

C.R.A.P. Classic Roleplaying Adaptation Project

Base Hak Documentation

General Installation Instructions:

The Classic Roleplaying Adaptation Project's Base Hak pack is three files, the 2da hak, the base hak, and the tlk file. We are now listing version numbers at the end of each file so that you will know exactly what version you are working with. All updates should be completely backwards compatible with previous hak releases, but if you find any errors, please report them to our forum page at <http://www.nwncrap.com>.

You must manually extract the files to the correct NWN folders. Files with a .hak extension must be placed in the C:\Neverwinter Nights\NWN\hak folder. The file with the .tlk extension must be placed in the C:\Neverwinter Nights\NWN\trlk folder. Note that 'C:\Neverwinter Nights' is the default installation directory for NWN. If you chose a custom install location for the game files, substitute that location accordingly.

Installation Instructions for Module Builders:

Follow the steps listed above for players in order to get the files into the correct locations on your computer's hard drive. Open up the module you want to work with, select edit, and then module properties. From the custom content tab on the resulting properties window, add the following entries in the order listed below. Finally at the bottom of the properties box is a selection for tlk file. In the dropdown menu select **crap305**.

NOTE - The base hakpack is now compatible with PRC 2.2 and Alternate Combat Animations v3.0. You can even use our haks as a merge hak for CEP and PRC. However, PRC and CEP update often so we can't support both, although we work as a merge hak for both, we do not intend to support or maintain PRC compatibility. Sorry, we have real lives too - we don't get paid to do this ;)

Hak attach order from top to bottom:

[crp_prc_merge](#) (ONLY if you are using PRC)

crp2da310

crpbase312

[Any and all crp tileset haks in any order \(optional\)](#)

[PRC 2.2 haks \(optional, unsupported\)](#)

[ACP 3.0 haks \(optional, unsupported\)](#)

[cep1patch152](#)

[cep1patch150](#)

[cep1patch](#)

[cep2da](#)

...

Introduction

As a long time pen & paper D&D enthusiast and computer geek, I loved the idea that a CRPG had finally been built with a somewhat user friendly toolset that would allow its gaming community the opportunity to build modules. That game was NWN. I was dismayed however, once I got into the toolset, to learn how time consuming putting together the little details could be. Sure, there were a lot of great community contributed haks, scripts, and what have you, to be found on the "Vault", but the majority of them required quite a bit of knowledge to implement and even more to integrate with one another. Not only that, but there was no general consensus about which to use and which not. Usually a player or DM can expect a different hak and different set of "house rules" for each and every module they download.

The CEP, Community Expansion Project, was the first major step in standardizing the player contributed content. This base module and the additional custom content used by it is the next step along that path.

The goal of this module is threefold. First, I hope to save my fellow module builders tons of tedious script work, by providing a pre-built base module that already has many of the most popular script systems (or variants thereof) already built in. Second, to provide module builders and their players several unique systems designed to increase the options available to players and to allow module builders greater flexibility in their designs. Finally, to provide extended standardized content that covers the many areas that CEP left out (tilesets, script systems, etc...)

Before we get into the depth of this document, I must point out a couple of things about this module. One, this document assumes that you are already familiar with the toolset. It is not designed to be a tutorial for new module builders, but rather as a resource that will allow experienced module builders to, not only save time, but to take their modules to the "next level". Two, the base module requires that you have the CEP content installed and the C.R.A.P. base content. At the time of this writing, this module requires CEP v1.52 and the C.R.A.P. Base v2.01. You can download CEP at the following link:

<http://nwwvault.ign.com/cep/downloads/>. You can download the C.R.A.P. Base content at the following link:
<http://nwwvault.ign.com/Files/hakpacks/data/1101100617000.shtml>.

In any case, I hope you find this module useful. If you do, please vote for it! If you would like to find out more information about the Classic Roleplaying Adaptation Project or if you would like to contribute to the project, please visit our webpage

at <http://www.realmsofmyth.org/crap/home.htm>. Full author credits can also be found there.

--KC Solberg

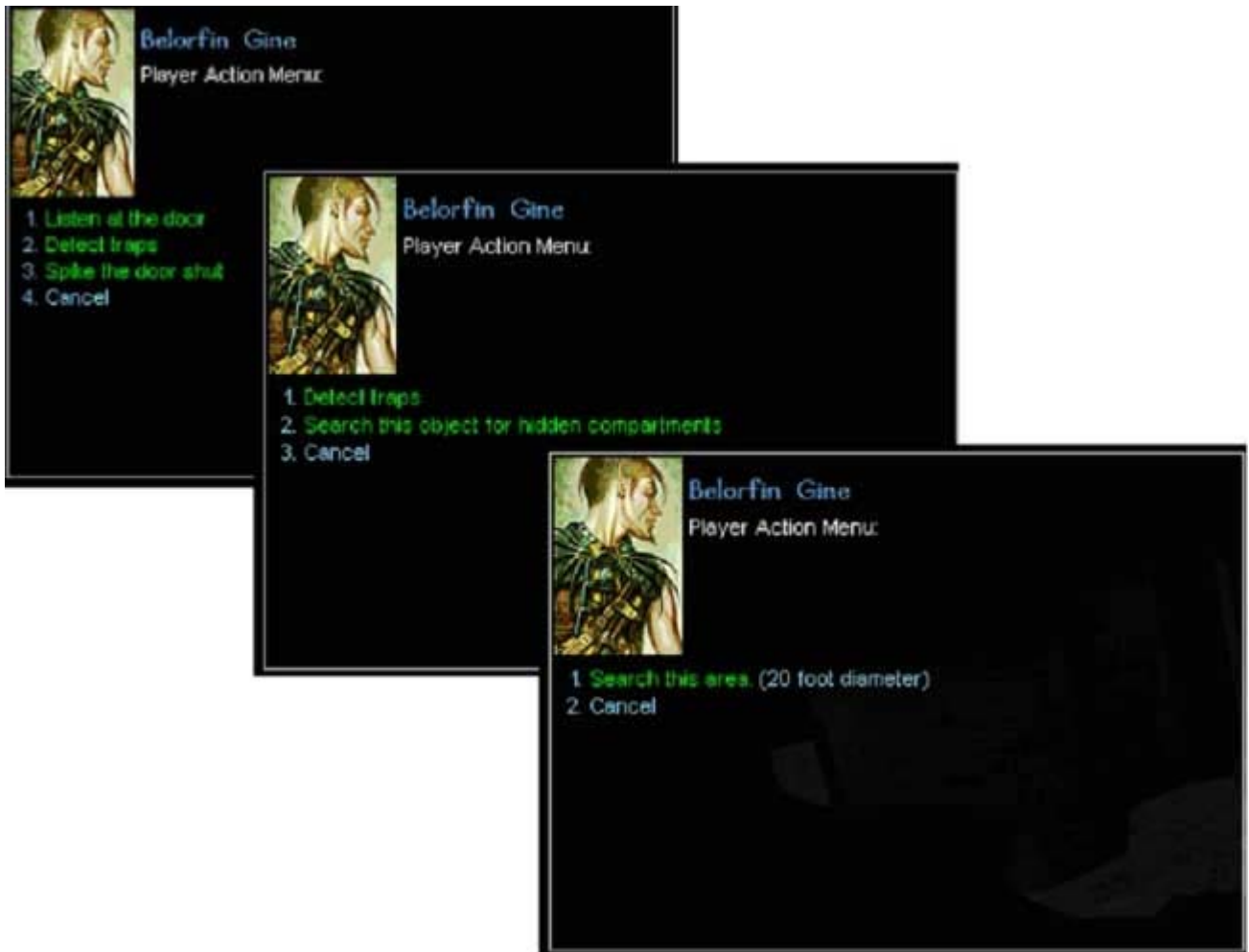
...

The Player Action Widget (PAW)

This module makes use of an innovative C.R.A.P. team original, called the Player Action Widget, from here out we'll refer to as the PAW. We have improved the widget dramatically in this release of the base module. An intelligent widget, the PAW offers players numerous play options that were 'standards' in PnP D&D but sadly lacking in NWN.

The base mod is pre-coded to give entering players a tool bag containing a PAW widget, DMFI player book, DMFI player dice bag, and DMFI player emote wand.

The PAW allows characters to actively search for traps, secret objects, and hidden compartments. They can also use it to leap gaps, listen at doors, use items in the inventories to spike doors open and shut, lower ropes into pits, etc... The best thing about the widget, though, is that it does not spam players with useless options. It only offers choices that make sense for the target selected.



The full list of options available to players through the PAW is as follows:

- Listen at the door
- Detect Traps
- Spike this door shut
- Spike this door open
- Remove the spikes from this door
- Search this object for hidden compartments
- Search this area (25 foot radius)
- Create a campsite here
- Apply Bandages
- Jump
- Drive a spike into the ground, secure a rope to it, and lower it into the pit.
- Retrieve rope
- Climb down the rope
- Listen

- Rest

It's easy to setup objects in your module so that the PAW can interact with them. There's NO scripting required at all. After you have done a couple items, you should be able to set up a pair of secret doors in about one minute. You should be able to create a hidden compartment in any container or placeable and setup the container's inventory in about 30 seconds. I can add a pit trap to an area in less than a minute. The rest of this module either lists rule sets for the mod and how to modify those rule sets to fit your needs or gives detailed instructions on setting up doors, traps, hidden items etc...

...

Secret & Hidden Objects

One of the design goals of this module was to allow a module builder to design faithful PnP module adaptations. That means 'active' searching. This is why your players are given a Player Action Widget. If you read above, you'll see that players can actively search an area, attempt to detect traps, and many other things with the PAW. Setting up secret doors or making active search traps is easy. Everything you need has already been pre-scripted and blueprints are in the palettes waiting for you to drop and use. All you'll have to do is right-click on your placed items and adjust a few variables. This section will list all the items/objects you need for building as well as any items that you should make available to your players. Each sub-sections detail exactly how to set your doors, traps, etc..

...

- **Creating Secret Doors**

Palette Locations of Objects you'll need:

Secret Doors	Placeables / Custom / Secret Objects
Dwarven Stonework Detector Trigger	Triggers / Custom / Secret Object Trigger
Elven Autosearch Trigger	Triggers / Custom / Secret Object Trigger

Each secret object should be created originally using one of the included placeable blueprints (custom-secret objects). You can edit these blueprints to create new blueprints but if you remove any of the attached scripts the system will not work.

1. Place two secret doors in your module.

These can both be hidden or one hidden and the other not. This also could be a

secret trapdoor and ladder combo but we'll cover those scenarios a little later. For now, just use two secret doors provided for you in the custom palette location specified above. Put the doors exactly where you want them to be found later.

2. Modify Door # 1 variable values.

First off, you must decide on two unique names (tags) for each door. Jot these down on a piece of paper but **DO NOT** adjust the actual door tags. Right-click on the first door and select variables from the popup list. You'll see a screen like the one shown in the screen below

Name	Type	Value
DC	int	0
DESTINATION	string	Enter destination door's unique tag here
RESET	int	0
TAG	string	Enter unique tag here

Name: Type: Value:

Adjust the variables as follows:

- **DC** - This is the detection DC of the hidden object. Set it to the appropriate value.
- **DESTINATION** - Enter the unique TAG name of the door that this will take you to. (You jotted this down on a piece of paper earlier. Remember? This is door number two's name.)
- **TAG** - Enter the unique name for this door (The other name that you jotted down on a piece of paper)
- **RESET** - This is an option value which we have defaulted to 0. If left at value 0, the door will remain revealed after players discover it. If you set this value to a positive number, the door will conceal itself again. The

number you enter into this value will be the number of seconds later that the door will re-hide itself.

3. Modify Door # 2 variable values.

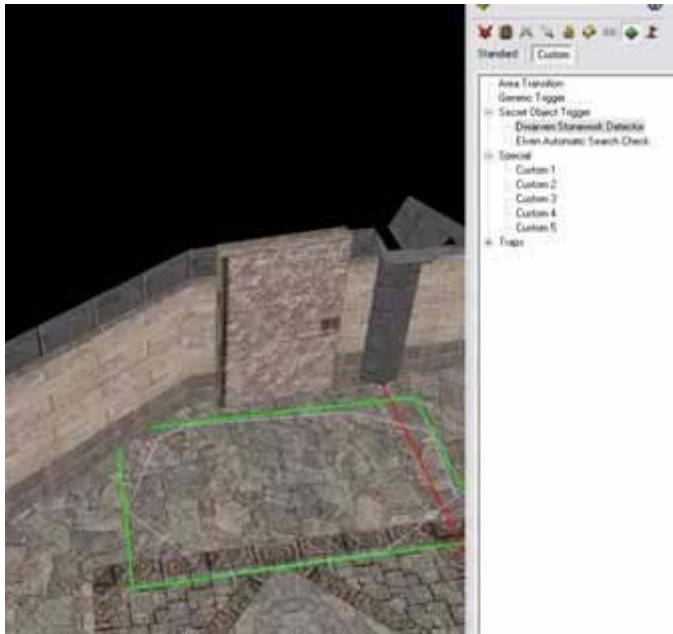
Now right-click on door number two and select variables. Repeat the steps detailed above. The tag and destination variables will be the exact opposites of door number one, of course.

If either door is not supposed to be hidden, just change it's actual "tag" to the value that you had recorded for its TAG variable and change its DC variable to 0. But if it IS supposed to be hidden, you must leave its actual tag as "secret" and enter its tag-name under the TAG variable only.

...

Advanced Hidden Door, Trapdoor, and Trap Options:

The Players Handbook states that Elves get to roll a search check just as if they were actively searching, just by coming within 5ft of a hidden object. Likewise, Dwarves get to roll a search check just for coming within 20ft of a hidden object that is built into stonework. You may use the two special triggers that are available in order to put these rules into play. They are under the custom trigger menu and are called, "Dwarven Stonework Detector" and "Elven Auto Search". To use, just draw the appropriate trigger around the area of your secret object (fig S2). The "Dwarven Stonework Detector" works for both Elves and Dwarves. The "Elven Auto Search" works only for Elves. When the appropriate racial type character walks into one of these triggers the game will do a hidden object search for that character in the background and reveal the object to them if they make the DC roll.



...

- **Creating (Secret) Trapdoors with Ladders**

**Palette Locations of
Objects you'll need:**

Secret Trap Doors	Placeables / Custom / Secret Objects
Ladders	Placeables / Custom / Miscellaneous
Dwarven Stonework Detector Trigger	Triggers / Custom / Secret Object Trigger
Elven Autosearch Trigger	Triggers / Custom / Secret Object Trigger

1. Place a trapdoor and ladder in your module.

The trapdoor can be a secret trapdoor or not. The ladder will not be hidden. Usually the ladder would be placed in a different area from the trapdoor but that is up to you, of course.

2. Modify the trapdoor's variable values.

First off, you must decide on two unique tags - one for the trapdoor, one for the ladder. Jot these down on a piece of paper but **DO NOT** adjust the trapdoor's tag if you wish for it to be a secret trapdoor. Right-click on the trapdoor first and select variables from the popup list. You'll see a screen like the one shown in the screen below.

Name	Type	Value
DC	int	0
DESTINATION	string	Enter destination door's unique tag here
RESET	int	0
TAG	string	Enter unique tag here

Name: Type: Value:

Adjust the variables as follows:

- **DC** - This is the detection DC of the secret trapdoor. Set it to the appropriate value. If the trapdoor is not secret, set this value to 0.
- **DESTINATION** - Enter the unique tag name of the ladder that this will take you to. (You jotted this down on a piece of paper earlier. Remember?)
- **TAG** - Enter the unique tag name for this trapdoor (The other name that you jotted down on a piece of paper)
- **RESET** - This is an option value which we have defaulted to 0. If left at value 0, the door will remain revealed after players discover it. If you set this value to a positive number, the trapdoor will conceal itself again. The number you enter into this value will be the number of seconds later that the trapdoor will re-hide itself. If this trapdoor is not secret, leave this value at 0.

3. Modify the ladder's actual tag.

Right-click on the ladder you placed earlier and select 'properties'. Change the ladder's actual tag to match the value of it's connecting trapdoor's DESTINATION value. Note that tag names are case sensitive.

4. Modify the ladder's variable values.

Right-click on the ladder again and select 'variables'. Change the ladder's

DESTINATION value to match the TAG variable value that you set on the trapdoor earlier.

...

- **Creating Hidden Tracks**

**Palette Locations of
Objects you'll need:**

Tracks

Placeables / Custom / Secret Objects

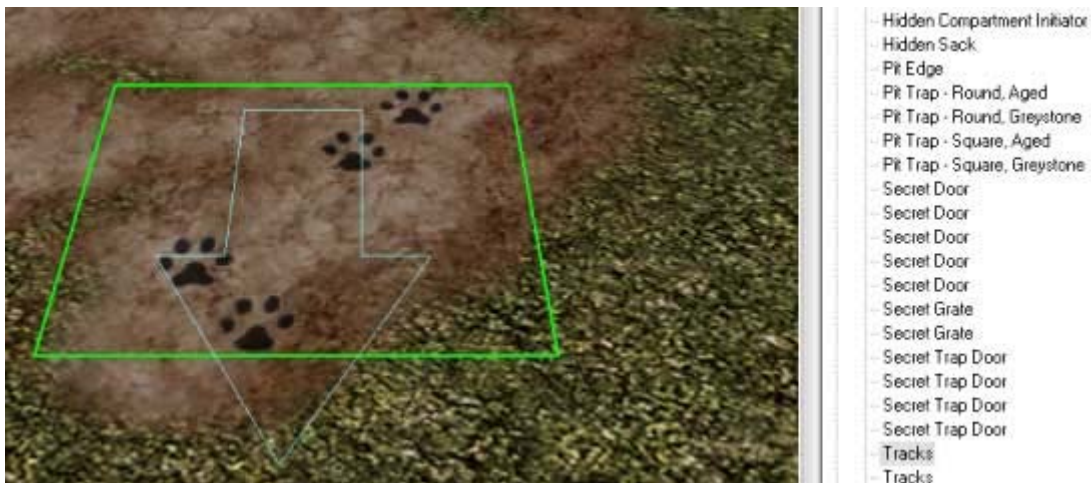
Hidden tracks have only a DC variable for detection purposes and they don't reset. However, they have three additional 'message' variables that determine what a player examining the track detects. The messages are set to allow three 'levels' of information: 1) a general low level message that anyone examining the tracks can gather, 1) amessage level that only rangers can gather from the tracks, and 3) a message level that only rangers 7th level and above can gather.

1. Place a set of tracks in your module.

2. Modify the track's variables

Right-click on the set of tracks after it has been placed and select variables. Adjust the variables as follows:

- **DC** - This is the detection DC vs a search check by the player to 'discover' the tracks. Set it to the appropriate value. If the tracks are not secret, set this value to 0 and change the track's real tag to anything other than 'secret'.
- **MSG1** - Low level message, readable by all
- **MSG2** - Mid level message - readable by any level ranger
- **MSG3** - High level message - readable by 7th level or higher ranger



...

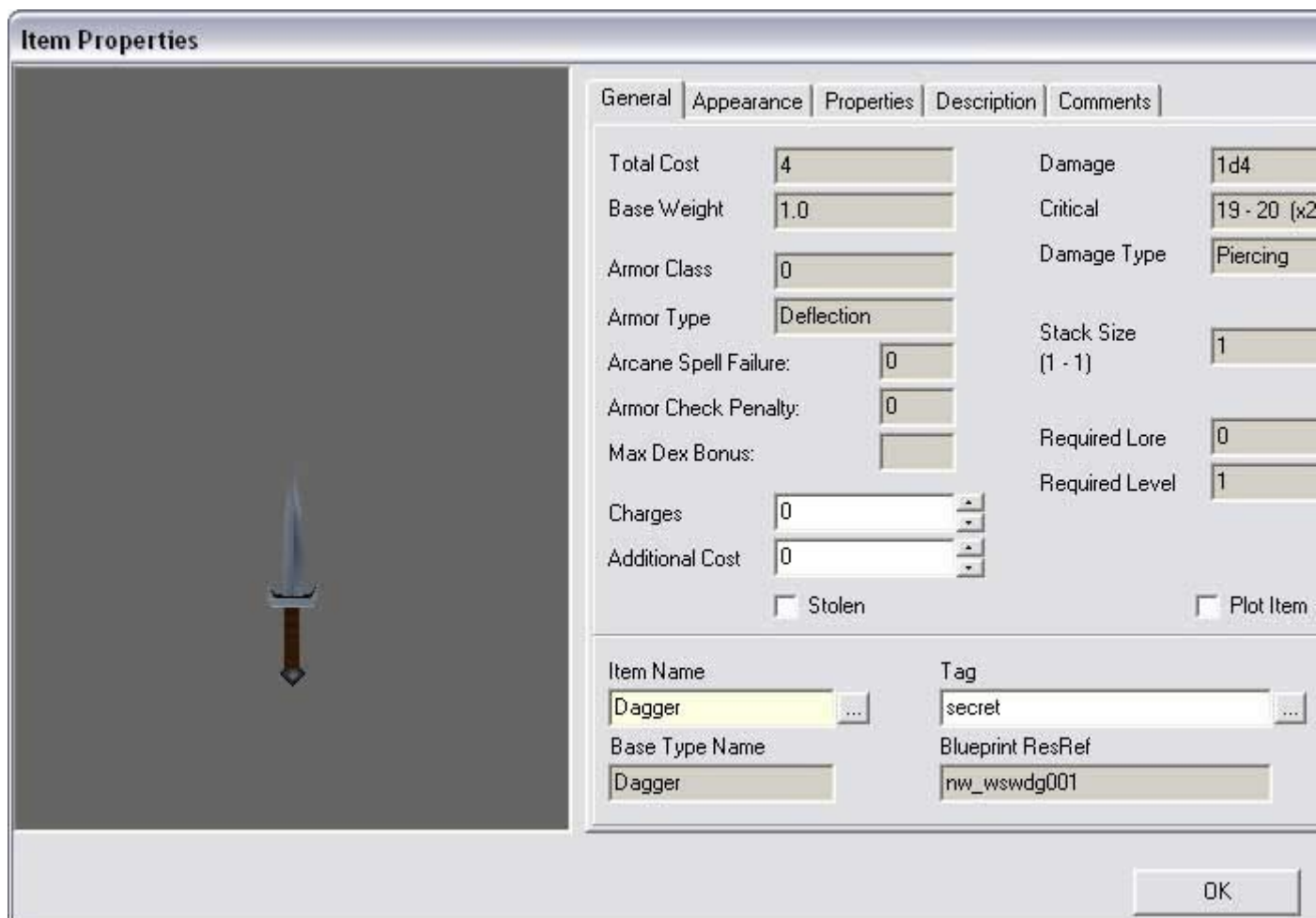
- **Creating Hidden Items**

ANY item can be hidden within your module. All you do is place the item where you want it to be found, open its properties page and make a couple quick edits.

1. **Place the item in your module.**

2. **Copy and edit the item's original tag**

Right-click on the item and copy down the item's original tag name. After you have copied it, change the item's tag to 'secret' - minus the quotes - case sensitive.



The screenshot shows the 'Item Properties' window with the following details:

- General Tab:**
 - Total Cost: 4
 - Base Weight: 1.0
 - Armor Class: 0
 - Armor Type: Deflection
 - Arcane Spell Failure: 0
 - Armor Check Penalty: 0
 - Max Dex Bonus: (empty)
 - Charges: 0
 - Additional Cost: 0
 - Damage: 1d4
 - Critical: 19 - 20 (x2)
 - Damage Type: Piercing
 - Stack Size (1 - 1): 1
 - Required Lore: 0
 - Required Level: 1
 - ☐ Stolen
 - ☐ Plot Item
- Item Name:** Dagger
- Tag:** secret
- Base Type Name:** Dagger
- Blueprint ResRef:** nw_wswdg001
- OK** button

3. **Add 'DC' and 'TAG' variables to item**

Right-click on the item and select variables. Add an int variable called DC to the item. Set this variable to the detection DC that a player must roll vs to find the

item. Add a second string variable call TAG. Copy the items original tag (you copied this in step 2) into the value field of this variable.

Name	Type	Value
DC	int	5
TAG	string	NW_WSWDG001

Name	Type	Value
TAG	string	NW_WSWDG001

Replace Add Delete

OK Cancel

...

- **Creating Hidden Containters (Placeables with Inventories)**

**Palette Locations of
Objects you'll need:**

Hidden Sack

Placeables / Custom / Secret Objects

Hidden containers are the easiest setup of all the hidden object types. Just drop a hidden sack from the placeables palette and put items into it's inventory. You then have one variable to set, DC. Set DC to the detection dc vs search that the player must roll to discover the container.

Feel free to alter the name of the hidden sack as you want. You can also alter it's appearance to suit your needs. But do not alter its tag.

...

- **Creating Hidden Compartments**

**Palette Locations of
Objects you'll need:**

Hidden Compartment
Initiator

Placeables / Custom / Secret Objects

Hidden compartments are also very easy to set up. Any placeable, whether it has an inventory or not, can conceal a hidden compartment.

1. Place the object which has a hidden compartment into your module

This can be any placeable, and as I stated before it doesn't matter whether the object has an inventory or not.

2. Add a DC variable to the placeable object.

Right-click on your newly placed object and select variables. Add an int variable called DC. Set DC to dc vs search that a player must roll in order to find the hidden compartment concealed within this placeable if they search.

3. Place a 'Hidden Compartment Initiator' on top of or right next to the placeable.

Don't worry, the initiator looks like a bag, but it won't ever be seen by your players. Place the treasure items that are supposed to be stashed within the hidden compartment into the hidden compartment initiator.

...

Traps

One of the design goals of this module was to allow a module builder to design faithful PnP module adaptations. That means 'active' searching. This is why your players are given a Player Action Widget. The CRP Base module supports active detect trap searches of doors and placeables. Game engine limitations don't allow for 'true' active searching of trigger traps, but in my opinion, it would be unweildy and ultimately annoying to players if they were forced to do full active searches for traps down every section of corridor. Instead, we allow for normal detection methods for trigger based traps with a 'bonus' for detection if a player actively searches a section of floor or corridor for traps and full active search trap detection on chests, doors, etc... In addition to the standard array of traps, I have added a super easy to configure pit trap system for your use. What good PnP conversion doesn't have pit traps, I ask you?!? The following subsections describe how to place and configure each typ of trap in your module.

...

- **Creating Pit Traps**

This base module includes easy to setup pit traps. These traps are created by placing a pit trap object in the location desired then drawing a pit trap trigger around the placeable pit's edges. Of course you'll have to create an area that the character can fall to, if they are unfortunate enough to trigger the trap and fail their reflex saving throw. Their fellow party members will also be able to lower a rope for them or climb down the rope, themselves, to the bottom of the pit.

Palette Locations of Objects you'll need:

Pit Trap Triggers	Triggers / Custom / Traps
Placeable Pit Traps	Placeables / Custom / Secret Objects

Palette Locations of Objects you should make available to players:

Rope	Items / Custom / Miscellaneous / Other
Iron Spikes	Items / Custom / Miscellaneous / Other
Spike Hammer	Items / Custom / Weapons / Blunt / Hammers

1. Drop a pit trap placeable object into your module

You have several pit trap placeable objects available to you in the custom menu. Place the type you want in the location where the pit will appear. The module will conceal the pit when the module loads.

2. Create the bottom of the pit area

Make an area that will be the bottom of the pit. This is where players will land if they fall or climb into a pit. Place a waypoint in this area and give the waypoint a unique tag.

3. Modify the pit placeable's variables

Return to the area where you placed the pit trap placeable, right-click on it and select variables. Change the DESTINATION variable of the pit to the tag of the waypoint you placed in the pit bottom area. ***NOTE* You will need to add an integer DC variable to the pit placeable, this is missing from the blueprints in BaseMod version 2.01**

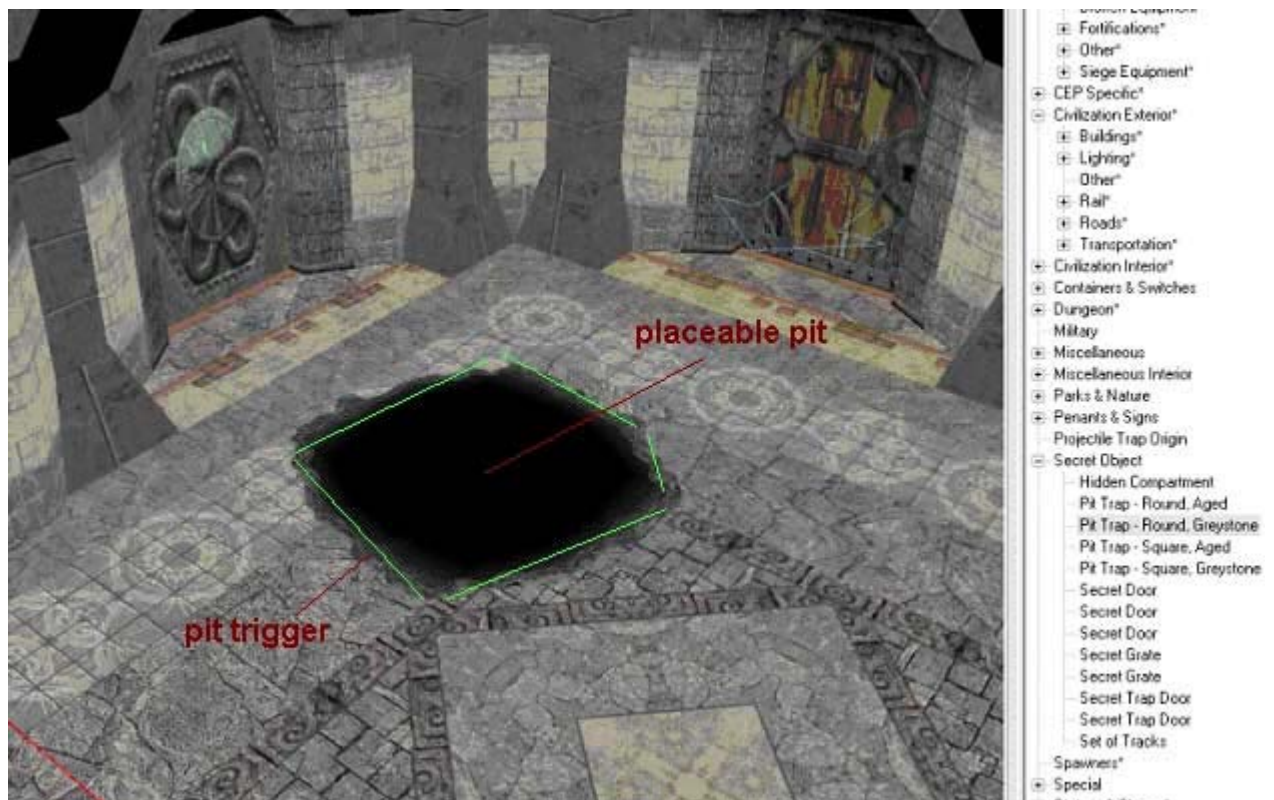
4. Draw the pit trap trigger

Outline your pit trap with a pit trap trigger. Five pit trap trigger blueprints are in your custom palette, ranging from easy to fatal. Each level of pit trap does 20% hitpoint damage to a player when they fall into the pit - Easy = 20% damage, ranging to Fatal = 100% damage. **NOTE: You may need to adjust the Detection DC under properties if the trigger shows up without searching with the PAW.**

5. (Optional) Draw a toggle jumping trigger

If player jumping is disabled by default in the [master control file](#) then you may

wish to draw a [toggle jumping trigger](#) around the general radius of the of the pit. The master control file and player jumping options are covered in later sections of this document.



Advanced Pit Options:

You can also allow your players to use a grappling hook and rope to get themselves out of pits. Enabling this option for your players requires you to place one placeable object at the pit's bottom and one waypoint at the pit's top near the edge.

Palette Locations of Objects you'll need:

Pit Edge Placeable

Placeables / Custom / Secret Objects

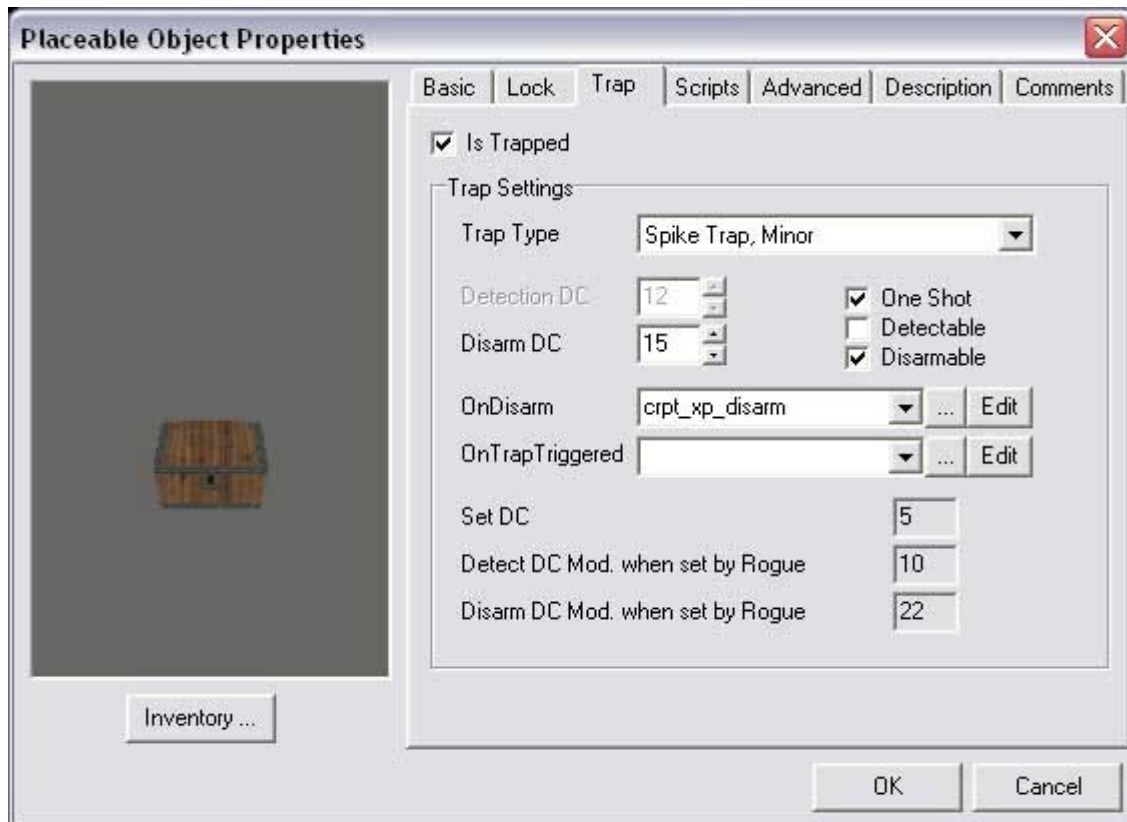
Put the invisible pit edge placeable somewhere near the drop waypoint at the bottom of the pit. Place a waypoint at the top of the pit and give the waypoint a unique tag. Go back to the invisible pit edge placeable, right-click it and select variables. Modify the DESTINATION variable so that its value is the tag (case-sensitive) of the waypoint at the top of the pit. Use of pit edge placeables and waypoints isn't limited to pits of course. You can do a similar setup in other areas

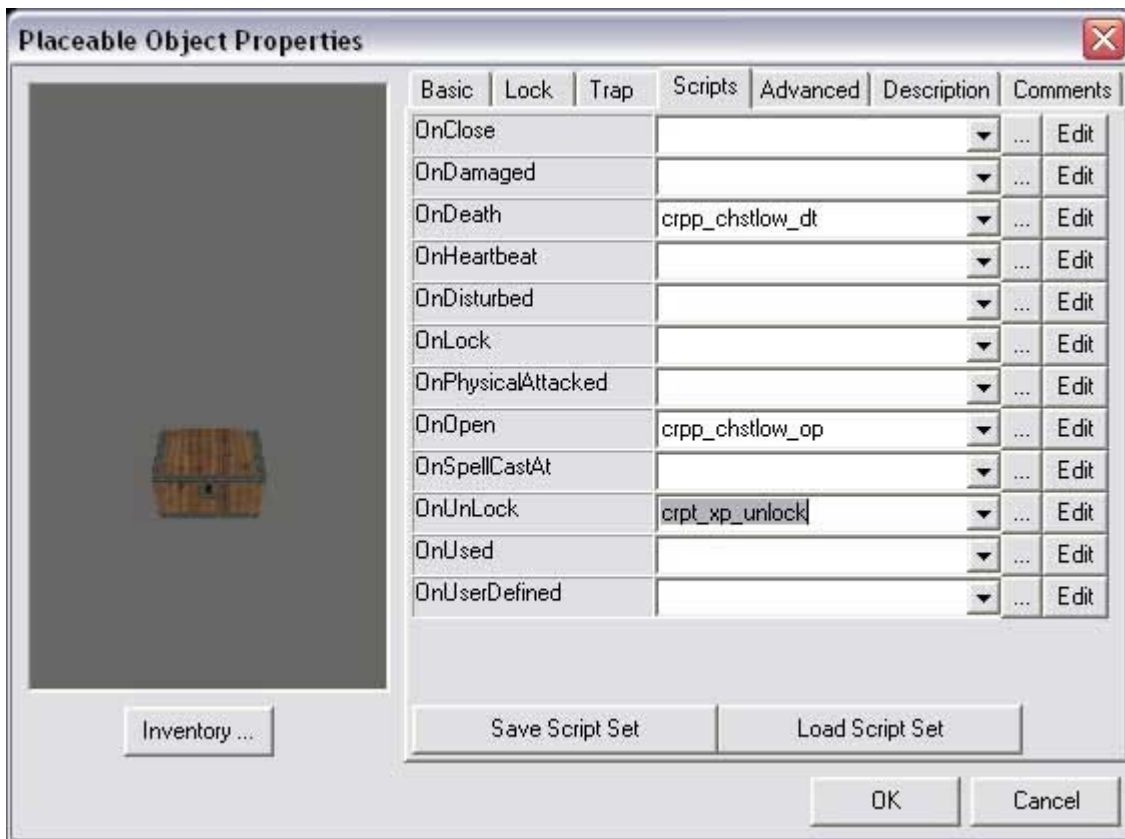
to allow players to scale to higher elevations or other inaccessible areas. You can rename the Pit Edge to anything you want, but don't modify its tag.

...

- **Setting up traps on placeable objects and doors**

Traps on doors and chests are just set up normally, with one additional step - after you set the trap's detection and disarm dcs, uncheck the trap's detectable checkbox. Players will still be able to use their PAW to actively detect the trap concealed on the object or door. At your option, you can also award experience points for disarming or unlocking a container or door. Use the scripts, `crpt_xp_disarm` and `crpt_xp_unlock` respectively. Your properties window for the trap on the placeable object should look something like this:





...

- **Setting up trigger based traps**

Just set these up normally. The PAW gives players a bonus to discover trigger traps if a player does active searching. If you want to offset this built-in bonus, up the detection dc on the trap by 5 points.

...

Doors

Many of the special functions that players can activate via the PAW involve doors (listening, detect traps, spike open/shut). In order for these functions to work correctly, you'll need to use the doors included in the custom palette. You can edit copies of any of these doors and save them as a new blueprint to suit your needs, but do not remove any of the variables or scripts attached or they will not function properly.

Besides the fact that the doors have been modified to work with the PAW, many new models and textures have been included as well, including curtain doors, web doors, and a whole host of dungeon and rustic doors. Just forget that any of the doors in the standard palette exist and drop all your doors from custom and there will be very little else for you to configure, however there may be situations where you wish to adjust the default behavior of a door either by removing the ability for it to be spiked or by adjusting the listen dc modifier that the doors adds to a player when they are attempting to listen for creatures on the other side.

Each door contains at least one variable. That variable will either be a LISTEN_DC_MODIFIER or a SPIKED variable. To adjust variables open the properties box for the door, click the advanced tab, and click the variables button. These variables may be adjusted as follows:

...

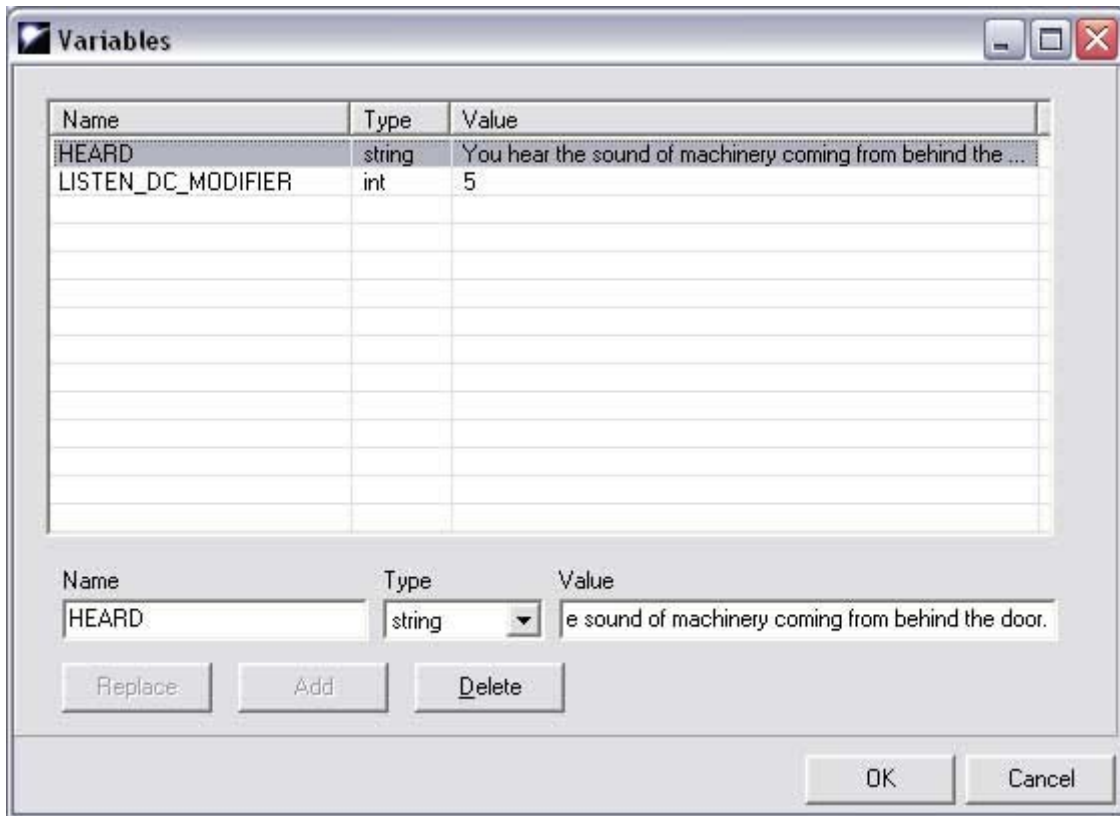
- ***Spiking Doors:***

To remove the ability to spike a door open or shut, a variable must be included called SPIKED (case sensitive). SPIKED must be set to "2" to disable spiking open or closed.

...

- ***Listening at Doors:***

The LISTEN_DC_MODIFIER determines the amount to add to the versus check on a listen check at the door. The higher this number, the harder it is to hear creatures on the other side of the door. Two factors determine the feedback a player gets when he or she listens at a door: 1) their skill checkroll, 2) the type of creature on the other side of the door. Default feedback messages have been preconfigured for you, but if you wish to add a custom message to any door, you can do so by editing the value of the door's HEARD variable. Whatever text you enter under HEARD will be returned to the player on a successful listen check at the door.



...

The Master Control Script

The rule systems implemented in this base module are customizable. Some rule sets can be completely disabled, others can be customized to suit your needs. Changes to all customizable rule systems are made within one master control file. The control file is a script called 'crp_inc_control'. Open it with your script editor. All the control lines are well documented in the file itself and the file is easy to edit (generally you'll either be setting a value to 0 or 1 to customize the behavior of the module), so I'll suffice with a brief listing here of customizable rule systems.

1. COINS - choose Bioware gold system or CRP copper, silver, electrum, gold, and platinum coin system.
2. BREAKABLE ITEMS - choose whether items in destroyed containers must roll a saving throw vs damage type to keep from being broken (or not)
3. DEATH SYSTEM - 5 different options that you can customize here (the death rules are covered later in this doc)

4. RACIAL MOVEMENT - choose whether to play with 3.5 ed racial movement rates or have all characters move at the same speed
5. ARMOR ENCUMBRANCE - 3 options - choose how medium and heavy armor affects player movement rates
6. LISTENING - choose whether players can listen for nearby creatures when outdoors or not - dungeon listening is only at doors and is always enabled
7. RESTING - choose how long a player must wait between rest periods - also enable or disable advanced resting rules (covered later in this doc)
8. TORCH LIFE - 2 options for controlling torch life and effects
9. WEATHER - 2 options for controlling the chance and type of precipitation in your module.

Remember, you don't have to script any of the above systems into this module. They are already pre-set and running for you. You can leave the options at default values (recommended) or change to suit your needs. When you modify the control file, changes do not actually take effect until you do a rebuild of the module.

...

Included Features

The base module included a number of enhancements to the general behavior of creatures and items in the game. These enhancements require no special configuration, for the most part. In other words, they just work. The enhancements are listed below with notes on configuring the ones that do have options associated with them.

**...Cloak
Appearances:**

Player armor appearances are automatically modified to show or not show a cloak when a player equips or unequips a cloak. Two non-magical cloaks are also included in the custom item palette under



miscellaneous - clothing - cloaks

...Rope Climbing:

Players can lower ropes to their party members or make use of grappling hooks attached to ropes to get themselves out of pits or to other normally inaccessible areas. When players climb a rope... well, a picture is worth a thousand words they say.

..Torch Life :

This base module uses LoCash's torchlife system. By default torches will fade and flicker out after 20 minutes. You can adjust this time limit in the [master control script](#), however.

..Poisons :

A slew of premade 3.5ed. poisons can be made available to players to add spice to your module.. These include poisons such as carrion crawler brainjuice,



...Jumping:

If jumping is **enabled** in the [master control script](#), players are able to jump any obstacle 1) to which they have line of sight, and 2) is within their race's jumping distance. When jumping is enabled, you can place a jump toggle trigger anywhere to **disable** it.

However, the recommended way to go is to **disable** jumping by default in the [master control script](#). When jumping is disabled, you can **enable** it in specific areas by placing a jump toggle trigger wherever jumping should be turned on. This allows for finer control over where your players can and can't go.

Jump toggle triggers are in the custom palette: **Trigger Custom / Generic**

greenblood oil, dragon bile, just to name a few. Players who attempt to poison a weapon have a slight chance of poisoning themselves (poison fumble). Assassins are exempt from this rule and do not have to roll a poison fumble skill check when applying poisons. In addition, weapons poisoned by assassins stay poisoned for longer than those poisoned by other character types. An assassin gains a bonus to poisoning length every 5 levels.

Poisons are located in the custom palette under: **Items / Custom / Miscellaneous / Poisons**

..Creature Morale :

This module uses Jeff Peterson's Fight or Flight creature morale system. There are really only two things you must know in order to use the system to its fullest potential. First is how to use a rally point. The rally point included in the custom waypoint palette is used by creatures who fail their morale check in combat. When a rally point is present, NPCs who have failed their morale check will run to it and rally there. The second thing you should know is how to disable the system and how to create a leader creature (one to whom the others will rally to). Leaders can be created or the system can be disabled on a creature by editing the script, `nw_c2_default9`. Read the script comments for more information.

..Weather :

The base module has a built in weather system already implemented. By default, the type of precipitation is set to RAIN. However, you can adjust this setting to SNOW for a winter setting through the [master control script](#). Change the values within the master script as specified by the comments. Remember to rebuild your module after any change to the master control script.

The weather system can only make it snow or rain. You must still set the lightning/thunder percentage chance on an area by area basis. You do this from the individual area's edit properties window.

..DMFI :

This base module includes DMFI v1.07. DMFI is a DM is an extremely useful DM wand package by Demetrius and HanSolo. It is beyond the scope of this document to cover all of DMFI and if you are not familiar with DMFI already, this section won't make much sense to you. Please download the DMFI documentation from [here](#).

DMFI overwrites several default Bioware scripts. Mainly it does this, for the emote functions that DMFI has. We have chosen not to overwrite as many default scripts in our implementation (for system performance reasons). We have, however, included 2 of DMFI's scripts, but named differently. These are

cz_listener_co & cz_listener_sp. These are included so that you can setup special NPC's to be emote listeners.

To setup an emote listener (necessary for the emote functions in DMFI), attach the script cz_listener_co to the NPC's OnConversation event; attach the script, cz_listener_sp to the same NPC's OnSpawn event. One or two listener NPCs per area should be sufficient.

...

Descriptions and Pop-up Messages

You may want to provide feedback to your players about things that the game's graphics or sounds can't convey - the smell of a place, the mood, whether it is warm or cold, etc... This type of feedback is normally provided through a message sent to the player's text window or via a pop-up text message within the module. We have two pre-configured (no scripting) solutions built into this module to allow you to give either type of feedback.

..Description Triggers :

Description triggers are in the custom trigger palette under: **Triggers / Custom / Generic Triggers**. Description triggers send a floaty notification to the player when he/she enters the trigger and sends the text of the description to their text window. To set the text of a description trigger, draw it then right-click on it and select variables. Enter the text of the description into the value field of the MESSAGE variable.

..Invisible Description Placeables :

Description placeables do pop-up text messages which can be read by all when a player approaches within a certain distance of them. The description placeable is invisible and can be placed anywhere. It is located in the custom palette under: **Placeables / Custom / Miscellaneous / Other**. Place the Description object and right-click on it. Select variables. Enter the text of the pop-up message under the value field of the MESSAGE variable.

...

Movement Rates

This base module uses KCS's Racial Movement Rates and Armor Encumbrance system. You can disable racial movement rates and armor encumbrance by adjusting settings in the master control script. By default:

- Halflings, Gnomes, and Dwarves move 20% slower than Elves, Humans and Halforcs.
- Halflings and Gnomes suffer an 8% medium armor encumbrance movement penalty or a 15% heavy armor encumbrance movement penalty.
- Humans, Elves, and Halforcs suffer a 10% medium armor encumbrance movement penalty and a 20% heavy armor encumbrance movement penalty.
- Dwarves have no armor encumbrance penalties (as specified by the D&D 3.5 edition rules).

Racial Movement Rates can be disabled in the master control script and armor encumbrance penalties are adjustable. Read the comments under the RACIAL MOVEMENT RATES & ARMOR ENCUMBRANCE section of the control file for details.

...

Death, Dying, & Respawning (DDR)

DDR adds a negative 10 bleed out system to the game per the 3.5 Players Handbook rules. Bleeding characters can be healed by other players, henchmen, or familiars or stabilized by bandages (see below).

Notes:

- Characters have a 10% +/- their constitution bonus to stabilize per round (go to 1 HP and stop bleeding).
- Bandages have been added which, when used by another player on a bleeding character, stabilizes that character on a successful healing skill check.
- DDR doesn't use the rules for disabled characters or death from massive damage.
- When bleeding, monsters assume the character is dead and move on to other targets or just move on about their business if no other targets are around.
- Nearby party members are sent a message each bleed round alerting them to the fact that their comrade is bleeding.

When a character respawns they are sent to Purgatory where they must pay the toll of the Reaper (XP Loss).

Adjustable DDR features

1. When a character dies, all non-plot equipment is dropped onto a corpse object and must be salvaged by the returning character or their party members. This feature can be disabled in the [master control script](#).
2. Item drops are disabled for lower level characters - third level or lower by default. This 'grace period' can be adjusted up or down in the [master control script](#).
3. Players who are in a party bleed slowly in order to give party members ample opportunity to apply bandages to them. By default the party bleed rate is 18 seconds between bleed rounds. Solo players bleed quickly since it is assumed that no one will apply a bandage to them. By default the solo bleed rate is 1.5 seconds between bleed rounds. Both bleed rates can be adjusted in the [master control script](#).
4. Players are charged a certain amount of xp for each level their character has attained when they respawn. The default is set to 25xp per level, but can be adjusted up or down to suit your needs in the [master control script](#).

..Setting up Respawn Locations :

You must place at least one respawn location waypoint somewhere within your module. Respawn waypoints are generally placed within temples, but that decision is up to you.

The default respawn waypoint is located in the custom palette: **Waypoints / Custom / Waypoints**

Place this waypoint wherever you want your players to return to after visiting the Realm of the Reaper.

..Optional Respawn Locations :

Players can also respawn to different temples based on their deity setting. To create optional deity respawn locations place waypoints in your module with tags based on the following naming convention: SR_deityname

examples.. SR_Lathander, SR_Torm, SR_Helm

...

Player Resting

There are two different ways that players can rest. First, by pressing the "R" key on their keyboard. Second, by selecting the option from the Player Action Menu

(initiated when players use the PAW). The PAW allows players to initiate resting, even if monsters are nearby, but give's no option to set AFK status. If players initiate resting by using the "R" key, they are presented with two options.

1. Rest
2. Set Status: Away From Keyboard -- (this uses LOK's AFK system)

By default - players can rest once every 4 game hours (8 minutes real time) and players can go AFK and Back, whenever they want. The period of time that must pass between rest periods can be adjusted in the [master control script](#). If you wish to disable timed resting completely, set the hour value in the control script to 0.

..Advanced Resting Rules :

Advanced resting rules can be enabled or disabled in the [master control script](#). If you are designing a heavy combat, hack and slash module, you might want to disable advanced resting rules. However, if your module's focus leans more towards exploration, puzzle solving, or roleplay (or a balance of any of these with combat), the advanced resting rules add a strategic element to the game that I recommend you take advantage of.

Palette Locations of Objects you'll need:

Invisible Campfire Object (optional)	Placeables - Custom - Parks & Nature - Campsite
Invisible Bed Object (optional)	Placeables - Custom - Civilization Int - Bedroom

Palette Locations of Objects you should make available to players:

Oilcloth Adventurers Tent	Items / Custom / Miscellaneous / Other
Burlap Adventurers Tent	Items / Custom / Miscellaneous / Other
Tinderbox	Items / Custom / Miscellaneous / Other
Bedroll	Items / Custom / Miscellaneous / Other
Planks	**obtained from smashed wooden doors**

Players can make use of the items listed above to create a more restful environment for their characters. The better the environment, the more hitpoints their character can recover from resting. Players can use different items individually to create a campsite, or they can use the PAW to scan their pack inventories and create the best camp possible with the items they are carrying.

Listed Below are two charts which, when used together, list the quality of possible campsites, both indoors and outdoors, and the resting benefits acquired from resting in each type of camp.

Conditions	Outside	Inside
no bedroll, campfire or tent - raining or snowing	0	NA
no bedroll, campfire, tent or bed	1	1
bedroll only	2	3
campfire only	2	2
tent only	2	1
tent & campfire	3	2
bedroll & tent	3	3
bedroll & campfire	3	4
bedroll, campfire, and tent	4	4
bed	NA	5

Result from Table One	Condition	Hitpoints Added after Rest
0	very poor	10% + (1/2 Con Bonus)
1	poor	20% + Constitution Bonus
2	fair	30% + Constitution Bonus
3	good	45% + Constitution Bonus
4	very good	60% + (2x Con Bonus)
5	bed	Full Hitpoint Return

- Players can use tinderboxes on plank to make campfires.
- Players can use oilcloth and burlap tents on an empty section of ground to raise tents.
- Bedrolls are automatically used if in a player's inventory.

..Controlled Resting Locations :

If CRP_CONTROLLED_RESTING_ONLY is enabled in the [master control script](#) (off by default), players will only be allowed to rest within the bounds of an 'allowed resting trigger'. This enables you to control where a player can and cannot rest. Allowed resting triggers can be found in the custom trigger palette under: **Triggers / Custom / Generic Triggers**

...

Listening Options

Palette Locations of Objects you'll need:

Listening Trigger

Sound Waypoint

Triggers / Custom / Generic Trigger

Waypoints / Custom / Waypoints

..Listening Triggers :

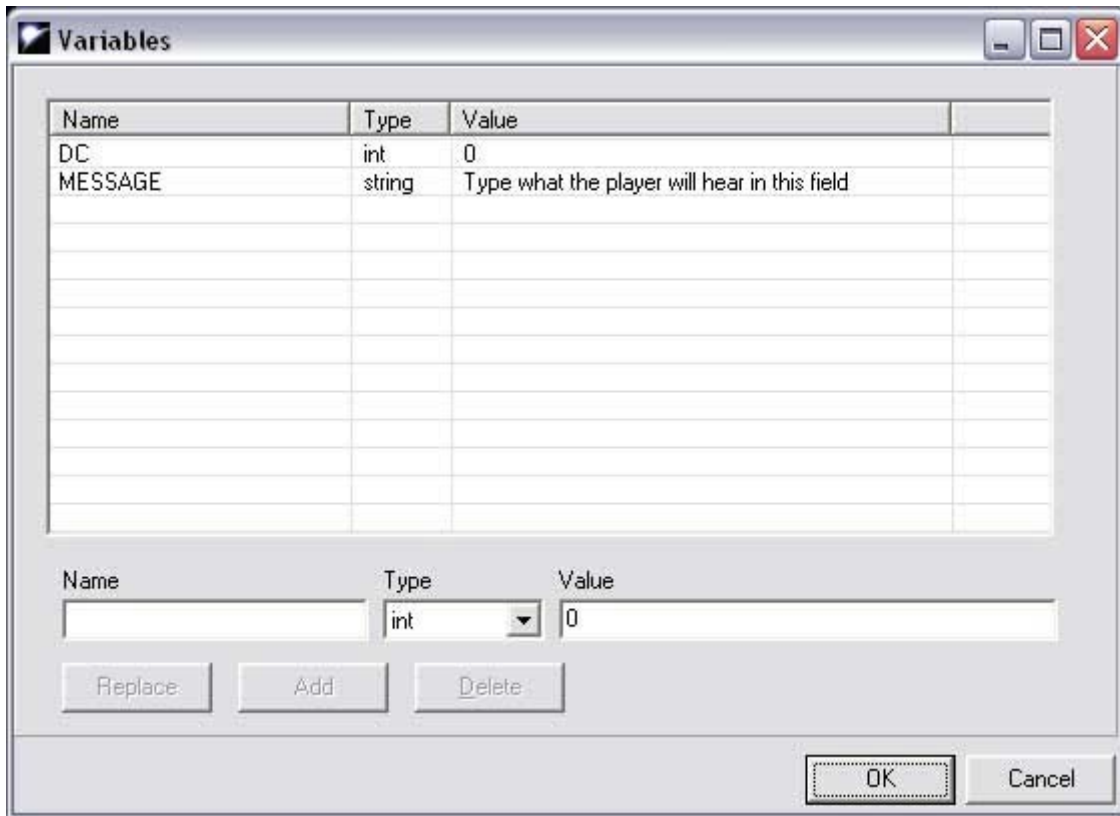
You can place listening triggers anywhere within your module. When players enter a listening trigger they make a behind the scenes listening check against the dc of the trigger. If they roll successfully, they get a floating text message that notifies them that they *'think they hear something'*. Activating the PAW will then give them the exact text describing what they hear. This text is provided by you as a variable on the listening trigger.

1. Draw a Listening Trigger

As stated above, players within the trigger will roll a listen check and receive an extra option when they activate the PAW within the trigger.

2. Modify the Listening Trigger's variables

Set a DC variable for the trigger (the dc is what the player must roll d20 vs to make a successful listen check). Then set the MESSAGE variable. The text of MESSAGE is a description of what the player hears when they check the PAW.



...

Treasure

..Coins :

If the CRP coin system has been selected in the [master control script](#) (it's on by default) player gold will be automatically converted to CRP gold coins. In addition, NPCs, including monsters, will generate random copper, silver, gold, electrum, and gold piece treasures instead of Bioware gold. NPC treasure scales upward based on the



upward based on the challenge rating of the creature. Likewise, CRP coins will generate in the CRP treasure chests instead of gold. CRP coins work with both Bioware random treasure generation systems, the default from the original game, plus the customizable treasure system included with the expansion packs.

Documenting Bioware's expansion treasure system is beyond the scope of this document, but if you want to learn it, you can do so by reading the comments in the script '*x0_i0_treasure*'.

If you are using CRP coins, it is important that you ALWAYS place your treasure chests from the blueprints provided for you in the custom palette:

Placeables / Custom / Treasure.

Sometimes your module may call for hundreds, if not thousands of coins to be in the inventory of a certain treasure chest. It would be very tedious to manually add that many coin stacks to the container's inventory, (and it's bad for the performance of the module). The treasure chests in the custom palette have an easy way to populate a specific number of coins into a chest if you prefer not to let a random treasure system generate the contents of the container.

To auto populate specific coin amounts, right click on

Name	Type	Value
CP	int	0
EP	int	0
GP	int	300
PP	int	90
SP	int	500

Name	Type	Value
SP	int	500

ReplaceAddDelete

..Breakable Items :

If
CRP_USE_BREAKABLE_ITEMS
has been selected in the [master control script](#) (it's on by default)
all fragile items within smashed
containers will have to save vs
the type of damage that
destroyed the chest or be
destroyed themselves. Suddenly
a good rogue becomes ALOT
more valuable to the party.



To the right you see the remains
of a treasure chest that was
blown to smithereines by a mage's
fireball. The chest originally
contained one arcane scroll, two
potions, and an emerald. All that
is left now is one pile of ashes
(the remains of the scroll which
failed its save vs fire damage),
two shattered items (the remains
of the potions), and the emerald,
which survived the blast.

If you are using Breakable Items,
it is important that you ALWAYS
place your treasure chests from
the blueprints provided for you in
the custom palette: **Placeables /
Custom / Treasure.**

You may modify the names and
appearances of the treasure
chests you place from the
custom palette, but you must
retain the scripts attached.

..Gems :



In order to allow you to more accurately duplicate treasure called for in your PnP conversion, approximately 50 custom gems types are included with this module. Custom gems can be found under the palette: **Items / Custom / Miscellaneous / Gems.**

The gems are simple items. Feel free to adjust the names and gp values of any to suit your needs.

If you select CRP_USE_GEMS in the [master control script](#), creatures and treasure chests will randomly generate CRP gems instead of the standard Bioware gems.

If you are using CRP gems, it is important that you ALWAYS place your treasure chests from the blueprints provided for you in the custom palette: **Placeables / Custom / Treasure.**

...

Merchants

For clarification, I'll be referring to two different types of merchants in this section. The first type is the actual NPC who you sell goods to a player. I'll refer to this type of merchant as 'merchant NPC'. The second type is the store waypoint that contains the merchant inventory. I'll refer to this type as 'store'.

If you are using the CRP coin system, you should ALWAYS create your stores using the CRP merchant template located in the custom merchant palette. If you intend to have your merchant NPC trade in goods that cost less than a gold piece, ie electrum, silver, and copper pieces, you should always create your merchant NPCs from the Generic Merchant template located under the custom creature palette: **Creatures / Custom / NPCs / Other**.

The template store has restrictions on it that only allow the store to buy gems. Edit the buying restrictions as needed, but if you are selling items for less than a gp, keep those item types in mind and do not allow your store to repurchase those item types or your players will be able to buy items over and over for less than a gp and then resell them to the same merchant for a gold piece or more. NEVER allow your store to purchase coins or you will have a major exploit on your hands. The CRP Store Template can be found under **Merchants / Custom / Merchants**.



..Customizing the Merchant NPC template :

You can edit a merchant NPC by either right clicking on the palette entry and selecting 'edit copy' to make a new blueprint, or by placing and directly editing an instance of the NPC in the game.

In either case, edit the merchant NPC's name, tag, appearance and other statistics to suit your needs. Do not change out the merchant's OnSpawn script however. You may modify the script if you feel comfortable doing so but be aware that you can break the custom merchant dialogs if you are not careful. The merchant has a default merchant conversation attached also.

Edit this conversation as needed for your new 'unique' merchant (although generic, it also works fine 'as is'). After you have made your changes to the conversation, save the modified conversation as a new name.

..Setting up Merchant Menus (as shown above):

You will have to go into the script editor to create merchant menus, but if you're not a scripter, don't worry! I've made it extremely easy for you to create the script you'll need. I've made an example script for you to work from and all you have to do is copy and paste a few lines of text and change out a couple values in those lines. For those of you who feel comfortable jumping right in, the example script is 'crp_merchant'. After modifying it you'll need to save your new script and name it the same as your merchant NPC's tag. Step by step instructions follow:

1. Note your Merchant NPC's tag

Jot down his or her tag onto a piece of paper. We'll need to know this tag name in just a minute.

2. Note the resrefs of the items your merchant will sell in the menu

You'll need these resref names when you create your new script, so put them on paper in advance.

3. Open 'crp_merchant' in the script editor

You'll enter one line of script for each item in your merchants menu. The lines are formatted exactly as follows:

```
SetMerchantItem( ListNumber, "ItemName", "Item resref", NumberOfCoins,  
TYPE_OF_COIN);
```

::Note -- the values of **TYPE_OF_COIN** can only be:
COIN_COPPER, COIN_SILVER, or COIN_GOLD.

Listed below is the exact script I used to make the menu shown in the picture above

```
#include "crp_inc_merchant"  
void main()  
{  
    SetMerchantItem(1, "Torch", "nw_it_torch001", 3,  
COIN_COPPER);  
    SetMerchantItem(2, "Iron Spike", "crpi_ironspikes",1,  
COIN_COPPER);  
    SetMerchantItem(3, "Rope", "crpi_rope", 3, COIN_SILVER);  
    SetMerchantItem(4, "Grappling Hook", "crpi_graphook", 5,  
COIN_SILVER);  
    SetMerchantItem(5, "Tinderbox", "crpi_tinderbox", 1,  
COIN_SILVER);  
    SetMerchantItem(6, "Bedroll", "crpi_bedroll", 2,  
COIN_SILVER);  
    SetMerchantItem(7, "Bandages", "crpi_bandages", 1,  
COIN_COPPER);  
}
```

4. Save the script

Save the script with the exact same name and the tag of your merchant NPC. You jotted this tag name down in step one. That's it, you're done. Now that wasn't so bad was it?

..Customizing the CRP store template :

A default store template has been provided in the custom merchant palette. If you are using CRP coins, you must build all your stores from this template. The CRP store template takes care of converting coins to Bioware gold while the player is browsing the merchant and converting them back when the player is done. You can right click on the template in the palette and select 'edit copy' to modify the merchant buy rates, the store's inventory, the tag of the store, and the

store's name to suit your needs. Then save the new store under a different resref.

The store template has buying restrictions set on it. These can be accessed from the restrictions tab of the store's properties window. Adjust as needed. NEVER allow your store to buy coins. You might also restrict the buying of items that you are selling for less than 1gp here or somewhere else.

Never directly edit the store template. Always select 'edit copy'.

...

Custom Player Activated Items

You will probably want to setup some custom player activated items in your module. This base module uses tag based scripting on player activated items (but not the full Bioware version of tag based scripting). You should not modify the script mod_activate. Instead, create your code in a script which is named exactly the same as the tag of your player activated item.

For example, you create a ring of recall. The ring's tag is "recall_ring". Just make a script called "recall_ring" as well. This script will hold your code that you would normally have added to the OnPlayerItemActivated event script.

...

Index of creature, placeable, and item palette locations

Secret Objects

Secret Doors	Placeables - Custom - Secret Objects
Secret Trapdoors	Placeables - Custom - Secret Objects
Ladders	Placeables - Custom - Miscellaneous
Dwarf Stonework Detector	Triggers - Custom - Secret Object Trigger
Elven Autosearch Check	Triggers - Custom - Secret Object Trigger

Tracks	Placeables - Custom - Secret Object
Hidden Sack	Placeables - Custom - Secret Objects
Hidden Compartment Initiator	Placeables - Custom - Secret Objects

Treasure

Treasure Containers	Placeables - Custom - Treasure
Coins	Items - Custom - Miscellaneous - Coins
Gems	Items - Custom - Miscellaneous - Gems

Merchants

Merchant NPC Template	Creatures - Custom - NPCs - Other
CRP Store Template	Merchants - Custom - Merchants

Doors

Wooden Doors	Doors - Custom - Universal - Wood
Stone Doors	Doors - Custom - Universal - Stone
Metal Doors	Doors - Custom - Universal - Metal
Special Doors	Doors - Custom - Tileset Specific

Respawning

Default Respawn Waypoint	Waypoints - Custom - Waypoints
-----------------------------	--------------------------------

Resting

Invisible Bed Object	Placeables - Custom - Civilization Int - Bedroom
Invisible Campfire Object	Placeables - Custom - Parks & Nature - Campsite
Allowed Resting Trigger	Triggers - Custom - Generic Trigger

Jumping

Toggle Player Jumping Trigger	Triggers - Custom - Generic Trigger
-------------------------------	-------------------------------------

Descriptions

Description Trigger	Triggers - Custom - Generic Trigger
Description Invisible Object	Placeables - Custom - Miscellaneous

Listening

Listening Trigger	Triggers - Custom - Generic Trigger
Sound Waypoint	Waypoints - Custom - Waypoints

Traps

Pit Trap Triggers	Triggers - Custom - Traps
Placeable Pit Traps	Placeables - Custom - Secret Object

DMFI

DM widgets	Items - Custom - Tutorial
Player widgets	Items - Custom - Tutorial

Player Items

Iron Spikes	Items - Custom - Miscellaneous - Other
Spike Hammer	Items - Custom - Weapons - Blunt - Hammers
Rope	Items - Custom - Miscellaneous - Other
Grappling Hook	Items - Custom - Miscellaneous - Other
Oilcloth Adventurers Tent	Items - Custom - Miscellaneous - Other
Burlap Adventurers Tent	Items - Custom - Miscellaneous - Other
Tinderbox	Items - Custom - Miscellaneous - Other
Bedroll	Items - Custom - Miscellaneous - Other
Bandages	Items - Custom - Miscellaneous - Other

Coin & Gem Pouch
Poisons
Cloth/Leather/Metal
Dyes

Items - Custom - Miscellaneous - Other
Custom - Miscellaneous - Poisons
Custom - Miscellaneous - Crafting/tradeskill