Developer Guide

Ayla Embedded Agent for QCA4010

Version: 3.0

Date Released: July 26, 2017

Document Number: AY006DQC6-3

Ayla Networks

Copyright Statement

Trademarks Statement

Referenced Documents

Ayla Networks does not supply all documents that are referenced in this document with the equipment. Ayla Networks reserves the right to decide which documents are supplied with products and services.

Contact Information

Ayla Networks TECHNICAL SUPPORT and SALES

Contact Technical Support: https://support.aylanetworks.com
or via email at support@aylanetworks.com

Contact Sales: https://www.aylanetworks.com/company/contact-us

Ayla Networks REGIONAL OFFICES

GREATER CHINA
Shenzhen
Room 310-311
City University of Hong Kong Research Institute Building
No. 8 Yuexing 1st Road
High-Tech Industrial Park
Nanshan District
Shenzhen, China
Phone: 0755-86581520

HEADQUARTERS
Silicon Valley
4250 Burton Drive, Suite 100
Santa Clara, CA 95054
United States
Phone: +1 408 830 9844
Fax: +1 408 716 2621

JAPAN
Wise Next Shin
Yokohama, 2-5-14
Shnyokohama, Kohokuku
Yokohama-shi, Kanagawa-ken
Yokohoma, 222-0033 Japan

EUROPE
London
30 Great Guildford St
London SE1 0HS
United Kingdom

TAIWAN
Taipei
5F No. 250 Sec. 1
Neihu Road, Neihu District
Taipei 11493, Taiwan

For a complete contact list of our offices in the US, China, Europe, Taiwan, and Japan, click:
https://www.aylanetworks.com/company/contact-us

Table of Contents

1 Introduction

The Ayla Embedded Agent enables modules to connect to the Ayla Cloud Services using a white box development model. When using the Ayla Embedded Agent software on modules that have the Qualcomm's QCA4010 or QCA4012 Software Development Kit (SDK), there are specific installation, setup, and implementation details to ensure success.

1.1 About this Document

This document describes the installation and use of the Ayla Embedded Agent software on modules using Qualcomm's QCA4010 or QCA4012 Software Development Kit (SDK). The Ayla Embedded Agent software was previously called the Ayla Device Agent (ADA), and there are still references to ADA in code and some of our documentation.

This is a companion document to *Ayla Embedded Agent Developer's Guide,* which covers all the portable interfaces and details not specific to the QCA4010 SDK.

NOTE	This version of the document is for use with ADA version 1.2 (or later) and with QCA4010 SDK version TX-2.0 CS 0.0.047.1.

1.2 Intended Audience

This document is for developers who are familiar with the Qualcomm QCA4010 SDK and are evaluating Ayla Embedded Agent software or integrating its functionality into a product.

1.3 Related Documentation

Refer to the following documentation as an additional resource to this document:

- Qualcomm's documentation covering the SDK and associated hardware.
- *Ayla Embedded Agent Developer's Guide (AY006DAR0),* for the portable aspects of the Ayla Embedded Agent software. Also the documents listed in the "*Related Documentation*" section of this guide.

1.4 Document Conventions

This document uses these Ayla documentation conventions:

- Text that you type (such as commands) and the contents of downloaded files are shown as:

```
cd wmsdk_bundle-3.1.16.1
tar xfz ada-wmsdk-src-1.0.tgz
```

- File names, scripts, names of commands, properties in a file, code, and the like are also in `courier new` font. For example: Use the `psm-dump` command to … .
- Network paths, file paths, menu paths and the like are shown in `courier new` font and each point that you have to click to navigate to the next is separated by "`/`."
- Ancillary information that is important to emphasize is shown as:

NOTE The commands provided in the example assume your evaluation board is `mw300_rd` and your chip is `mw300`. If otherwise, make the appropriate substitutions.

- Information on hazards that could damage a product or cause data loss is shown as:

Ensure that the appropriate data buffering is accounted for in deployed devices, where loss of data is critical to the core functionality or the services provided by the systems.

1.5 Revision History

Table 1 provides a summary of updates to the content of this document.

Table 1: Revision History of Content Updates and Changes

Revision #	Date (yyyy-mm-dd)	Change
1.0	2016-01-05	Initial version
2.0	2016-01-14	Minor updates
3.0	2016-11-21	Updates for ADA 1.2 (and later), QCA-TX SDK, ADW

1.6 Abbreviations and Acronyms

The following acronyms are used in this document.

ADA	Ayla Device Agent. This is a legacy term for Ayla Embedded Agent. The ADA acronym is still used in CLI commands and sources, and descriptions of either of these may refer to the Ayla Embedded Agent as the "device agent."
ADW	Ayla Device Wi-Fi
DNS	Domain Name Server
EVB	Evaluation Board
JTAG	Joint Task Action Group
mDNS	multicast Domain Name Server
NTP	Network Time Protocol
OTA	Over-the-Air
PWM	Pulse Width Modulation
SDK	Software Developer Kit
SNTP	Simple Network Time Protocol
SSID	Service Set Identifier
UDP	User Data Protocol
UUID	Universally Unique Identifier

1.7 Glossary

White Box	This is a type of Ayla endpoint that allows for a more complex and versatile device than the Black-Box class of devices. However, the development effort is significantly longer for OEMs and therefore results in longer time to market than the Black-Box modules. Some primary characteristics are: • Available for embedded or LINUX solutions. • The Ayla Embedded Agent is available as a library or source. • Well-equipped for applications with existing RTOS and networking. • The Ayla White-Box-based Cloud Agent's modular design allows code for additional functions to be included as needed.

Black Box	This is a fully-managed, Ayla-enabled module intended to be used as-is by the manufacturer. Some primary characteristics are: • Available for embedded solutions. • Provides the fastest time to market for OEMs • No custom gateway or other forms of communication agent software, including QA required regardless of the type of end-device. • Any microcontroller-based system can be enabled with cloud connectivity.

2 SDK Installation

Before installing the Ayla Embedded Agent, follow the instructions provided by Qualcomm to install:

- The Software Development Kit (SDK) and the cross-compilation tools required.
- The drivers to access the serial port and JTAG devices for your evaluation board or device.

3 Ayla Embedded Agent Installation

Once you have installed everything outlined in Section 2 for the SDK, follow the steps in this section to install the Ayla Embedded Agent.

If you copy and paste commands from this document, the results may be UTF-8 characters that look the same as the simple ASCII ones, but may not work. To avoid problems with the commands, make sure that they contain minus signs (dashes), not other symbols, like the em dash.

1. Extract the Ayla Embedded Agent package into the directory created when you extracted the SDK as follows:

NOTE This is the top-level directory, which contains the directory `target`.

```
cd qca4010
tar xfz ada-qca4010-src-1.2.tgz
```

2. Copy the files into the SDK tree as follows:

```
cp -R ada-qca4010-src-1.2/* .
```

3. Build the demo application using the `make` command in the top-level directory:

```
make
```

The resulting files will be in `target/bin/*ayla_demo.bin`.

4. Download the demo to the evaluation board as shown below:

```
make download
```

5. Press the Reset button to ensure that the device boots into the demo.

NOTE Everytime you run `make download`, the configuration of the module is cleared and must be reset as described below.

4. Setting up the Demo for the First Time

The first time you use the demo, you have to set the serial number and key for the device, as described in this section.

4.1 Obtain the Device Serial Number (DSN) and Key XML Files

Upon request, we provide the DSN and private/public key pair from the Ayla Factory Service (also referred to as AFS). The main purpose of AFS is to generate this information to authenticate the device with the cloud service. Please contact your Ayla Customer Support representative to obtain the Open Agent XML Provisioning files, which are used in the examples shown throughout this section. Make sure you specify that your modules use the Ayla part number, AY008QCM1.

4.2 Set the Device Serial Number and Key

A different Ayla DSN and key must be configured for each device. Each DSN and key comes in a separate XML file like the following:

```
<?xml version=”1.0 encoding=”UTF-8?>
<f-device>
 <dsn>AC000W00012346</dsn>
 <public-key>-----BEGIN RSA PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA4uBo4pDNe7aS4CJS0ck1
pKaHCS1fv2Oqr+DnPiA5Ue/6Htac7j9DQyc4qD8sxHip/7v9M2HqGKv
FPhkb7SDjbj+v/qYca2cWXhqUYv9H5wEeCUG2MaBYGUZ0HfnBiD/PIULWmYBLZ6D
4yF3dQIi+MCgWJU9GCVR6pCt+02SUfIxOQWPslx5nlkMMpj+E1ORjYPSa7kIeoik
waoPc85UfTA+1TSOq5tNHZu+CuoYBkRPkbXM1bhrQx98TY8FQG/zS+kqmedpV73S
xotItJjtlupKQI6C42Bg867ncuvNufUF+58WZGobaZZIu3I3ouXxGqIgqdOMYi+q
kqidaqab
-----END RSA PUBLIC KEY-----
</public-key>
</f-device>
```

Use the `conf show` command to see if these values have already been set up for you, or to verify your entries afterwards

To set the DSN and key into PSM for the Ayla Embedded Agent, the key must be collapsed into a single line, using the part between the dashed lines and eliminating blanks.

The factory log also requires the `mfg_model` and `mfg_serial number`. These values are available for product tracking purposes and should be short strings, not containing commas. The `mfg_model` must be 23 characters or less, and the `mfg_serial` must be 31 characters or less. They may not be empty, but do not have to be unique.

The resulting commands are shown below (the prompts and most of the key are omitted).

```
id dev_id AC000W000123456
id mfg_model EVB
id mfg_serial EVB1
file start 0
file add MIIBIjANBgkq … DNe7aS4CJS0ck1
file add pKaHCS1fV2Oq … ESEd7v9M2HqGKv
file add FPhkb7SDjbj+ … D/PIULWmYBLZ6D
file add 4yF3dQIi+MCg … ORjYPSa7kIeoik
file add waoPc85UfTA+ … /zS+kqmedpV73S
file add xotItJjtlupK … XxGqIgqdOMYi+q
file add kQIDAQAB
conf save
```

The modules may immediately connect to the Ayla Device Service (ADS) if the modules indicate that the connection attempt is part of manufacturing test. To configure the modules to indicate this, you must internally set the `ada_conf.test_connect` field to `1`, or use the `client test` CLI or another similar method of your preference. This indication is not persisted across reboots. It is, however, the most convenient way to provision devices.

- The `test_connect` field should ***not*** be set on every boot to avoid issues.
- The Ayla module part number (AY008QCM1 in this case) must be specified when the DSN is reserved to avoid connection issues.

For the easiest demo, run the `client test` command the first time you connect, as shown below:

```
client test
```

If `test_connect` is not used, the alternative is to enable the device on the ADS. This provides information about the manufacturer and the device, including its MAC address and hardware ID from the flash. To generate the log line, set the time and use the `factory-log CLI` command as shown below:

```
shell> time
[ada] 2015-01-01T00:00:03

shell> time 2015-11-28T20:47:20
[ada] time set

shell> factory-log
[ada] factory-log line:
[ada] 3,1448302853,2015/11/23 18:20:53
UTC,label,0,AY008QCM1,AC000W000441610,00037f2e035a,EVB,EVB1,42,,0dfc7
900,smartplug1,0,ADA demo customer,ayla_outlet1_demo 1.0 Nov 23 2015
10:15:43
```

Send the last line, starting with the "3," to your Ayla Customer Technical Lead (often referred to as CTL). Several device logs can be sent in the same file. Please be careful to omit the "[ada]" prefix, and not to add spaces or new lines to that single line.

4.3 Set the OEM Secret

All devices using the Ayla cloud have an OEM ID, an OEM model, and an OEM key (also called the OEM secret). The OEM ID and OEM model identify the type of device and are provided by the demo code.

NOTE The values for the OEM ID and model in the demo are for an Ayla smart plug and should be changed to the values assigned for your product.

The OEM key (secret) is a secret string that is provided with each OEM ID and used to authenticate the OEM when communicating and transmitting information with the Ayla cloud. To authenticate the device from your OEM, you must enter the OEM secret. This secret is saved in encrypted form on the device and must be entered for each device after entering the public key. The purpose of the OEM secret is to prevent others from creating similar devices under your OEM, and should therefore be kept secret. Authentication is not required for devices used on the Ayla Networks developer site, including those using the demo. However, as a best practice, you should authenticate these devices since authentication is required when the devices are switched from the Ayla Developer Service to the Ayla Field Service.

If you have the OEM key, you can set it with the following command, which should be done before the device contacts the ADS. Use the `oem key` command as follows:

```
shell> oem key <secret-string>
```

The OEM model cannot be changed from the one that is set using the `oem key` command, unless the command string is as follows. The asterisk at the end means that any OEM model can be used with the encrypted string.

```
shell> oem key <secret-string> *
```

You can also view and set the OEM Secret in the OEM Profile page of the OEM Dashboard, as shown in Figure 1.

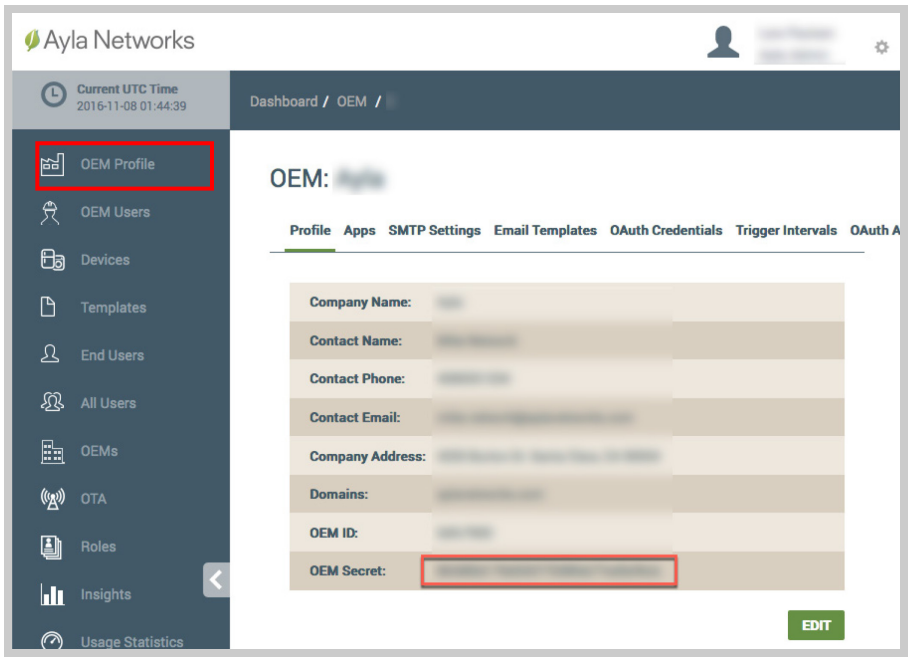

Figure 1: Setting the OEM Secret

NOTE Refer to the *OEM Dashboard User's Guide* (on support.aylanetworks.com) for more information on making changes to the OEM Profile.

4.4 Set up Wi-Fi

NOTE This section assumes that you use the Ayla Device Wi-Fi (ADW) facilities to manage your Wi-Fi since the demo uses ADW. ADW maintains the Wi-Fi configuration, keeping the device connected if possible, and supports Wi-Fi setup either with a web page or a secure Wi-Fi setup mobile application.

The Wi-Fi configuration is typically done using a smart phone application, such as Ayla's Aura or AMAP. However, Wi-Fi may also be configured directly from the console interface of the device. The following example shows the commands involved when configuring the Wi-Fi Profile from the console.

```
shell> wifi my-ssid my-wpa2-passphrase
shell> conf save
shell> wifi join
```

In the command example above:

- The first line sets the SSID and passphrase. At this time, only WPA2-Personal security is supported by the demo.
- The second line saves the configuration for subsequent runs.
- The last line actually starts the Wi-Fi connection.

5 Running the Demo

When you first run the demo, you must register the device and set up the device properties as described in this section.

When the demo starts, its output consists of several debug and other log messages on the serial console, as shown in the following example:

```
_WAKEMGR_Register  item:0x43a4fc
cmnos_allocram_init: start=0x455d98 size=696812
NUM_DEV=2 FWMODE=5 FWSUBMODE=0 FWBR_BUF 1
Mac address is : 00:03:7f:2e:03:5a

…

CLI:
shell> [ada] 3192 i client: ADA-qca4010 1.1-eng 2015-??1-23 13:27:46
jre/ada/fcd48a6+
demo: connecting to Ayla2
[ada] 4339 i client: ada_client_up: IP 192.168.1.10
ads_demo_connect_sts: 0
[ada] 4340 i client: get DSN AC000W000441610

[ada] 4948 i client: client_connect_cb: 1
[ada] 5007 i client: client_connect_cb: 16
_dhc_setip Got IP address 172.16.11.204
Can't get IP addr by host name smartplug1-0dfc7900-
device.aylanetworks.com
[ada] 30038 i client: ada_client_up: IP 172.16.11.204
…
ads_demo_connect_sts: 1
…
[ada] 21:32:06.507 i client: get DSN AC000W000441610
…
[ada] 21:32:07.685 i notify: ANS server ans.aylanetworks.com at
54.209.68.173
[ada] 21:32:09.500 i client: clock was 2015-11-23T21:32:07 UTC
[ada] 21:32:09.501 i client: clock now 2015-11-23T21:32:09 UTC
[ada] 21:32:09.502 i client: module name "QCA4010_evb2" key 65655.
```

Once connectivity with the Ayla Device Service is established, the message, “ads_demo_connect_sts: 1” prints. The production device may also indicate connectivity by illuminating an LED.

5.1 Register the Device

Create a user account on https://developer.aylanetworks.com. There you can register the device, or you can use the AMAP or Aura application for iOS or Android. Before completing the registration or using the application:

- Make sure that the computer you are using for the browser or your mobile device is on the same local network as the device.
- Verify that you are using the same wireless access point (AP).

After creating a user account, log in to the developer site and click **Register New Device**. This is a link that takes you to the instructions to register a new device

5.2 Set Up Device Properties

The demo in `target/demo/ayla_demo` implements:

- The `outlet1` property of a smart plug
- The pair of string properties called `string_in` and `string_out`
- A pair of integer properties called `input` and `output`

You can set these properties from the Ayla Developer Portal http://developer.aylanetworks.com, or using the Ayla Control mobile app. Following are descriptions of the properties:

- The `outlet1` property to the device controls the PWM LED (DS-4 on the SP240 EVB) using the green PWM output. Be sure to connect the SENSOR jumper (JP17) and turn on switch S12-2 to enable the LED.
- The `string_in` and `string_out` properties are linked. Whenever a string value is sent to `string_in`, the value is sent back to the service as `string_out`.
- The `input` and `output` properties are linked. Whatever integer value is sent to `input`, this value is sent back to the service as `output`.
- The `oem_host_version` property is a string that gives the software version of the demo that is running on the device. Currently, the `version` property gives the same information.

6 Sample Application Code

The sample applications on are provided as working examples of how the Ayla Embedded Agent APIs are used. These applications should be evaluated only as examples and not considered the only way or even the best way to use the APIs.

The sample application source is in `target/demo/ayla_demo` and is composed of the following C files:

- `ada_demo.c`
- `conf.c`
- `demo_ota.c`

Additionally, the following files are copied to `target/demo/ayla_demo` from `target/demo/sdk_shell` by the build.

- `app_start.c`
- `sdk_shell.c`
- `swat_shell.c`
- `swat_task_manage.c`

This section provides descriptions of the files and how they should work.

6.1 ada_demo.c

This file contains the code specific to the demo and its properties.

The demo starts in `ada_demo_start()`, which calls `client_conf_init()` in `conf.c` and then `ada_init()`. This initializes the demo and the Ayla Embedded Agent.

The demo then registers two property managers: `ads_demo_mgr` and `ada_demo_sprops`. The property manager `ads_demo_mgr` is used to show the status of connectivity to the Ayla cloud. The property manager `ada_demo_sprops` is used in the registration of demo properties as described below.

`ada_demo_start()` then calls `demo_ota_init()` to prepare to handle OTA update downloads.

`ada_demo_start()` calls `ada_demo_ntp()` to start the SDK's SNTP client to get the current time. NTP is optional. You can configure the list of NTP servers (`DEMO_NTP_SERVERS`) in the `conf.h` file or via another application-specific method; however, you should confirm that the servers are accepted by the device manufacturer.

`ada_demo_start()` then calls `ada_client_up()` to start the Ayla Embedded Agent.

`ada_demo_start()` uses the `ada_sprop_mgr_register()` API to register the properties handled by the device through the simple property manager. The table used is called `ada_demo_sprops`. That table describes the various properties by name and gives functions to handle the set and get operations.

`ada_demo_outlet1_set()` is used to handle the setting of the `outlet1` property. On some evaluation boards, `ada_demo_outlet1_set()` also controls an LED, for example, on PWM output 2 (green LED).

`ada_demo_input_set()` is used to handle the setting of the `input` property by reflecting its value back to the cloud as the `output` property.

`ada_demo_string_set()` is used to handle the setting of the `string_in` property by reflecting its value back to the cloud as the `string_out` property.

6.2 conf.c

This file initializes parts of the ADA client configuration from the SDK and compiled-in defaults.

`client_conf_init()` reads the MAC address and the system UUID, which ADA sends to the Ayla Device Service (ADS) to confirm that the device is not using the same serial number as another device. `client_conf_init()` sets the `enable` and `get_all` flags in the `client_conf` structure, as well as the `poll_interval`, which is used in cases where connectivity to the Ayla Notification Service (ANS) is lost.

6.3 demo_ota.c

This file contains code to demonstrate the OTA firmware download interfaces. A host OTA can be deployed from the developer's site, and the code in `demo_ota.c` receives the host OTA and loads it into the best available partition provided by the SDK. After a successful download, the module reboots into the new image. The customer application may want to modify this reboot process by adding additional checks on the image during the download or by adding additional decisions about when to apply and reboot into the new image.

To use the OTA facility, build your application using the SDK or the provided `Makefile`. The resulting OTA image file is placed in `target/bin/ota*.bin`. Using the Ayla Developer Site, navigate to the device's Host image page and create, upload, and deploy the image.

NOTE Refer to the *Ayla Embedded Agent Developer's Guide* (on support.aylanetworks.com) for information on the OTA interfaces.

7 Source Code Organization

The library source code for the device agent, not including `sample_app`, is delivered in the subdirectory `target/ayla`. This directory contains a `Makefile.inc` file and the libraries `ext_xyssl`, `libayla`, and `libnet`. These libraries are described in detail in the *Ayla Device Agent for Embedded Systems Developer's Guide*. Details specific to QCA4010 are documented in this section of this document.

7.1 Library ext_xyssl

This is an open-source SSL library from which the device agent uses some encryption routines. This library is not used for SSL/TLS itself. XYSSL is copyrighted by Christophe Devine and provided under the BSD license, which is included in the source.

7.2 Library libada

This library contains the Ayla Embedded Agent. Most of the library is portable and described in the *Ayla Device Agent for Embedded Systems Developer's Guide*. The platform-specific portions are under the `al` subdirectory. The abbreviation "al" is for adaptation layer.

The descriptions of the files in this library are as follows:

al/qca4010/ada_lock.c

This file implements an interface to the Threadx mutexes.

al/qca4010/client_task.c

This file contains code for the client thread, its initialization, timers, locking, and work queue.

The function `ada_init()` does the following:

- Initializes the logging subsystem, including setting severity levels to be logged for each subsystem.
- Initializes the timers, and the `client_mutex`, which protects most of the client data structures.
- Calls `conf_load_config()`, `clock_init()`, `client_init()`, `net_init()`, `stream_init()`, `client_init)` and then creates the `client_task_queue` and the client thread.
- Calls `client_mdns_init()`.

The client thread runs in the function `client_idle()`. This initializes takes the client lock and then enters the idle loop.

The idle loop handles any pending timers that need to be handled in the client thread while still holding the client lock. Then, the idle loop waits for callbacks on the

`client_task_queue`. This waits until the next scheduled timer or until an event occurs. If a callback is found, its handler is called and the loop repeats.

The function `client_task_cm()` runs from a periodic timer approximately once per second to monitor the network status. If the status of the Wi-Fi interface changes, the ADA client and internal web server are stopped or restarted as appropriate.

The remainder of the code in this file implements client timers and synchronization needed with the server thread.

The functions `client_lock_int()` and `client_unlock_int()` implement the locking around the client structures. These functions maintain some debugging information that can be useful if there is trouble in this area. The global string pointer `client_mutex_func` holds the name of the function that last successfully locked the `client_lock`. The global integer `client_mutex_line` gives the line number in that function. The global pointer `client_mutex_owner` is a pointer to the OS task that holds the mutex and is used to detect recursive attempts on the lock.

al/qca4010/http_client.c

This file implements the interface between the HTTP client API of the device agent and the TCP/TLS sockets in the `libnet` library.

al/qca4010/notify_task.c

This file implements timers for the notification subsystem, which interacts with the Ayla Notification Service (ANS).

al/qca4010/server.c

This file implements the HTTP server for the Ayla Embedded Agent.

al/qca4010/stubs.c

This file contains miscellaneous, required routines, some for features that are not functional in this architecture, and some that are simple interfaces between the Ayla Embedded Agent and the SDK.

7.3 Library libayla

The sources in this library are portable. These are described in the *Ayla Device Agent for Embedded Systems Developer's Guide*. The exceptions to this are as follows:

arch/qca4010/assert.c

This file implements the action to be taken upon assertion failure. Currently, this action stops in the debugger before resetting the system. The application can re-implement this function, overriding the library's supplied function, to take its own action upon assertion failure.

arch/qca4010/clock.c

This file implements a soft real-time clock for the Ayla Embedded Agent, using `time_ms()` from the SDK.

arch/qca4010/conf_flash.c

This file implements access to the flash for configuration files used by the Ayla Embedded Agent. The file uses the DSET subsystem in the SDK with dataset IDs from `DSETID_VENDOR_START + 0 thru 7` currently.

arch/qca4010/malloc.c

This file is a wrapper around the memory allocation interfaces for the SDK.

arch/qca4010/malloc_hist.c

This file implements debug tracking for memory allocations by the Ayla Embedded Agent. The file cannot track any other memory allocations.

7.4 Library libnet

This library contains a thin layer of code and macros that interface from the Ayla Embedded Agent to the SDK. The library contains the following files:

dnss.c

This file implements a DNS and mDNS server.

dnss_util.c

This file is part of the DNS and mDNS server.

ipaddr_fmt.c

This file contains some IP address formatting functions.

net_crypto.c

This file provides interfaces between the Ayla Embedded Agent and the SDK and XYSSL for encryption and message digest functionality.

net_mbuf.c

This file implements network buffers.

net_qca4010.c

This file provides asynchronous UDP and TCP interfaces on top of the SDK.

ssl.c

This file interfaces with the SDK's TCP and SSL support for HTTP client sockets.

7.5 JSMN Parser

JSMN (pronounced Jasmine) is a small JSON parser copyrighted by Serge A. Zaitsev and distributed under a generous source license. The source is placed in `target/jsmn` along with the LICENSE file.

4250 Burton Drive, Santa Clara, CA 95054
Phone: +1 408 830 9844
Fax: +1 408 716 2621