Effect Database for RoundMaster API

Effect-DB is a database character sheet created, used and updated by the **RoundMaster API** (see separate documentation). The database holds macros as Ability Macros that are run when certain matching statuses are placed on or removed from tokens (see Roll20 Help Centre for information on Ability Macros and Character Sheet maintenance). The macros have certain defined parameters dynamically replaced when run by RoundMaster, providing the token & character IDs and names, and values such as AC, HP and Thac0, available for manipulation.

Effect macros are primarily intended to act on the Token and its variables, but can also act on the represented Character Sheet. A single Character Sheet can have multiple Tokens representing it, and each of these are able to do individual actions using the data on the Character Sheet jointly represented. However, if such multi-token Characters / NPCs / creatures are likely to encounter effects that will affect the Character Sheet they must be split with each Token representing a separate Character Sheet, or else the one effect will affect all tokens associated with the Character Sheet, whether they were targeted or not! In fact, **it is recommended that tokens and character sheets are 1-to-1** to keep things simple.

The recommended Token Bar assignments for all APIs in the Master Series are:

Bar1 (Green Circle): Armour Class (AC field) – only current value  
 Bar2 (Blue Circle): Base Thac0 (thac0-base field) before adjustments – only current value  
 Bar3 (Red Circle): Hit Points (HP field) – current & max

It is recommended to use these assignments, and they are the bar assignments set by the **CommandMaster API** if its facilities are used to set up the tokens. All tokens must be set the same way, whatever way you eventually choose.

These assignments can be changed in each API, by changing the *fields* object near the top of the API script. See individual API documentation for details of how to do this.

**Macro Parameter Fields**

Dynamic parameters are identified in the macros by bracketing them with two carets: **^^parameter^^**. The standard Roll20 syntax of @{selected|…} is not available, as at the time the macros run the targeted token may not be selected, and @{character\_name|…} will not enable the token to be affected (especially where the Character Sheet is represented by more than one token). The ^^…^^ parameters always relate to the token on which a status has been set, and the Character Sheet it represents. Currently available parameters are:

**^^tid^^** TokenID  
**^^tname^^** Token\_name  
**^^cid^^** CharacterID  
**^^cname^^** Character\_name  
  
**^^ac^^** Armour Class value (order looked for: a token bar, Character Sheet AC field, MonsterAC)  
**^^ac\_max^^** Maximum value of AC, wherever it is found  
**^^token\_ac^^** The token field name for AC value field, if set as a token bar  
**^^token\_ac\_max^^** The token field name for AC max field, if set as a token bar  
  
**^^thac0^^** Thac0 value (order looking: a token bar, Character Sheet Thac0\_base field, MonsterThac0)  
**^^thac0\_max^^** Maximum value of Thac0, wherever it is found  
**^^token\_thac0^^** The token field name for Thac0 value field, if set as a token bar  
**^^token\_thac0\_max^^** The token field name for Thac0 max field, if set as a token bar  
  
**^^hp^^** HP value (order looked for: a token bar, Character Sheet HP field)  
**^^hp\_max^^** Maximum value of HP, wherever it is found  
**^^token\_hp^^** The token field name for HP value field, if set as a token bar  
**^^token\_hp\_max^^** The token field name for HP max field, if set as a token bar  
  
**^^bar1\_current^^** Value of the token Bar1\_value field  
**^^bar2\_current^^** Value of the token Bar2\_value field  
**^^bar3\_current^^** Value of the token Bar3\_value field

This allows most data on both the token and the character sheet to be accessed. For example **@{^^cname^^|strength}** will return the strength value from the represented character sheet. Of course all loaded RPGMaster series API commands are available.

Two other APIs from the Roll20 Script Library are extremely useful for these macros, and indeed are used by many of the provided APIs: **ChatSetAttr API** from joesinghaus allows easy and flexible setting of Character Sheet attributes. **Tokenmod API** from The Aaron supports easy setting and modifying of Token attributes. Combined with the dynamic parameters above, these make for exceptionally powerful real-time effects in game-play.

**Effect Macro qualifiers**

Each effect macro runs when a particular status event occurs. Here is the complete list of effect macro status name qualifiers that can be used. Each of these is appended to the status whenever the status experiences the relevant event, and an effect macro with that name searched for and run if found:

statusname**-start** The status is created on a token  
statusname**-turn** Each round the status has a duration that is not zero  
statusname**-end** The status duration reaches zero

These effect across are triggered for weapons when certain events take place:

weaponname**-inhand** A weapon is taken in-hand (triggered by AttackMaster API --weapon command)  
weaponname**-dancing** A weapon starts dancing (triggered by AttackMaster API --dance command)  
weaponname-**sheathed** A weapon is sheathed (out of hand - triggered by AttackMaster --weapon cmd)

**Examples of Effect Macros**

Here is an example of an effect macro that runs when a Faerie fire (twilight form) status is placed on a token. The following --target command might be run to set this status, with the caster token selected:

!rounds --target area|@{selected|token\_id}|&{target|Select first target|token\_id}|Faerie-Fire-twilight|[[4\*@{selected|Casting-Level}]]|-1|Outlined in dim Faerie Fire, 1 penalty to AC|aura

This will result in the following effect macro being run when the first token is targeted:

**Faerie-fire-twilight-start**

!token-mod --ignore-selected --ids ^^tid^^ --set ^^token\_ac^^|+1  
^^tname^^ is surrounded by Faerie Fire, and becomes easier to hit

This uses the Tokenmod API to increase the AC number of the targeted token by 1 (making it 1 wose), and then display a message to all Players stating the name of the targeted token, and the effect on it. This will be run for each token targeted, and will be individual to each. **Note:** the tokens are not ‘selected’ in Roll20 terms, and so *@{selected|…}* will not work

When the Faerie Fire status counts down to zero, the following effect macro will be run on each of the tokens it was applied to:

**Faerie-fire-twilight-end**

!token-mod --ignore-selected --ids ^^tid^^ --set ^^token\_ac^^|-1  
^^tname^^ has lost that glow and is now harder to aim at

Again, the Tokenmod API is used to decrease the token AC and a message issued confirming what has happened. If messages should only be sent to the Player(s) controlling the character represented by the token, use */w “^^cname^^”* before the message. If the message is only for the gm, use */w gm*.

A more complex example is a Quarterstaff of Dancing, that uses the complete suite of possible effect macros and certain aspects of the **AttackMaster API** functionality triggered by Weapon table field settings. The first macro is triggered by AttackMaster API when a Character takes a Quarterstaff-of-Dancing in hand to use as a weapon:

**Quarterstaff-of-Dancing-inhand**

!rounds --addtargetstatus ^^tid^^|Quarterstaff-of-Dancing|4|-1|Quarterstaff not yet dancing so keep using it|stopwatch

This command sets a status marker on the Token of the Character taking the Quarterstaff in hand, and sets a countdown of 4 rounds, running the next effect macro in each of those rounds:

**Quarterstaff-of-Dancing-turn**

!attk --quiet-modweap ^^tid^^|quarterstaff-of-dancing|melee|+:+1 --quiet-modweap ^^tid^^|quarterstaff-of-dancing|dmg|+:+1  
/w “^^cname^^” Updating the quarterstaff +1 to attk & dmg

This command then runs each round as the Quarterstaff-of-Dancing status counts down, and uses the *!attk --quiet-modweap* command to gradually increment the magical to-hit and dmg plus, round by round. Once the countdown reaches zero, the next effect macro is run:

**Quarterstaff-of-Dancing-end**

!attk --dance ^^tid^^|Quarterstaff-of-Dancing

This calls an AttackMaster API command to start the weapon dancing, resets the weapon to its specs that it starts dancing with, and calls the next effect macro:

**Quarterstaff-of-Dancing-dancing**

!rounds --addtargetstatus ^^tid^^|Dancing-Quarterstaff|4|-1|The Quarterstaff is Dancing by itself. Use this time wisely!|all-for-one  
!attk --quiet-modweap ^^tid^^|quarterstaff-of-dancing|melee|sb:0 --quiet-modweap ^^tid^^|quarterstaff-of-dancing|dmg|sb:0

This places a new status marker on the token representing the Character with the dancing weapon (note the new status name *Dancing-Quarterstaff*), and resets the Strength Bonus flags for the weapon - a dancing weapon can’t have the Strength Bonus of the wielder. As each round now passes, the following different status effect macro is run:

**Dancing-Quarterstaff-turn**

!attk --quiet-modweap ^^tid^^|quarterstaff-of-dancing|melee|+:+1 --quiet-modweap ^^tid^^|quarterstaff-of-dancing|dmg|+:+1

As per the previous *-turn* effect macro, this increments the magical plusses on To-Hit and Dmg, round by round. It has to have a different name, as the *-end* effect macro does different actions:

**Dancing-Quarterstaff-end**

!attk --dance ^^tid^^|Quarterstaff-of-Dancing|stop

This uses the AttackMaster API command to stop the Quarterstaff from dancing. As can be seen from the above, quite complex sequences of effect macros can be created.