Revision

Version 1.0 of the Guide to Creating Vendors.

Introduction

This guide provides instructions for creating and stocking server vendors. These instructions include the templates, strings, and tables containing the stock, payment methods, and making an event to turn the vendors on and off.

In this example, we will be creating three vendors and a gift box, the latter being a generic container that can hold simple items. The first vendor will sell boxes of marauder tokens for cash, the second sells chronicler tokens in exchange for marauder tokens, and the third sells heroic tokens in exchange for chronicler tokens. This approach intends to demonstrate the payment methods, which will become clear as you read this guide.

I provide a working example in my fork (HeronAlexandria · GitHub), and while I have performed basic testing, I cannot guarantee that the example is bug-free. Use my examples at your own risk, and remember, snapshot your VM before beginning!

You will need Sitners, but we will not be building tre files. I will show you how to place files in the client directory, so constructing a tre file is unnecessary, making development much easier. Programming is required, but it is all in Java, and for the most part, you will be able to use what others have written and make some simple changes.

There are two commits for this guide. The first is Vendor Demonstration · HeronAlexandria/dsrc@aa06fbf · GitHub, which contains the code for the vendors and the gift box. The second is Fixes to vendor demo. Added event to turn vendors on and off · HeronAlexandria/dsrc@3e99932 · GitHub, which contains some minor fixes and changes to allow the vendors to be turned on and off in server configuration.

The Generic Gift Box

The vendors only sell one item at a time. For example, during a holiday event, you earn tokens, and you trade those tokens in for a reward. You get a single item from the vendor. Some things sold on vendors might be a small stack, but that is one item with charges, and the object was defined that way. The generic gift box allows the creation of multiple items without changing the time or providing additional definitions.

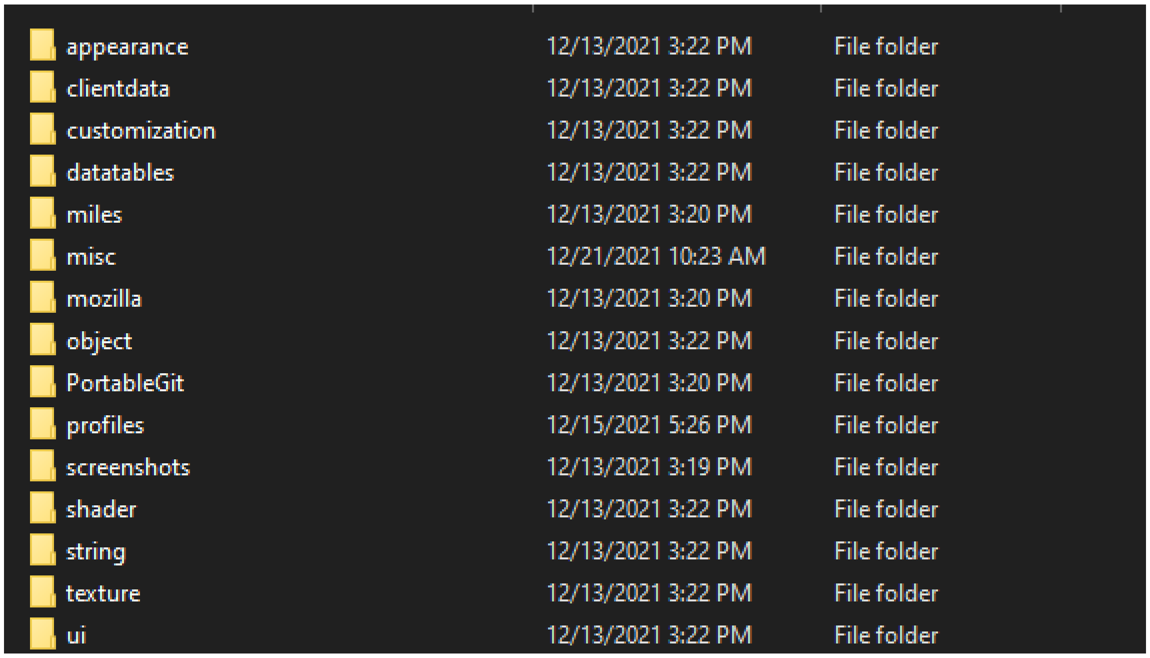
A functioning gift box requires a script, an entry in the master items table, a string file for the box, and an entry in the static items table. Since we added an object, we will copy the CRC file from the server to the client. This is a lot to take in, so let's go through it.

To make a gift box, you add an item to the master items table, attach the script, and set the object variables. After compiling, you move the crc file from the server to the client. To see strings for the responses from the box and an actual name, you must change string files; we will go over that soon. When the object gets created, the object variables are assigned to the object, and the script is attached.

The box will not work without the script, so let's begin there. I have already written the script, which you can find at dsrc/sku.0/sys.server/compiled/game/script/item/gift_boxes at master · HeronAlexandria/dsrc · GitHub. The name of the script is generic_gift_box.java.

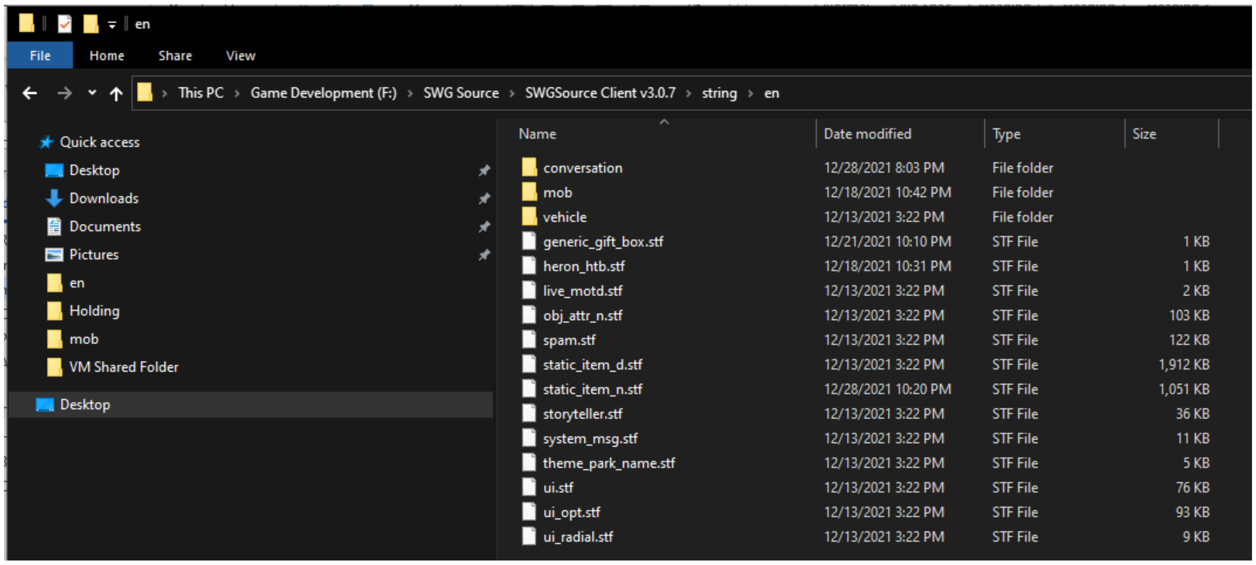
The gift box is modeled after the standard uniform box (item_npe_uniform_crate_01_01). The model script is npe.uniform_crate. The uniform box has a simple menu, including an open action. When opened, items are created and placed in the character's inventory, and the box is destroyed. The gift box works the same way, except that the contents are programmable.

The script expects strings to be stored in a file named generic_gift_box.stf. I created this file with Sitners and placed it in the client folder in string/en. While we are on this subject, let's take a quick look at where to store client-side resources. You are probably familiar with tre files, but you do not have to store assets in tre files; you can place them directly in the client folder if you follow the file structure. Below is a partial screenshot of my client folder. Any assets in this structure override tre files, so if you place an asset in a tree file and it is not working, check the folder structure for a copy of the file.



appearance	12/13/2021 3:22 PM	File folder
clientdata	12/13/2021 3:22 PM	File folder
customization	12/13/2021 3:22 PM	File folder
datatables	12/13/2021 3:22 PM	File folder
miles	12/13/2021 3:20 PM	File folder
misc	12/21/2021 10:23 AM	File folder
mozilla	12/13/2021 3:20 PM	File folder
object	12/13/2021 3:22 PM	File folder
PortableGit	12/13/2021 3:20 PM	File folder
profiles	12/15/2021 5:26 PM	File folder
screenshots	12/13/2021 3:19 PM	File folder
shader	12/13/2021 3:22 PM	File folder
string	12/13/2021 3:22 PM	File folder
texture	12/13/2021 3:22 PM	File folder
ui	12/13/2021 3:22 PM	File folder

The string file for the box is expected to be stored in the English string folder, as shown below. I have placed this file in my client assets folder in my fork; you can download it from there.



Name	Date modified	Type	Size
conversation	12/28/2021 8:03 PM	File folder	
mob	12/18/2021 10:42 PM	File folder	
vehicle	12/13/2021 3:22 PM	File folder	
generic_gift_box.stf	12/21/2021 10:10 PM	STF File	1 KB
heron_htb.stf	12/18/2021 10:31 PM	STF File	1 KB
live_motd.stf	12/13/2021 3:22 PM	STF File	2 KB
obj_attr_n.stf	12/13/2021 3:22 PM	STF File	103 KB
spam.stf	12/13/2021 3:22 PM	STF File	122 KB
static_item_d.stf	12/13/2021 3:22 PM	STF File	1,912 KB
static_item_n.stf	12/28/2021 10:20 PM	STF File	1,051 KB
storyteller.stf	12/13/2021 3:22 PM	STF File	36 KB
system_msg.stf	12/13/2021 3:22 PM	STF File	11 KB
theme_park_name.stf	12/13/2021 3:22 PM	STF File	5 KB
ui.stf	12/13/2021 3:22 PM	STF File	76 KB
ui_opt.stf	12/13/2021 3:22 PM	STF File	93 KB
ui_radial.stf	12/13/2021 3:22 PM	STF File	9 KB

The script adds the menu to take the string from the string file; you can see this in OnObjectMenuRequest. When the box is used, OnObjectMenuSelect is called. This method parses the object variables and checks for validity, and if the variables are valid, the items are created, and the box is destroyed.

You do not have to restart the server if you change a script, and sometimes you can reload a table. If you edit a script, recompile it, and then reload the script from the client using "/script reload" For example, "/script reload item.gift_boxes.generic_gift_box" will reload the giftbox script.

You can reload tables, such as "server reloadTabe datatables/item/master_item/master_item.iff" will reload the master item table. Most of the time this works, but I have seen this report success but not actually take effect.

Take a look at the master items table located in dsrc/sku.0/sys.server/compiled/game/datatables/item/master_item at master · HeronAlexandria/dsrc (github.com). Find the entry for the item item_heron_gift_pgc_01_10. Note that this is assigned the generic uniform template (column B), has some object variables (column G), which we will discuss in a moment, and assigns the generic gift box script (column K) to the object.

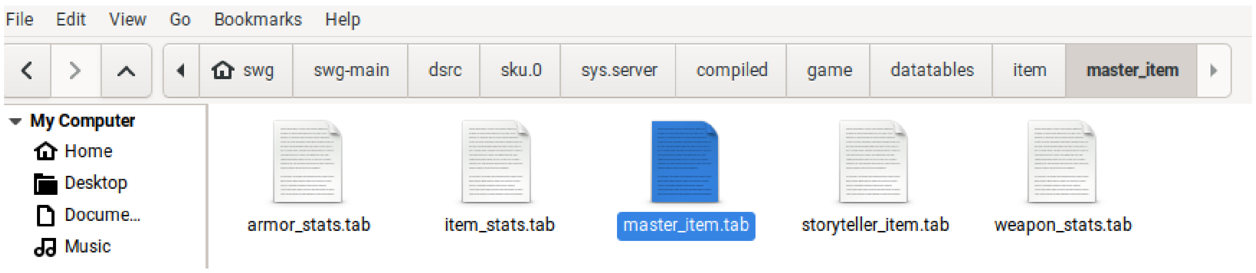
When the object is created, the object variables and script are assigned. Note that you can edit the script, then reload the script, and you do not have to update the object. If you edit the object variables, recompile the master items table and reload the data table, this does not update any object variables for existing objects. You would have to destroy any existing items and recreate them or edit the object variables manually.

The provided script recognizes four object variables: mode, source, selections, and items, and they are case sensitive. Mode, source, and selections have defaults, but items must be defined. Mode is either "all" or "random" and defaults to "all." When mode is set to "all," everything in the items list is generated; when the value of selections defines "random", the number of items generated, selections are ignored if the mode is "all" and must be at least one when the mode is "random."

The value of source is either "list" or "table" and defaults to "list." When the value is "list," the items are in a semi-colon delimited list. Take a look a the object variables for item_heron_gift_pgc_01_10 for an example. Note the colon after each entry; this tells the script to create 10 of the item. The script will create ten copper, silver, and gold chronicler tokens in this example. Note that mode is not defined, so it defaults to all.

If the source is "table," items refer to a table. Placing items in a table allows them to be restricted by class if desired. Refer to the object item_heron_gift_test in the master item table for an example. For the most part, a list will do what you need.

The procedure for creating a new gift box is simple. Copy one of the entries I have made in the master items table, change the template in column B if you like, and then change the object variables. Edit the table "static_item_n.sft" located in the client directory string/en using Sitners and add your object name and string description. The following screenshot shows the location of the master items table:



armor_stats.tab item_stats.tab master_item.tab storyteller_item.tab weapon_stats.tab

I placed my static items string file in [client-assets/string/en at master · HeronAlexandria/client-assets (github.com)](), named "heron_static_item_n.stf"for reference.

Recompile, start your server, and try creating the object you defined using the createstaticitem command. You should be able to open the box and claim your goodies.

Creating the First Vendor

The first vendor will offer a gift box containing marauder tokens for credits. These tokens will be used to pay the second vendor, who will sell chronicler tokens, and the third vendor will accept the chronicler tokens and sell heroic tokens. Creating a vendor is not difficult, but there are a lot of steps, and it is easy to make a mistake.

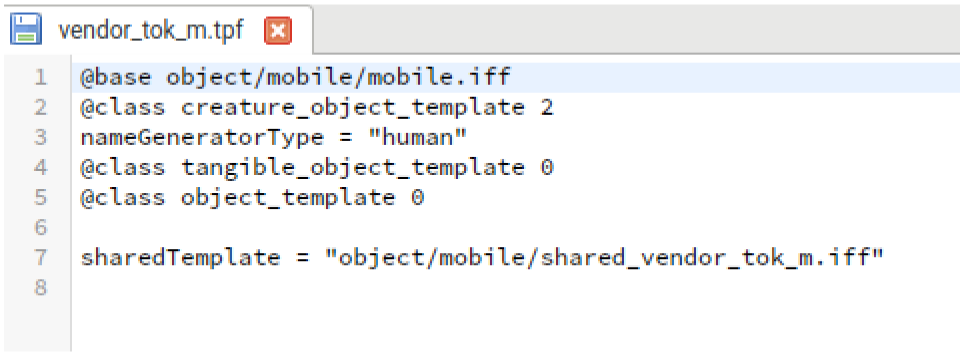
To create a vendor, we perform the following steps:

1. Create the templates for the NPC.
2. Create a conversation script for the NPC.

3. Create the inventory.
4. Add the entry to the creature's table.
5. Add the spawner to the proper build-out file.
6. Recompile and copy the files.

Creating the Templates

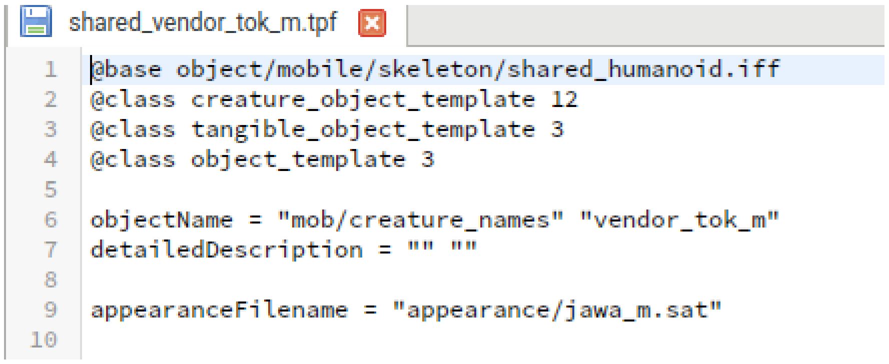
The first vendor is the marauder token vendor, which I named vendor_tok_m. There are thus two template files to create: vendor_tok_m.tpf and shared_vendor_tok_m.tpf. The first file, vendor_tok_m.tpf is located in dsrc/sku.0/sys.server/compiled/game/object/mobile. I based this on vendor_tcg_1.tpf; I copied the file and made the changes. You can use mine for a template or copy another. The only information that you need to change is the location of the shared template, as shown below:


vendor_tok_m.tpf

```
@base object/mobile/mobile.iff
@class creature_object_template 2
nameGeneratorType = "human"
@class tangible_object_template 0
@class object_template 0

sharedTemplate = "object/mobile/shared_vendor_tok_m.iff"

```

The reference to the shared template is the compiled file, which is named shared_vendor_tok_m.iff, not shared_vendor_tok_m.tpf. Remember that file names in the Linux world are case-sensitive.

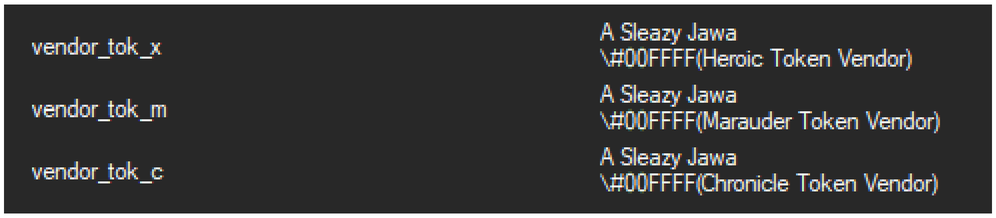
The shared template file is located in dsrc/sku.0/sys.shared/compiled/game/object/mobile. Below is a screenshot of shared_vendor_tok_m.tpf:


shared_vendor_tok_m.tpf

```
@base object/mobile/skeleton/shared_humanoid.iff
@class creature_object_template 12
@class tangible_object_template 3
@class object_template 3

objectName = "mob/creature_names" "vendor_tok_m"
detailedDescription = "" ""

appearanceFilename = "appearance/jawa_m.sat"

```

Take careful note of the objectName. This entry defines the name of the NPC in the string file and the name of the string file to look in. I tried using a file other than creature_names, but it would not work.

This string file is built on the client and copied to the server. Without the string file, the server will randomly generate a name for the NPC. I have added my string file to client-assets/string/en/mob at master · HeronAlexandria/client-assets (github.com), named heron_creature_names.stf for reference. Here is a screenshot and that shows the names of the NPCs and their string names:

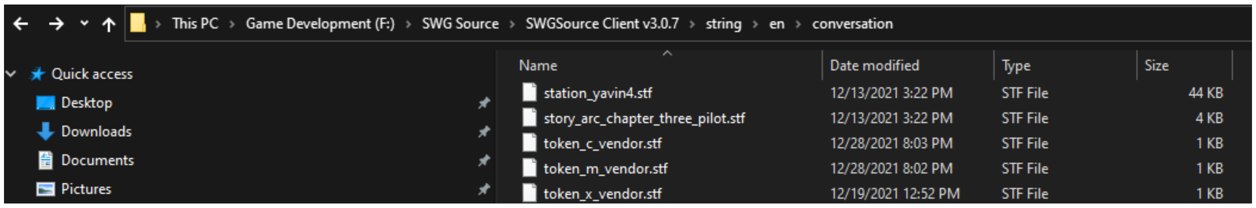


vendor_tok_x	A Sleazy Jawa \#00FFFF(Heroic Token Vendor)
vendor_tok_m	A Sleazy Jawa \#00FFFF(Marauder Token Vendor)
vendor_tok_c	A Sleazy Jawa \#00FFFF(Chronicle Token Vendor)

Create the Conversation Script

The conversation scripts for vendors are straightforward. I used a TCG vendor for the base, but you can use my token_m_vendor.java script. You will find the conversation script at dsrc/token_m_vendor.java at master · HeronAlexandria/dsrc (github.com).

You need to change the reference to the conversation file on line 14 and change everything named token_m_vendor to your vendor's name. These references include the method names, such as token_m_vendor_condition__defaultCondition on line 14 and the strings on line 30. Just use the find function of an editor to move through the file and change the references. The string file referenced on line 14 goes in the client folder as shown below:



This PC > Game Development (F:) > SWG Source > SWGSource Client v3.0.7 > string > en > conversation

Name	Date modified	Type	Size
station_yavin4.stf	12/13/2021 3:22 PM	STF File	44 KB
story_arc_chapter_three_pilot.stf	12/13/2021 3:22 PM	STF File	4 KB
token_c_vendor.stf	12/28/2021 8:03 PM	STF File	1 KB
token_m_vendor.stf	12/28/2021 8:02 PM	STF File	1 KB
token_x_vendor.stf	12/19/2021 12:52 PM	STF File	1 KB

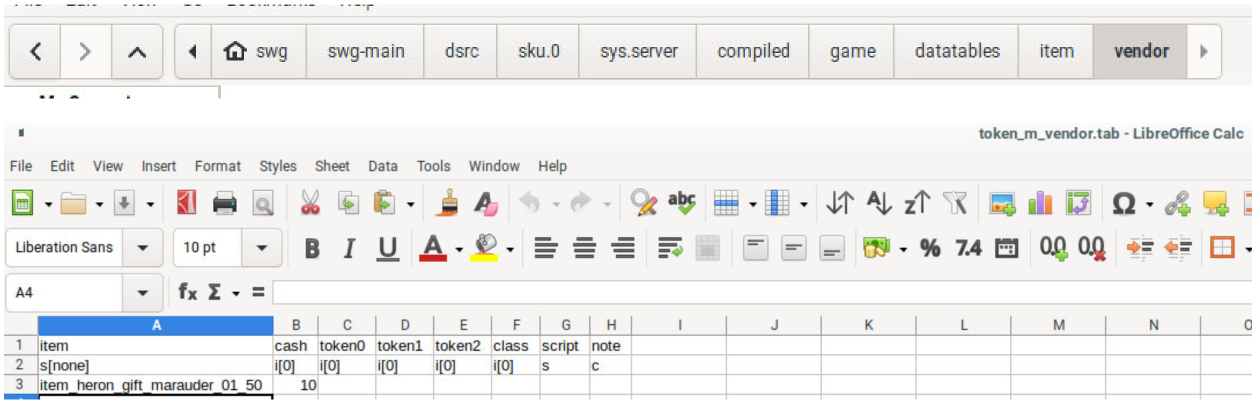
I created these from a TCG vendor conversation, but you are welcome to use mine and edit them as you wish. These files are included at client-assets/string/en/conversation at master · HeronAlexandria/client-assets (github.com).

I have different conversation files for my three vendors, but they are simple enough to share the same conversation script and string files. Just copy files, change the names, and make the changes you need.

Create Inventory

The vendor's inventory is stored in a data table. There are two steps for the vendor's inventory; the first is to create the gift box, and the second is to add it to the vendor's data table. Refer to the master item table I provided and look for item_heron_gift_marauder_01_50. Take a look at the attached script and the object variables. The object variables instruct the script to create marauder tokens when the box is opened.

Now go look at the vendor's data table token_m_vendor.tab, which you will find in [dsrc/sku.0/sys.server/compiled/game/datatables/item/vendor at master · HeronAlexandria/dsrc (github.com)](). Here is a screenshot from my VM:



swg
swg-main
dsrc
sku.0
sys.server
compiled
game
datatables
item
vendor
token_m_vendor.tab - LibreOffice Calc
File Edit View Insert Format Styles Sheet Data Tools Window Help
Liberation Sans
10 pt
A4

	A	B	C	D	E	F	G	H
1	item	cash	token0	token1	token2	class	script	note
2	s[none]	i[0]	i[0]	i[0]	i[0]	i[0]	s	c
3	item_heron_gift_marauder_01_50	10						

This vendor only sells the marauder token box I defined in the master items table for ten credits.

Add the Vendor to the Creature's Table

The creature's table, creatures.tab, is located in dsrc/sku.0/sys.server/compiled/game/datatables/mob; refer to [dsrc/sku.0/sys.server/compiled/game/datatables/mob at master · HeronAlexandria/dsrc (github.com)]() for mine.

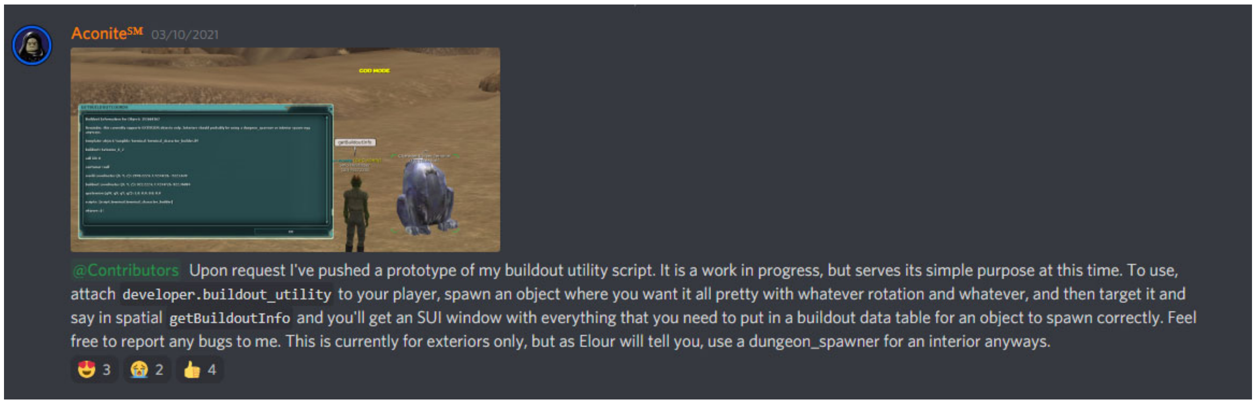
There are many entries, so the easiest way to look at the vendor definition is to look at my table. Find "vendor_tok_m.tab" Use this as a guide when making your own. In general, you want to copy my entry, change the name in column A then make changes to columns N (the NPC template), BM (sets the object variables), BN (attaches the scripts).

The NPC template is the compiled template you defined above. The object variables set the table name from which the vendor's inventory is taken. The scripts in column BN are the vendor script npc.vendor.vendor, which provides the vendor functionality, and the conversation script we created earlier.

The creature's table entry binds the NPC template, vendor table, and the scripts together.

Add Spawner

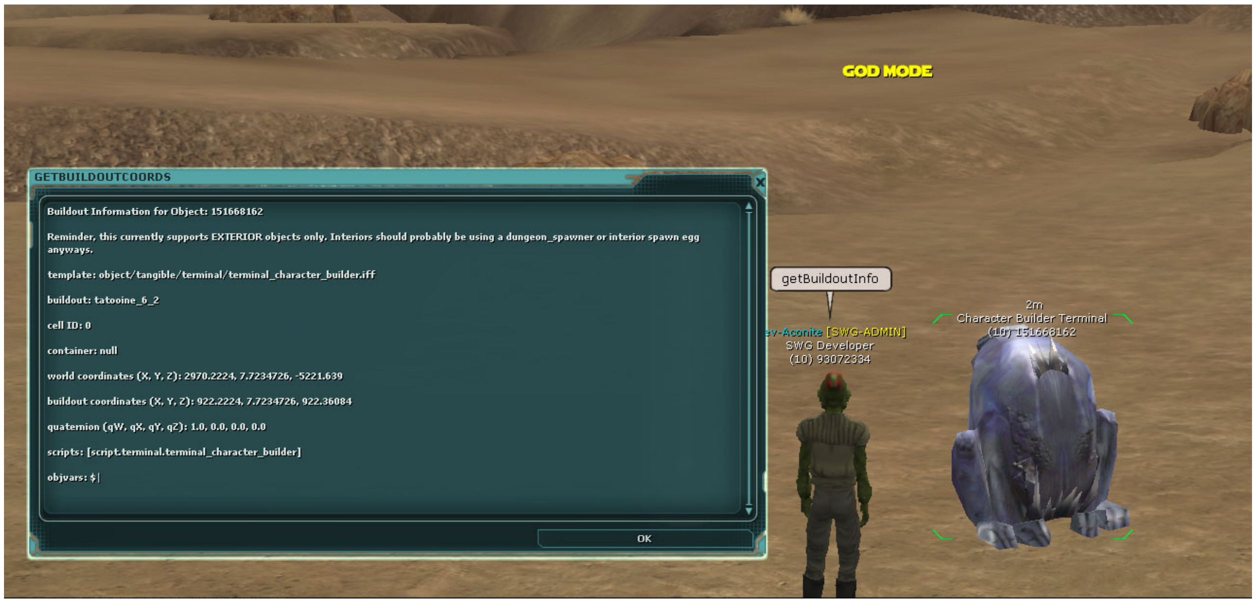
Now that we have the definitions done, we need to add the spawner so the NPC will spawn. The NPC is created with an area spawner, but you must know which build-out file to place the spawner in, the coordinates, and the facing. Buildout coordiantes are not the same as world coordiantes. Fortunately, Aconite has provided a utility for doing just that:



AconiteSM 03/10/2021

@Contributors Upon request I've pushed a prototype of my buildout utility script. It is a work in progress, but serves its simple purpose at this time. To use, attach `developer.buildout_utility` to your player, spawn an object where you want it all pretty with whatever rotation and whatever, and then target it and say in spatial `getBuildoutInfo` and you'll get an SUI window with everything that you need to put in a buildout data table for an object to spawn correctly. Feel free to report any bugs to me. This is currently for exteriors only, but as Elour will tell you, use a dungeon_spawner for an interior anyways.

😍 3 😭 2 👍 4



```
GETBUILDOUTCOORDS

Buildout Information for Object: 151668162

Reminder, this currently supports EXTERIOR objects only. Interiors should probably be using a dungeon_spawner or interior spawn egg
anyways.

template: object/tangible/terminal/terminal_character_builder.iff

buildout: tatooine_6_2

cell ID: 0

container: null

world coordinates (X, Y, Z): 2970.2224, 7.7234726, -5221.639

buildout coordinates (X, Y, Z): 922.2224, 7.7234726, 922.36084

quaternion (qW, qX, qY, qZ): 1.0, 0.0, 0.0, 0.0

scripts: [script.terminal.terminal_character_builder]

objvars: $|
```

The original link is found here: https://discord.com/channels/366560008068005892/366576100941234177/819406962613420062.

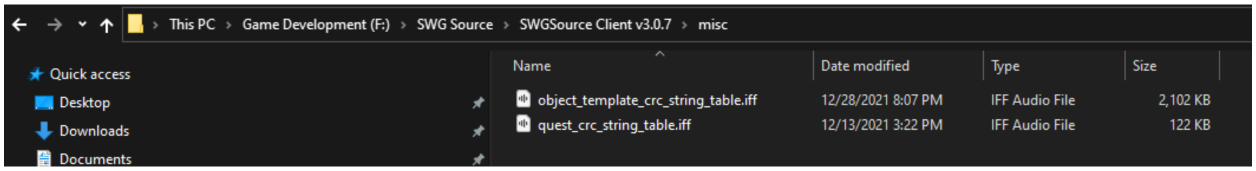
Use the above script to get the build-out coordinates, the quaternion (facing), and the build-out file. You will place the spawner in the build-out file. Take a look at dsrc/sku.0/sys.server/compiled/game/datatables/buildout/tatooine at master · HeronAlexandria/dsrc (github.com), and find the file vendor_demo_mos_eisley.tab. I placed the definitions for my vendors in that file. If you treat this guide as a tutorial, the spawners go in tatooine_6_2.tab. The file vendor_demo_mos_eisley.tab is a separate build-out file used to turn the vendors on and off. I will cover that topic last. If you plan to make your vendors switchable, I suggest getting them working and then adding the configuration last.

Please take a look at the build-out file I provided and use it as a guide for making your spawners.

Recompile and Copy Files

Now recompile. Check the crc files. They are located in swg-main/data/sku.0/sys.client/build/game/misc, swg-main/data/sku.0/sys.server/build/game/misc, and swg-main/data/sku.0/serverdata/misc. The files locate din client and serverdata/misc are the same file. If the crc did not rebuild, then run run ant build_object_template_crc to force a crc rebuild.

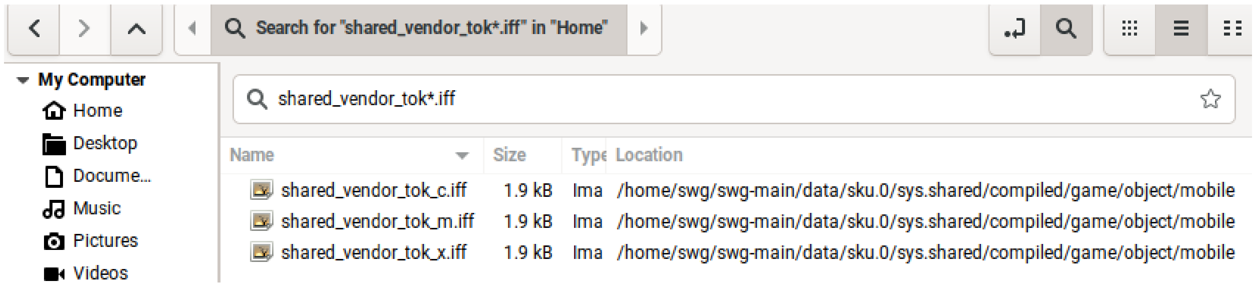
Copy the client crc file to the client directory as shown below:


This PC › Game Development (F:) › SWG Source › SWGSource Client v3.0.7 › misc

Name	Date modified	Type	Size
object_template_crc_string_table.iff	12/28/2021 8:07 PM	IFF Audio File	2,102 KB
quest_crc_string_table.iff	12/13/2021 3:22 PM	IFF Audio File	122 KB

If you do not copy the crc file, the client will not find the vendor, and it will not spawn correctly.

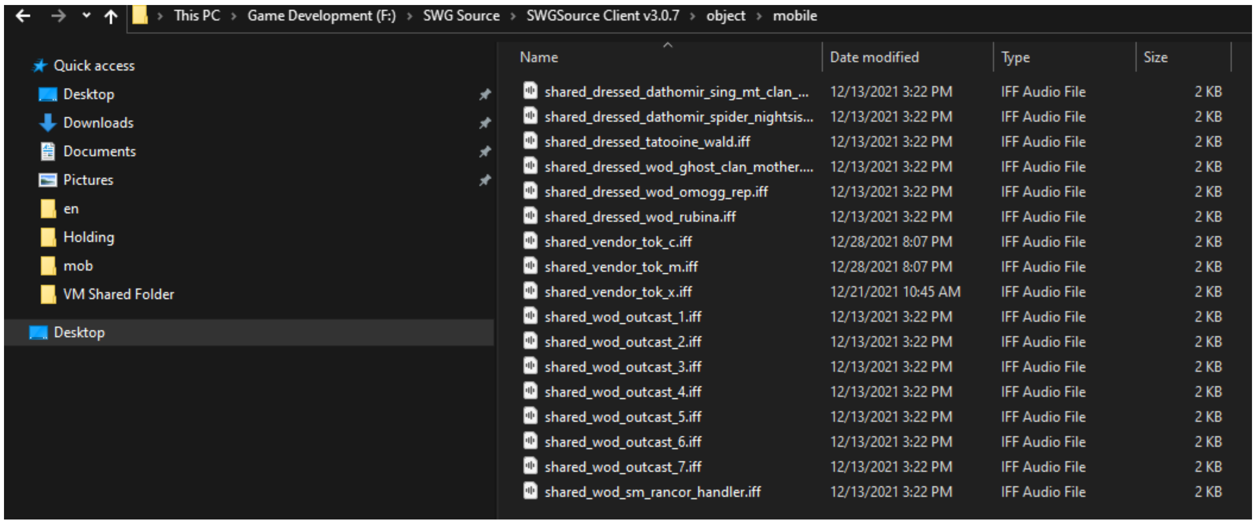
Next, copy the shared vendor templates from the server to the client. You will find them on the server here:


Search for "shared_vendor_tok*.iff" in "Home"

shared_vendor_tok*.iff

Name	Size	Type	Location
shared_vendor_tok_c.iff	1.9 kB	Ima	/home/swg/swg-main/data/sku.0/sys.shared/compiled/game/object/mobile
shared_vendor_tok_m.iff	1.9 kB	Ima	/home/swg/swg-main/data/sku.0/sys.shared/compiled/game/object/mobile
shared_vendor_tok_x.iff	1.9 kB	Ima	/home/swg/swg-main/data/sku.0/sys.shared/compiled/game/object/mobile

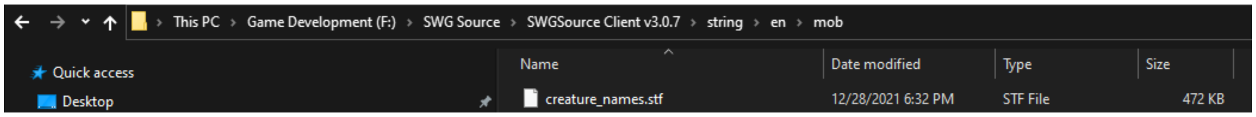
They go in the client folder here:


This PC › Game Development (F:) › SWG Source › SWGSource Client v3.0.7 › object › mobile

Name	Date modified	Type	Size
shared_dressed_dathomir_sing_mt_clan_...	12/13/2021 3:22 PM	IFF Audio File	2 KB
shared_dressed_dathomir_spider_nightsis...	12/13/2021 3:22 PM	IFF Audio File	2 KB
shared_dressed_tatooine_wald.iff	12/13/2021 3:22 PM	IFF Audio File	2 KB
shared_dressed_wod_ghost_clan_mother....	12/13/2021 3:22 PM	IFF Audio File	2 KB
shared_dressed_wod_omogg_rep.iff	12/13/2021 3:22 PM	IFF Audio File	2 KB
shared_dressed_wod_rubina.iff	12/13/2021 3:22 PM	IFF Audio File	2 KB
shared_vendor_tok_c.iff	12/28/2021 8:07 PM	IFF Audio File	2 KB
shared_vendor_tok_m.iff	12/28/2021 8:07 PM	IFF Audio File	2 KB
shared_vendor_tok_x.iff	12/21/2021 10:45 AM	IFF Audio File	2 KB
shared_wod_outcast_1.iff	12/13/2021 3:22 PM	IFF Audio File	2 KB
shared_wod_outcast_2.iff	12/13/2021 3:22 PM	IFF Audio File	2 KB
shared_wod_outcast_3.iff	12/13/2021 3:22 PM	IFF Audio File	2 KB
shared_wod_outcast_4.iff	12/13/2021 3:22 PM	IFF Audio File	2 KB
shared_wod_outcast_5.iff	12/13/2021 3:22 PM	IFF Audio File	2 KB
shared_wod_outcast_6.iff	12/13/2021 3:22 PM	IFF Audio File	2 KB
shared_wod_outcast_7.iff	12/13/2021 3:22 PM	IFF Audio File	2 KB
shared_wod_sm_rancor_handler.iff	12/13/2021 3:22 PM	IFF Audio File	2 KB

Remember, any files in the client folder override any entries in tre files. We could put the shared vendor files in a tre file without fear of being overridden since they do not exist, but it is easier to develop without making tre files.

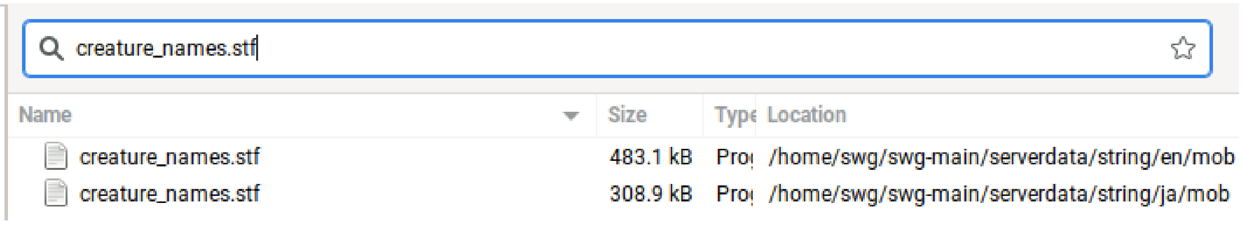
Finally, we want to copy the creature names string file from the client to the server. If you do not do this, the random name generator will generate a name for your vendor. The file is named creature_names.stf and is located in the client folder here:



This PC › Game Development (F:) › SWG Source › SWGSource Client v3.0.7 › string › en › mob

Name	Date modified	Type	Size
creature_names.stf	12/28/2021 6:32 PM	STF File	472 KB

The file goes on the server, as shown below. Note that we only need the English string file.



creature_names.stf

Name	Size	Type	Location
creature_names.stf	483.1 kB	Pro	/home/swg/swg-main/serverdata/string/en/mob
creature_names.stf	308.9 kB	Pro	/home/swg/swg-main/serverdata/string/ja/mob

You are Done!

Once the above steps are completed, start your server and test your vendor. If you did everything correctly, it would work just fine. If not, go back through the steps one at a time, or ask for help on Discord.

Creating the Second Vendor

The second vendor will offer chronicler tokens in exchange for marauder tokens. Repeat the steps for the first vendor; remember that the vendor's name is vendor_tok_c. Note that in the creature's table, the object variable defines a token type, which means that token0 in the vendor table refers to the marauder token, and the others are unused. Even though the other tokens were not used, I had to define through token4 to avoid an index exception.

Refer to the files I provided in my fork as examples, using the steps to create the first vendor as a guide.

Creating the Third Vendor

The third vendor sells heroic tokens in exchange for chronicler tokens. You will notice that I did not define boxes for the echo base tokens; I leave that as an exercise for the reader. This vendor takes only tokens, not cash. The default types are used since we did not define a token type. The default tokens are defined in the file trail.java located in dsrc/sku.0/sys.server/compiled/game/script. The default types are defined in the array HEROIC_TOKENS. Thus, token0 is Axkva Min, token1 is Tusken, and so on.

Refer to the files I provided in my fork as examples, using the steps to create the first vendor as a guide.

Toggling Vendors

To toggle vendors on and off, like the holiday vendors, you have to place the spawners in a standalone build-out file, add that build-out file to the planet's areas, and then bind that area to an event. You then update the holiday controller to start and stop the event. Once this is done, you can turn the vendors on and off in local options. It is not that complicated, but we will go through the steps briefly.

Note that you should implement this functionality when you need to turn vendors on and off. Remove the spawners if you want to remove vendors and never see them, such as the TCG vendors. This approach really should only be used for events, such as Halloween, Lifeday, or your server anniversary event.

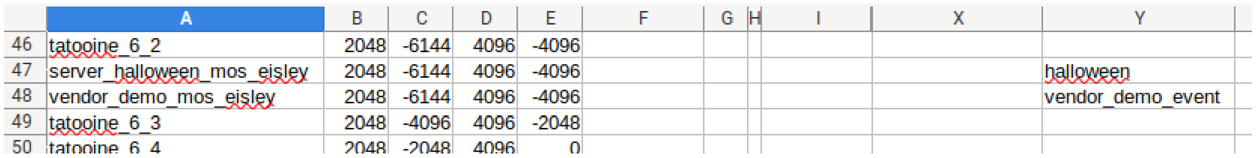
1. Define the build-out file.
2. Add the build-out file to the areas.
3. Update the holiday controller.
4. Rebuild and copy the area file
5. Turn the event on or off in local options.

Define Build-Out

Remember when we defined the vendor spawners and ignored the vendor_demo_mos_eisley.tab file and placed the spawners in tatooine_6_2.tab? Now you know why the spawners for my vendors are in a separate file. Take the spawners out of tatooine_6_2.tab, and put them in vendor_demo_mos_eisley.tab. You can find the build-out files in dsrc/sku.0/sys.server/compiled/game/datatables/buildout/tatooine.

Add the Area

The area file is shared (meaning it does on the client and server), which means it is located in dsrc/sku.0/sys.server/compiled/game/datatables/buildout/Tatooine. The area file is named areas_tatooine.tab. Here is a screenshot of the relevant area:



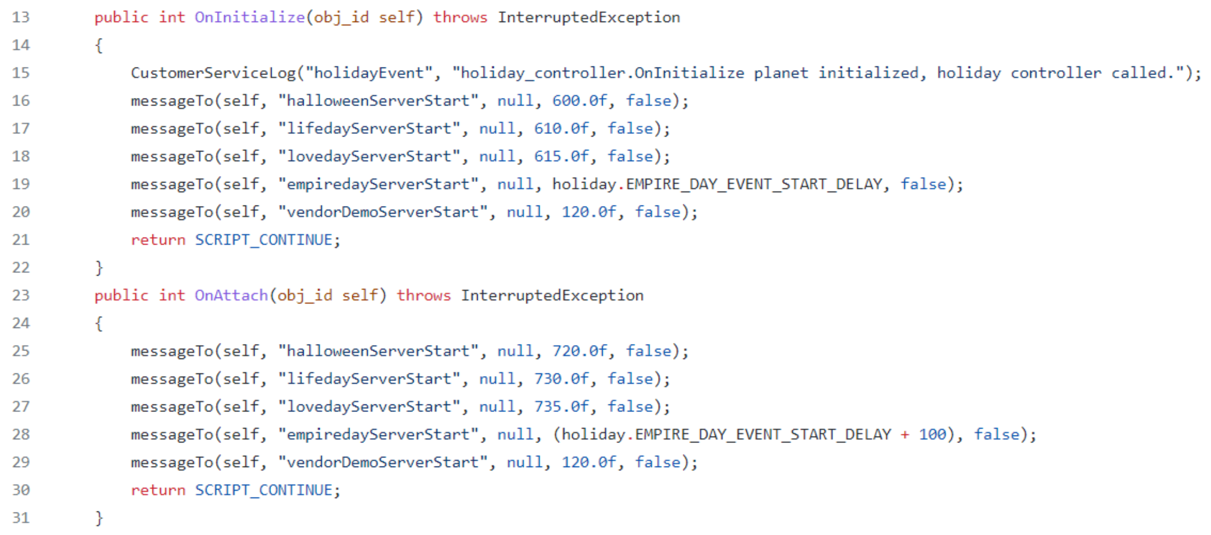
	A	B	C	D	E	F	G	H	I	X	Y
46	tatooine_6_2	2048	-6144	4096	-4096						
47	server_halloween_mos_eisley	2048	-6144	4096	-4096						halloween
48	vendor_demo_mos_eisley	2048	-6144	4096	-4096						vendor_demo_event
49	tatooine_6_3	2048	-4096	4096	-2048						
50	tatooine_6_4	2048	-2048	4096	0						

I set the area for the vendor demo to be the same as the Halloween area. Note column Y, which sets the name of the event. This means that "vendor_demo_event" must be running for this area to be active.

Update Holiday Controller

We need to update the holiday controller to create the vendor demo event. You will find the holiday controller in dsrc/sku.0/sys.server/compiled/game/script/event, named holliday_controller.java. The link to my fork is dsrc/sku.0/sys.server/compiled/game/script/event at master · HeronAlexandria/dsrc (github.com).

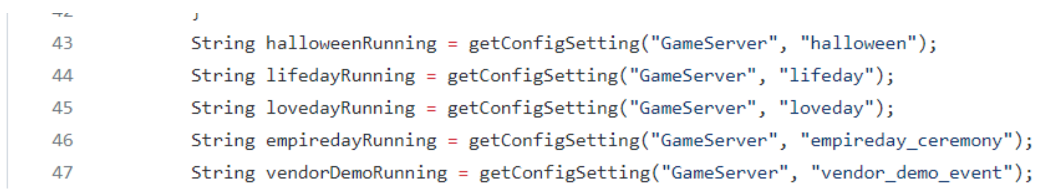
Look for the event I added, vendorDemoServer, and use it as a guide. There are a few things that require an additional explanation. Take a look at the following screenshot:



```
13      public int OnInitialize(obj_id self) throws InterruptedException
14      {
15          CustomerServiceLog("holidayEvent", "holiday_controller.OnInitialize planet initialized, holiday controller called.");
16          messageTo(self, "halloweenServerStart", null, 600.0f, false);
17          messageTo(self, "lifedayServerStart", null, 610.0f, false);
18          messageTo(self, "lovedayServerStart", null, 615.0f, false);
19          messageTo(self, "empiredayServerStart", null, holiday.EMPIRE_DAY_EVENT_START_DELAY, false);
20          messageTo(self, "vendorDemoServerStart", null, 120.0f, false);
21          return SCRIPT_CONTINUE;
22      }
23      public int OnAttach(obj_id self) throws InterruptedException
24      {
25          messageTo(self, "halloweenServerStart", null, 720.0f, false);
26          messageTo(self, "lifedayServerStart", null, 730.0f, false);
27          messageTo(self, "lovedayServerStart", null, 735.0f, false);
28          messageTo(self, "empiredayServerStart", null, (holiday.EMPIRE_DAY_EVENT_START_DELAY + 100), false);
29          messageTo(self, "vendorDemoServerStart", null, 120.0f, false);
30          return SCRIPT_CONTINUE;
31      }
```

See the messageTo calls on lines 20 and 29? Note the 120; this is the number of seconds before the event starts. Vendors will not appear until the server runs for two minutes. Note that the delays for the other events are much longer; I set this demo to two minutes for easier testing.

An example of checking to see if the event is running appears on line 47 in the following screenshot:



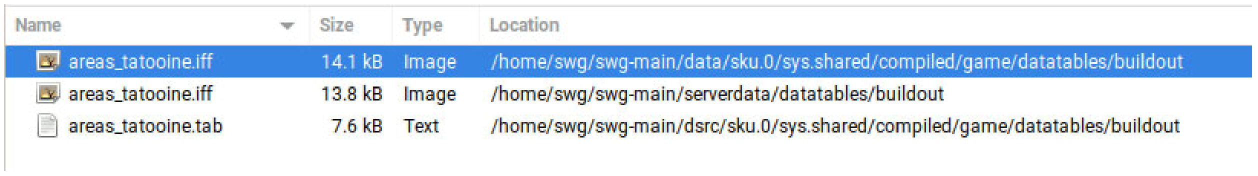
```
43          String halloweenRunning = getConfigSetting("GameServer", "halloween");
44          String lifedayRunning = getConfigSetting("GameServer", "lifeday");
45          String lovedayRunning = getConfigSetting("GameServer", "loveday");
46          String empiredayRunning = getConfigSetting("GameServer", "empireday_ceremony");
47          String vendorDemoRunning = getConfigSetting("GameServer", "vendor_demo_event");
```

Note that these checks do appear in more than one place; search the script for them to get a clear picture.

Finally, leave the OnHearSpeach method alone. In general, it does not work correctly. You will turn your vendors on or off in a server restart, not talking to the holiday controller.

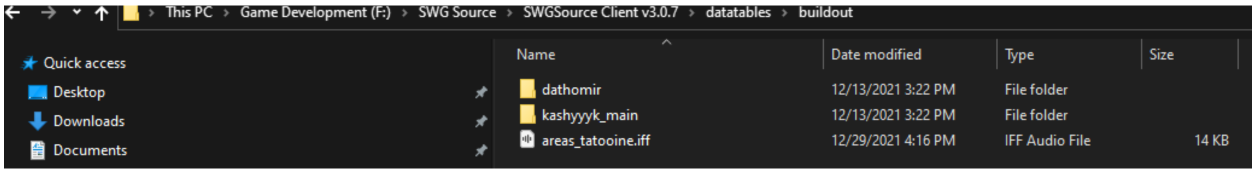
Rebuild and Copy

Now rebuild, and then find the area file and copy it to the client. In this example, you will find the area file in the following location on the VM:



Name	Size	Type	Location
areas_tatooine.iff	14.1 kB	Image	/home/swg/swg-main/data/sku.0/sys.shared/compiled/game/datatables/buildout
areas_tatooine.iff	13.8 kB	Image	/home/swg/swg-main/serverdata/datatables/buildout
areas_tatooine.tab	7.6 kB	Text	/home/swg/swg-main/dsrc/sku.0/sys.shared/compiled/game/datatables/buildout

Copy that file to the following location on the client:



This PC › Game Development (F:) › SWG Source › SWGSource Client v3.0.7 › datatables › buildout

Name	Date modified	Type	Size
dathomir	12/13/2021 3:22 PM	File folder	
kashyyyk_main	12/13/2021 3:22 PM	File folder	
areas_tatooine.iff	12/29/2021 4:16 PM	IFF Audio File	14 KB

Turn on the Event

To activate the event, add vendor_demo_event=1 or vendor_demo_event=true to your localOptions.cfg, as you would any other event. The event will start when you restart the server, and the vendors will respawn after the delay you specified when you updated the holiday controller.