Open-CMSIS-Pack

Technical Project Meeting 2021-07-13

This meeting is recorded !

Open-CMSIS Pack

Linaro

Agenda

- Welcome and review of the agenda
- Actions from last week
- CLI tool for assisted Pack creation of Cmake based projects
    - CMSIS-13
- Feedback collection
- Project Generation – potential tool flow (discussion)
    - Issue #12
- Wrap Up

Actions from last week

- Create: CMSIS-Build Gap Analysis (Issue #11) – Joachim Krech
    - Summary from Shadowfax Componentization Exploration Team

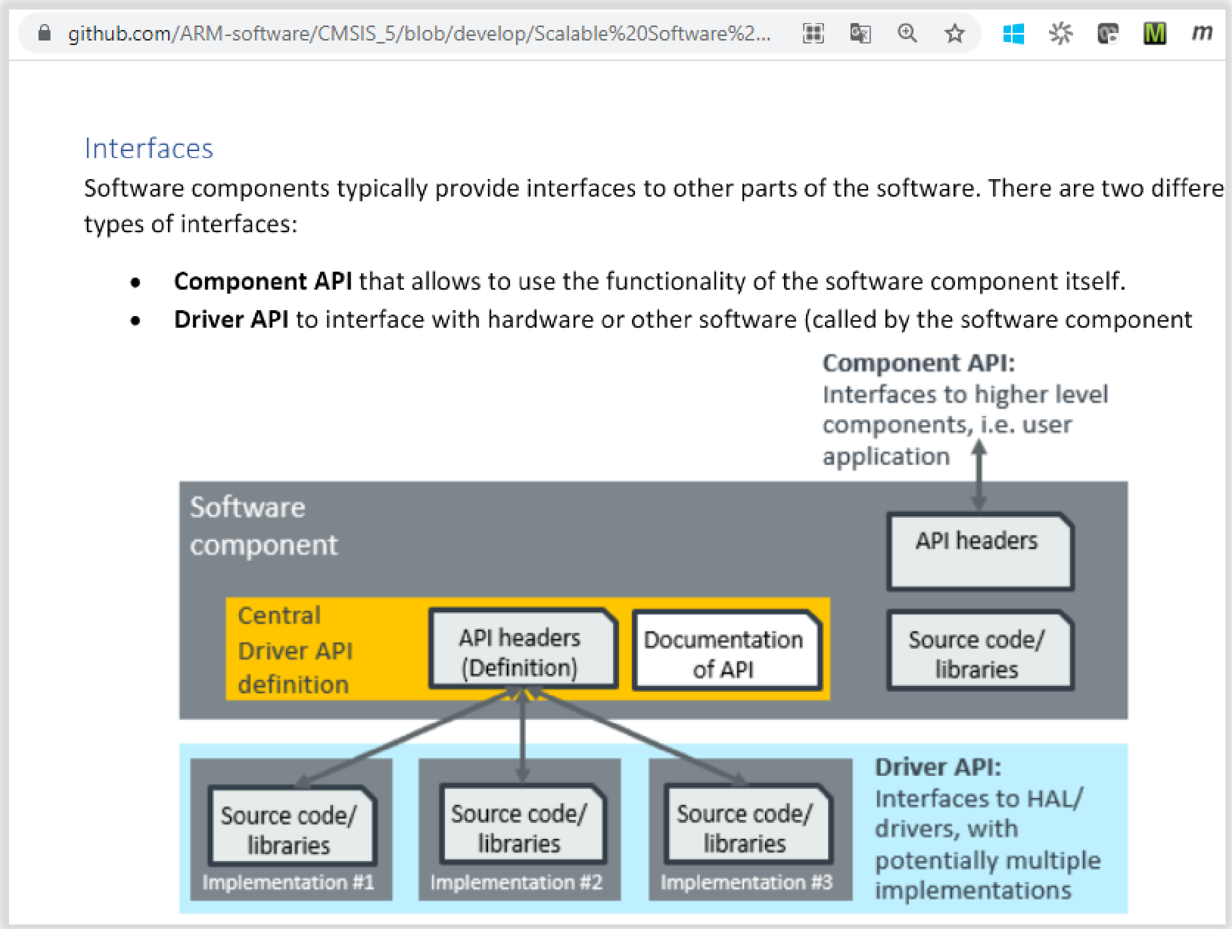
Assisted Pack creation based on CmakeLists

CMSIS Pack Generator

Daniel Brondani
13th July 2021

Background: CMSIS-Pack generation is a manual process

This document explains how to structure a software stack
https://github.com/ARM-software/CMSIS_5/blob/develop/Scalable%20Software%20Stack.pdf



github.com/ARM-software/CMSIS_5/blob/develop/Scalable%20Software%2...

Interfaces

Software components typically provide interfaces to other parts of the software. There are two differe
types of interfaces:

- **Component API** that allows to use the functionality of the software component itself.
- **Driver API** to interface with hardware or other software (called by the software component

A pack is described in XML format. Today there is limited tool support for pack creation.

To create a pack, the pack provider must:

- **Define the User View of the Components**
  - Component description and component version
  - Link to documentation
  - Dependency to other components
  - Configuration file(s) for the component
  - Etc.

- **Maintain a list of source files for the Components**

Can we simplify pack creation for Cmake based projects?

Delivering embedded software components in a CMSIS Pack has several advantages over CMakeLists.
Software component providers can specify the interfaces and relationship to other software components.

CMake is used to maintain many projects, hence automate pack creation is desired. We explored two ways:

- **TF-M Approach:** takes as input a PDSC template, a python settings file with a list of CMake target build configurations
  - Python settings file (manually written) maps CMake targets and components to be generated.
  - Python uses the CMake File API to retrieve build info for every CMake target build configuration.
  - and combines this info with the maps provided in the Python settings file to generate components and conditions.

- **AWS Approach:** uses a manifest.yml template and CMakeLists:
  - manually written CMake variables describing source files and include paths.
  - A CMake script appends the build info to the manifest.yml file.
  - The updated manifest.yml file is used to generate a PDSC component.

These trails resulted in the following proposal for **packgen utility**, an assisted pack generation process from Cmake:

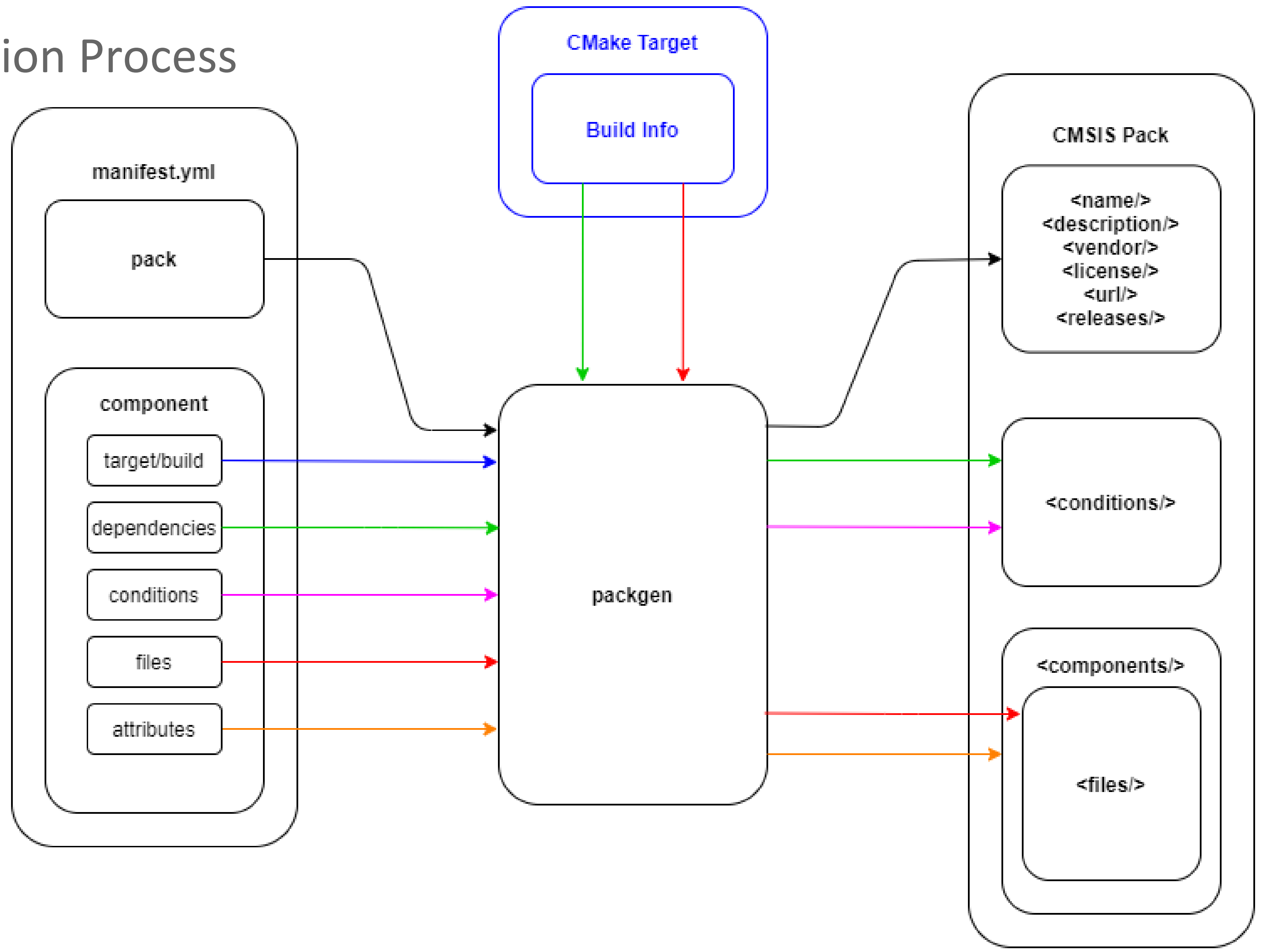
  - The **packgen** utility reads a manifest YAML file with metadata
  - It then runs the CMake generation step to retrieve targets build information (source files and include paths)
  - It finally generates the pack description, copy pack files, compress it and check its validity.

Demo: https://github.com/brondani/aws-iot-device-sdk-embedded-C/tree/packgen-cmake/tools/cmsis

CMSIS Pack Generation Process

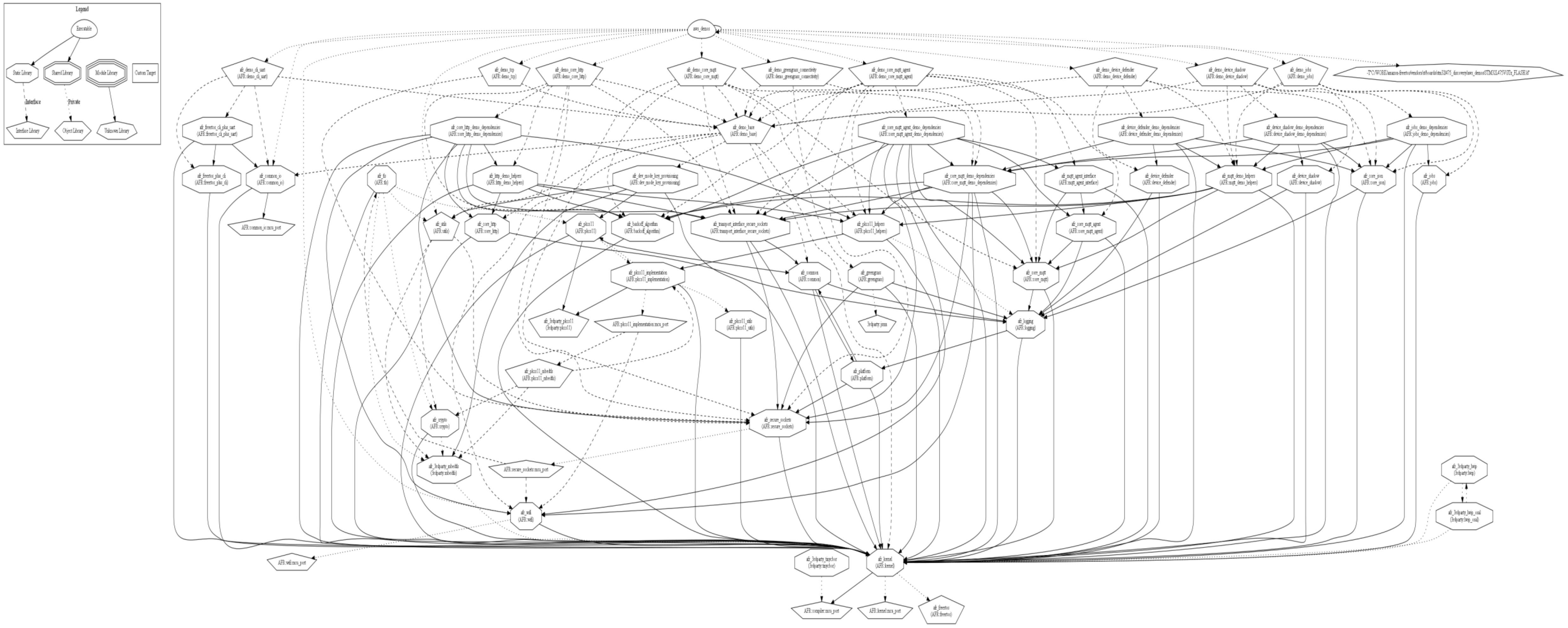
Input:
- CMakeLists.txt
- manifest.yml

Output:
- PDSC file
- Compressed Pack



Example of CMakeLists dependencies

Graphviz generated by CMake for https://github.com/aws/amazon-freertos



Proof of Concept

The PoC with **packgen** binaries and a getting started guide can be found in the following forked branch: https://github.com/brondani/aws-iot-device-sdk-embedded-C/tree/packgen-cmake/tools/cmsis

CMakeLists.txt

```
add_library(cmsis_aws_iot_ota)
target_sources(cmsis_aws_iot_ota PUBLIC ${OTA_SOURCES})
target_include_directories(cmsis_aws_iot_ota PUBLIC
${OTA_INCLUDE_PUBLIC_DIRS})
set_target_properties(cmsis_aws_iot_ota PROPERTIES DEFINE_SYMBOL "")
set_target_properties(cmsis_aws_iot_ota PROPERTIES LINKER_LANGUAGE C)

add_library(cmsis_aws_iot_json)
target_sources(cmsis_aws_iot_json PUBLIC ${JSON_SOURCES})
target_include_directories(cmsis_aws_iot_json PUBLIC
${JSON_INCLUDE_PUBLIC_DIRS})
set_target_properties(cmsis_aws_iot_json PROPERTIES DEFINE_SYMBOL "")
set_target_properties(cmsis_aws_iot_json PROPERTIES LINKER_LANGUAGE C)

add_library(cmsis_tinycbor)
target_sources(cmsis_tinycbor PUBLIC ${TINYCBOR_SOURCES})
target_include_directories(cmsis_tinycbor PUBLIC
${TINYCBOR_INCLUDE_DIRS})
set_target_properties(cmsis_tinycbor PROPERTIES DEFINE_SYMBOL "")
set_target_properties(cmsis_tinycbor PROPERTIES LINKER_LANGUAGE C)
```

```
Green   1. component in the pack
Red     2. component in the pack
Blue    External component
```

CMake Targets

```
TARGET: cmsis_aws_iot_ota
src: libraries/3rdparty/tinycbor/src/cborencoder.c
src: libraries/3rdparty/tinycbor/src/cborencoder_close_container_checked.c
src: libraries/3rdparty/tinycbor/src/cborerrorstrings.c
src: libraries/3rdparty/tinycbor/src/cborparser.c
src: libraries/3rdparty/tinycbor/src/cborparser_dup_string.c
src: libraries/3rdparty/tinycbor/src/cborpretty.c
src: libraries/3rdparty/tinycbor/src/cborpretty_stdio.c
src: libraries/coreJSON/source/core_json.c
src: libraries/ota.c
src: libraries/ota_base64.c
src: libraries/ota_interface.c
inc: libraries/3rdparty/tinycbor/src
inc: libraries/coreJSON/include
inc: libraries/include
inc: libraries/portable

TARGET: cmsis_aws_iot_json
src: libraries/coreJSON/core_json.c
inc: libraries/coreJSON/include

TARGET: cmsis_tinycbor
src: libraries/3rdparty/tinycbor/src/cborencoder.c
src: libraries/3rdparty/tinycbor/src/cborencoder_close_container_checked.c
src: libraries/3rdparty/tinycbor/src/cborerrorstrings.c
src: libraries/3rdparty/tinycbor/src/cborparser.c
src: libraries/3rdparty/tinycbor/src/cborparser_dup_string.c
src: libraries/3rdparty/tinycbor/src/cborpretty.c
src: libraries/3rdparty/tinycbor/src/cborpretty_stdio.c
inc: libraries/3rdparty/tinycbor/src
```

Proof of Concept

cmsis.yml

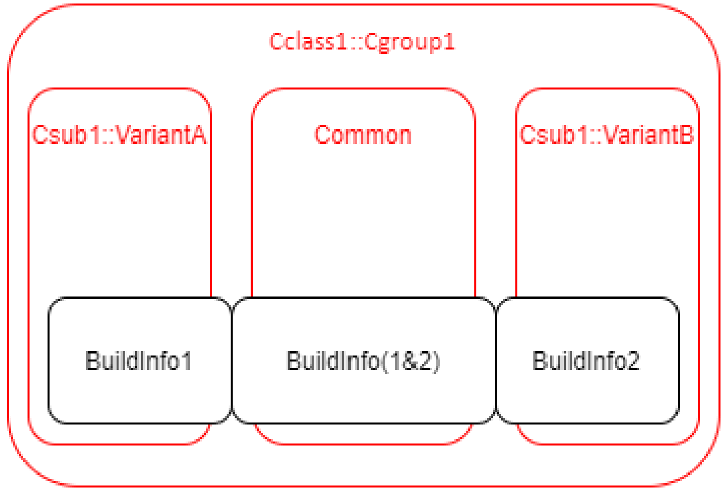
```
components:

  - name: cmsis_aws_iot_ota
    target: cmsis_aws_iot_ota
    attributes: {Cclass: "AWS IoT", Cgroup: "AWS IoT OTA", Cversion: "1.0.0"}
    description: "Client library for Device Over-the-air Update service"
    dependencies: [cmsis_tinycbor, cmsis_aws_iot_json]
    conditions:
      - require: {Cclass: "Data Exchange", Cgroup: "CBOR", Csub: "TinyCBOR"}

  - name: cmsis_aws_iot_json
    target: cmsis_aws_iot_json
    attributes: {Cclass: "AWS IoT", Cgroup: "coreJSON", Cversion: "1.0.0"}
    description: "Parser for ECMA-404 JSON standard"
```

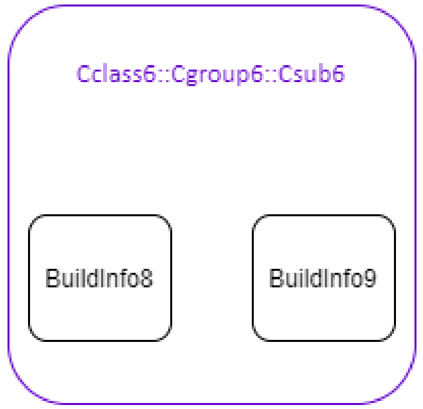
PDSC

```
 <conditions>
  <condition id="cmsis_aws_iot_ota Condition">
    <require Cclass="AWS IoT" Cgroup="coreJSON"/>
    <require Cclass="Data Exchange" Cgroup="CBOR" Csub="TinyCBOR"/>
  </condition>
 </conditions>

<components>
  <component Cclass="AWS IoT" Cgroup="AWS IoT OTA" Cversion="1.0.0"
   condition="cmsis_aws_iot_ota Condition">
    <description>Client library for Device Over-the-air</description>
    <files>
      <file category="source" name="libraries/ota.c"/>
      <file category="source" name="libraries/ota_base64.c"/>
      <file category="source" name="libraries/ota_interface.c"/>
      <file category="include" name="libraries/include/"/>
      <file category="include" name="libraries/portable/"/>
    </files>
  </component>

  <component Cclass="AWS IoT" Cgroup="coreJSON" Cversion="1.0.0">
    <description>Parser for ECMA-404 JSON standard</description>
    <files>
      <file category="source" name="libraries/coreJSON/core_json.c"/>
      <file category="include" name="libraries/coreJSON/include/"/>
    </files>
  </component>
</components>
```

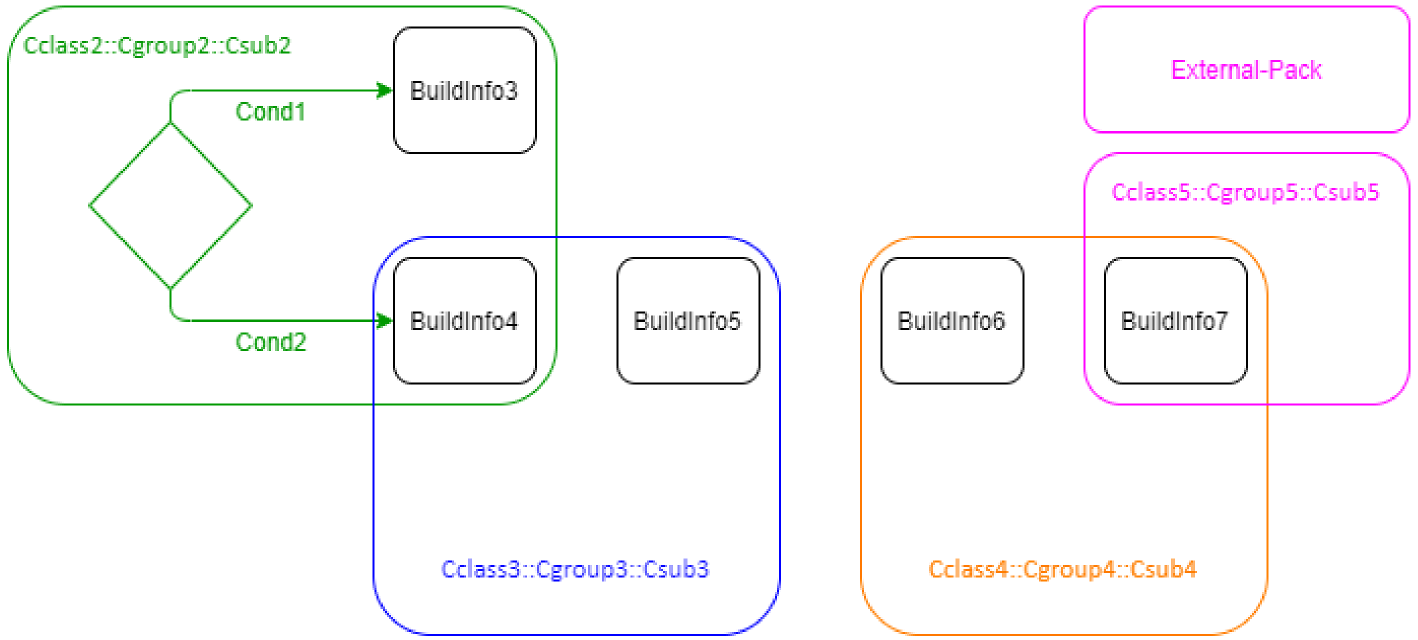
Green	1. component in the pack
Red	2. component in the pack
Blue	External component

Further work: handling different use cases





Use case	Component	CMake Target	Build Configuration	Build Info	File Condition
1	Cclass1::Cgroup1::Csub1::VariantA	Target1	BuildConfig1	Build1	
	Cclass1::Cgroup1::Csub1::VariantB		BuildConfig2	Build2	
	Cclass1::Cgroup1::Common		Common	Build(1&2)	
2	Cclass2::Cgroup2::Csub2	Target2	BuildConfig3	Build3	Cond1
			BuildConfig4	Build4	Cond2
3	Cclass3::Cgroup3::Csub3	Target3	Any	Build5	
4	Cclass4::Cgroup4::Csub4	Target4	Any	Build6	
	External-Pack Cclass5::Cgroup5::Csub5	Target5	Any	Build7	
5	Cclass6::Cgroup6::Csub6	Target6	Any	Build8	
		Target7		Build9	



Thank You
Danke
Gracias
谢谢
ありがとう
Asante
Merci
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה

Feedback

- 5th meeting of the Open-CMSIS-Pack technical project meeting – Time to reflect
- What is working well?
- What is not working for you?
- What should we do differently?
- What can we do to improve?

- Checking availability during summer vacation period (July / Aug 2021)
    - Action all: Please send email to Joachim.Krech@arm.com listing the weeks you are unavailable (wk29 – wk35)

Discussion Topic

- Project Generation – potential tool flow
  - Issue #12

Wrap Up

Issue Overview

Next week:

- Introduction to CMSIS-Zone methodology

August:

- Protecting CMSIS-Pack from malicious tempering (TBD)
- Kick-off development for project creation and maintenance MVP CMSIS-12

- Next Meeting: Tuesday July 20th 2021, 15:00 – 16:00 (UK)

Thank you

Linaro