Unit File

This is an explanation of how units work in Payday 2. This may not be 100% perfect, but it should give the basic idea of what's going on.

A "unit" in Payday 2 is an object like a mask or an enemy. It basically sets properties, applies effects, and specifies textures for the units.

Files

- **.unit** - Points to a *.object file for further loading data about the unit. Specifies dependencies like sounds, effects, or animations. Establishes the extensions, which consist of properties for the unit like inventory, AI, damage, movement, sounds, or interactions. Sometimes can specify a path for the .unit file that is supposed to be loaded when the unit is spawned over network. And finally can specify the location of sound sources for the unit.
- **.object** - Points to a **.material_config** file, **.sequence_manager** file and the animations. For some units, there are "bodies" that establish which parts of bodies are enabled, and sets properties for them like friction, collision class, and a template. Then, there are "constraints", which mostly establish how far limbs can rotate. After that, "decal_surfaces" are established, which basically assigns a specific decal material to parts of bodies. And finally, "graphics" are specified, they set which parts of model are enabled, and sometimes sets a LOD for them depending on distance from the unit.
- **.model** - This is the model for the unit.
- **.material_config** - Contains a list of materials with specified templates that are applied to the textures. Points to *.texture files to use for reflections, diffused textures and normal maps.
- **.cooked_physics** - Not much is known about this file but it is related to physics of the unit.
- **.sequence_manager** - This file contains sequences which can be run on the unit.
- **.texture** - This is a renamed **.dds** texture file.

Loading Order

Units in Payday are loaded in a specific order. If a file in this chain has an error, the unit will not work correctly and potentially crash the game.

The loading order is as follows: .unit file is loaded first (this file has to point to the .object file to load it next), next the .object file is loaded (this file often points to the .material_config file, which contains textures for the model to use), next either .model or .material_config is loaded (.material_config has specified texture files that it loads and applies a specific template with effects to them. And .model contains the model and retrieves materials from the .material_config).

.unit Details

The .unit file is the first to get loaded when accessing the unit. This file is in .xml format, with a fairly simple structure. Most of the things in these .unit files are pretty self explanatory. However, the weapon units, mask units, and character units are all different. As they are obviously none of them are the same at all! Here is a sample of a .unit file.

Sample Breakdown of a .unit file

As an example, I will be looking at the cloaker's (spook's) unit files. First, the .unit file of a cloaker are located at (units\payday2\characters\ene_spook_1\ene_spook_1.unit).

```
<unit type="being" slot="12">
  <anim_state_machine name="anims/units/enemies/cop/cop_machine" />
  <object file="units/payday2/characters/ene_spook_1/ene_spook_1" />

  <dependencies>
    <depends_on animation_state_machine="anims/units/enemies/cop/cop_machine"
animation_def="anims/units/enemies/cop/cop_def" />
    <depends_on bnk="soundbanks/regular_vox" />
    <depends_on effect="effects/particles/character/cloaker_goggle" />
    <depends_on unit="units/payday2/characters/ene_acc_baton/ene_acc_baton" />
  </dependencies>

  <extensions>
    <extension name="unit_data" class="ScriptUnitData" />
    <extension name="base" class="CopBase" >
      <var name="_tweak_table" value="spooc" />
      <var name="_default_weapon_id" value="mp5_tactical" />
    </extension>
    <extension name="inventory" class="CopInventory" />
    <extension name="brain" class="CopBrain" />
```

```xml
      <extension name="anim_data" class="PlayerAnimationData" />
      <extension name="character_damage" class="CopDamage">
        <var name="_head_body_name" value="head" />
        <var name="_death_sequence" value="kill_spook_lights" />
      </extension>
      <extension name="movement" class="CopMovement" >
        <var name="_footwear" value="boots" />
        <var name="_anim_global" value="cop" />
      </extension>
      <extension name="interaction" class="IntimitateInteractionExt" >
        <var name="tweak_data" value="intimidate" />
      </extension>
      <extension name="network" class="NetworkBaseExtension" />
      <extension name="damage" class="UnitDamage" >
        <var name="_skip_save_anim_state_machine" value="true" />
      </extension>
      <extension name="contour" class="ContourExt" />
      <extension name="sound" class="CopSound" />
    </extensions>

    <network sync="spawn" remote_unit="units/payday2/characters/ene_spook_1/ene_spook_1_husk"/>

    <sounds>
      <default_soundsource source="Hips"/>
    </sounds>
  </unit>
```

Please notice the structure: anim_state_machine, then object, then dependencies, then extensions, then network, then sounds. Sometimes, if this structure is not followed, the game will crash. (Note, Payday: The Heist sometimes does not follow this structure, so PD1 .unit files would often result in a crash).

Let's breakdown this file by sections.

```xml
<unit type="being" slot="12">
```

The first section states that this unit is a human being (I am assuming when you hit it, blood will come out). And that it's in slot 12 (I believe slot number can be ignored).

```xml
<anim_state_machine name="anims/units/enemies/cop/cop_machine" />
```

The second section establishes the animations that this unit will use. In this case, the cloaker will be using cop animations.

```
<object file="units/payday2/characters/ene_spook_1/ene_spook_1" />
```

The third line establishes where the next file is, the .object file. This file will be loaded after the .unit (Please note that the path does not contain an extension, the game already knows that you pointed at a .object file.

```
<dependencies>
    <depends_on animation_state_machine="anims/units/enemies/cop/cop_machine"
animation_def="anims/units/enemies/cop/cop_def" />
    <depends_on bnk="soundbanks/regular_vox" />
    <depends_on effect="effects/particles/character/cloaker_goggle" />
    <depends_on unit="units/payday2/characters/ene_acc_baton/ene_acc_baton" />
</dependencies>
```

This block of code establishes the dependencies that this unit has. This unit is dependent on the cop animations using the cop animation definitions. Next, this unit is dependent on the sound bank "soundbanks/regular_vox" (this soundbank is related to speech). Next, this unit is dependent on an effect, the "effects/particles/character/cloaker_goggle", in the .object file this effect will be assigned to a specific location. And finally, this unit is dependent on another unit "units/payday2/characters/ene_acc_baton/ene_acc_baton" (this counts as an enemy accessory, thus the name "ene_acc"). And once again, not a single path has an extension, the game knows.

```
<extensions>
    <extension name="unit_data" class="ScriptUnitData" />
    <extension name="base" class="CopBase" >
        <var name="_tweak_table" value="spooc" />
        <var name="_default_weapon_id" value="mp5_tactical" />
    </extension>
    <extension name="inventory" class="CopInventory" />
    <extension name="brain" class="CopBrain" />
    <extension name="anim_data" class="PlayerAnimationData" />
    <extension name="character_damage" class="CopDamage">
        <var name="_head_body_name" value="head" />
        <var name="_death_sequence" value="kill_spook_lights" />
    </extension>
    <extension name="movement" class="CopMovement" >
        <var name="_footwear" value="boots" />
        <var name="_anim_global" value="cop" />
    </extension>
```

```
    <extension name="interaction" class="IntimitateInteractionExt" >
      <var name="tweak_data" value="intimidate" />
    </extension>
    <extension name="network" class="NetworkBaseExtension" />
    <extension name="damage" class="UnitDamage" >
      <var name="_skip_save_anim_state_machine" value="true" />
    </extension>
    <extension name="contour" class="ContourExt" />
    <extension name="sound" class="CopSound" />
  </extensions>
```

This chunk of code assigns basic unit things, like AI, inventory, sounds, etc. Most of them stay the same, but variables change. The "unit_data" extension is present practically in every .unit file and does not seem to change. The "base" extension is usually present on characters or usable objects. In "base" the class changes depending on the unit, and the variables in the extension also change. For this unit, the variables set the identity of this unit as "spooc" and assigns it "mp5_tactical" as a default weapon. The "inventory" seems to be only present on units that can carry items (like ammo or other weapons). The class usually stays as "CopInventory", but there could be multiple kinds of "inventory". The "brain" assigns AI to the unit, in this case "CopBrain" is assigned. The "anim_data" is currently unknown to me, but I am guessing that these are the kinds of animations a unit can perform, "PlayerAnimationData" is assigned. The "character_damage" assigns various things regarding the damage the unit will take, "CopDamage" is assigned. Two variables are assigned as well, "_head_body_name" signifies what part of body (according to model) is considered to be the head, the "_deadh_sequence" signifies what sequence this unit will perform at death. The "movement" assigns what kind of movement this unit will perform, "CopMovement" is assigned, as well as two other variables. The first variable "_footwear" states what kind of shoes the unit will have, "boots" are assigned. Second variable "_anim_global" states what kind of movement animations this unit will perform, "cop" animations are assigned. The "interaction" is usually present with units that can be interacted with, I am lacking detail as to what kind of interaction, "IntimateInteractionExt" is assigned, with one variable. The variable "tweak_data" is pretty much always present with "interaction", "intimate" is assigned. The "network" extension is usually present on units that can be spawned or changed during the game, "NetworkBaseExtension" is assigned. The "damage" extension is usually present with units that can deal damage, "UnitDamage" is assigned with one variable. The variable "_skip_save_anim_state_machine" is related to animations and I am unsure about the exact usage of this, variable is set to "true". The "contour" extension is usually present with units that can have an outline, class of "ContourExt" is assigned. The "sound" extension determines what kind of sounds this unit can make, "CopSound" is assigned.

```
  <network sync="spawn" remote_unit="units/payday2/characters/ene_spook_1/ene_spook_1_husk"/>
```

This section is usually present with units that can be spawned or changed during the game. This is responsible for syncing the spawn, by specifying the .unit file to be loaded on the clients. (Lobby host does not use this, only the client).

```
<sounds>
  <default_soundsource source="Hips"/>
</sounds>
```

This section specifies the source of sound for the unit. Apparently the cloaker (and all other enemies) make sounds from their "Hips".

After the .unit file is loaded, the .object gets loaded. Not all commands were listed in this example, so other commands will be listed below in the "Other .unit commands" section with their explanations.

Other .unit commands

Other .unit commands will be added here as research progresses.

.object Details

The .object file is second to get loaded when accessing the unit. This file is in .xml format, with a fairly simple structure. Most of the things in these .object files are pretty self explanatory, and about 80% unique to the unit, as it heavily replies on the model and the names of body parts in the model. The .object file usually contains properties of model parts like materials, sequence_manager, bodies, constraints, effects, graphics, and lights. Here is a continuation of the unit breakdown from previous section.

Sample Breakdown of a .object file

Following from the previous section, the .object file path of a cloaker was assigned as "units/payday2/characters/ene_spook_1/ene_spook_1" (that's without the .object at the end). The .object files tend to be repetitive, as they assign each "body" in a model, specific settings. And for sake of space, the .object file will be cut down to include as little repetition as possible.

```
<dynamic_object>
  <diesel materials="units/payday2/characters/ene_spook_1/ene_spook_1" orientation_object="root_point" />
  <sequence_manager file="units/payday2/characters/ragdoll" /> <animation_def
name="anims/units/enemies/cop/cop_def" />

  <bodies>
```

```xml
    <body name="body" enabled="true" template="character" friction="0.6" collision_class="ragdoll">
        <object name="Spine1"/>
        <object name="c_capsule_body" collision_type="capsule"/>
    </body>
    <body name="mover_blocker" enabled="true" template="mover_blocker" keyframed="true"
collision_class="ragdoll">
        <object name="root_point"/>
        <object name="c_capsule_mover_blocker" collision_type="capsule"/>
    </body>
            ***PART OF THE FILE WAS SNIPPED HERE***


<!-- RAGDOLL -->
    <body name="rag_Head" enabled="false" template="corpse" friction="0.01" sweep="true"
collision_class="ragdoll" keyframed="false" collision_script_quiet_time="0.5" collision_script_tag="small"
ray="block" lin_damping="0.6" ang_damping="20" collides_with="0" tag="flesh" restitution="0">
        <object name="Neck" />
        <object collision_type="sphere" mass="4" padding="-15" name="c_sphere_head_ragdoll"/>
    </body>


    <body name="rag_Hips" enabled="false" template="corpse" friction="0.6" sweep="true"
collision_class="ragdoll" keyframed="false" collision_script_quiet_time="0.5" collision_script_tag="large"
ray="block" lin_damping="0.4" ang_damping="20" collision_group="1" collides_with="0" tag="flesh"
restitution="0">
        <object name="Hips" />
        <object collision_type="capsule" mass="22" padding="-5" name="c_sphere_Hips" />
    </body>
    ***PART OF THE FILE WAS SNIPPED HERE***
  </bodies>

  <constraints>
    <constraint type="ragdoll" name="RightArm" enabled="false">
        <param body_a="rag_Spine2" body_b="rag_RightArm"/>
        <param pivot="position:RightArm"/>
        <param twist_axis="yaxis:RightArm" twist_min="-60" twist_max="70" twist_freedom="20"/><!-- X axis --
>
        <param plane_axis="xaxis:RightArm"/><!-- Y axis -->
        <param cone_y="35" cone_z="40" cone_freedom="10"/>
        <param damping="1" spring_constant="200" min_restitution="0"/>
    </constraint>
```

```xml
        <constraint type="limited_hinge" name="RightForeArm" enabled="false">
            <param body_a="rag_RightArm" body_b="rag_RightForeArm"/>
            <param pivot="position:RightForeArm"/>
            <param min_angle="-60" max_angle="60" axis="yaxis:RightForeArm" twist_freedom="5"/> <!-- X axis -->
            <param plane_axis="xaxis:RightForeArm"/> <!-- Y axis -->
            <param damping="1" spring_constant="200" min_restitution="0"/>
        </constraint>
        ***PART OF THE FILE WAS SNIPPED HERE***
    </constraints>

    <decal_surfaces default_material="flesh" />

    <effects>
        <effect_spawner name="es_light" enabled="false" object="e_light"
effect="effects/particles/character/cloaker_goggle" />
    </effects>

    <graphics>
        <graphic_group name="character" enabled="true" culling_object="g_body">

            <lod_object name="lod_body">
                <object name="g_body" ⬜⬜enabled="true"  max_distance="3000"  max_draw_lod="0" />
                <object name="g_body_lod1" ⬜enabled="true" lod="1" />
            </lod_object>

            <object name="s_body" enabled="true" shadow_caster="true"/>

            <object name="g_il" ⬜⬜enabled="false" />

        </graphic_group>
    </graphics>

    <lights>
        <light name="point_light" enabled="false" multiplier="reddot" far_range="25" near_range="1" falloff="4.0"
type="omni|specular" />
    </lights>

</dynamic_object>
```

Please notice the structure: diesel materials, then sequence_manager, then animation_def, then bodies, then constraints, then decal_surfaces, then effects, then graphics, and then lights. Sometimes, if this structure is not followed, the game will crash. (Note, Payday: The Heist sometimes does not follow this structure, so PD1 .object files would often result in a crash).

Like before, Let's breakdown this file by sections.

```
<diesel materials="units/payday2/characters/ene_spook_1/ene_spook_1" orientation_object="root_point" />
```

This section specifies the location of the the .material_config file, along with the orientation position. This is present in most units that have a model (some weapons don't seem to specify this). The .material_config file is specified to be "units/payday2/characters/ene_spook_1/ene_spook_1" (once again, no .material_config, game knows) with the orientation at "root_point" of the model.

```
<sequence_manager file="units/payday2/characters/ragdoll" />` `<animation_def
name="anims/units/enemies/cop/cop_def" />
```

This section specifies what sequence_manager file to use. And what animations to use. (Sometimes these are separated into two lines). The sequence_manager file is specified to be located at "units/payday2/characters/ragdoll" (no .sequence_manager extension). And the "anims/units/enemies/cop/cop_def" animation definition is set to be used.

```
<bodies>
  <body name="body" enabled="true" template="character" friction="0.6" collision_class="ragdoll">
    <object name="Spine1"/>
    <object name="c_capsule_body" collision_type="capsule"/>
  </body>
  <body name="mover_blocker" enabled="true" template="mover_blocker" keyframed="true"
collision_class="ragdoll">
    <object name="root_point"/>
    <object name="c_capsule_mover_blocker" collision_type="capsule"/>
  </body>
            ***PART OF THE FILE WAS SNIPPED HERE***

  <!-- RAGDOLL -->
  <body name="rag_Head" enabled="false" template="corpse" friction="0.01" sweep="true"
collision_class="ragdoll" keyframed="false" collision_script_quiet_time="0.5" collision_script_tag="small"
ray="block" lin_damping="0.6" ang_damping="20" collides_with="0" tag="flesh" restitution="0">
    <object name="Neck" />
    <object collision_type="sphere" mass="4" padding="-15" name="c_sphere_head_ragdoll"/>
  </body>
```

```
　　<body name="rag_Hips" enabled="false" template="corpse" friction="0.6" sweep="true"
collision_class="ragdoll" keyframed="false" collision_script_quiet_time="0.5" collision_script_tag="large"
ray="block" lin_damping="0.4" ang_damping="20" collision_group="1" collides_with="0" tag="flesh"
restitution="0">
　　　<object name="Hips" />
　　　<object collision_type="capsule" mass="22" padding="-5" name="c_sphere_Hips" />
　　</body>
            ***PART OF THE FILE WAS SNIPPED HERE***
　</bodies>
```

This section pretty much defines collisions and ragdolls per body parts in the model. For the first
"body" is enabled (as it's set to true), the templace for "character" is used with friction of "0.6",
and a collision class of "ragdoll". This seems to include the object "Spine1" and "c_capsule_body"
of collision type "capsule". Both of those objects are most likely defined in the .model file. **THIS IS
NOT FINISHED!!!**

```
<constraints>
  <constraint type="ragdoll" name="RightArm" enabled="false">
    <param body_a="rag_Spine2" body_b="rag_RightArm"/>
    <param pivot="position:RightArm"/>
    <param twist_axis="yaxis:RightArm" twist_min="-60" twist_max="70" twist_freedom="20"/><!-- X axis --
>
    <param plane_axis="xaxis:RightArm"/><!-- Y axis -->
    <param cone_y="35" cone_z="40" cone_freedom="10"/>
    <param damping="1" spring_constant="200" min_restitution="0"/>
  </constraint>

  <constraint type="limited_hinge" name="RightForeArm" enabled="false">
    <param body_a="rag_RightArm" body_b="rag_RightForeArm"/>
    <param pivot="position:RightForeArm"/>
    <param min_angle="-60" max_angle="60" axis="yaxis:RightForeArm" twist_freedom="5"/> <!-- X axis -->
    <param plane_axis="xaxis:RightForeArm"/> <!-- Y axis -->
    <param damping="1" spring_constant="200" min_restitution="0"/>
  </constraint>
  ***PART OF THE FILE WAS SNIPPED HERE***
</constraints>
```

This section deals with constraints of rotations and movement. **THIS SECTION NEEDS MORE
EXPLANATION, BUT IS SELF EXPLANATORY!!!**

```
<decal_surfaces default_material="flesh" />
```

This little section states that the default material for the unit is "flesh". There are a few other default materials besides flesh, and sometimes they're even applied per body part in this section.

```
<effects>
 <effect_spawner name="es_light" enabled="false" object="e_light"
 effect="effects/particles/character/cloaker_goggle" />
</effects>
```

This section applies effects to the unit. I am not 100% certain on the application process. I believe that the effect under the name "es_light" is being applied to the object "e_light" (probably stated in .model) from effect file "effects/particles/character/cloaker_goggle" (once again, .effect extension is not needed).

The name of the effect does not matter; it can be set to anything you want. It seems to be only for referential purposes. The effect does not necessarily need to be applied to an "e_light" object, as other objects in the file will work as well (such as "g_body" or "root_point").

```
<graphics>
   <graphic_group name="character" enabled="true" culling_object="g_body">

     <lod_object name="lod_body">
       <object name="g_body"  enabled="true"  max_distance="3000"  max_draw_lod="0" />
       <object name="g_body_lod1" enabled="true" lod="1" />
     </lod_object>

     <object name="s_body" enabled="true" shadow_caster="true"/>

     <object name="g_il"  enabled="false" />

   </graphic_group>
</graphics>
```

In this section, there are two things happening. First, the LOD is being set per distance (in centimeters). So at distance < 3000 cm the default LOD model with be drawn to screen. If distance > 3000 cm then the LOD1 will be drawn. Second, some elements of the model are enabled/disabled in this section. As you can see, the "s_body" is enabled (with "true") and is set to cast a shadow with shadow_caster set to "true". And then there is "g_il" which is disabled. Please note that not all elements of the .model can be disabled here. Only the ones you know (i.e. the ones already listed in this .object file) or the ones you know from the model (currently there is no way of looking up element names from models.

```
<lights>
    <light name="point_light" enabled="false" multiplier="reddot" far_range="25" near_range="1" falloff="4.0"
type="omni|specular" />
</lights>
```

This section is for creating a light on the unit. For this unit in particular, it creates a glow around them that gets enabled via the sequence_manager. This "point_light" has a range of 1 - 25 with the falloff of "4.0" (I think the falloff is for the type) and type of "omni|specular". I am not certain about what this type specifically does, but it certainly acts as an effect on this "point_light".

After the .object file is loaded, either the .model or the .material_config file get loaded. Not all commands were listed in this example, so other commands will be listed below in the "Other .object commands" section with their explanations.

Other .object commands

Other .object commands will be added here as research progresses.

.model Details

Currently there are no details on the .models, as the filetype has not been completely reverse engineered.

Research Notes:

.model contains hashed names of objects using Hash64, uint64. (Currently looking into editing elements)

Bones have been redone since Payday:`` ``The`` ``Heist , they now don't include 4th elements of fingers. Each bone is specified as a 3D Object, which contains rotation matrix, position, and a parent ID.

Observations:
*It's near impossible to find model names, as they are hashed and unhashing them would be near impossible. They are not part of idstring.
*I'm assuming .material_config hashes the name of material, and applies it to the model. If it doesn't exist, it still applies (to nothing).
*It would be easier to create models from scratch, as you will know the names of all objects and materials, so you would have full control over the model.

.material_config Details

The .material_config file is loaded sometime after the .object file. This file is in .xml format, with a fairly simple structure. Most of the things in these .material_config files are pretty self explanatory, and is unique to the unit, as it heavily replies on the model and the names of body parts in the model. The .material_config file contains texture paths (diffused and bump map textures), sometimes reflection textures, sometimes material_textures, and sometimes some variables for the render_template. Here is a continuation of the unit breakdown from previous section.

Sample Breakdown of a .material_config file

In this example we will be using the cloaker. The .material_config file path of a cloaker was assigned as "units/payday2/characters/ene_spook_1/ene_spook_1" (that's without the .material_config at the end). The .material_config files tend to be repetitive, as they assign each the requested material names in the model, specific textures and effects. And for sake of space, the .material_config file will be cut down to include as little repetition as possible.

```
<materials version="3" group="ene_spook_1">
  <material name="mtr_body"
render_template="generic:DIFFUSE_TEXTURE:NORMALMAP:RL_COPS:SKINNED_3WEIGHTS" version="2">
    <bump_normal_texture    file="units/payday2/characters/shared_textures/spook_heavy_nm"/>
    <diffuse_texture     file="units/payday2/characters/shared_textures/spook_heavy_df"/>
  </material>
  <material name="mtr_il"
render_template="generic:ALPHA_MASKED:DIFFUSE_TEXTURE:OPACITY_TEXTURE:RL_COPS:SELF_ILLUMINATION"
version="2">
    <diffuse_texture     file="units/payday2/characters/shared_textures/spook_il"/>
    <self_illumination_texture   file="units/payday2/characters/shared_textures/spook_il"/>
    <opacity_texture     file="units/payday2/characters/shared_textures/spook_il"/>
    <variable      value="reddot" type="scalar" name="il_multiplier"/>
  </material>
    <material name="shadow_caster" render_template="shadow_caster_only:SKINNED_1WEIGHT" version="2"/>
  </materials>
```

There is no specific structure to follow. This file seems to be a list of materials with some variables attached. The only real problems that can occur are incorrect textures, broken model, no model at all (but a floating blob of gray).

Let's breakdown this file by sections.

```
<materials version="3" group="ene_spook_1">
```

This establishes the group that these materials belong to. I am not sure as to what group names can be given, but it's best to keep them similar to the original model names. The version does not seem to matter.

```
<material name="mtr_body"
render_template="generic:DIFFUSE_TEXTURE:NORMALMAP:RL_COPS:SKINNED_3WEIGHTS" version="2">
    <bump_normal_texture ⬜⬜file="units/payday2/characters/shared_textures/spook_heavy_nm"/>
    <diffuse_texture ⬜⬜⬜file="units/payday2/characters/shared_textures/spook_heavy_df"/>
</material>
```

This section identifies a material "mtr_body" with a render_template of "generic:DIFFUSE_TEXTURE:NORMALMAP:RL_COPS:SKINNED_3WEIGHTS" and version of "2" (once again, does not seem to matter). The material name has to match the one listed in the .model file, otherwise the model will be broken. The render_template is a predefined template, and **CANNOT SIMPLY BE APPENDED**, there is a list of available render_templates with explanations [HERE](). This material has two variables included (these are present with almost every material). The "bump_normal_texture" specifies where the bump map of this material is, for this example it's located at "units/payday2/characters/shared_textures/spook_heavy_nm" (once again, the .texture extension is not needed). Followed by a "diffuse_texture", which is the actual texture of the material, located at "units/payday2/characters/shared_textures/spook_heavy_df". Both the "bump_normal_texture" and the "diffuse_texture" are passed to the render_template to be handled.

```
<material name="mtr_il"
render_template="generic:ALPHA_MASKED:DIFFUSE_TEXTURE:OPACITY_TEXTURE:RL_COPS:SELF_ILLUMINATION"
version="2">
    <diffuse_texture ⬜⬜⬜file="units/payday2/characters/shared_textures/spook_il"/>
    <self_illumination_texture ⬜⬜file="units/payday2/characters/shared_textures/spook_il"/>
    <opacity_texture ⬜⬜⬜file="units/payday2/characters/shared_textures/spook_il"/>
    <variable ⬜⬜⬜⬜value="reddot" type="scalar" name="il_multiplier"/>
</material>
```

This section right here is almost identical to the previously viewed material. To differentiate this new material, it has a different, uses a different render_template, and has a few new variables. The diffuse texture serves the same purpose as before. The new, "self_illumination_texture" points to the path of "units/payday2/characters/shared_textures/spook_il". This "self_illumination_texture" is related to the render_template. Same with "opacity_texture " and the "variable". All of them are passed to to the render_template to be handled.

```
<material name="shadow_caster" render_template="shadow_caster_only:SKINNED_1WEIGHT" version="2"/>
```

This last section is responsible for casting shadows. With name "shadow_caster" and render_template of "shadow_caster_only:SKINNED_1WEIGHT". A list of available render_templates with explanations HERE.

After the .material_config file is loaded, nothing else loads. Not all commands were listed in this example, so other commands will be listed below in the "Other .material_config commands" section with their explanations.

Other .material_config commands

Other .material_config commands will be added here as research progresses.