Creating a new weapon for SWAT 4

Shawn Robertson aka: IG_Pappy
Weapon, FX, character artist/animator
Irrational Games, Boston

Disclaimer: I work in 3D studio Max, I know this will disappoint a few of you out there. All instructions are for Max users. I have never used Maya, so I can't instruct Maya users on which plug-ins to get, or which orientation is correct for mesh exports. I wish I could, and hopefully someone out there can add some Maya instructions to this document!

I am writing this on my own time, in hopes of helping the SWAT mod community with adding new weapons to SWAT 4. As a result, this isn't an "official" SWAT 4 document that will have continued support through official channels. Any questions should be directed to the VUG forums, and I will answer them when I can.

If you need ActorX help, visit the Unreal website. I am also assuming that most modders will have a basic understanding of editing in Unreal. If not, I again point you in the direction of Unreal's website.

We use DXT texture compression in SWAT. For the plug-ins appropriated to the program you are using to export DXT textures in .dds format, see nVidia's website.

These are also instructions that I more or less followed to add weapons to the shipping version of SWAT 4. I am unsure of how exactly the mods will work in SWAT 4, so you may want to create new files, instead of editing shipping files. I'll leave it up to the more programmer savvy types to figure out what might happen if you have a system file mismatch when trying to play on-line.

Files Provided:

ActorX.dlu	-3D Max plug-in for exporting animation files (.PSA (animation), .PSK (mesh))
Hand_withHandgun.max	-example Max file
Hand_withRifle	-example Max file
HandgunScaleExample	-example Max file
RifleScaleExample	-example Max file
HandUV.tga	-texture map for hands

New Asset Checklist:

New content:
1st Person Weapon Model
1st Person Weapon Textures
1st Person Weapon Animations (if necessary)
1st Person Hand Animations (if necessary)
1st Person Weapon FX
3rd Person Weapon Model
3rd Person Weapon Textures
3rd Person Weapon Animations (if necessary)
3rd Person Character Animations (if necessary)
3rd Person Weapon FX
GUI weapon image
GUI ammo image (if necessary)

Files to be edited within the SWAT4 editor:

In Content/Classes/:
SwatEquipmentModels.u
SwatEffects.u (if new effects created)
SwatAmmo.u (if new ammo added)

In Content/Animations/
FP_Hand.ukx (if necessary)

Files to be edited with Content/System/IGEffectsConfigurator.exe

In Content/System/:
VisualEffects.ini
SoundEffects.ini

Files to be edited with a text editor:

In Content/System/:
SwatEquipment.ini

Step by step instructions:

1st Person Weapon Model and Textures

Create a model of the weapon you want. A scale example has been included in this package for a handgun and an assault rifle. At this point, make the weapon look good; we'll get to syncing next. If the weapon is going to animate, this is the time to set your hierarchy up. Make the gun body, or handgrip, the parent.

Triangle Count: <1000
Texture Size: 512x512
I usually had a diffuse with alpha (for specular mask), and a normal map.
Orientation: Muzzle is facing Positive X (3Dmax).

Save the texture as a .dds file, and import into Unreal. See Unreal's website to learn how to create materials in a texture package.

1st Person Weapon and Hand Animation

Bring your new weapon into the provided Hand_withXXX.max. Position the weapon in the hands. When the position is correct, attach the weapon to the R_Grip dummy on the right hand. Now you can start animating, if you need to. The pose provided should be the generic aiming forward position. In this position, the gun should be aiming straight forward.

The generic aiming forward pose will be the last frame of equip, the first frame of unequip, the first frame of fire, etc…

NOTE: The 9mm pistol model in game uses the same hand animations as the 1911, but uses different weapon animations. All grenades in SWAT 4 use the same hand animations. If you are creating a new handgun, it might fit right into the already created 1911 hand animations. You can test this out pretty quickly in-game, see the notes section at the end of this document for instructions.

1st Person Weapon Animation
The following animations are supported for weapons and hands in SWAT 4.
Idles –animation for when weapon isn't in use.
Idle Accents –adds flavor to idles, can be weighted to play less than main idle, optional.
Equipping –bringing the weapon into view.
Unequipping –putting the weapon away.
Fire –pull the trigger.
FireBurst –pull the trigger in auto mode.
FireEmpty –pull the trigger, click!
FireLast –for weapons that change state when empty (like the 1911's slide staying back when empty).
ReloadFull –reloading with a bullet still in the chamber.
ReloadEmpty –reloading when weapon is empty.
LowReady –single frame, weapon lowers for team member or proximity to wall.
CantShoot -for multi-player when you are LTL and can't fire, single frame.

Things to remember when animating:

The camera and aim-spot set up in the Max file gives you a ROUGH idea of what you will see on screen.

I provided a handgun and rifle pose for the hands, to get you started on rough position.

Most animations in SWAT 4 are at 30fps.

The 1st person and 3rd person animations have to be the same length! (weapons, hands, characters)

I export Fire and FireBurst as the same animation. In the ActorX panel, you can set the fps of the animation. Set the FireBurst to a higher fps for a smoother firing animation. i.e. 60fps. This way, single shots don't look so jittery, and auto-fire can keep up with real world rounds per minute numbers.

FireLast will play when the weapon plays the fire animation when it only has one bullet left. This is optional, as not all weapons are visibly empty. If this is left blank, the gun will play the regular fire animation.

ALL animations are optional for the weapon. If no animations exist for the actual weapon, the weapons base pose will be used. You can also have animations for some things and not others. Most weapons don't have an equip animation, but will have a reload animation.

Exporting 1st Person Animations

When you are ready to export the 1st Person animations, you are going to need to separate the weapon from the hands. I always keep a Max file handy with both connected, so I can rework animations, but I also have my export files.

Hands
For the hands, delete the entire weapon, keeping the R_Grip dummy in the scene. Export each animation via ActorX into a .psa file. Call the .psa file something easy to remember, like **M4A1hands** you'll need this name later. If you have animated the R_Grip dummy, keep the keyframes in this file.

Weapons
When the hands are in their base pose, delete everything except the weapon and the R_Grip dummy. The weapon should be facing forward on positive X. Select R_Grip and move it to 0,0,0 in the world. Now you can export the skeletal mesh file (.psk). Use this same position to export any animations the weapon may have. Make sure the animation length is the same as the hand. I use the naming scheme **FP_weapon** for exporting 1st person weapons. Name the .psk and .psa file the same name. If you have animated the R_Grip dummy, delete the keyframes for this file.

At this point, you should have a 1st person hands animation file, a 1st person weapon mesh file, and a 1st person weapon animation

file.

Importing the 1st Person Weapon Mesh and Hand Animations

Import the animations and the weapon mesh with it's animations into Unreal. Now would be a good time to apply your shiny, new texture to the new weapon. Make sure to link the animations to the mesh, or you will never see them play! (blue chain link button in animation browser)

If you have created a new animation package for your hands, you will need to create a socket for the hands that the weapon will attach to.
-Open the "mesh®attach" tab on the right.
-Add a socket.
-Attach alias can be anything you want, i.e. FredsAK74suSocket.
-BoneName should be R_Grip.

To see in the animation browser how your hands line up with the weapon, attach the weapon to the hands:
-Open the hands animation package.
-Open the "mesh®attach" tab on the right.
-Find the appropriate socket. If you haven't done anything crazy, try **1911Socket** as a default.
-Find the field for "TestMesh".
-Put in the skeletal mesh data for your weapon. For example, the 9mm pistol would be: **SkeletalMesh'FP_9mmPistol.FP_9mmPistol'** the naming scheme is: SkeletalMesh'PackageName.MeshName' as seen in the animation browser.

If the weapon doesn't line up the way you want, you can play with the rotation and translation of the socket, or go back to Max and re-calculate how the hands and weapons went together. If you need to rotate the socket, make a new one, so as not to mess up existing weapons.

You won't see the weapon animate in the animation browser when it is an attachment to the hands. This is only good for initial weapon and hand synching.

If everything is good to go, you are ready to set up the class files. Skip to that section if you aren't dealing with 3rd person animations and models.

3rd Person Weapon Model and Textures

Like the 1st person model, create your 3rd person weapon. Most 3rd person weapons in SWAT 4 are static meshes, but a couple, like the Colt 1911 do animate. SWAT 4 can handle both situations. 3rd person examples can be found in the Max weapon scale files I have provided. At this point, make the weapon look good; we'll get to syncing next. If the weapon is going to animate, this is the time to set your hierarchy up. Make the gun body, or handgrip, the parent.

Triangle Count: <500
Texture Size: 256x256
Orientation: Muzzle is facing Negative Y (3Dmax).

Save the texture as a .dds file, and import into Unreal.

If your weapon isn't animated, export it as a .ASE file that you can import into Unreal. This will be helpful to line up character animations.

3rd Person Weapon and Character Animation

SWAT 4 comes with generic weapon aimposes and animations that will be appropriate for some weapons. If you export your weapon as an .ASE file, and import it into a static mesh package, you will be able to see it in a characters hand in the animation browser.

If you aren't animating the 3rd person weapon, export it as an .ASE file into Unreal. This would be the time to set up your textures, too. Save the static mesh package.

Open the animation browser and open SWATMaleAnimation. Under the mesh tab, open the GripRhand socket. Under TestStaticMesh, enter your weapon's info. i.e. The 3rd person AK47 would be written as: **StaticMesh'SwatGear_sm.MG_ak47'**, the naming scheme being StaticMesh'PackageName.MeshName'. With this, you can test various sockets and animations to see if the weapon you created fits within an animation set. For example, the AK47 static mesh will fit the MG (Machine Gun) animation set. All animations that end with the letters MG will be correct for the AK47. The 1911 will be correct for the HG (Handgun) set. The M4A1 has a unique set, and fits all animations ending with M4A1.

SG is shotgun
SMG is Sub-Machine Gun
Grenade is Grenades
TA is Tactical Aid
Spray is Pepperspray
OW is Optiwand
UMP is the 45 SMG

If your new 3rd person weapon fits within an existing socket and animation set, then skip the rest of this section.

Aimposes
The first set of animations that a new weapon will need is aimposes. These are single frame animations that show the character

aiming a weapon in space. They cover standing and crouching positions, leaning and not leaning, and low ready positions. In the file SWATofficer_withRifle.max provided, there is a protractor that shows the angles you are going to want to animate. For aimposes, you are going to want Center, High, Low, Left, Right for multiple stances.

SWATofficer_withRifle.max example aimpose list:

frame	Animation Name
0	AimCenter_M4A1
1	AimHigh_M4A1
2	AimLow_M4A1
3	AimLeft_M4A1
4	AimRight_M4A1
5	
6	LeanLeftAimCenter_M4A1
7	LeanLeftAimHigh_M4A1
8	LeanLeftAimLow_M4A1
9	LeanLeftAimLeft_M4A1
10	LeanLeftAimRight_M4A1
11	
12	LeanRightAimCenter_M4A1
13	LeanRightAimHigh_M4A1
14	LeanRightAimLow_M4A1
15	LeanRightAimRight_M4A1
16	LeanRightAimRight_M4A1
17	
18	LowReady_M4A1
19	LowReadyLookHigh_M4A1
20	LowReadyAimLow_M4A1
21	LowReadyLookLeft_M4A1
22	LowReadyLookRight_M4A1
23	
24	LowReadyMP_M4A1 -this is the extreme low ready to help prevent guns sticking through doors.

When animating the aimposes, don't animate anything below the spine. These aimposes are only applied to the upper body in-game.

You will also need a crouching set of aimposes. Crouch base is found on frame 30.

When exporting an aimpose via ActorX, use the "Animation Range" field to limit the animation to 1 frame, i.e. 0-0 or 12-12 in the animation range will only export 1 frame of animation

Actions
Actions are things you perform with the weapon, equipping, loading, etc.

Example Actions for the M4A1:
equipM4A1
unequipM4A1
reloadM4A1full - reloading with a bullet in chamber
reloadM4A1empty - reloading with no bullets in chamber
fireM4A1
fireM4A1auto - 60fps animation for auto fire only.

Again, make sure that any 3rd person animations match their 1st person counter-parts. 33 frames for a reload in 1st Person means 33 frames for a reload in 3rd person.

When importing these aimpose animations, they should be imported to the SWATMaleAnimation package.

Fidgets
You won't need fidgets if this is a multi-player only weapon. Aimposes are used during MP when a player isn't moving.

Export these animations into an animation package. I split them up into Aimposes and Actions, but for a mod, you may want everything in one group within an animation package.

3rd Person Weapon Animation
The 3rd person Colt 1911 is animated in game, mostly because I wanted to see when someone ran out of bullets in a MP game. Most weapons aren't animated in their 3rd person model, but it is possible.

Similar to animating the weapon in 1st person, the 3rd person weapon needs to be synced to the hands and exported correctly. The provided Max file, HandgunScaleExample.max has a group called 3rdPerson1911(animated) in it that shows the correct orientation and position of a 3rd person weapon to be exported. I don't have any examples of animated rifles. If an AI will use this weapon in a single player game, you will need a static mesh version of it. This will be explained in the **Classes** section.

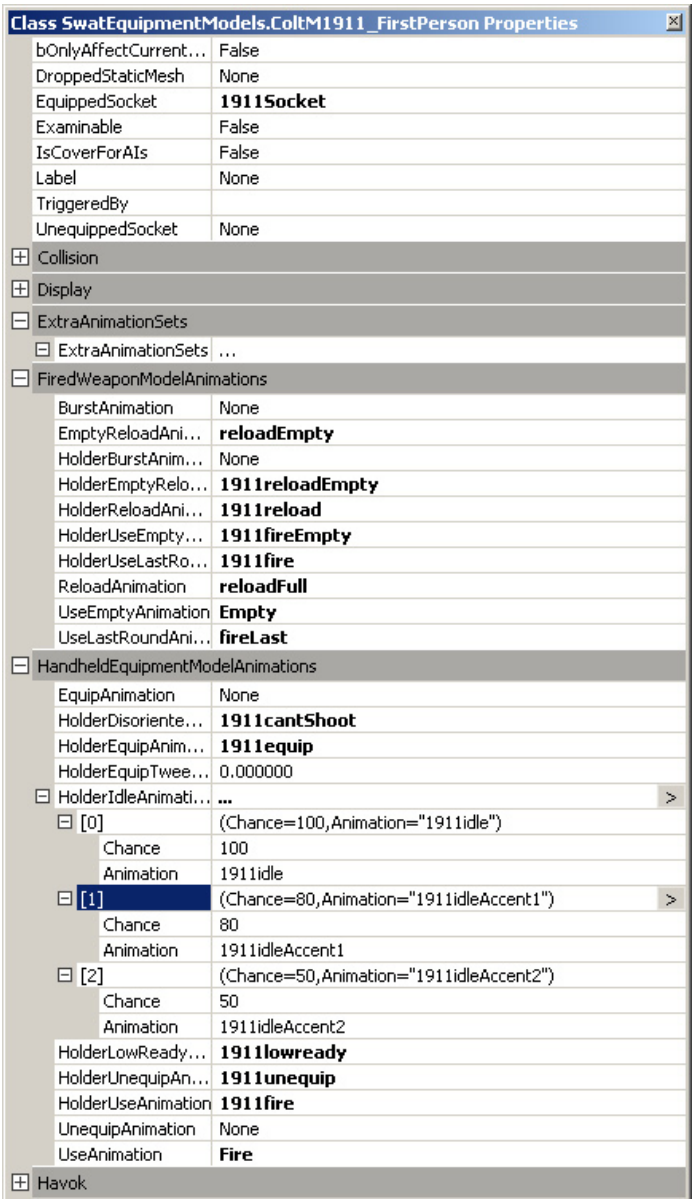
You don't need to do all animations for the gun. If there is no movement on a particular action, the gun will stay on the last frame of the last animation it played.

Classes

Classes are the files that tell the weapons how to behave. For setting up a weapon model class, you will need to have your 1st and 3rd models and animations imported into Unreal.

Any weapon that is going to be used by a person in a MP game will need a 1st and 3rd person weapon model class set up for it. Any weapon that is AI only in a SP game, will only need a 3rd person class set up.

1st Person Class
Open the SWAT 4 editor. Click on the Class Browser icon (Chess Pawn) up top to open the class browser. Within the class browser, open SwatEquipmentModels.u. This is the file that contains all the SWAT 4 weapons. Again, I don't know if you want to edit this, or create a new .u file for your particular mod. For examples, I will use the Colt1911_FirstPerson class. If you double click on this file, the properties will come up.



Not shown in this image is the Display Properties. In this case the 1911 is a mesh (it is animated) and points to SkeletalMesh'FP_coltHG.1st_1911' which is the 1st person animation file I created for this gun. If the gun were a static mesh the mesh field would be set to "none", and the StaticMesh field would be set to the correct static mesh.

This is where you associate actions to animations. The Holder animations refer to the pawn, in the the case of a 1st person weapon, this would be the hands. In this example the holder animations are all prefixed with 1911, that how I labeled my hand animations within the 1st person hand model to organize all the motions.

Equipped socket is the attach point of the gun to the hand.

Dropped Static Mesh – In this example it is blank, but for a 3rd person animated weapon that will be dropped (dropping only happens in SP games with AI's), this will need to point to the static mesh version of the animated gun. This is for Havok collision, when the gun falls it is a rigid body, not a skeletal ragdoll.

FX

Weapon effects are created in the editor and saved in the SWATeffects.u package. They are accessed through VisualEffects.ini, which, in turn is edited via the IGEffectsConfigurator.exe.

I create separate FX for 1st and 3rd person effects. The weapons are oriented differently, and have some size variation. To create an effect, I suggest placing an existing effect emitter in a test level with your new weapons. Use an emitter that is appropriate to the weapon you created. i.e. The 1911 muzzle effects would be a good start for a Beretta or USP40, where the M4A1 effects would be a good start for a Steyr Aug.

Place the emitter and weapon in the level without applying rotation to either of them. Place them in the exact same coordinates. Start editing the emitter to line the muzzle flash up with your new weapon. If you need help with Unreal emitters, please visit their website!

Once you have an effect that you are happy with, give it a new name in the Label field, and save it into either the SwatEffects.u file, or a new .u file.

To do this, have the emitter selected and choose "New Sibling From Viewport Selection" in the class browser. This will add the emitter to the class file you have open. Make sure that once the emitter is in the class file that the global settings are AutoDestroy=True and AutoReset=False, or the emitter will keep going off once triggered.

Once you have your 1st and 3rd person emitters saved, you are ready to move on to the next step.

IGEffectsConfigurator.exe

Open this executable. This helps you edit sound and visual effects in SWAT4. What you do in this program will affect VisualEffects.ini and SoundEffects.ini. For now we are concerned with Visual Effects only. The bottom of the screen has a line that reads "Effects Subsystem" with a drop box beside it. Set the drop box to "Visual" to narrow down the fields to display visual effects only.

Now you are ready to create a new event response. Click on New. In Source Class, write the name of the weapon model class you are setting up. For instance, **BobsBigGun_ThirdPerson**. Now you need to fill in the event. In this case we're going to use "**Fired**". Now when you fire the 3rd person weapon **BobsBigGun**, something happens, but we still need to tell the game what happens.

Go to the bottom center and create a new Effect Specification. You should name it something similar to what effect you are trying to create, **BobsBigGun3rdPersonMuzzleFlash**. Give it a chance of 100.

Now on to the third column. This is where you tell the event response what special effect to use when it knows that the gun has been fired. For these weapon FX, leave the MVT setting to Default. In the text field next to the MVT_Default box, type in the emitter name, i.e. **class'SWATeffects.BobsBigGun3rdPersonMuzzleFlash'**. This assumes that the emitter is saved in SWATeffects.u and the name of the emitter is BobsBigGun3rdPersonMuzzleFlash.

Save the file. Now the gun should call "fired" when it is fired and call for the effect.

If this gun is an automatic weapon, you will need to set up the "BurstFired" event response, too. BurstFired is called when a weapon is fired in auto or burst mode. If you set up a BurstFired event response, you can use the same effect for the muzzle flash, so you can use the same Effect Specification. Click "Add" instead of "New" and scroll down the list of available specifications and pick **Bobs BigGun3rdPersonMuzzleFlash**. The emitters will already be filled out.

If you want multiple event responses on one weapon, use the suffix field. i.e. You want a dynamic light tied to the fire event of BobsBigGun_ThirdPerson, but you don't want it to be in the same specification as the muzzle smoke, type "dynamicLight" into the suffix field. Now you can either use an existing specification for a dynamic light, or create a new one.

Make sure you set up the 1st and 3rd person effects for the weapon, then save the file. Test, and enjoy.

GUI art

If you want people to select your gun, you will need to create gui art for it. Current GUI images of guns are 512x256 dds images with no mip maps compressed to DXT1.

Gui ammo images are 256x256 dds images with no mip maps compressed to DXT1.

Code work needed

First, you need to have the UnrealScript files, which are provided with the SWAT SDK. Add a file to Game/SwatEquipment/Classes/Weapons/{NewWeaponName}.uc. You can use one of the existing weapons as an example. Just change the name of the class to the name of your new weapon. Then, from content/system, run "ucc make" to compile your new weapon. This will create a new SwatEquipment.u file in content/system.

To have 3rd person characters use the aimpose animations you've created, you'll need to first create a new animation set for those aimposes. Open SwatPawnAnimationSets.ini and look at the [AnimationSetSubMachineGun] section for an example. You should give your weapon's animation set a unique name., and replace the animation names with your new aimpose animations.

Next, to have the game load in your aimpose animation set, open Game/SwatGame/Classes/Animation/AnimationSetManager.uc. You'll need to add your animation set to the EAnimationSet enum, and add a CreateSet() call for your animation set in Construct().

In order to make characters use your aimpose animation set when your new weapon is equipped, open Game/SwatGame/Classes/SwatPawn.uc, and search for the GetEquipmentAimSet() function. This function has a series of conditionals that decides what animation set to use, based on the character's active item. You should add a specific if() statement for your new weapon, and return your new aimpose animation set if your new weapon is the active item.

Ammo, Ballistics, etc.

You will need to open the SwatEquipment.ini in your Content/System directory.

In this .ini you'll find all of the settings and variables for each weapon. If you don't like how one weapon is behaving here is where you can tweak it to meet your expectations.

For your new weapon you will need a new entry. You should Copy and paste an entry that already exists and modify it to match your new weapon.

So, first copy and paste the entry for a similar weapon. Then change the entry name to match the name of the class that you created for the new weapon. The entry name looks like this:

[SwatEquipment.PythonRevolverHG]

Change the portion after the dot to match the name of the class for the new weapon.

Then, change the first and third person model class entries to match the location of the new models.

If you created a weapon that doesn't use a caliber of ammo that one of the weapons in SWAT4 uses you will need to create a new ammo class for the weapon for complete accuracy. (See below) If you created a weapon that is the same caliber you should point the PlayerAmmoOption to the location of the ammo class that matches the caliber of the weapon you made.

SWAT4 shipped with the following calibers modeled:

9mm
.223 (5.56 mm)
.357 Magnum
.45
308 (sniper rifle)
00 buck
Shotgun Sabot

If your new weapon is one of these calibers you can copy the ammo class from one of the other weapon entries.

Tuning the Weapon

The most important number to get right when creating a new weapon is the muzzle velocity. All of the muzzle velocities in SWAT4 are based on the muzzle velocity of the real life counterpart. You should do the same if you want the weapon to feel right in comparison to the other weapons. If you don't know the muzzle velocity of the weapon you can usually find out on the internet with a few web searches.

Once you have found the real world muzzle velocity you need to convert it into Units that the engine understands. Here is the conversion formula:

X ft/sec * 15.385

NOTE: muzzle velocity must first be in feet per second.

Once you have this number plug it into the MuzzleVelocity in your weapons SwatEquipment.ini entry.

The muzzle velocity is the base number that the ballistics system uses to calculate the damage your weapon will do. If this is way off your weapon might be too lethal or hardly damage anyone at all.

Dissecting the SwatEquipment.ini entries.
This section attempts to describe what each entry in a weapon entry means.

PlayerAmmoOption - This is a list of the different types of ammo that the player is allowed to use in the weapon. It usually has two entries. One for the JHP and one for the FMJ.

EnemyUsesAmmo - This entry tells the enemy which ammo to use and what is the chance that it will use it. If you have multiple entries the chance values determine the Percent chance that a particular ammo type will be chosen by the enemy when it spawns. We always had them use FMJ or they would rarely hurt the SWAT team or player.

Range - The range of the weapon in Unreal units. Default is 4000. That is usually plenty long.

AvailableFireMode - There can be multiple entries for this data. This lists the available fire modes for a weapon. The first mode listed is the weapon's default mode. The supported modes are Firemode_Single, FireMode_Burst, and FireMode_Auto. You must add an AvailableFireMode entry for each firemode separately. If a mode is not listed the weapon doesn't use that mode.

BurstRateFactor - This is the rate at which to play the auto-fire animation for the weapon. If not listed it plays at a rate of 1. Numbers higher than one play it slower and numbers lower than one play it faster. The game was tuned with the AK47s rate of fire being equal to 1. All other weapons were tuned to play the autofire animation faster or slower than the AK to tune the Rates of Fire to be relatively accurate to each other.

BurstShotCount - If not listed the default burst shot count is 3. You can add this to any entry that doesn't have it listed or change this number to change the amount of shots per burst on a weapon that uses FireMode_Burst. (All weapons in SWAT that have burst fire use the default 3 round burst except the GB36s which uses a 2 round burst.)

ShouldHaveFirstPersonModel - This is a boolean (True or False). If the weapon is just for use with enemies/SWAT AI and not players you should set it to false.

OverrideArmDamageModifier - The arms kind of work like poor man's armor in the ballistics and body part targeting systems making it difficult to cause weapons to always deal lethal damage when shooting at the chest. (This is because you hit the arms that are holding the weapon.) This issue made shotguns feel really underpowered at close range. To solve this issue (in the 11th hour) we added this value. It is a damage modifier to make shots to the arm more potent. We only wanted to use this on the shotguns so we made it tunable on a per weapon basis. Use this field to override the arm damage modifiers from SWATSkeletalRegions.ini to make a weapon deal more damage to the arms than normal.

Description - This is the text description that appears in the Equipment screen

FriendlyName - This is the short name that appears in your loadout readout in the equipment screen and in the caption of the GUI image on the equipment screen. If the friendly name is too long these captions can get cut off.

GUIImage - This is the location of the 2D image for the equipment screen.

GUIPaperDollImage - This is legacy and is ignored.

EquipAnimationRate - How fast the equip animation is played. Affects how fast you can equip. Higher numbers mean faster equipping. Remember this is a RATE. 2 means twice as fast as the animation was built. 0.5 means half as fast as the animation was built.

UnequipAnimationRate - How fast to play the unequip animation

UseAnimationRate - How fast to play the Use animation (Fire animation for weapons)

ReloadAnimationRate - How fast to play the reload animation

ACCURACY:

All of the accuracy numbers are in degrees. This numbers affect the reticule in first person view. 0 degrees is perfect accuracy.

MaxAimError - This is the maximum number of degrees that the reticule can ever reach for this weapon.

LargeAimErrorRecoveryRate - The reticule recovers at two different rates. One for when it is large. Once it gets to a certain degree threshold (AimErrorBreakingPoint) it then uses a second rate to return to Base Accuracy. LargeAimErrorRecoveryRate is the rate of recovery from when the reticule is large down to the AimErrorBreakingPoint.

SmallAimErrorRecoveryRate - This is the recovery rate from the AimErrorBreakingPoint down to the base accuracy.

StandingAimError - This is the accuracy of the weapon when you are standing.

WalkingAimError - This is the accuracy of the weapon when you are walking.

RunningAimError - This is the accuracy of the weapon when you are running.

CrouchingAimError - This is the accuracy of the weapon when you are crouching.

Penalties are added to the current Aim error when a certain event occurs. These are assessed instantly to bloom the reticule.

EquippedAimErrorPenalty - Penalty added when you equip the weapon.

StandToWalkAimErrorPenalty - This is the penalty added to the accuracy when you go from standing to walking

WalkToRunAimErrorPenalty - This is the penalty added to the accuracy when you go from walking to running

DamagedAimErrorPenalty - This is the penalty added when you get damaged. This is zero on all weapons that ship with SWAT. If you want the weapon to have "Reticule knock" this is the number to tune. When you get shot your accuracy will bloom using this number. We kept the number at zero because when it was on it made it too difficult (read: not fun) to fight back if someone else got the first shot off.

FiredAimErrorPenalty - This number causes blooming when you fire the weapon. This helps make a weapon feel like it is harder to handle.

InjuredAimErrorPenalty - The InjuredAimErrorPenalty is a bit different than the other penalties. When you are shot in the arm and torso but don't die this number is added to your base aim errors. This DOES NOT recover. Increasing this makes your accuracy worse for every hit to the arm and torso.

MaxInjuredAimErrorPenalty - This is the maximum the InjuredAimErrorPenalty will accumulate to. Subsequent injury after the accuracy is already modifed to this number has no effect.

LookAimErrorPenaltyFactor - This number affects how fast you need to move your mouse to cause the reticule to bloom. This number is the same for every weapon and is very difficult to tweak. Change at your own risk.

MaxLookAimErrorPenalty - The largest penalty that will be added to the base aim error from looking around too quickly.

RECOIL

The recoil is modeled by moving the reticule up from the initial shot location and then returning it to that location at different rates.

RecoilBackDuration - This is how long in seconds it takes for the reticule to move up when you fire.

RecoilForeDuration - This is how long in seconds it takes for the reticule to return to the original position. It should be significantly longer than the back duration.

RecoilMagnitude - How much the reticule moves up when you fire. Not sure of the units, but 1000 is huge and 10 is too small to notice. Good numbers are usually between 100 and 500.

AutoFireRecoilMagnitudeIncrement - To get the feeling of the weapon getting wilder the longer you fire, the recoil magnitude gets larger for every shot after the first when firing in fully automatic or burst modes. This number is incremented to the regular recoil magnitude in a += fashion. So if the RecoilMagnitude is 100 and the AutoFireRecoilMagnitudeIncrement is 100, the magnitude will increase by 100 for every shot. Shot1 = 100, Shot2 = 200, Shot3=300, etc. This makes the recoil magnitude grow quickly the more you shoot if this value is high.

HasAttachedFlashlight - This is a Boolean (must be True or False). This just indicates if the weapon has a flashlight. If False, pressing the flashlight key does nothing. NOTE: The 1st and 3rd person models don't need to display a flashlight for it to have one, although it is recommended for feedback issues.

FlashlightPosition_1stPerson
FlashlightRotation_1stPerson
FlashlightPosition_3rdPerson
FlashlightRotation_3rdPerson - These are a bunch of location offsets so that the flashlight emits from the correct location on your first person and third person models

ZOOM

ZoomedFOV - This is the Field of View in degrees of the weapon while zoomed. Base FOV of the 1st person is 85. Smaller than 85 looks zoomed in, larger than 85 creates a "Fish Eye" lens effect.

ZoomTime - The amount of time that it takes for the weapon to transition from the base FOV to the Zoomed FOV.

Creating Ammo

If you are making a weapon that has a new type of ammo than those in SWAT you need to make new ammo if you want to be completely accurate. If you don't care you can just tell the gun to use one of the ammos that shipped with the game, although this might make the weapon tuning a bit out of whack.

To create new ammo you must open the SWAT editor.

Click on the class browser from the tool bar.

Click on the Open button and choose SWATAmmo.u

On the toolbar there are 3 buttons that display circles. One is filled in green, one is empty and one has a Red box around it. Make sure the two left hand circle buttons are depressed. (everything but the RED one).

You should be seeing the list of all of the ammo types in the game. If you are seeing a monumentally huge list, make sure the "All" button is not depressed in the browser tool bar.

If you are making a new caliber of FMJ or JHP right click on the entry for Full Metal Jacket or Jacketed Hollow Point. If you are making ammo of another type just right click in the empty space in the browser window.

Choose "New" from the menu.

A pop-up appears. Enter the name of the new ammo you want to create. Click OK.

Your new ammo will appear in the list. If you added it by right clicking on Full Metal Jacket or Jacketed Hollow Point the new ammo will appear as a sub-class and you will need to expand the entry by clicking on the + sign to see it.

Once the ammo is created you need to set up some numbers in it. If you made a new type of JHP or FMJ many of the numbers will already be filled in since the new class inherits from the parent.

Tuning the Ammo

Right click on the ammo and select "Default Properties..."

This brings up a window with a ton of properties. You don't need to do anything with most of these properties. The important ones are discussed below.

Expand the "Ammo" section. You will see the data important to ammo.

Clip size - this is how many bullets are in a magazine of this ammo. If you want different size magazines for a weapon you need separate ammo for each size.

Default Enemy Clip Count - How many clips an enemy has when using this ammo in his weapon

Default Officer Clip Count - How many clips an officer or player has when using this ammo in his weapon.

Description - The text that appears in the equipment screen to describe this ammo.

Friendly Name - The short name of the ammo that appears in the loadout display and as the captions to the GUI Image

GUI Image - Location of the texture where the 2D GUI image lives.

Impact Momentum - This number does nothing. Ignore it.

Internal Damage - This number is added to the damage when a bullet gets buried in the victim. This number is used to make JHP more damaging to unarmored targets. It basically represents the bullets tendency to shatter and spread when entering the body. It also can be a good fudge factor if you need one.

Shots per Round - This only really applies to shotguns and represents how many pellets are in a shotgun round. You could tell any weapon to fire multiple shots with every shot, but that's not very realistic is it?

The only other important number is located in the **Movement** section. Expand the Movement section from the big list.

Mass is the only value to change in here. This represents the mass of the bullet. Together with the muzzle velocity of the weapon the bullet's momentum is calculated. The momentum is the most important determining factor of the damage a weapon will deal. Momentum also determines the bullet's ability to pierce objects and body armor.

To determine the correct mass you must first find out how much the bullet you are modeling weighs in real life. Convert that weight into Kilograms. (Note: 15432.4 Grains = 1kg or 1grain = 0.0000648 kg)

Once you have the kg mass of the bullet enter it in the Mass field.

Making the ammo usable
There is one final thing that must be done to get the ammo working in the game. You must open the SWATEquipment.ini file and tell the game that the ammo is a viable choice for Primary and Secondary weapons. This is done in the section regarding PocketSpecs.

Search the ini file for the string Pocket_PrimaryAmmo. We will make your ammo work in Primary weapons.

You will need to add an entry for the new ammo.

An entry consists of the EquipmentClassName, Validity, and bSelectable.

Add an entry for EquipmentClassName and set it equal to the name of the class that you create. Be sure to put SwatAmmo. in front of the name. This tells the game which class package the ammo lives in. So a proper entry looks like this:

EquipmentClassName=SwatAmmo.UMP45SMG_FMJ

Then add Validity under that line and set it equal to the Netmode you want to use it in. If you want it to be only available in SinglePlayer type "NETVALID_SPOnly", if you want MP only type "NETVALID_MPOnly", if you want both type "NETVALID_All" (No quotes)

A proper entry looks like this:

Validity=NETVALID_All

Finally, mark if it is selectable by setting bSelectable to True or False. You almost certainly want this to be true.

A complete entry looks like this:

EquipmentClassName=SwatAmmo.MP5SMG_JHP
Validity=NETVALID_All

bSelectable=True

Copy your completed entry (all 3 lines) into the Pocket_SecondaryAmmo section. This will make it viable for use in back up weapons.

Example section from SwatEquipment.ini

```
;*****************************
;COLT M1911
;*****************************

[SwatEquipment.ColtM1911HG]
;FiredWeapon configuration
PlayerAmmoOption=SwatAmmo.ColtM1911HG_JHP
PlayerAmmoOption=SwatAmmo.ColtM1911HG_FMJ
EnemyUsesAmmo=(AmmoClass="SwatAmmo.ColtM1911HG_JHP",Chance=100)
Range=4000
MuzzleVelocity=12770
;HandheldEquipment configuration
FirstPersonModelClass=class'SwatEquipmentModels.ColtM1911_FirstPerson'
ThirdPersonModelClass=class'SwatEquipmentModels.ColtM1911_ThirdPerson'
Description="The Colt M1911 Handgun has been a mainstay in law enforcement for decades. This .45 caliber weapon packs a powerful punch and is the most common backup weapon for SWAT officers. This semi-automatic weapon has an 8 round magazine and can fire both full metal jacket and jacketed hollow point ammunition."
FriendlyName="M1911 Handgun"
GUIImage=material'gui_tex.equip_1911'
EquipAnimationRate=1
UnequipAnimationRate=1.5
UseAnimationRate=1.0
ReloadAnimationRate=1
;Since zero is perfect AimError, Max AimError is the worst AimError
MaxAimError=60.0
;AimError recovered per second until base AimError is achieved
LargeAimErrorRecoveryRate=10.0
SmallAimErrorRecoveryRate=1.0
AimErrorBreakingPoint=1.0
StandingAimError=0.75
WalkingAimError=0.95
RunningAimError=6.5
CrouchingAimError=0.5
;Instantaneous penalties
EquippedAimErrorPenalty=5.0
StandToWalkAimErrorPenalty=2.0
WalkToRunAimErrorPenalty=5.0
DamagedAimErrorPenalty=0.0
FiredAimErrorPenalty=0.75
InjuredAimErrorPenalty=0.5
MaxInjuredAimErrorPenalty=1.0
;AimError penalty for one unit of mouse movement per second
LookAimErrorPenaltyFactor=0.000125
MaxLookAimErrorPenalty=7.0
RecoilBackDuration=0.02
RecoilForeDuration=0.45
RecoilMagnitude=400
HasAttachedFlashlight=true
FlashlightPosition_1stPerson=(X=1,Y=-1.5,Z=5)
FlashlightRotation_1stPerson=(Pitch=0,Yaw=0,Roll=0)
FlashlightPosition_3rdPerson=(X=0,Y=-11.8,Z=1.9)
FlashlightRotation_3rdPerson=(Pitch=0,Yaw=-16384,Roll=0)
;Weapon Zoom
ZoomedFOV=60.0
ZoomTime=0.5
```

You will need to modify the [Pocket_PrimaryWeapon] or [Pocket_SecondaryWeapon] sections in SwatEquipment.ini to refer to their new weapon so that it can be selected in the GUI.

NOTES

VisualEffectsConfig.u and SoundEffectsConfig.u are built by the game when it starts, not by the configurator. Basically, modders shouldn't worry about changing them.

Quick prototyping:
If you have a model and texture you want to see in-game quickly, follow these instructions.

Export the gun as a static mesh. Use the orientation of the weapons provided to properly align your gun. If it's a 1st person weapon, use the 1st person model as a template, same goes for 3rd person.

Import the model and textures into Unreal, set up your materials, and save out the static mesh package.

Find a weapon model class of a gun similar to yours and open up the class file. Under display, switch the display to be your weapon, save out the class file.

Open the game, and in the loadout screen, pick the weapon you edited and load up a level. When you equip the weapon, you should see your model.

This is a quick and easy method to see how your gun will look in game. JUST BE SURE TO BACK UP ALL FILES YOU EDIT!!!!