Yocto Recipe - CloudServiceNew

hcloudnew.bb

```
./meta-ccos-avn/meta-ccos-avn/recipes-ccos-interface/hcloudnew/hcloudnew.bb

SUMMARY = "ccOS refactored hcloud lib"
DESCRIPTION = "ccOS refactored hcloud lib"
SECTION = "ccOS/interface"

inherit ccos-base-interface
inherit ccos-cmake

DEPENDS += "hcommon hutil cloudservicenew"
DEPENDS += "glibmm"

CCOS_REPO_NAME = "ccos.api.hcloudnew"

CCOS_VERSION = "2.4.0_6a5c5043deb3981b8e215210dce9378e3d068c2d"

CXXFLAGS_append = " -DLOGGINGCONTEXT"

PR = "r2"
```

hprovisioning.bb

```
./meta-ccos-avn/meta-ccos-avn/recipes-ccos-interface/hprovisioning/hprovisioning.bb

SUMMARY = "ccOS hprovisioning lib"
DESCRIPTION = "hprovisioning lib is to provisioning service"
SECTION = "ccOS/interface"

inherit ccos-base-interface
inherit ccos-cmake

DEPENDS += "hcommon hutil"
DEPENDS += "dbus glib-2.0 jsoncpp"

CCOS_REPO_NAME = "ccos.api.hprovisioning"
CCOS_VERSION = "2.4.0_a9880a5ca2070de73b0312cab43cbcc8aae841b9"

PR = "r2"
```

hnetwork.bb

```
meta-ccos-avn/meta-ccos-avn/recipes-ccos-interface/hnetwork/hnetwork.bb

SUMMARY = "This library(ccos::network) controls the network service group"
DESCRIPTION = "it supports to control network service group"
SECTION = "ccOS/interface"

inherit ccos-base-interface
inherit ccos-cmake
inherit ccos-gdbus-codegen-glibmm

DEPENDS += "hcommon hsystem hutil networkservice"
DEPENDS += "glib-2.0 glibmm"
DEPENDS += "gtest"

CCOS_REPO_NAME = "ccos.api.hnetwork"

CCOS_VERSION = "2.0.5_678b29f40f4b357cd7a1625d1ba6c5eefecc07d5"
PR = "r3"
```

cloudservicenew.bb

./meta-ccos-avn/recipes-ccos-service/cloudservicenew/cloudservicenew.bb

```
SUMMARY = "recipe for refactored ccOS cloudservice"
DESCRIPTION = "recipe for refactored ccOS cloudservice"
SECTION = "ccOS/service"

inherit ccos-base-service
inherit ccos-cmake
inherit ccos-gdbus-codegen-glibmm
inherit ccos-systemd

DEPENDS += "mosquitto"
DEPENDS += "info-cloud ccu-cloud"
DEPENDS += "systemd"
DEPENDS += "hcommon hutil"

DEPENDS += "huser hsysteminfo htelephony hlocation"
DEPENDS += "servicehub hsmcpp"
#DEPENDS += "htrace"

CCOS_REPO_NAME = "ccos.service.cloudservicenew"

CCOS_VERSION = "100_42d66310e3e633a181e569bab8246f369374ed22"

PR = "r2"
```

./recipes-ccos-service/cloudservicenew/cloudservicenew/cloudservicenew.service

```
[Unit]
Description=[ccos::service] %N
Before=ccos-service.target
After=ccos-init-service.target systemservice.service locationservice.service telephonyservice.service diagnosisservice.service
JoinsNamespaceOf=netns@ccs.service

[Service]
Type=notify
PrivateNetwork=yes
BindPaths=/ccos/data/services/networkservice/etc/netns/ns-ccs/resolv.conf:/etc/resolv.conf
ExecStart=/ccos/services/cloudservicenew/cloudservicenew
Restart=always
RestartSec=5
TimeoutSec=10

[Install]
WantedBy=ccos-service.target
```

servicehub.bb

./meta-ccos-avn/recipes-ccos-service/servicehub/servicehub.bb

```
SUMMARY = "recipe for refactored ccOS servicehub"
DESCRIPTION = "recipe for refactored ccOS servicehub"
SECTION = "ccOS/service"

inherit ccos-library
inherit ccos-cmake

DEPENDS += "nlohmann-json openssl zlib curl"

CCOS_REPO_NAME = "ccos.service.cloudservice-libservicehub"

CCOS_VERSION = "100_73b70f2aa723c73a3b6e800555cc8dbed0ef9fe6"

PR = "r2"
```

```
./meta-ccos-avn/recipes-support/curl/curl_%.bbappend

EXTENDPRAUTO_append = "ccosref1"

PACKAGECONFIG_remove = "gnutls"

PACKAGECONFIG_append = " ssl"
```

libservicehub - CMakeLists.txt

```
cmake_minimum_required(VERSION 3.0)
```

```
project(servicehub VERSION 3.0.0)
set(LIB_NAME servicehub)

# set compile options
set(CXX_STANDARD_REQUIRED YES)
set(CXX_STANDARD 11)
set(POSITION_INDEPENDENT_CODE ON)
add_compile_options(-Wall -Wextra -pedantic)
add_compile_options(-Wno-unused-function -Wno-unused-parameter) # temporary off during initial implementation
add_link_options(-Wl,--no-undefined)
add_link_options(-Wl,--no-allow-shlib-undefined)

if(NOT CCOS_LIB_VERSION)
if(DEFINED ENV{CCOS_LIB_VERSION})
set(CCOS_LIB_VERSION "$ENV{CCOS_LIB_VERSION}")
set(PROJECT_VERSION "$ENV{CCOS_LIB_VERSION}")
else()
set(CCOS_LIB_VERSION "3.0.0")
set(PROJECT_VERSION "3.0.0")
# message(FATAL_ERROR "add build option : ₩n -DCCOS_LIB_VERSION=X.X.X or export CCOS_LIB_VERSION=X.X.X")
endif()
endif()
string(REPLACE "." ";" VERSION_LIST ${CCOS_LIB_VERSION})
list(GET VERSION_LIST 0 LIB_MAJOR_VERSION)
list(GET VERSION_LIST 1 LIB_MINOR_VERSION)
set(LIB_VERSION "${CCOS_LIB_VERSION}")
set(PROJECT_VERSION "${CCOS_LIB_VERSION}")

if(NOT $ENV{CCOS_LIB_DIR} STREQUAL "")
set(CCOS_LIB_DIR "./.")
endif()
if(NOT $ENV{CCOS_INC_DIR} STREQUAL "")
set(CCOS_INC_DIR "./.")
endif()
set(CCOS_LIB_DIR "./.")
set(CCOS_INC_DIR "./.")

# set logging option
option(LOGGING_MODE "option: configure log level")
if(NOT DEFINED LOGGING_MODE)
set(LOGGING_MODE "DEFAULT")
elseif(LOGGING_MODE STREQUAL NICE)
add_definitions(-DLOGGING_MODE_NICE)
elseif(LOGGING_MODE STREQUAL QUIET)
add_definitions(-DLOGGING_MODE_QUIET)
endif()
message("Config: logging mode ${LOGGING_MODE}")

# dependency packages
find_package(PkgConfig REQUIRED)
find_package(nlohmann_json REQUIRED)
pkg_check_modules(CURL REQUIRED libcurl)
pkg_check_modules(SSL REQUIRED libssl)
pkg_check_modules(CRYPTO REQUIRED libcrypto)

# ASIO
add_definitions(-DASIO_STANDALONE)

find_package(websocketpp REQUIRED)
find_package(OpenSSL REQUIRED)
pkg_check_modules(ZLIB REQUIRED zlib)

# Include custom base64 from apr-util
set(APR_UTIL_BASE64_SRC
${CMAKE_CURRENT_SOURCE_DIR}/3rdparty/apr-util/apr_base64.c
)

# build library
set(SRC_COMMON_FILES
${CMAKE_CURRENT_SOURCE_DIR}/src/ServiceHub.cpp
)
set(SRC_BASE64_FILES
${CMAKE_CURRENT_SOURCE_DIR}/src/base64/Base64.cpp
)
set(SRC_CERTIFICATE_FILES
${CMAKE_CURRENT_SOURCE_DIR}/src/certificate/CertificatesStorage.cpp
)
set(SRC_CONFIGURATION_FILES
${CMAKE_CURRENT_SOURCE_DIR}/src/configuration/Configuration.cpp
)
set(SRC_HTTP_FILES
${CMAKE_CURRENT_SOURCE_DIR}/src/http/HttpClient.cpp
${CMAKE_CURRENT_SOURCE_DIR}/src/http/HttpConnectionSettings.cpp
${CMAKE_CURRENT_SOURCE_DIR}/src/http/HttpGetRequest.cpp
```

```
${CMAKE_CURRENT_SOURCE_DIR}/src/http/HttpMessageBase.cpp
${CMAKE_CURRENT_SOURCE_DIR}/src/http/HttpPostRequest.cpp
${CMAKE_CURRENT_SOURCE_DIR}/src/http/HttpHeadRequest.cpp
${CMAKE_CURRENT_SOURCE_DIR}/src/http/HttpDeleteRequest.cpp
${CMAKE_CURRENT_SOURCE_DIR}/src/http/HttpPatchRequest.cpp
${CMAKE_CURRENT_SOURCE_DIR}/src/http/HttpPutRequest.cpp
${CMAKE_CURRENT_SOURCE_DIR}/src/http/HttpRequest.cpp
${CMAKE_CURRENT_SOURCE_DIR}/src/http/HttpResponse.cpp
${CMAKE_CURRENT_SOURCE_DIR}/src/http/private/CurlHelper.cpp
${CMAKE_CURRENT_SOURCE_DIR}/src/http/private/HttpClientImpl.cpp
${CMAKE_CURRENT_SOURCE_DIR}/src/http/private/AtomicHttpRequestsQueue.cpp
${CMAKE_CURRENT_SOURCE_DIR}/src/http/private/AsyncHttpPerformer.cpp
${CMAKE_CURRENT_SOURCE_DIR}/src/http/private/HttpPerformanceHelper.cpp
)
set(SRC_SMS_FILES
${CMAKE_CURRENT_SOURCE_DIR}/src/sms/SmsHttpCodec.cpp
${CMAKE_CURRENT_SOURCE_DIR}/src/sms/SmsManager.cpp
)
set(SRC_LOGGING_FILES
${CMAKE_CURRENT_SOURCE_DIR}/src/logging/LogDaily.cpp
${CMAKE_CURRENT_SOURCE_DIR}/src/logging/LogGeneric.cpp
${CMAKE_CURRENT_SOURCE_DIR}/src/logging/LogSequential.cpp
)
SET(SRC_THREADS_FILES
${CMAKE_CURRENT_SOURCE_DIR}/src/threads/Thread.cpp
)
set(SRC_UTILS_FILES
${CMAKE_CURRENT_SOURCE_DIR}/src/utils/StringUtil.cpp
${CMAKE_CURRENT_SOURCE_DIR}/src/utils/TimeUtil.cpp
${CMAKE_CURRENT_SOURCE_DIR}/src/utils/UriCodec.cpp
${CMAKE_CURRENT_SOURCE_DIR}/src/utils/EncryptionAesUtil.cpp
${CMAKE_CURRENT_SOURCE_DIR}/src/utils/CompressionDeflateUtil.cpp
)
set(SRC_WEBSOCKETS_FILES
${CMAKE_CURRENT_SOURCE_DIR}/src/websockets/WebSocketClient.cpp
${CMAKE_CURRENT_SOURCE_DIR}/src/websockets/private/WebSocketClientImpl.cpp
${CMAKE_CURRENT_SOURCE_DIR}/src/websockets/private/WebsocketppClient.cpp
${CMAKE_CURRENT_SOURCE_DIR}/src/websockets/private/WebsocketppConnection.cpp
)

add_library(${LIB_NAME} SHARED
${SRC_COMMON_FILES}
${SRC_BASE64_FILES}
${SRC_CERTIFICATE_FILES}
${SRC_CONFIGURATION_FILES}
${SRC_GZIP_FILES}
${SRC_HTTP_FILES}
${SRC_LOGGING_FILES}
${SRC_SMS_FILES}
${SRC_THREADS_FILES}
${SRC_UTILS_FILES}
${SRC_WEBSOCKETS_FILES}

${APR_UTIL_BASE64_SRC}
)

target_include_directories(${LIB_NAME} PUBLIC
${CMAKE_CURRENT_SOURCE_DIR}/include/servicehub
${CMAKE_CURRENT_SOURCE_DIR}/3rdparty
${CMAKE_CURRENT_SOURCE_DIR}/src
${ZLIB_INCLUDE_DIRS}
)

target_link_libraries(${LIB_NAME}
pthread
${SSL_LDFLAGS}
${CRYPTO_LDFLAGS}
${ZLIB_LDFLAGS}
${CURL_LDFLAGS}
websocketpp::websocketpp
rt
)

set_target_properties(${LIB_NAME} PROPERTIES VERSION ${LIB_VERSION} SOVERSION ${LIB_MAJOR_VERSION})

set(PROJECT_VERSION "${LIB_VERSION}")
set(PROJECT_DESCRIPTION "Helper library for cloud features")
configure_file(servicehub.pc.cmakein servicehub.pc @ONLY)

install(TARGETS ${LIB_NAME} LIBRARY DESTINATION ${CCOS_LIB_DIR})
install(DIRECTORY ${CMAKE_CURRENT_SOURCE_DIR}/include/ DESTINATION ${CCOS_INC_DIR})
install(FILES ${CMAKE_BINARY_DIR}/servicehub.pc DESTINATION ${CCOS_LIB_DIR}/pkgconfig)

# build test app
```

```
option(BUILD_WITH_TEST_APP "option: configure log level")
if(BUILD_WITH_TEST_APP OR (NOT $ENV{BUILD_WITH_TEST_APP} STREQUAL ""))
message ("Config: ENABLE tests")
add_subdirectory(${CMAKE_CURRENT_SOURCE_DIR}/test)
else ()
add_subdirectory(${CMAKE_CURRENT_SOURCE_DIR}/test/tools/sslkeylog)
endif()
```

ccos-cmake.bbclass

```
inherit cmake

#inherit ccos-sanitize

EXTRA_OECMAKE_append += " ₩
    -DCCOS_LIB_VERSION=${PV} ₩
    -DCCOS_COMPONENT_NAME=${BPN} ₩
    -DCCOS_MACHINE_NAME=${MACHINE} ₩
    -DCCOS_TARGET_ARCH=${TARGET_ARCH} ₩
    -DCCOS_BUILD_TYPE=${BUILD_TYPE} ₩
    -DCCOS_DISTRO=${DISTRO} ₩
"
```

ccos-library.bbclass

```
#
# This bbclass is for ccOS libraries.
#

inherit ccos-base
#inherit ccos-repo
#inherit ccos-version

FILES_${PN}_append += " ₩
    ${CCOS_LIB_DIR}/*${SOLIBS} ₩
"

FILES_${PN}-dev_append += " ₩
    ${CCOS_LIB_DIR}/*${SOLIBSDEV} ₩
    ${CCOS_LIB_DIR}/pkgconfig ₩
    ${CCOS_INC_DIR} ₩
"

FILES_${PN}-staticdev_append += " ₩
    ${CCOS_LIB_DIR}/*.a ₩
"

# Pass $CCOS_LIB_DIR to CMakeLists.txt if this app uses cmake
EXTRA_OECMAKE_append += " ₩
    -DCCOS_LIB_DIR=${CCOS_LIB_DIR} ₩
    -DCCOS_INC_DIR=${CCOS_INC_DIR} ₩
"

# Pass $CCOS_LIB_DIR to $BPN.pro if this app uses qmake
EXTRA_QMAKEVARS_PRE += "₩
    CCOS_LIB_DIR=${CCOS_LIB_DIR} ₩
    CCOS_INC_DIR=${CCOS_INC_DIR} ₩
"

SYSROOT_DIRS += " ₩
    ${CCOS_LIB_DIR} ₩
    ${CCOS_INC_DIR} ₩
"
```

ccos-base.bbclass

```
# All ccOS components have "CLOSED" license.
LICENSE ?= "CLOSED"

# All ccOS components are installed under $CCOS_ROOT_DIR
CCOS_ROOT_DIR = "/ccos"

# Applications
CCOS_APP_DIR = "${CCOS_ROOT_DIR}/apps"
CCOS_APP_HMI_DIR = "${CCOS_APP_DIR}/hmi"
CCOS_APP_MANAGER_DIR = "${CCOS_APP_DIR}/manager"
CCOS_APP_INTERFACE_DIR = "${CCOS_APP_DIR}/interface/${BPN}"
CCOS_APP_SHARE_DIR = "${CCOS_APP_DIR}/share"
CCOS_APP_IMAGES_DIR = "${CCOS_APP_SHARE_DIR}/images"
CCOS_APP_IMAGES2_DIR = "${CCOS_APP_SHARE_DIR}/images-2"
CCOS_APP_PLUGINS_DIR = "${CCOS_APP_SHARE_DIR}/plugins"
CCOS_APP_QML_DIR = "${CCOS_APP_SHARE_DIR}/qml"
CCOS_APP_WEB_DIR = "${CCOS_APP_DIR}/web"

# Interfaces
CCOS_INTERFACE_DIR = "${CCOS_ROOT_DIR}/interfaces/${BPN}"

# Services
CCOS_SERVICE_ROOT_DIR = "${CCOS_ROOT_DIR}/services"
CCOS_SERVICE_DIR = "${CCOS_SERVICE_ROOT_DIR}/${BPN}"
CCOS_SERVICE_SHARE_DIR = "${CCOS_SERVICE_ROOT_DIR}/share"

# Cloud Interface
CCOS_SERVICE_CLOUD_CONF_DIR = "${CCOS_SERVICE_ROOT_DIR}/cloudservicenew/etc/metadata"

# HAL and Daemons
CCOS_HAL_DIR = "${CCOS_ROOT_DIR}/hal/${BPN}"
CCOS_DAEMON_DIR = "${CCOS_ROOT_DIR}/daemons/${BPN}"

# Libraries and Headers
CCOS_LIB_DIR = "${CCOS_ROOT_DIR}/lib"
CCOS_INC_DIR = "${CCOS_ROOT_DIR}/include/${BPN}"

CCOS_PKG_CONFIG_DIR = "${STAGING_DIR_TARGET}${CCOS_LIB_DIR}/pkgconfig:${STAGING_DATADIR}/pkgconfig"

# ccOS uses $WORKDIR/git as a source directory by default.
S = "${WORKDIR}/git"

export PKG_CONFIG_PATH = "$PKG_CONFIG_DIR:${CCOS_PKG_CONFIG_DIR}"
inherit pkgconfig

# If you want to pass some environment variables to devshell,
# you need to define it as below. As of now, only CCOS_LIB_VERSION
# is passed to devshell.
python do_devshell_prepend() {
    os.environ['CCOS_LIB_VERSION'] = d.getVar("PV", True)
}

# ccOS build QA applies to all ccOS components by default.
#inherit ccos-insane

#inherit ccos-ofbuild
```

ccos-gdbus-codegen-glibmm.bbclass

```
inherit ccos-base

DEPENDS += "glibmm gdbus-codegen-glibmm-native"

FILES_${PN}-dev += " ₩
    ${CCOS_INC_DIR} ₩
    ${CCOS_LIB_DIR}/pkgconfig ₩
"

FILES_${PN}-staticdev += "₩
    ${CCOS_LIB_DIR}/*.a ₩
"
```

gdbus-codegen-glibmm_3.0.0.bb

```
require gdbus-codegen-glibmm.inc

SRC_URI = "git://github.com/Pelagicore/gdbus-codegen-glibmm.git;protocol=https"
SRCREV = "dcb1d20d6b5253a3e02a711131397e5725be8e89"

do_install_append() {
    cd ${D}${bindir}
    ln -sf ${BPN}${PV_MAJOR} ${BPN}
}
```

gdbus-codegen-glibmm.inc

```
SUMMARY = "This is a code generator for generating D-Bus stubs and proxies from XML introspection files"
DESCRIPTION = "This is a code generator for generating D-Bus stubs and proxies ₩
from XML introspection files. The generated stubs and proxies are implemented ₩
using glibmm and giomm. This generator is based on the gdbus-codegen code ₩
generator which ships with glib."
HOMEPAGE = "https://github.com/Pelagicore/gdbus-codegen-glibmm"
LICENSE = "LGPLv2.1"
LIC_FILES_CHKSUM = "file://LICENSE;md5=4fbd65380cdd255951079008b364516c"

DEPENDS = "python3-jinja2-native python3-markupsafe-native"

S = "${WORKDIR}/git/"

inherit setuptools3

PV_MAJOR = "${@d.getVar('PV', True).split('.')[0]}"

do_install_append_class-native() {
    sed -i '1 c₩#!/usr/bin/env nativepython3' ${D}${bindir}/${BPN}${PV_MAJOR}
}

# Dependencies for building this software.
RDEPENDS_${PN} = "python3 python3-setuptools python3-jinja2"

BBCLASSEXTEND = "native nativesdk"

PACKAGE =+ "${PN}"
```

```
root@p2382t186:~/ivis/cloudservice# cd ../servicehub/
root@p2382t186:~/ivis/servicehub# ls
connection_test.conf   libglibmm-2.4.so                        libservicehub.so.2
connection_test.conf~  libglibmm-2.4.so.1                      libservicehub.so.2.0.0
http_client.log        libglibmm-2.4.so.1.3.0                  libsigc-2.0.so
libcurl.so             libglibmm_generate_extra_defs-2.4.so    libsigc-2.0.so.0
libcurl.so.4           libglibmm_generate_extra_defs-2.4.so.1  libsigc-2.0.so.0.0.0
libcurl.so.4.7.0       libglibmm_generate_extra_defs-2.4.so.1.3.0  servicehubTestConnection
libgiomm-2.4.so        libmosquitto.so.1                       testLibServiceHub
libgiomm-2.4.so.1      libmosquittopp.so.1
libgiomm-2.4.so.1.3.0  libservicehub.so
root@p2382t186:~/ivis/servicehub#
```