0	26/09/16	Mohammad Abo Mokh	First draft.

COPYRIGHT © 2015 ARM.
ALL RIGHTS RESERVED.

Document Revision History

Rev	Date	Description
1.0	Not yet released	First release version

Contents

COPYRIGHT © 2015 ARM.
ALL RIGHTS RESERVED.

List of Figures

No table of figures entries found.

Copyright © 2015 ARM.
All rights reserved.

List of Tables

1 Introduction

PAL Porting Guide explains the steps of porting the PAL Platform APIs to a new target platform.

This document contains explanation regarding each of PAL modules and layers.

In order to ensure correct porting, expected tests results are attached.

Notes:

- Currently mbedTLS library is a requirement from the underlying platform to support mbed client porting.
- To ensure mbed TLS is configured securely in the customer platform as part of porting, mbed TLS entropy should be bound to it (see https://docs.mbed.com/docs/mbed-os-handbook/en/5.1/advanced/tls_porting/) and implement seed persistence - `mbedtls_nv_seed_read/mbedtls_nv_seed_write`.

1.1 Intended Audience

1.2 Referenced Documents

Table 1: Referenced Documents

Ref	Name
[PROD_DOCNAME1]	
[PROD_DOCNAME2]	

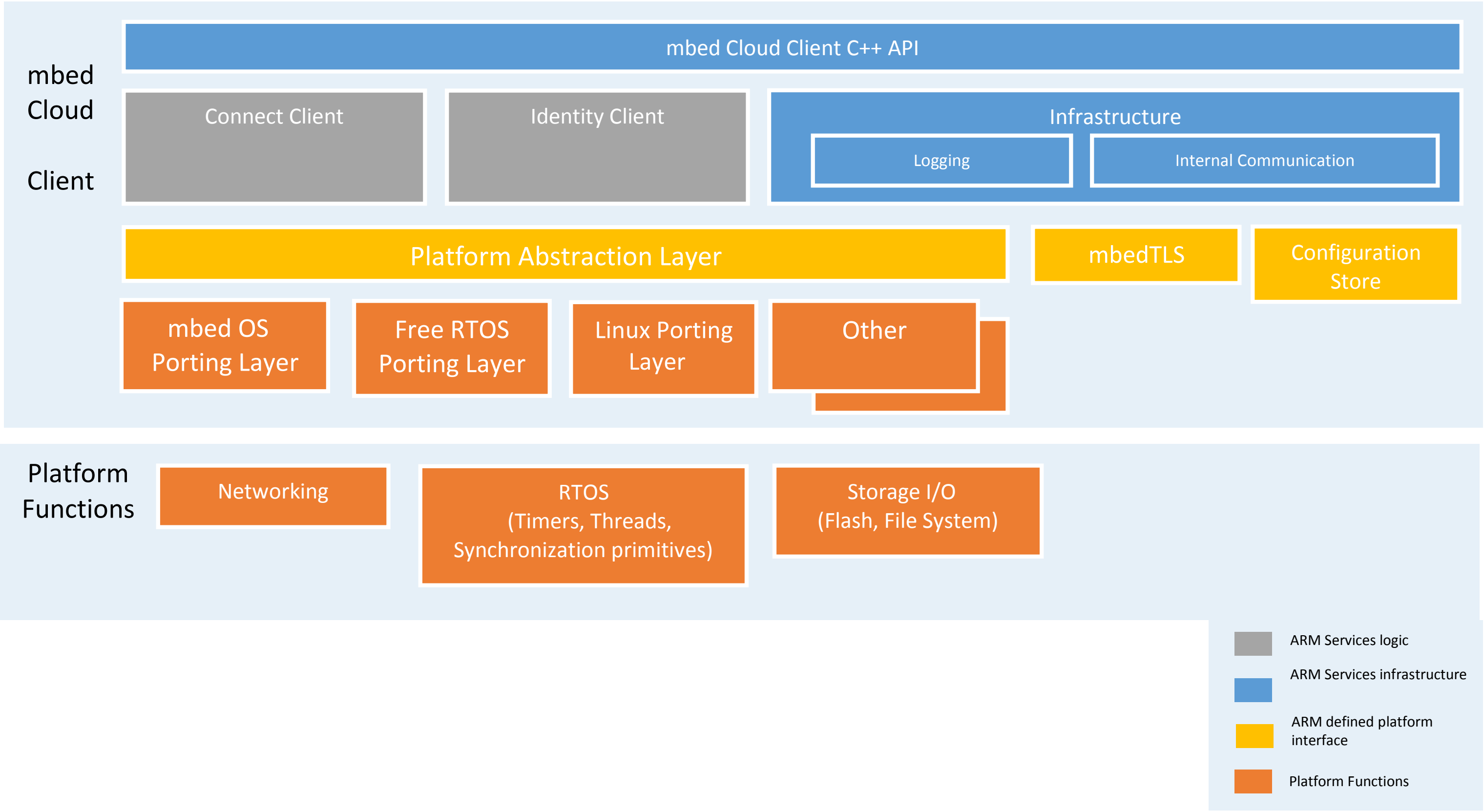
1.3 Terms and Abbreviations

Table 2: Glossary

Term	Description

2 PAL Introduction

A Platform Abstraction Layer connects the mbed-client with the underlying platform.



The main purpose of PAL is to enable easy and fast mbed client services portability. Allowing them to operate over wide range of ARM Cortex based platforms running different operating systems with various libraries (Networking for example).

3 PAL Layers

3.1 Services APIs Layer

These APIs are exposed to the mbed client services; Implemented by the PAL team.
Customer should not deal with those APIs.

3.2 Platform APIs Layer

This layer provides a standard set of base line requirements from the platform side that should be implemented in order to make the mbed client ready for running on the target platform.
This layer is implemented differently for each OS or library;
PAL provides a reference implementation over mbedOS which was developed and tested by the PAL team.

Note: for detailed Platform API documentation, please refer to the 'PAL.chm' file under PAL/Docs/ or 'index.html' under PAL/Docs/html

4 PAL Modules

4.1 Networking

The Networking module provides a POSIX like abstraction layer which provides access to TCP and UDP based communication using sockets. Platform layer should implement both synchronous and asynchronous sockets as well as the address resolution APIs (DNS).

The APIs which MUST be implemented by the platform are listed in:
PAL/Source/Port/Platform-API/pal_plat_network.h
The mbedOS reference implementation can be found in:
PAL/Source/Port/Reference-Impl/mbedOS/Networking/pal_plat_network.cpp

4.2 RTOS: (real-time operating-system)

RTOS module is responsible for providing RTOS functionality; which includes primitives such as: (Threads, Mutexes and Semaphores) also it provides an API for kernel ticks, timers, memory-pool and message-queue.

The APIs which MUST be implemented by the platform are listed in:
PAL/Source/Port/Platform-API/pal_plat_rtos.h.
Reference implementation can be found in:
PAL/Source/Port/Reference-Impl/mbedOS/RTOS/pal_plat_rtos.c

5 How to Build PAL

Currently PAL does not supply its own build system. PAL is built as an integrated source part of the mbed client solution.

For tests build, please see the Tests section.

6 How to port PAL to a new platform

In order to port for new platform, customer needs to implement the "Platform-API" interfaces (or use one of the existing reference implementations, if applicable).
It is important to add the customer files to the relevant existing build solution.

6.1 Modules Porting Notes

6.1.1 RTOS

The major assumption exists in PAL RTOS design that each thread has a unique priority and the number of threads is limited by a predefined value "**PAL_MAX_NUMBER_OF_THREADS**" (defined in PAL/Source/PAL-Impl/Services-API/pal_configuration.h).

Platform layer must define:
uint8_t g_palThreadPriorities[PAL_MAX_NUMBER_OF_THREADS];

An array which will store information about each thread priority whether it is in use or not. This array is used by the following API functions:

- pal_plat_osThreadCreate()
- pal_plat_osThreadTerminate()

NOTE: Adding a wrapper function for the main thread function will help to clear data from this array for the specific priority in case of normal thread exit, without calling the terminate API.
Please see the existing reference implementation in the PAL repository under: "Source\Port\Reference-Impl\mbedOS\RTOS\pal_plat_rtos.c".

RTOS Initialization:
The initialization function (pal_plat_RTOSInitialize) receives via mbed client context required to initialize the underlying OS functionality, for cases where one is required. (i.e. in mbedOS it is not required). If this is not required, please ignore this parameter (assuming it’s content is set to NULL). If it is required, please clearly specify what exactly is the context and how it is obtained.

COPYRIGHT © 2015 ARM.
ALL RIGHTS RESERVED.

6.1.2 Networking

The Networking module has three different feature sets that can be selected at compile time:

- TCP socket support (#if PAL_NET_TCP_AND_TLS_SUPPORT)
- Asynchronous socket support (#if PAL_NET_ASYNCHRONOUS_SOCKET_API)
- DNS name resolution support (#if PAL_NET_DNS_SUPPORT)

Please implement **all** the three sets for full service functionality.

Some important points to consider when porting the network module

- The API of the PAL network module is based on the POSIX sockets API (uses similar usage semantics) but has some significant changes:
    - All error values are returned via the return value of the function (there is no errno usage). Please consult the list of errors (in pal_errors.h) and try to return the most precise error value available.
    - The 'select' like function API has a different usage semantics than the original Posix function (no reliance on FD_SET for portability) and a more limited scope (supports very limited number of sockets).
- The initialization function (pal_plat_sockets_init) receives a context for cases where one is required (i.e. in mbedOS it is not required). If this is not required, please ignore this parameter (assuming it's content is set to NULL). If it is required, please clearly specify what exactly is the context, and how it is obtained. You may assume that the same context passed to the initialization function will also be passed to the termination function.
- The networking module should maintain or have access to an enumerated list of the network adaptors available for use by the user. The services can interact with the network adaptor list in three ways:
    - The services may query information about the available network adaptors using the 'getNumberOfNetInterfaces' and 'getNetInterfaceInfo functions'.
    - The services may bind a socket to a particular network adaptor by specifying its index in the list during socket creation (communication from this socket should then use the selected adaptor).
    - The service may populate the list with network adaptors using the 'RegisterNetworkInterface' function (note: this may not be required by all ports of the PAL. e.g. in Linux the Porting layer may enumerate the current network adaptors on its own during initialization). If this is required, please specify exactly what network interface context will be needed and how it is obtained (e.g. in mbedOS this is a pointer to the Network Interface object of the added interface).
- The internal structure of the PAL address structure is defined in the pal service layer (see pal_socketAddressInternal and pal_socketAddressInternal6). Services are not expected to modify the address structures directly (instead they use the

functions below). To make manipulation of address simpler you may use the following functions :

- pal_setSockAddrPort
- pal_setSockAddrIPV4Addr
- pal_setSockAddrIPV6Addr
- pal_getSockAddrIPV4Addr
- pal_getSockAddrIPV6Addr
- pal_getSockAddrPort

7 PAL Tests

As a part of PAL code, PAL provides a Unity Test based Framework to test the existing APIs. This framework exists under:
{PAL}/Test/

7.1 PAL Tests Sources

Pal Tests code exists in: {PAL}/Test/Unitest/ for each module there is a test file, test runner and test main for the specific OS.

7.2 PAL Tests Build & Run

In order to build PAL test executable, user needs to do the following: (for mbedOS5.1.3 over Freescale-K64F board)

1. Define the environment variable: MBEDOS_ROOT to be the father folder of "mbed-os[1]".
2. cd $(PAL_FOLDER)/Test/
3. make mbedOS_all - This will build the tests for mbedOS5.1 over Freescale-K64F board.
4. In order to build and run the tests over the platform please run:

   ```
   $ make mbedOS_check
   ```
5. In order to see debug prints please set the following flag "DEBUG=1" in compilation command:

   ```
   $ make mbedOS_check DEBUG=1
   ```

6. In order to build tests for a specific module, please edit "$(PAL_FOLDER)/Test/makefile" under mbedOS platform. Please change the value of the "TARGET_CONFIGURATION_DEFINES" to the desired module: (default value is for all existing PAL modules)

[1] Folder which holds the mbedOS.

COPYRIGHT © 2015 ARM.
ALL RIGHTS RESERVED.

* HAS_RTOS → RTOS module APIs
* HAS_SOCKET → Networking module APIs

7.3 Notes:

1. HAS_RTOS flag compiles basic RTOS test scenarios, in order to run tests for Threads, Semaphores and Mutexes and additional flag should be added to the make command:
   a. $ make mbedOS_check PAL_TEST="THREAD_UNITY_TEST"
   b. $ make mbedOS_check PAL_TEST="MUTEX_UNITY_TEST"
   c. $ make mbedOS_check PAL_TEST="SEMAPHORE_UNITY_TEST"

COPYRIGHT © 2015 ARM.
ALL RIGHTS RESERVED.

7.4 Expected tests results output

Here are captured results from successful test run:

7.4.1 RTOS

```
==== mbedos_pal_rtos ====
--------   ----------------------------------   ----
pal_rtos   pal_osKernelSysTick_Unity            PASS
pal_rtos   pal_osKernelSysTick64_Unity          PASS
pal_rtos   pal_osKernelSysTickMicroSec_Unity    PASS
pal_rtos   pal_osKernelSysMilliSecTick_Unity    PASS
pal_rtos   pal_osKernelSysTickFrequency_Unity   PASS
pal_rtos   pal_osDelay_Unity                    PASS
pal_rtos   BasicTimeScenario                    PASS
pal_rtos   TimerUnityTest                       PASS
pal_rtos   MemoryPoolUnityTest                  PASS
pal_rtos   MessageUnityTest                     PASS
pal_rtos   AtomicIncrementUnityTest             PASS
pal_rtos   PrimitivesUnityTest1                 PASS
pal_rtos   PrimitivesUnityTest2                 PASS
pal_rtos   pal_init_test                        PASS
--------   ----------------------------------   ----
TOTAL   PASS    FAIL    IGNORE
-------  ------  ------  --------
  14     14      0       0

Final status: PASS.
```

7.4.2 Network

```
==== mbedos_pal_socket ====
----------   ----------------------------   ----
pal_socket   socketUDPCreationOptionsTest   PASS
pal_socket   basicTCPclinetSendRecieve      PASS
pal_socket   basicUDPclinetSendRecieve      PASS
pal_socket   basicSocketScenario3           PASS
pal_socket   basicSocketScenario4           PASS
----------   ----------------------------   ----
TOTAL   PASS    FAIL    IGNORE
-------  ------  ------  --------
  5      5       0       0

Final status: PASS.
```

COPYRIGHT © 2015 ARM.
ALL RIGHTS RESERVED.