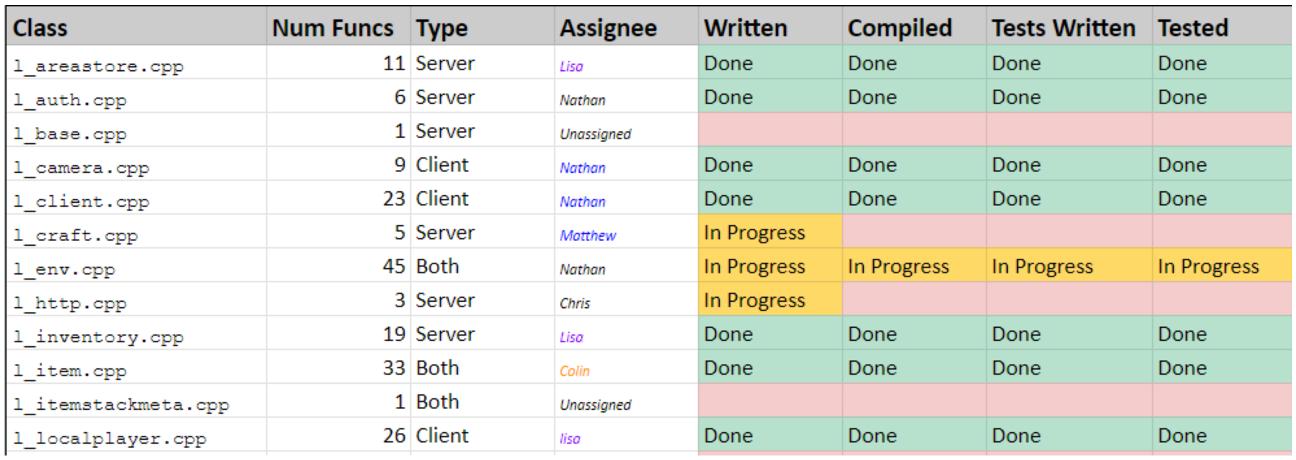
Creating a Native Mod - Standard Cacti Council Workflow

By Sean Ferguson

1. **Check the MineTest Lua API Class Assignments Spreadsheet for available functions:**
    a. https://docs.google.com/spreadsheets/d/1qx9d23yNtywQsov7tHmJWJOzh0SsFeWLeg9_c2n6DTc/edit#gid=703209104

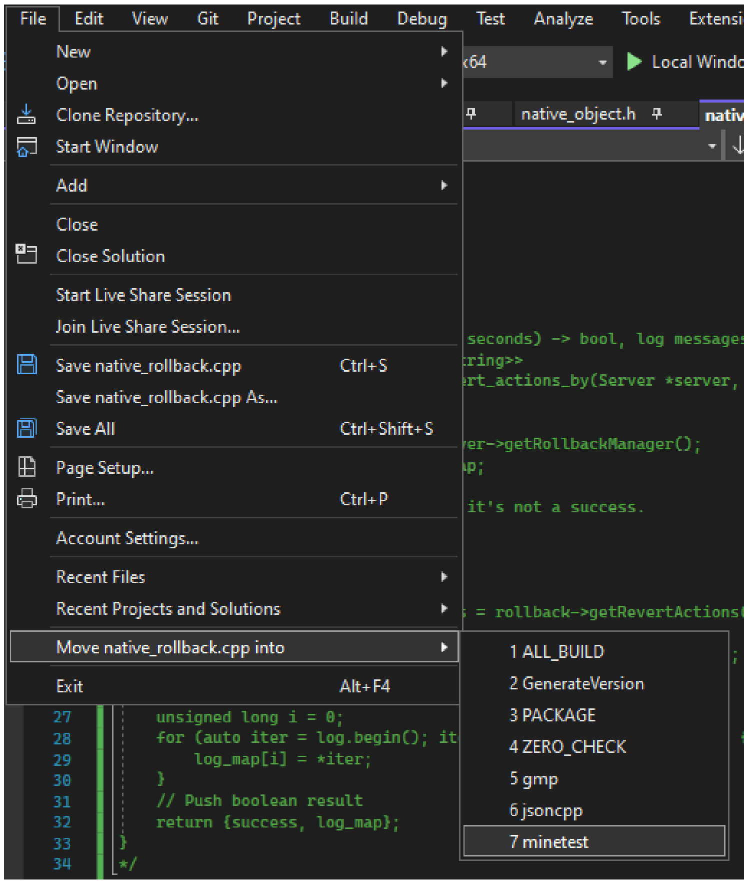
    b. Be sure to view the most recent semester's work (as of writing this document it is SPRING 2023)
    c. In Column "D", "Assignee" states the current person's working class, you may fill in your name into any square with "Unassigned"
    d. ***Fill in your progress with your own class:***
        i. Column E - Written
            1. Create the "l_native_<class>" function within the lua_api directory and the "n_<class>" within the native_api directory
        ii. Column F - Compiled
            1. Minetest runs with newly created functions
        iii. Column G - Tests Written
            1. Created Tests for lua, native and comparing results of the two
        iv. Column H - Tested
            1. All tests run and compare results are verified to be the same
    e. ***Class separation progress states:***
        i. N/A
            1. This part of the project has not started yet
        ii. In Progress
            1. The Assignee in Column D is currently working on this step
        iii. Done
            1. This part of the step has been completed



Class	Num Funcs	Type	Assignee	Written	Compiled	Tests Written	Tested
l_areastore.cpp	11	Server	Lisa	Done	Done	Done	Done
l_auth.cpp	6	Server	Nathan	Done	Done	Done	Done
l_base.cpp	1	Server	Unassigned				
l_camera.cpp	9	Client	Nathan	Done	Done	Done	Done
l_client.cpp	23	Client	Nathan	Done	Done	Done	Done
l_craft.cpp	5	Server	Matthew	In Progress			
l_env.cpp	45	Both	Nathan	In Progress	In Progress	In Progress	In Progress
l_http.cpp	3	Server	Chris	In Progress			
l_inventory.cpp	19	Server	Lisa	Done	Done	Done	Done
l_item.cpp	33	Both	Colin	Done	Done	Done	Done
l_itemstackmeta.cpp	1	Both	Unassigned				
l_localplayer.cpp	26	Client	lisa	Done	Done	Done	Done

Stage 1 - Written

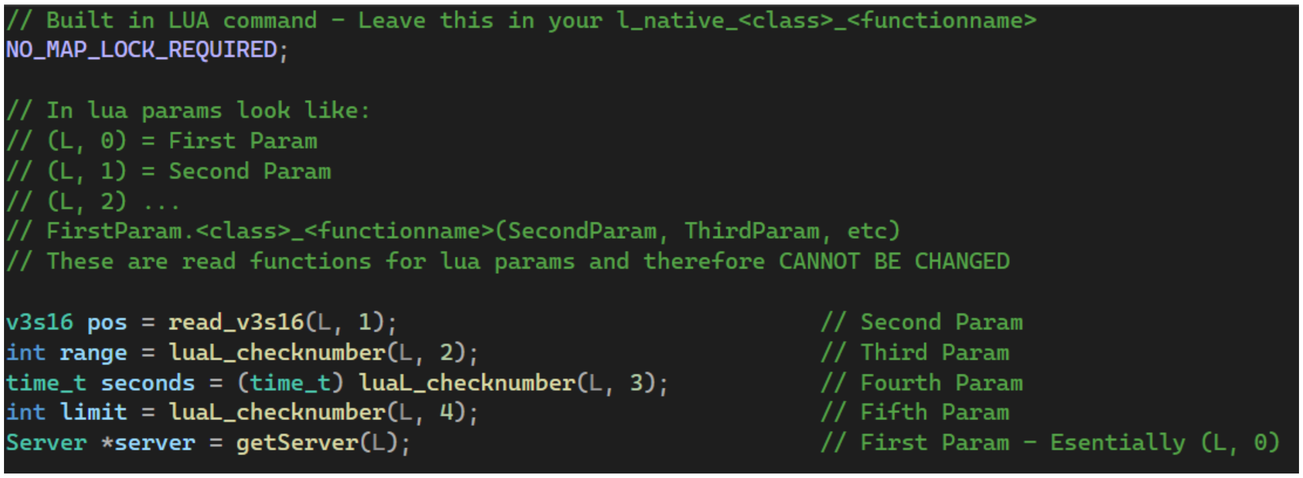
2. ***Create the native files for your class. We will reference this class as <class> from now on.***
    a. In "C:\Users\YourName\DIR\minetest\src\script\native_api" and create 2 files
        i. native_<class>.cpp
        ii. native_<class>.h
    b. Link these files to the project by selecting the file in Visual Studio and go to File > Move native_<class>.<fileextension> > minetest



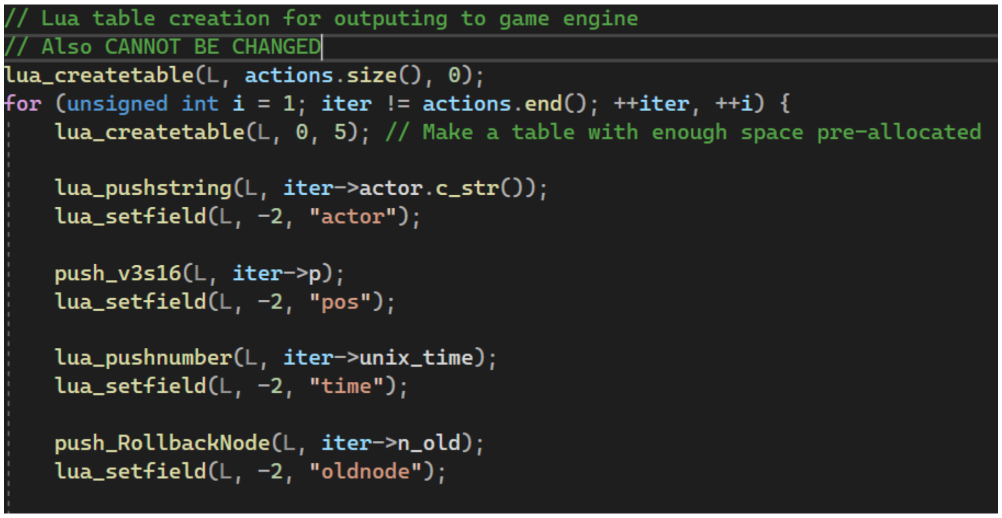
```
seconds) -> bool, log message
ring>>
rt_actions_by(Server *server,
er->getRollbackManager();
p;
it's not a success.
; = rollback->getRevertActions(
    unsigned long i = 0;
    for (auto iter = log.begin(); it
        log_map[i] = *iter;
    }
    // Push boolean result
    return {success, log_map};
}
*/
```

3. ***In this stage, you will be creating the separated lua and c++ functions within the class of your choice.***
    a. First, to locate your class go to "C:\Users\YourName\DIR\minetest\src\script\lua_api". Here all the classes are listed along with their header files. You can also use the Solution Explorer within Visual Studios but header files are hidden, therefore you may need to open them through a file explorer. The Solution Explorer directory is "mintest\Source Files\script\lua_api".
4. ***When viewing the class, you will sometimes see helper functions at the top of the file and functions with naming conventions such as:***
    a. `<returnType> <classObj>::l_<class>_<functionname>(lua_State *L)`
5. Example:
    a. `int ModApiRollback::l_rollback_get_node_actions(lua_State *L)`

6. ***You will create a function that contains all called functions dependent on the lua_State under the original function with the naming scheme:***
    a. `<returnType> <classObj>::l_`**`native`**`_<class>_<functionname>(lua_State *L)`
7. Example:
    a. `int ModApiRollback::l_`**`native`**`_rollback_get_node_actions(lua_State *L)`
8. Lua dependent C++ code that **CANNOT BE CHANGED (Needs to stay in your native seperation):**

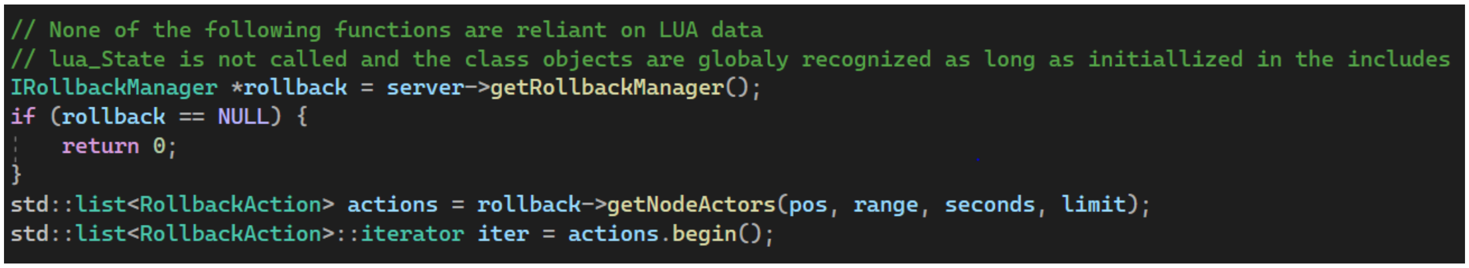


```
// Built in LUA command - Leave this in your l_native_<class>_<functionname>
NO_MAP_LOCK_REQUIRED;

// In lua params look like:
// (L, 0) = First Param
// (L, 1) = Second Param
// (L, 2) ...
// FirstParam.<class>_<functionname>(SecondParam, ThirdParam, etc)
// These are read functions for lua params and therefore CANNOT BE CHANGED

v3s16 pos = read_v3s16(L, 1);                           // Second Param
int range = luaL_checknumber(L, 2);                     // Third Param
time_t seconds = (time_t) luaL_checknumber(L, 3);       // Fourth Param
int limit = luaL_checknumber(L, 4);                     // Fifth Param
Server *server = getServer(L);                          // First Param - Esentially (L, 0)
```



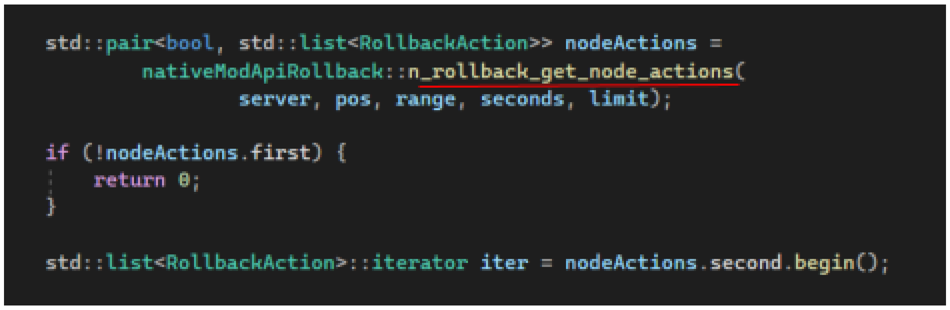
```
// Lua table creation for outputting to game engine
// Also CANNOT BE CHANGED
lua_createtable(L, actions.size(), 0);
for (unsigned int i = 1; iter != actions.end(); ++iter, ++i) {
    lua_createtable(L, 0, 5); // Make a table with enough space pre-allocated

    lua_pushstring(L, iter->actor.c_str());
    lua_setfield(L, -2, "actor");

    push_v3s16(L, iter->p);
    lua_setfield(L, -2, "pos");

    lua_pushnumber(L, iter->unix_time);
    lua_setfield(L, -2, "time");

    push_RollbackNode(L, iter->n_old);
    lua_setfield(L, -2, "oldnode");
```

9. Examples of code that **CAN BE CHANGED:**



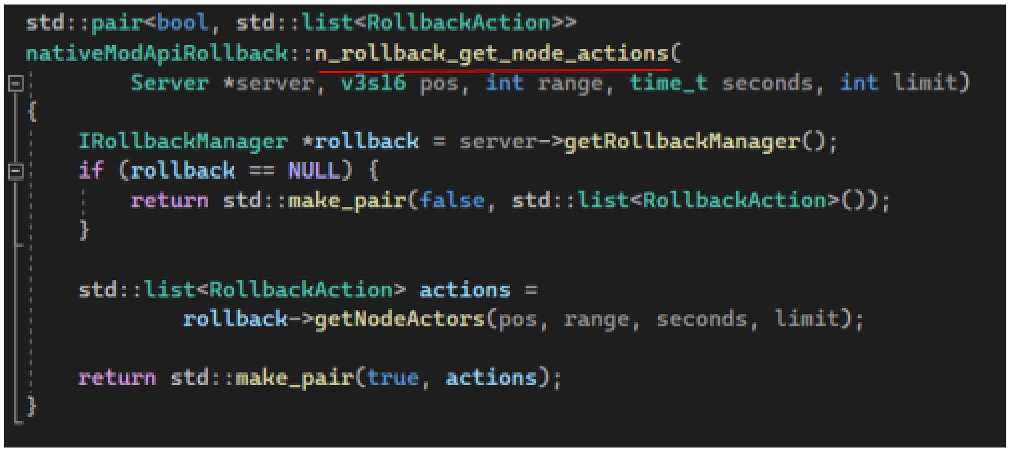
```
// None of the following functions are reliant on LUA data
// lua_State is not called and the class objects are globaly recognized as long as initiallized in the includes
IRollbackManager *rollback = server->getRollbackManager();
if (rollback == NULL) {
    return 0;
}
std::list<RollbackAction> actions = rollback->getNodeActors(pos, range, seconds, limit);
std::list<RollbackAction>::iterator iter = actions.begin();
```

10. ***Separate the C++ code from the Lua dependent code:***
    a. Lua file:



```
std::pair<bool, std::list<RollbackAction>> nodeActions =
        nativeModApiRollback::n_rollback_get_node_actions(
                server, pos, range, seconds, limit);

if (!nodeActions.first) {
    return 0;
}

std::list<RollbackAction>::iterator iter = nodeActions.second.begin();
```

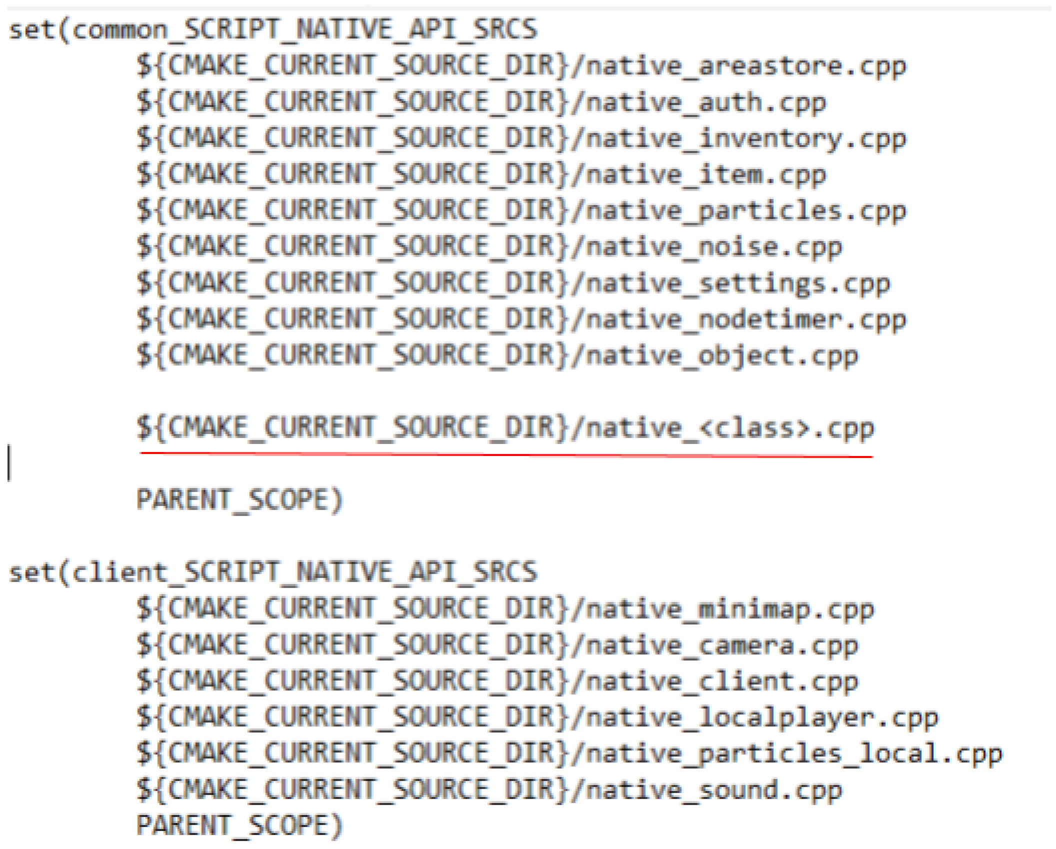
b. Native file:



```
std::pair<bool, std::list<RollbackAction>>
nativeModApiRollback::n_rollback_get_node_actions(
        Server *server, v3s16 pos, int range, time_t seconds, int limit)
{
    IRollbackManager *rollback = server->getRollbackManager();
    if (rollback == NULL) {
        return std::make_pair(false, std::list<RollbackAction>());
    }

    std::list<RollbackAction> actions =
            rollback->getNodeActors(pos, range, seconds, limit);

    return std::make_pair(true, actions);
}
```

11. Add your new class to the CMakeLists

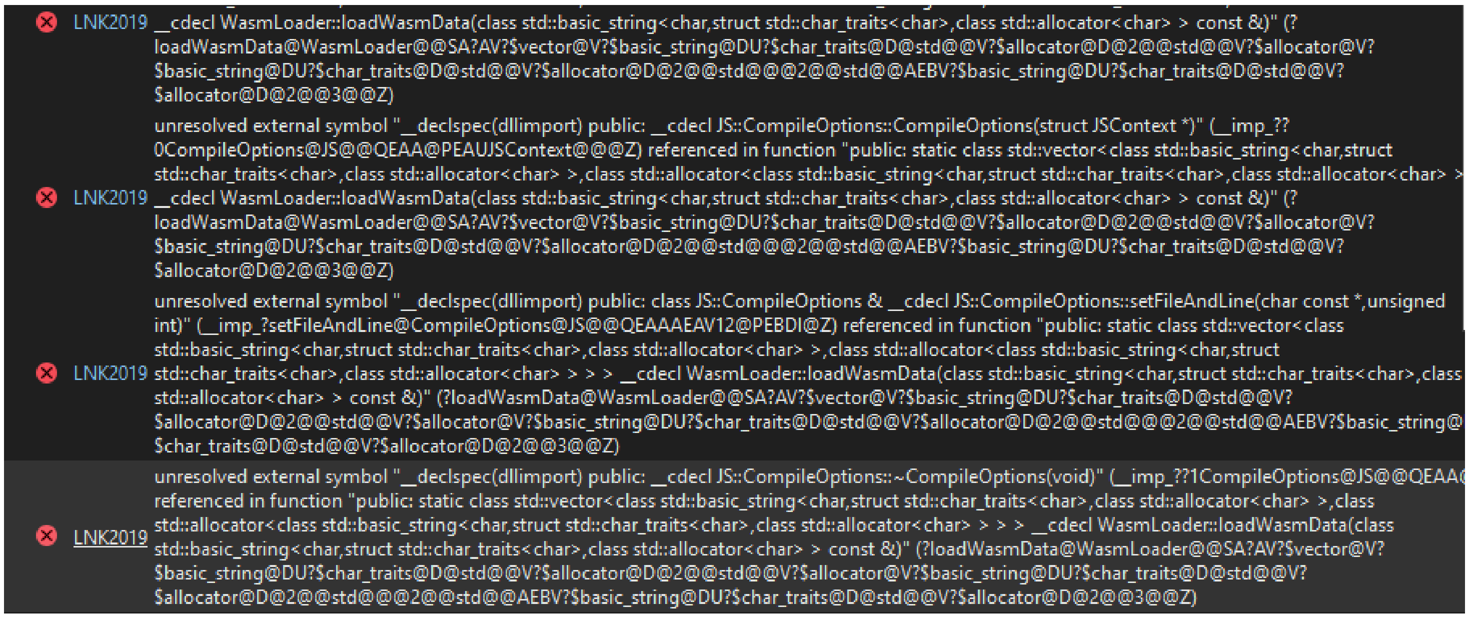
a. In “C:\Users\YourName\DIR\minetest\src\script\native_api” there is a text file call “CMakeLists.txt” and you must add your new native class file using the line:
    i. ${CMAKE_CURRENT_SOURCE_DIR}/native_<class>.cpp
b. This is either a server (a.k.a Common) mod or a client mod. To check which set you should put this line in, you can check the CMakeLists.txt in “C:\Users\YourName\DIR\minetest\src\script\lua_api” and you should place it in the same set as your class.



```
set(common_SCRIPT_NATIVE_API_SRCS
        ${CMAKE_CURRENT_SOURCE_DIR}/native_areastore.cpp
        ${CMAKE_CURRENT_SOURCE_DIR}/native_auth.cpp
        ${CMAKE_CURRENT_SOURCE_DIR}/native_inventory.cpp
        ${CMAKE_CURRENT_SOURCE_DIR}/native_item.cpp
        ${CMAKE_CURRENT_SOURCE_DIR}/native_particles.cpp
        ${CMAKE_CURRENT_SOURCE_DIR}/native_noise.cpp
        ${CMAKE_CURRENT_SOURCE_DIR}/native_settings.cpp
        ${CMAKE_CURRENT_SOURCE_DIR}/native_nodetimer.cpp
        ${CMAKE_CURRENT_SOURCE_DIR}/native_object.cpp

        ${CMAKE_CURRENT_SOURCE_DIR}/native_<class>.cpp

        PARENT_SCOPE)

set(client_SCRIPT_NATIVE_API_SRCS
        ${CMAKE_CURRENT_SOURCE_DIR}/native_minimap.cpp
        ${CMAKE_CURRENT_SOURCE_DIR}/native_camera.cpp
        ${CMAKE_CURRENT_SOURCE_DIR}/native_client.cpp
        ${CMAKE_CURRENT_SOURCE_DIR}/native_localplayer.cpp
        ${CMAKE_CURRENT_SOURCE_DIR}/native_particles_local.cpp
        ${CMAKE_CURRENT_SOURCE_DIR}/native_sound.cpp
        PARENT_SCOPE)
```

Stage 2 - Compiled

12. When compiling, you may get errors like the following (LNK2019):



```
LNK2019 __cdecl WasmLoader::loadWasmData(class std::basic_string<char,struct std::char_traits<char>,class std::allocator<char> > const &)" (?
        loadWasmData@WasmLoader@@SA?AV?$vector@V?$basic_string@DU?$char_traits@D@std@@V?$allocator@D@2@@std@@V?$allocator@V?
        $basic_string@DU?$char_traits@D@std@@V?$allocator@D@2@@std@@@2@@std@@AEBV?$basic_string@DU?$char_traits@D@std@@V?
        $allocator@D@2@@3@@Z)
        unresolved external symbol "__declspec(dllimport) public: __cdecl JS::CompileOptions::CompileOptions(struct JSContext *)" (__imp_??
        0CompileOptions@JS@@QEAA@PEAUJSContext@@@Z) referenced in function "public: static class std::vector<class std::basic_string<char,struct
        std::char_traits<char>,class std::allocator<char> >,class std::allocator<class std::basic_string<char,struct std::char_traits<char>,class std::allocator<char> >
LNK2019 __cdecl WasmLoader::loadWasmData(class std::basic_string<char,struct std::char_traits<char>,class std::allocator<char> > const &)" (?
        loadWasmData@WasmLoader@@SA?AV?$vector@V?$basic_string@DU?$char_traits@D@std@@V?$allocator@D@2@@std@@V?$allocator@V?
        $basic_string@DU?$char_traits@D@std@@V?$allocator@D@2@@std@@@2@@std@@AEBV?$basic_string@DU?$char_traits@D@std@@V?
        $allocator@D@2@@3@@Z)
        unresolved external symbol "__declspec(dllimport) public: class JS::CompileOptions & __cdecl JS::CompileOptions::setFileAndLine(char const *,unsigned
        int)" (__imp_?setFileAndLine@CompileOptions@JS@@QEAAAEAV12@PEBDI@Z) referenced in function "public: static class std::vector<class
        std::basic_string<char,struct std::char_traits<char>,class std::allocator<char> >,class std::allocator<class std::basic_string<char,struct
LNK2019 std::char_traits<char>,class std::allocator<char> > > > __cdecl WasmLoader::loadWasmData(class std::basic_string<char,struct std::char_traits<char>,class
        std::allocator<char> > const &)" (?loadWasmData@WasmLoader@@SA?AV?$vector@V?$basic_string@DU?$char_traits@D@std@@V?
        $allocator@D@2@@std@@V?$allocator@V?$basic_string@DU?$char_traits@D@std@@V?$allocator@D@2@@std@@@2@@std@@AEBV?$basic_string@
        $char_traits@D@std@@V?$allocator@D@2@@3@@Z)
        unresolved external symbol "__declspec(dllimport) public: __cdecl JS::CompileOptions::~CompileOptions(void)" (__imp_??1CompileOptions@JS@@QEAA
        referenced in function "public: static class std::vector<class std::basic_string<char,struct std::char_traits<char>,class std::allocator<char> >,class
LNK2019 std::allocator<class std::basic_string<char,struct std::char_traits<char>,class std::allocator<char> > > > __cdecl WasmLoader::loadWasmData(class
        std::basic_string<char,struct std::char_traits<char>,class std::allocator<char> > const &)" (?loadWasmData@WasmLoader@@SA?AV?$vector@V?
        $basic_string@DU?$char_traits@D@std@@V?$allocator@D@2@@std@@V?$allocator@V?$basic_string@DU?$char_traits@D@std@@V?
        $allocator@D@2@@std@@@2@@std@@AEBV?$basic_string@DU?$char_traits@D@std@@V?$allocator@D@2@@3@@Z)
```

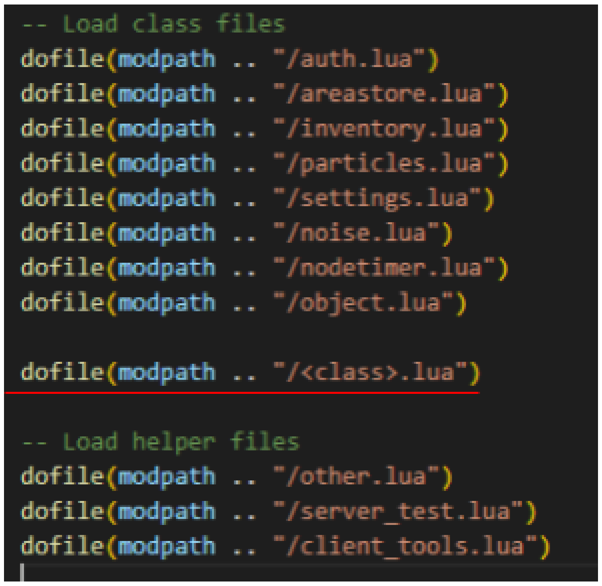
a. This is a linker error and it is caused by some files not being linked to the minetest project correctly.
b. To fix this you either have to:
    i. Check that your files are in Minetest (Step 2.b)
    ii. Relink your Minetest project (Step 3.b of Project Minetest Setup Guide)

13. If the Game Compiles, Compile Stage is complete!

Stage 3 - Tests Written

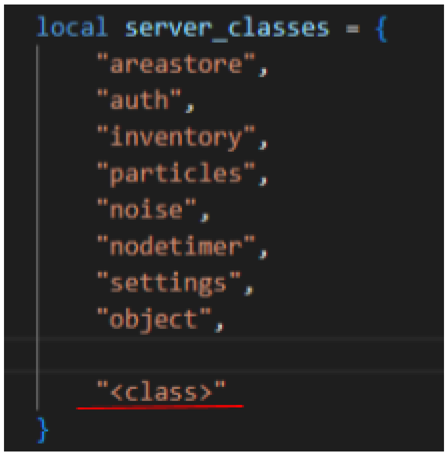
14. ***Now we are going to make the mods required for testing our new functions. First we must create and link the mod to the project.***
    a. As long as you are on the correct build for the CactiCouncil's Minetest Native-API build (or a branch from this!!) you should have a mods folder:
        i. "C:\Users\YourName\DIR\minetest\mods\testingnativeapi_server"
    b. Add a new file:
        i. "<class>.lua"
    c. Example:
        i. "rollback.lua"

15. ***Link your mod to the project.***
    a. In "init.lua" in the same file path, add your class mod to the dofile list



```lua
-- Load class files
dofile(modpath .. "/auth.lua")
dofile(modpath .. "/areastore.lua")
dofile(modpath .. "/inventory.lua")
dofile(modpath .. "/particles.lua")
dofile(modpath .. "/settings.lua")
dofile(modpath .. "/noise.lua")
dofile(modpath .. "/nodetimer.lua")
dofile(modpath .. "/object.lua")

dofile(modpath .. "/<class>.lua")

-- Load helper files
dofile(modpath .. "/other.lua")
dofile(modpath .. "/server_test.lua")
dofile(modpath .. "/client_tools.lua")
```

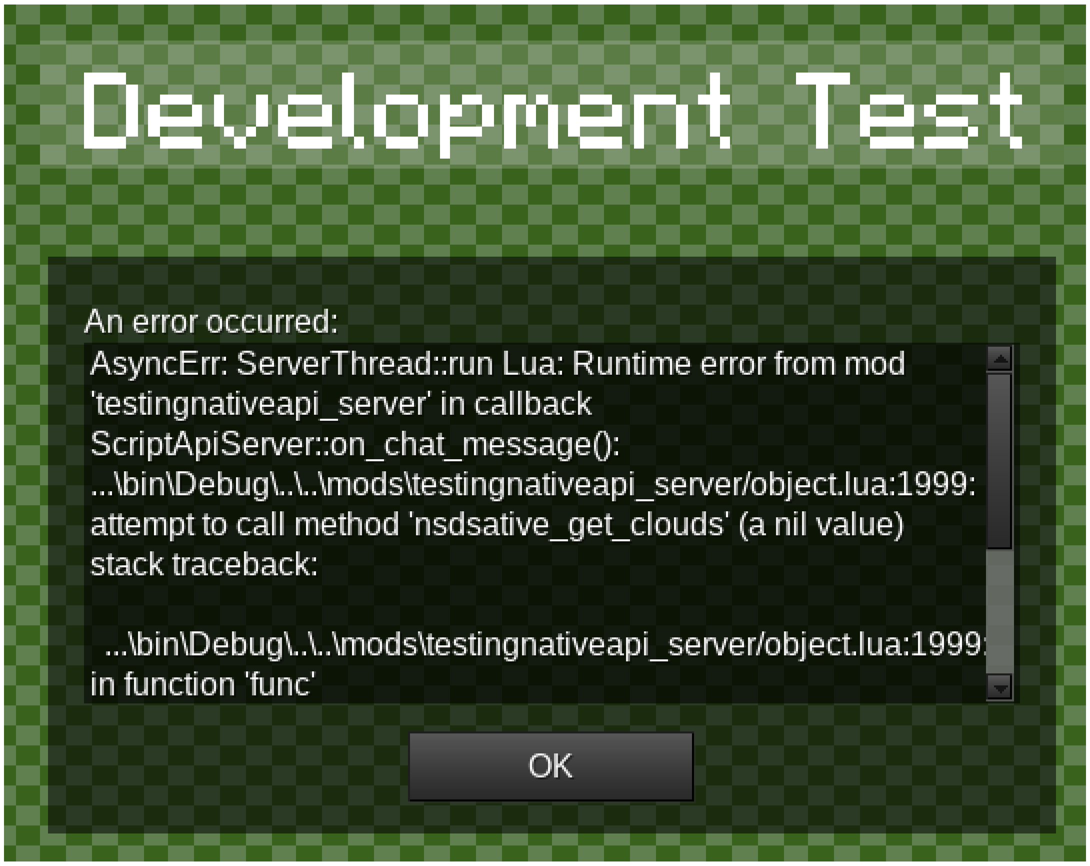
    b. In "server_test.lua" add your class to the server_classes (remember put a comma after the last object)



```lua
local server_classes = {
    "areastore",
    "auth",
    "inventory",
    "particles",
    "noise",
    "nodetimer",
    "settings",
    "object",

    "<class>"
}
```

16. There are 3 types of mods that we use:

a. /lua_<class>_<functionname> <param>
    i. This tests the original functionality of the class function you edited. You need to make this so you know what input parameters generate a testable output.
b. /native_<class>_<functionname> <param>
    i. This is implemented once your lua class is working and should output a similar looking result. If it doesn't and you crash, use the Development Test error report to diagnose the issue:



```
An error occurred:
AsyncErr: ServerThread::run Lua: Runtime error from mod
'testingnativeapi_server' in callback
ScriptApiServer::on_chat_message():
...\bin\Debug\..\..\mods\testingnativeapi_server/object.lua:1999:
attempt to call method 'nsdsative_get_clouds' (a nil value)
stack traceback:

  ...\bin\Debug\..\..\mods\testingnativeapi_server/object.lua:1999:
in function 'func'
```

c. /test_<class>_<functionname> <param>
    i. Compares the lua stack push from both of the functions and determines if the outputs are the same

17. Lua/Native Mod:

a. As stated in the comments in the figure on 8, Lua retrieves sent in params via a call (L, <#>)
b. In lua params look like:
    i. (L, 0) = <FirstParam>
    ii. (L, 1) = <SecondParam>
    iii. (L, 2) = <ThirdParam>
c. To assign these params in Lua
    i. <FirstParam>.<class>_<functionname>(<SecondParam>, <ThirdParam>, etc)

d. The difficulty of this project arises from matching lua, “local” variables to C++ variable types.

18. Tips when writing mods

a. The chat command needed to run is set using the

```
  i.  minetest.register_chatcommand(“ChatCommand”, {
 ii.  func = function(input params)
        1. return
iii.  end
 iv.  })
```

b. and can be called IN-GAME using “/ChatCommand” in the ingame chat

c. In “function()” you can either

   i. do function(self) and refer to player as “singleplayer” or self and no input params OR

   ii. do function(name, param) where name is the player and param is the input

d. Position typically refers to a 3 float array of coords

   i. For testing you could get the the player’s position:

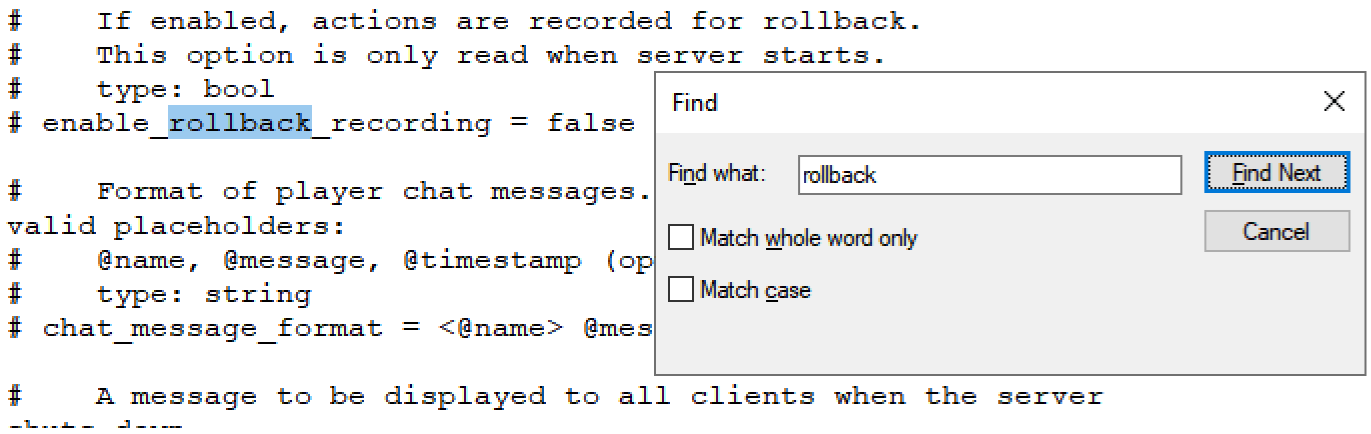
```
1.  local player = minetest.get_player_by_name(name)
2.  local pos = player:get_pos()
```

   ii. Or you could hard code a set of coords:

```
1. local pos = {x = 0, y = 32, z = 0}
```

e. If your function seems like it should be working but isn’t you may have to check the mintest.conf.example file to add a command into the minetest.conf

   i. For example, in rollback the functions were not activating because rollback recording was not enabled



```
#    If enabled, actions are recorded for rollback.
#    This option is only read when server starts.
#    type: bool
# enable_rollback_recording = false

#    Format of player chat messages.
valid placeholders:
#    @name, @message, @timestamp (op
#    type: string
# chat_message_format = <@name> @mes

#    A message to be displayed to all clients when the server
```

f. Refer to the Minetest forums if you are stuck on data types, odds are your question has been answered in one line of text or another:

   i. https://forum.minetest.net/

Stage 4 - Tested

19. ***Run /test_server_classes to ensure that all server classes work.***
20. ***You have finished your first class seperation!***
21. ***Be sure to update the class assignment spreadsheet to “Done”***