

Open TV One - Build Tree

Scope

Purpose of this document is to present necessary information for build tree supplied to third parties responsible for implementing HAL.

Introduction

Delivery package for board vendor consists of precompiled binaries of Open TV One software, precompiled libraries for linking with hardware abstraction layer (HAL) libraries implemented by board vendors. Part of the package is a build tree used while implementing HAL and validating it.

Tree parties are identified in porting Open TV One software to a new board:

- Chip vendor
- Board manufacturer
- Open TV team

Chip vendor is responsible for delivering vanilla SDK and vanilla HAL port. In case that **chip vendor** delivers reference platform, SDK and HAL implementation must be ported to run on reference platform. HAL validation test report must be supplied at the end of porting process. Delivery package must at least contain HAL modules necessary for the Free-to-Air application to successfully run. Depending on agreement, it may be extended to more HAL modules.

Board manufacturer is responsible of adjusting SDK and HAL implementation per specific requirement of the board. It must ensure that reference implementation of SDK and HAL is possible to be used at any point in development process. Any changes done in shared code must be surrounded with proper conditional compilation primitives in order to maintain original state. It is permitted that **Board manufacturer** creates its own copy of HAL and SDK along with configuration files. In that case **Board manufacturer** responsible for adjusting its SDK and HAL copy when **Chip vendor** updates vanilla HAL and SDK.

Open TV One team is responsible for delivering necessary binaries, libraries and resources in order to test and prepare complete Open TV One solution. It needs to deliver and maintain following Open TV One packages:


- Free to Air distribution
- CAS abstraction layer test suite distribution
- CAS certification distribution
- Customization distribution for target operator

Depending on the agreement, **Open TV One team** may share reference HAL code to **Chip vendor** or **Board manufacturer** prior the start of the HAL implementation for specific platform and board.

Structure and organization given in previous sections must be respected. If at any point a need for change or update arises for structure outside of HAL and SDK folder, **OpenTV One team** must be consulted.

Build tree organization

Build tree is divided into following folder structure:

-  build

- `chal`
- `external_libs`
- `sdk`
- `test`

Folder `build` contains built binaries and libraries. Also it contains configuration code and makefiles necessary for building. Those files are not meant to be edited by the HAL development team. Structure is as following

- `bin` - built binaries and libraries (HAL validator and DTV application)
- `inc` - configuration headers used by HAL/HAL validator
- `src` - configuration source code
- `make` - makefiles necessary for building
- `set` - configuration scripts used for building (refer to building section)
- `dist` - distributions of OpenTV One software stack (libraries and resources)

Folder `chal` contains code for Hardware Abstraction Layer modules. It is divided in tree components TKEL, TDAL and TBOX. Vendor is responsible of implementing these modules. Testing basic modules is done with HAL validator.

Folder `sdk` contains boards SDK package. It contains source of SDK, precompiled libraries and related tools for particular platform.

Folder `test` contains HAL validator code with all tests.

Development team for Open TV One may change structure and function of those files over time.

Distrubitions

Distribution is package delivered to a 3rd party along with the build tree in order to port HAL layer, pass CAS certification, troubleshoot issues in HAL or verify HAL using DTV application.

Folder `dist` under `build` folder contains one or more distributions. Folder structure is as following:

- `dist/`
 - `vendor_chip_manufacturer_model_application_...`
 - `binaries`
 - `libraries`
 - `resources`

Folder `binaries` contains precompiled Open TV One software ready for running on board. Each version may contain `debug` and `release` version.

Folder `libraries` contains Open TV One middleware libraries used for linking with 3rd party libraries (e.g CAS libraries) and HAL libraries.

Folder `resources` contains resources for delivered types of application. Sub-folders if exists, should match the ones presented in `libraries` folder. Resources are compressed assets that will be extracted to the storage medium at the boot time.

Vendor SDK

Vendor SDK folder is located under root folder of build tree. It is named using nomenclature





```
<chip_vendor>_<chip_model>_<board_manufacturer>_<board_model>_<target_os>
```

where :

Nomenclature name	Configuration variable	Description
<chip_vendor>	PLATFORM_CHIP_VENDOR	Platform chip vendor
<chip_model>	PLATFORM_CHIP_MODEL	Chip model
<board_manufacturer>	PLATFORM_BOARD_VENDOR	Board manufacturer
<board_model>	PLATFORM_BOARD_MODEL	Board model
<target_os>	PRODUCT_OS	Target OS

Folder name may not contain all the data from above table, meaning that name could be generalized if possible (by reducing names from right hand side to left). Same rule is used for HAL naming. Variables from table above are defined in configuration file (see Building software section).

Folder for vendor SDK has following structure:



-  `inc` - include files available to the HAL source code.
-  `src` - vendors SDK source, used to build sdk library
-  `libs` - prebuilt `libsdk.a` libraries in debug or release version
-  `tools` - tools used for generating image, signing and so

For building SDK there is main `Makefile` . Targets that need to be defined are

- `all` - used to build vendor SDK
- `clean` - used to clean vendor SDK
- `image` - creates compressed image (executable + resources)
- `sign` - signs compressed image (to be recognized by downloader)

This makefile is used to build SDK without setting environment variables (see Building software section), in which case it should be called directly from folder where makefile resides. Result of vendor SDK build is `libsdk.a` library. It should be located under `libs/` folder of vendor sdk. Also as a result of build process `inc/` folder must contain necessary headers for building HAL.

There are 2 additional makefiles used for building Open TV One software:

-  `tools.mak`
-  `drivers.mak`

First file `tools.mak` defines toolchain for building and shell tools on host operating system that will be used during build process.

Second file `drivers.mak` contains switches that are used while building DTV software. It is divided in two sets for variables, OS specific variables and driver specific variables.

All variables described bellow contain **space** separated arguments, so they may not be seen as single option variables.

Variable	Description
OS_C_FLAGS	Target OS specific C flags
OS_CXX_FLAGS	Target OS specific C++ flags
OS_AS_FLAGS	Target assembler flags
OS_LD_FLAGS	Target OS linker flags
OS_INCLUDE_PATH	Target OS include paths
OS_LIBS_PATH	Target OS libraries
OS_LIBS	Target OS libraries for linking

Variable	Description
SDK_C_FLAGS	Driver specific C flags
SDK_CXX_FLAGS	Driver specific C++ flags
SDK_LD_FLAGS	Driver linker flags
SDK_AS_FLAGS	Driver assembler flags
SDK_INCLUDE_PATH	Driver include paths
SDK_LIBS_PATH	Driver libraries
SDK_LIBS	Driver libraries for linking

SDK flags are used only when building HAL, exception is `SDK_LD_FLAGS` that is needed for linking stage. Target OS flags are used when building Open TV One software (including HAL, HAL validator, etc).

Under `drivers.mak` one additional global variable `PRODUCT_PREBUILT_SDK` is defined that tells build system whether to build SDK from source or use prebuilt library `libsdk.a`.

Flags defined in above tables determine what will be available at compile and linking stage to the HAL (and HAL validator) source code. Build system does NOT look in any predefined places for files. It uses 3 makefiles in root folder of vendor SDK, and predefined flags in them.

HAL - Hardware abstraction layer

Hadrware abstraction layer consists of 3 components:

- TKEL - provides an OS independent kernel abstraction (example : OS20, OS21, eCos, Linux, ...)
- TBOX - used to add debug traces into the code and to get input from user for test applications.
- TDAL - driver abstract layer, covering access to hardware devices.

Refer to HAL documentation for more information about each of the modules.




















Each of the implementations comply to naming scheme:

```
<chip_vendor>_<chip_model>_<board_manufacturer>_<board_model>_<target_os>
```

where :

Nomenclature name	Configuration variable	Description
<chip_vendor>	PLATFORM_CHIP_VENDOR	Platform chip vendor
<chip_model>	PLATFORM_CHIP_MODEL	Chip model
<board_manufacturer>	PLATFORM_BOARD_VENDOR	Board manufacturer
<board_model>	PLATFORM_BOARD_MODEL	Board model
<target_os>	PRODUCT_OS	Target OS

Top level folder organization:

-  `tbody`
 -  `inc`
 -  `src`
 -  `specific_version`
 -  `lib`
 -  `specific_version`
 -  `debug`
 -  `release`
-  `tkel`
 -  `inc`
 -  `src`
 -  `specific_version`
 -  `lib`
 -  `specific_version`
 -  `debug`
 -  `release`
-  `tdal`
 -  `inc`
 -  `src`
 -  `specific_version`
 -  `lib`
 -  `specific_version`
 -  `debug`
 -  `release`

Additional global variable `PRODUCT_PREBUILT_HAL` is defined that tells build system whether to build HAL from source or use prebuilt libraries located under `lib` folder. Picking proper library is done using nomenclature stated at the beginning of the section and additionally libraries may contain arbitrary suffix defined by `PRODUCT_HAL_SUFFIX` in configuration makefile. Arbitrary suffix allows modules to be distinguished in case they differ in some minor feature or setting.

HAL Validator

HAL validator is located under `test/chalvalidator` folder. It contains two types of tests unitary and scenario tests. Unitary tests check only API of each of the functions, where scenario tests cover more functional testing.

Tests are grouped in 3 sections:

- TKEL tests
- TBOX tests
- TDAL tests

Each group contains unitary and scenario tests.

HAL validator does not cover CA tests or software update tests. Those tests are covered with CAS specific test suites and SSU tests using final DTV application.

HAL validator uses serial interface for communication, so it must be built with `TRACES=UART`. In case there is intention to build DTV application without TRACES from CHAL this switch should be omitted.

Building software

To start build run command from **project's root folder**:

```
make SETENV=<configuration_file> <target>
```

Alternative way is to set configuration variable first in host OS

1. Windows: `set SETENV=<configuration_file>`
2. Linux: `export SETENV=<configuration_file>`

and then run build with `make <target>`.

In case SETENV variable is wrongly set or not set at all, error will be printed on first run of build, stating that it can not find configuration file.

Targets:

To print help and see available configurations type `make help`.

For building software use following commands:

1. Build SDK libraries: `make SETENV=<configuration_file> sdk`
 - `TRACES=UART` will build SDK, with logging enabled, otherwise it will be without any log output from SDK.
2. Build HAL libraries: `make SETENV=<configuration_file> chal`
 - `TRACES=UART` will select enable logging in HAL, refer to `build/src/traces_cfg.c` for logging settings
 - `DEBUG=1` will build HAL with debug symbols, allowing it to be analyzed with specialized debugger.
3. Build HAL validator: `make SETENV=<configuration_file> chalvalidator TRACES=UART`
4. Building Open TV One software: `make SETENV=<configuration_file> comedia`
 - `DEBUG=1` will select debug version of prebuilt OpenTV One middleware library, otherwise it will be release version

In order to build HAL validator, SDK must be already built. In order to build Open TV One software both SDK and HAL libraries must be built.

To adjust toolchain and related command line tools refer to `tools.mak` for appropriate vendor's sdk. See section following section for more details.