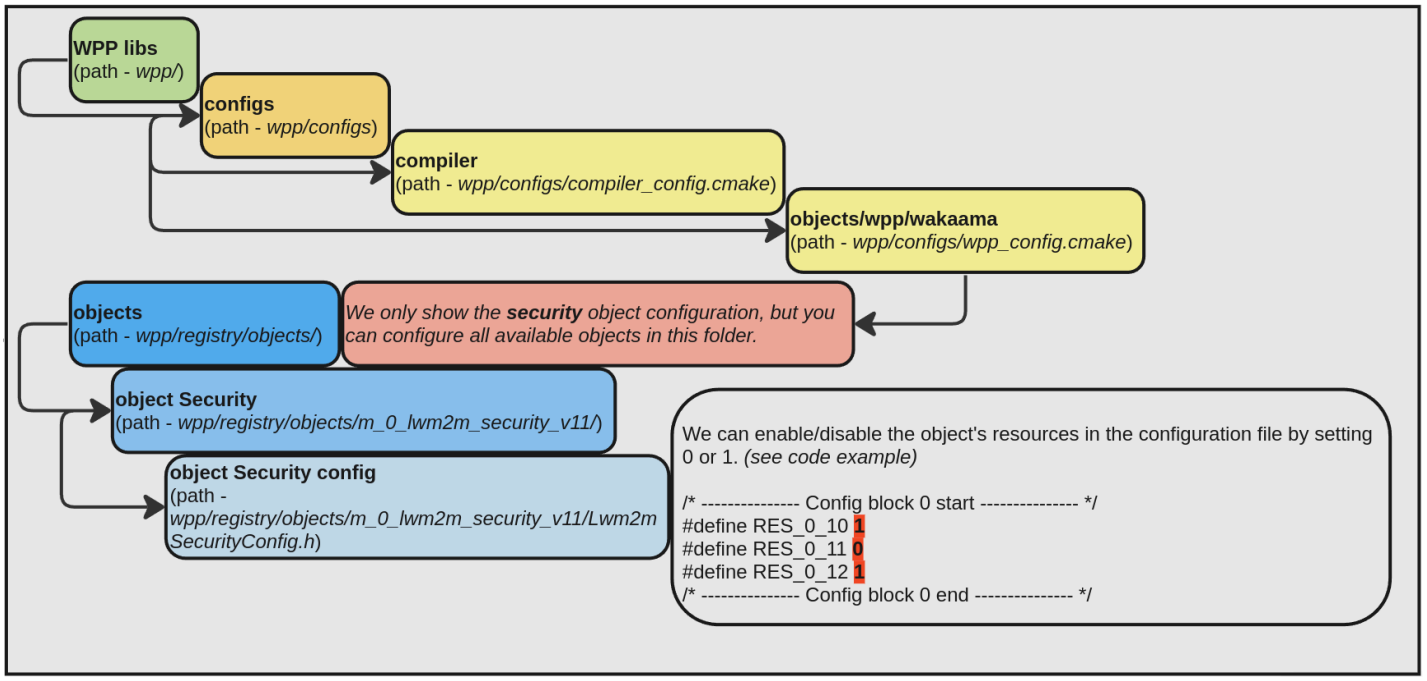
Wpp_Configure

Definitions and configurations

We have 4 configuration types. And all of these configs (defines) developer can redefine/modify or implement to project by another method. Be free and careful.
Important, if you change configs in wpp/configs - all changes are applied to another part of the repository.



Compiler configuration

`wpp/configs/compiler_config.cmake`

Build options

WPP_BUILD_WITH_EXCEPTIONS - enable support of Exceptions (default: **OFF**)
WPP_BUILD_WITH_RTTI - enable support of RTTI (default: **OFF**)
WPP_BUILD_FOR_64_BIT - build for 64-bit system or 32-bit (default: **ON**)
CMAKE_POSITION_INDEPENDENT_CODE - whether to create a position-independent target (default: **ON**)

Compiler options

Waggregate-return - warn if any functions that return structures or unions are defined or called.
Wall - this enables all the warnings about constructions that some users consider questionable, and that are easy to avoid, even in conjunction with macros.
Wcast-align - warn whenever a pointer is cast such that the required alignment of the target is increased.
Wextra - this enables some extra warning flags not enabled by -Wall.

Wfloat-equal - warn if floating-point values are used in equality comparisons.
Wpointer-arith - warn about anything that depends on the “size of” a function type or of void.
Wshadow - warn whenever a local variable or type declaration shadows another variable, parameter, type, class member (in C++), or instance variable (in Objective-C) or whenever a built-in function is shadowed.
Wswitch-default - warn whenever a switch statement does not have a default case.
Wwrite-strings - these warnings help you find at compile time code that you can try to write into a string constant, but only if you have been very careful about using const in declarations and prototypes.
Wno-unused-parameter - unused parameters are common in this ifdef-littered code-base but of no danger.
Wno-uninitialized - too many false positives.
Wno-gnu-zero-variadic-macro-arguments - allow usage ##__VA_ARGS__ in macros.
pedantic - issue all the warnings demanded by strict ISO C and ISO C++; diagnose all programs that use forbidden extensions, and some other programs that do not follow ISO C and ISO C++.
Werror - turn (most) warnings into errors.
Wno-error=cast-align - disabled because of existing, non-trivially fixable code.

Object configuration

`wpp/configs/wpp_config.cmake`

Mandatory objects config

OBJ_M_3_DEVICE - include mandatory Device object in the build (default: **ON**)
OBJ_M_1_LWM2M_SERVER - include mandatory Lwm2mServer object in the build (default: **ON**)
OBJ_M_0_LWM2M_SECURITY - include mandatory Lwm2mSecurity object in the build (default: **ON**)

Optional objects config

OBJ_O_4_CONNECTIVITY_MONITORING - include optional ConnectivityMonitoring object in the build (default: **ON**)
OBJ_O_2_LWM2M_ACCESS_CONTROL - include optional Lwm2mAccessControl object in the build (default: **ON**)
OBJ_O_5_FIRMWARE_UPDATE - include optional FirmwareUpdate object in the build (default: **ON**)

Wakaama configuration

`wpp/configs/wpp_config.cmake`

LWM2M_CLIENT_MODE - Wakaama should be always in the client mode
LWM2M_BOOTSTRAP - enable LWM2M Bootstrap support in a LWM2M Client (default: **OFF**)
LWM2M_SUPPORT_SENML_JSON - enable SenML JSON payload support (default: **OFF**)
LWM2M_SUPPORT_JSON - enable JSON payload support (default: **OFF**)
LWM2M_SUPPORT_TLV - enable TLV payload support (default: **ON**)
LWM2M_SUPPORT_CBOR - enable CBOR payload support (default: **OFF**)
LWM2M_SUPPORT_SENML_CBOR - enable SenML CBOR payload support (default: **OFF**)
LWM2M_OLD_CONTENT_FORMAT_SUPPORT - support the deprecated content format values for TLV and JSON (default: **OFF**)
LWM2M_BS_PREFERRED_CONTENT_TYPE - to set preferred content type for bootstrap server (default: **110**)
LWM2M_REG_PREFERRED_CONTENT_TYPE - to set preferred content type for registration (default: **110**)
LWM2M_VERSION_1_0 - support only version 1.0 (default: **OFF**)
LWM2M_RAW_BLOCK1_REQUESTS - ror low memory client devices where it is not possible to keep a large post or put request in memory to be parsed. Control over such operations is provided entirely to the user. At the moment, there are certain restrictions regarding the use of this mode, only two operations are supported BLOCK_EXECUTE without restrictions, and BLOCK_WRITE with the following restrictions: recording only one SINGLE resource, recording is possible in the following formats: TEXT, OPAQUE, TLV. (default: **OFF**)
LWM2M_WITH_LOGS - enable logs for wakaama core (default: **OFF**)
LWM2M_COAP_DEFAULT_BLOCK_SIZE - CoAP block size used by CoAP layer when performing block-wise transfers. Possible values: 16, 32, 64, 128, 256, 512 and 1024 (default: **1024**)

Define your own endian if the endian is different from the platform default.
set(WPP_DEFINITIONS ${WPP_DEFINITIONS} LWM2M_BIG_ENDIAN) - big-endian format
set(WPP_DEFINITIONS ${WPP_DEFINITIONS} LWM2M_LITTLE_ENDIAN) - little-endian format

WPP configuration

`wpp/configs/wpp_config.cmake`

WPP_ENABLE_LOGS - enable logs for WakaamaPlus (default: **ON**)
WPP_LOGS_LEVEL - set logs detalization for WPP_ENABLE_LOGS ON (default: **0**)

For setup server, ports, security, and bootstrap server make changes in the file
`./2305-WakaamaPlus/examples/objects.cpp`

In the function **serverInit** you can set the following:
SHORT_SERVER_ID_0 (example: 123)
BINDING_7 (example: WPP_BINDING_UDP / WPP_BINDING_TCP / WPP_BINDING_SMS / WPP_BINDING_NON_IP)
LIFETIME_1 (example: 25)
NOTIFICATION_STORING_WHEN_DISABLED_OR_OFFLINE_6 (example: false/true)

objects.cpp 4 × WppBindings.h 2

2305-WakaamaPlus > examples > objects.cpp > securityInit(WppClient &)

```cpp
/* ------------- Methods to init objects ------------- */

void serverInit(WppClient &client) {
    client.registry().registerObj(Lwm2mServer::object(client));
    Instance *server = Lwm2mServer::createInst(client);

    server->set<INT_T>(Lwm2mServer::SHORT_SERVER_ID_0, TEST_SERVER_SHORT_ID);
    server->set<STRING_T>(Lwm2mServer::BINDING_7, WPP_BINDING_UDP);
    server->set<TIME_T>(Lwm2mServer::LIFETIME_1, 25);
    server->set<BOOL_T>(Lwm2mServer::NOTIFICATION_STORING_WHEN_DISABLED_OR_OFFLINE_6, false);

    #if OBJ_O_2_LWM2M_ACCESS_CONTROL
    Lwm2mAccessControl::create(Lwm2mServer::object(client), Lwm2mAccessControl::ALL_OBJ_RIGHTS);
    Lwm2mAccessControl::create(*server, TEST_SERVER_SHORT_ID);
    #endif
}
```

In the function **securityInit** you can set the following:
if defined LWM2M_BOOTSTRAP (in ./2305-WakaamaPlus/wpp/configs/wpp_config.cmake)

- URL and port (example: "coap://friendly-tech.com:5680")
- BOOTSTRAP_SERVER_1 (example: false/true) (*security on/off)
- CLIENT_HOLD_OFF_TIME_11 (example: 10)

for security DTLS_WITH_PSK

- pskId (example: "SINAI_TEST_DEV_ID")
- URL and port (example: "coap://friendly-tech.com:5681")
- SECURITY_MODE_2 (example: LWM2M_SECURITY_MODE_PRE_SHARED_KEY)
- SECRET_KEY_5 (example: {0x00, 0x11, 0x22})

if not defined

- URL only (example: "coap://friendly-tech.com:")

for security DTLS_WITH_PSK

- port (example: 5684)
- pskId (example: "SINAI_TEST_DEV_ID")
- SECURITY_MODE_2 (example: LWM2M_SECURITY_MODE_PRE_SHARED_KEY)
- SECRET_KEY_5 (example: {0x00, 0x11, 0x22})

for security DTLS_WITH_RPK

- port (example: 5684)
- SECURITY_MODE_2 (example: LWM2M_SECURITY_MODE_RAW_PUBLIC_KEY)
- SECRET_KEY_5 (example: {0x00, 0x11, 0x22})

in default variant, without a bootstrap server of security

- port (example: 5683)
- SECURITY_MODE_2 (example: LWM2M_SECURITY_MODE_NONE)
- BOOTSTRAP_SERVER_1 (example: false)

objects.cpp 4 ● main.cpp 6

2305-WakaamaPlus > examples > objects.cpp > deviceInit(WppClient &)

```cpp
void securityInit(WppClient &client) {
    client.registry().registerObj(Lwm2mSecurity::object(client));
    wpp::Instance *security = Lwm2mSecurity::createInst(client);

    #ifdef LWM2M_BOOTSTRAP
        string url = "coap://friendly-tech.com:5680";
        security->set<BOOL_T>(Lwm2mSecurity::BOOTSTRAP_SERVER_1, true);
        security->set<INT_T>(Lwm2mSecurity::CLIENT_HOLD_OFF_TIME_11, 10);
        #if DTLS_WITH_PSK
        string pskId = "SINAI_TEST_DEV_ID";
        url = "coap://friendly-tech.com:5681";
        security->set<INT_T>(Lwm2mSecurity::SECURITY_MODE_2, LWM2M_SECURITY_MODE_PRE_SHARED_KEY);
        security->set(Lwm2mSecurity::PUBLIC_KEY_OR_IDENTITY_3, OPAQUE_T(pskId.begin(), pskId.end()));
        security->set(Lwm2mSecurity::SECRET_KEY_5, OPAQUE_T {0x00, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66});
        #endif
    #else
        string url = "coaps://friendly-tech.com:";
        #if DTLS_WITH_PSK
            url += "5684";
            string pskId = "SINAI_TEST_DEV_ID";
            security->set<INT_T>(Lwm2mSecurity::SECURITY_MODE_2, LWM2M_SECURITY_MODE_PRE_SHARED_KEY);
            security->set(Lwm2mSecurity::PUBLIC_KEY_OR_IDENTITY_3, OPAQUE_T(pskId.begin(), pskId.end()));
            security->set(Lwm2mSecurity::SECRET_KEY_5, OPAQUE_T {0x00, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66});
        #elif DTLS_WITH_RPK
            url += "5684";
            security->set<INT_T>(Lwm2mSecurity::SECURITY_MODE_2, LWM2M_SECURITY_MODE_RAW_PUBLIC_KEY);
            security->set(Lwm2mSecurity::PUBLIC_KEY_OR_IDENTITY_3, OPAQUE_T {0x04, 0xba, 0xda, 0x54, 0x75});
            security->set(Lwm2mSecurity::SECRET_KEY_5, OPAQUE_T {0x92, 0x04, 0x53, 0x22, 0xa5, 0xb3, 0x45});
        #else
            url += "5683";
            security->set<INT_T>(Lwm2mSecurity::SECURITY_MODE_2, LWM2M_SECURITY_MODE_NONE);
        #endif
        security->set<BOOL_T>(Lwm2mSecurity::BOOTSTRAP_SERVER_1, false);
    #endif

    security->set<STRING_T>(Lwm2mSecurity::LWM2M_SERVER_URI_0, url);
    security->set<INT_T>(Lwm2mSecurity::SHORT_SERVER_ID_10, TEST_SERVER_SHORT_ID);
}
```

In the function **deviceInit** you can set the following:

- SUPPORTED_BINDING_AND_MODES_16 (example: WPP_BINDING_UDP)
- MANUFACTURER_0 (example: "Wakaama Plus")
- MODEL_NUMBER_1 (example: "Lightweight M2M Client")
- SERIAL_NUMBER_2 (example: "0123456789")

objects.cpp 4 ● main.cpp 6

2305-WakaamaPlus > examples > objects.cpp > fwUpdaterInit(WppClient &, FirmwareUpdater &, FwUriDownloader &, FwAutoDownloader &)

```cpp
void deviceInit(WppClient &client) {
    client.registry().registerObj(Device::object(client));
    wpp::Instance *device = Device::createInst(client);

    device->set<EXECUTE_T>(Device::REBOOT_4, [](Instance& inst, ID_T resId, const OPAQUE_T& data) {
        cout << "Device: execute REBOOT_4" << endl;
        _rebootDevice = true;
        return true;
    });
    device->set<INT_T>(Device::ERROR_CODE_11, 0, Device::NO_ERROR);
    device->set<STRING_T>(Device::SUPPORTED_BINDING_AND_MODES_16, WPP_BINDING_UDP);
    device->set<STRING_T>(Device::MANUFACTURER_0, "Wakaama Plus");
    device->set<STRING_T>(Device::MODEL_NUMBER_1, "Lightweight M2M Client");
    device->set<STRING_T>(Device::SERIAL_NUMBER_2, "0123456789");

    #if OBJ_0_2_LWM2M_ACCESS_CONTROL
    Lwm2mAccessControl::create(Device::object(client), Lwm2mAccessControl::ALL_OBJ_RIGHTS);
    Lwm2mAccessControl::create(*device, TEST_SERVER_SHORT_ID);
    #endif
}
```