Cre library

Action functions

**clone\_creature (*list* creature, [*str* edits, *arrowlist* editstring, *bool* allow\_missing])**

**edit\_all\_creatures ([*str* edits, *arrowlist* editstring])**

**edit\_creature (*list* creature,[*str* edits, *arrowlist* editstring, *bool* tv, allow\_missing])**

**install\_creature (*list* creature, [*str* edits, location, locbase, *arrowlist* editstring, *bool* tv, allow\_missing]**

Standard editors; no special features

**make\_creature (*str* creature, [*str* edits, *arrowlist* editstring])**

The creature created has its animation, saving throws, and THAC0 set to the appropriate value for a creature of its type (unless they are manually set in the creation process). If an animation cannot be found, a warning will be displayed. (For non-PC classes we try RACE\_GENDER, and then just RACE, in ANIMATE.ids.)

Standard-form patch functions

Basic patch functions (absolute)

**name1\_string**  (0x8,LONG)

**name2\_string** (0xc,LONG)

**xp\_value** (0x14,LONG)

**hp\_current** (0x24,SHORT)

**hp\_max** (0x26,SHORT)

**animation\_code** (0x28,SHORT)

**effect\_type** (0x33,BYTE)

**color\_metal** (0x2c,BYTE)

**color\_minor** (0x2d,BYTE)

**color\_major** (0x2e,BYTE)

**color\_skin** (0x2f,BYTE)

**color\_leather** (0x30,BYTE)

**color\_armor** (0x31,BYTE)

**color\_hair** (0x32,BYTE)

**hide\_in\_shadows** (0x45,BYTE)

**ac\_natural** (0x46,SHORT)

**ac\_effective** (0x48,SHORT)

**thac0** (0x52,BYTE)

**attacks** (0x53,BYTE)

**save\_vs\_death** (0x54,BYTE)

**save\_vs\_wand** (0x55,BYTE)

synonym: **save\_vs\_wands**

**save\_vs\_poly** (0x56,BYTE)

synonym: **save\_vs\_polymorph**

**save\_vs\_breath** (0x57,BYTE)

**save\_vs\_spell** (0x58,BYTE)

synonym: **save\_vs\_spells**

**detect\_illusions** (0x64,BYTE)

**set\_traps** (0x65,BYTE)

**lore** (0x66,BYTE)

**open\_locks** (0x67,BYTE)

**move\_silently** (0x68,BYTE)

**find\_traps** (0x69,BYTE)

**pick\_pockets** (0x6a,BYTE)

**resist\_fire** (0x59,BYTE)

**resist\_cold** (0x5a,BYTE)

**resist\_electricity** (0x5b,BYTE)

**resist\_acid** (0x5c,BYTE)

**resist\_magic** (0x5d,BYTE)

**resist\_magic\_fire** (0x5e,BYTE)

**resist\_magic\_cold** (0x5f,BYTE)

**resist\_slashing** (0x60,BYTE)

**resist\_crushing** (0x61,BYTE)

**resist\_piercing** (0x62,BYTE)

**resist\_missile** (0x63,BYTE)

**level**  (0x234,BYTE) –synonym: **level1**

**level2**  (0x235,BYTE)

**level3**  (0x236,BYTE)

**str**  (0x238,BYTE) – synonym: **strength**

**str\_ex** (0x239,BYTE) – synonym: **strength\_ex**

**int** (0x23a,BYTE) – synonym: **intelligence**

**wis**  (0x23b,BYTE) –synonym: **wisdom**

**dex** (0x23c,BYTE) – synonym: **dexterity**

**con** (0x23d,BYTE) – synonym: **constitution**

**cha** (0x23e,BYTE) – synonym: **charisma**

**script\_override** (0x248,ASCII)

**script\_class** (0x250,ASCII)

**script\_race** (0x258,ASCII)

**script\_general** (0x260,ASCII)

**script\_default** (0x268,ASCII)

**specifics**  (0x274,BYTE)

**dv** (0x280,ASCII20)

**dialog**  (0x2cc,ASCII)

**morale** (0x23f,BYTE)

**morale\_break** (0x240,BYTE)

Basic patch functions (relative to offset)

*For items:*

**item\_resource** (0x0,ASCII)

**item\_charges\_1** (0xa, SHORT)

**item\_charges\_2** (0xc,SHORT)

**item\_charges\_3** (0xe, SHORT)

*For spells:*

**spell\_resource** (0x0,ASCII)

*For effects:*

**opcode** (0x8,LONG)

**effect\_resource** (0x28,ASCII)

–synonym: resource

**target** (0x1c,LONG)

**probability1** (0x24,SHORT)

**probability2**  (0x26,SHORT)

**duration**  (0x20,LONG)

**parameter1** (0x14,LONG)

**parameter2** (0x18,LONG)

*For others:*

**memorized\_spell** (0x0,ASCII)

**item\_slot** (0x0,SHORT)

IDS-based patch functions (all absolute)

**animation**  (0x28,SHORT) – from ANIMATE.ids

**allegiance** (0x270,BYTE) – from EA.ids

**general** (0x271,BYTE) – from GENERAL.ids

**race** (0x272,BYTE) – from RACE.ids

**class** (0x273,BYTE) – from CLASS.ids

**alignment** (0x27b,BYTE) – from ALIGN.ids

**kit\_vanilla** (0x244,LONG\_REVERSED) – from KIT.ids

(use kit instead)

Flag-based patch functions (all absolute)

**state\_sleeping** (0x20,0)

**state\_berserk** (0x20,1)

**state\_panic** (0x20,2)

**state\_stunned** (0x20,3)

**state\_invisible** (0x20,4)

**state\_helpless** (0x20,5)

**state\_frozen\_death** (0x20,6)

**state\_stone\_death** ( 0x20,7)

**state\_exploding\_death** (0x21,0)

**state\_flame\_death** (0x21,1)

**state\_acid\_death** (0x21,2)

**state\_dead** (0x21,3)

**state\_silenced** (0x21,4)

**state\_charmed** (0x21,5)

**state\_poisoned** (0x21,6)

**state\_hasted** (0x21,7)

**state\_slowed** (0x22,0)

**state\_infravision** (0x22,1)

**state\_blind** (0x22,2)

**state\_hidden** (0x22,3)

**state\_feebleminded** (0x22,4)

**state\_nondetection** (0x22,5)

**state\_improved\_invisibility** (0x22,6)

**state\_bless** (0x22,7)

**state\_chant** (0x23,0)

**state\_drawuponholymight** (0x23,1)

**state\_luck** (0x23,2)

**state\_aid** (0x23,3)

**state\_chantbad** (0x23,4)

**state\_blur** (0x23,5)

**state\_mirrorimage** (0x23,6)

**state\_confused** (0x23,7)

Composite functions

**hitpoints** (sets both current and total hp)

**level** (sets all three levels)

**ac** (sets both ac settings)

**saves** (takes as input five numbers separated by commas; sets the five saving throws in order death, wands, poly, breath, spells)

**gender** (takes as input an entry from GENDER.ids. Uses it to set the gender byte (0x275) and, if the entry is “FEMALE”, also sets the “sex” byte (0x237) to female (2)).

**read\_gender** also available, based only on 0x275.

**kit** (takes as input a kit from kit.ids; uses it to correctly set the kit.)

**read\_kit** also available.

Script-setting functions

**enforce script\_order (*str* arguments)**

“Arguments” should be either “script1 above script2” or “script1 below script2”. The positions in which those scripts occur are swapped, if necessary, to ensure that they occur in the specified order.

**insert\_script (*str* arguments)**

“arguments” should be either “script1 above script2”, or “script1 below script2” (“before” and “after” are acceptable synonyms for “above” and “below”). script1 is slotted in directly before or after script2, or not at all if script2 is not present. If necessary, other scripts are moved to make space. If there are insufficient free slots (i.e., none, and script1 does not occur) no insertion occurs and a warning is printed.

**insert\_script\_high (*str* arguments)**

Inserts “arguments” in the top (override) script slot, moving other scripts down if necessary; does nothing except print a warning if space is not available.

**insert\_script\_low (*str* arguments)**

Inserts “arguments” in the bottom (default) script slot, moving other scripts up if necessary; does nothing except print a warning if space is not available.

**locate script (*list* arguments -> *int* value)**

If the listed script is present, returns its location (0=override, etc). If not, returns -1.

**match\_script (*str* arguments -> *bool* value)**

Returns 1 if the argument is present in a script slot.

**strip script (*list* arguments)**

Removes all scripts in the list. If “arguments” is set to “all”, removes all scripts.

**swap\_script (*arrowlist* arguments)**

For each pair of scripts in “arguments”, replace all occurrences of the first with the second and vice versa.

Enforcement of features

**enforce\_animation (*str* arguments)**

SFO attempts to set an animation type for the creature, based on the following rules:

1. If [class]\_[gender]\_[race] is an ANIMATE.ids entry, use that. (Classes are mapped to their standard representations – so for Paladins, [class]=FIGHTER.)
2. Failing that, if [race]\_[gender] is an ANIMATE.ids entry, use that. (e.g., VAMPIRE\_FEMALE.)
3. Failing that, if [race] is an ANIMATE.ids entry, use that.
4. Failing that, leave the existing setting unchanged and print a warning.

If “arguments” is set to “only\_if\_new”, the animation type is only set if its existing value is -1.

**enforce\_hp (*str* arguments)**

The creature’s hit points (current and total) are adjusted to come in line with its class and level. The value calculated is chosen to lie between the minimum and maximum possible hit points: the position in that range is determined (as a % of distance towards the maximum) by the ini file setting “hitpoint\_percentage”, or 75% if that setting is not present. The syntax use for “arguments” is as follows:

* if “arguments” is “at\_worst”, hit points will only be adjusted if they are less good than the calculated value
* if “arguments” is “at\_best”, hit points will only be adjusted if they are better than the calculated value
* if “arguments” is “only\_if\_new”, hit points will only be adjusted if they are set to 65535. (You probably won’t use this; it’s called by make\_creature.)
* if “arguments” has any other value or is blank, hit points will be adjusted whenever they differ from the calculated value.

Note that the algorithm used in this function does not deal correctly with dual-classed creatures (they are treated as if multi-classed).

**enforce\_saves (*str* arguments)**

The creature’s saving throws are adjusted to come in line with the legal values for its class and level, using this syntax:

* if “arguments” is “at\_worst”, saving throws will only be adjusted if they are less good than the legal value
* if “arguments” is “at\_best”, saving throws will only be adjusted if they are better than the legal value
* if “arguments” is “within\_tolerance”, saving throws will only be adjusted if they are within a distance “tolerance” of their correct value, where “tolerance” is set by the “save\_tolerance” entry in the ini file. (The idea here is to correct small errors while leaving intentionally wildly-different values alone.)
* if “arguments” is “only\_if\_new”, saving throws will only be adjusted if they are set to -1. (You probably won’t use this; it’s called by make\_creature.)
* if “arguments” has any other value or is blank, saving throws will be adjusted whenever they are incorrect.

**enforce\_thac0 (*str* arguments)**

The creature’s THAC0 is adjusted to come in line with the legal value for its class and level, using the same syntax as enforce\_saves. (“tolerance” is set by the “thac0\_tolerance” entry in the ini file.)

Spell functions

**add\_known\_spells (*list* arguments)**

Each item on the list is added as a known spell, using the following syntax:

1. if the item is the IDS reference of a spell file, add that spell
2. if adding one *but not both* of “CLERIC” or “WIZARD” to the item produces the IDS reference of a spell file, add that spell (e.g., there is WIZARD\_FIREBALL but not CLERIC\_FIREBALL)
3. if both “CLERIC” and “WIZARD” can be added to produce an IDS reference, add the WIZARD one and print a warning
4. if the item is the resource name of an existing spell, add that spell.
5. Otherwise, fail.

Level and type are inferred from the spell file.

If the argument is “all”, instead give the creature all cleric/druid spells appropriate to their class, level and alignment, deleting any illegal cleric/druid spells in the process.

**add\_spells (*list* arguments)**

Each item on the list is added as a memorized spell, using the same syntax as **add\_known\_spells.**

If a nonnegative integer is added in parentheses (as in: FIREBALL(3)), add that many copies of the spell.

**remove\_spells (*list* arguments)**

Process each element in “arguments” as follows:

1. If set to “all”, remove all known and memorized spells.
2. If set to “priest”, “wizard” or “innate”, remove all known and memorized spells of that type. (“cleric” and “mage” are acceptable synonyms for “priest” and “wizard, respectively).
3. Otherwise, treat the element as an individual spell, using the same syntax as **add\_known\_spells**, and remove it from the known and memorized list.

Immunity functions

**immunity\_to\_icon (*list* arguments)**

The creature is made immune (opcode 169) to the display icons in the list.

**immunity\_to\_opcode (*list* arguments)**

The creature is made immune (opcode 101) to each opcode on the list.

**immunity\_to\_spell (*list* arguments)**

The creature is made immune (opcode 206) to each spell on the list, using the same syntax as **add\_known­\_spells.** (i.e., “FIREBALL”, “WIZARD\_FIREBALL”, and “SPWI304” are all legal ways to give immunity to fireball.

**immunity\_to\_string (*list* arguments)**

The creature is made immune (opcode 267) to each strref in the list.

Others

**movement (*int* arguments)**

Sets the movement rate to “arguments” via opcode 176.

say\_sounds (*str* arguments)

**steal\_colors\_from (*str* arguments)**

If “arguments”.cre exists, sets the colorscheme of the creature being patched to match that creature. If not, prints a warning.

**steal\_sounds\_from (*str* arguments)**

If “arguments”.cre exists, sets the soundste of the creature being patch to match that creature. If not, prints a warning.