DC Script Package - DC Vars Ver 2

Caskey, Damon V.

2015-12-23

For use with the OpenBOR engine.
Source License (required for use)

This script package adds wrappers for retrieval (get) and storage (set) for all non-inline OpenBOR variant scopes. Set and get functions are type specific, and typing is strongly enforced. That is to say, attempting to set a string with a float set function will produce an error alert and a default string will be used instead.

This package was crafted with the following goals:

- Insert a more C authentic, strongly typed behavior into OpenBOR script.
- Providing for specific error trapping to replace the unforgiving, generic, and sometimes difficult to decipher errors from OpenBOR's scripting engine when dealing with erroneous values.
- Provide useful default values for each variable type.

Change Log

Version 2 – 2016-05-20

- Remove all indexed scope functions. I was entirely mistaken on indexed variable operations. Indexed variables are depreciated as of OpenBOR version 3.902 and are now simply wrappers for the global scope. There's no point supporting them any further.
- All script variable functions replaced with local functions. Similar to global verses local, script variable functions in OpenBOR are depreciated wrappers as of version 3.902.
- Updated instructions.

To Do

- Real time default value override for all variable types.
- Add a persistent scope variable set by wrapping text file streams.

Installation

1. Install any required dependencies (see requirements). See their instructions for specifics.
2. Download *dc_vars* and unzip. Place the *dc_vars* folder into your *data/scripts* folder.
3. Adjust the constants located in *data/scripts/dc_vars/settings.h* to suit your needs. Most constants are self-explanatory, or will have explanations and instructions as needed in the file itself.

4. Add the following to any script you want to use this package with: *#include data/scripts/dc_vars/main.c*
5. All functions and constants are now available in your scripts. Call them as needed.

Note that if you choose to place the *dc_vars* folder or any dependencies in alternate locations, you will have to adjust the *#include* and *#import* directives in each individual file to match.

Requirements

- OpenBOR 3.922+
- dc_error

USE

Use is straight forward. Call the functions provided to convert values or get and set variables among the various scopes. A suite of validation functions is also provided. If you are unsure about the scopes for variables within openBOR, please see here for a full explanation.

Conversion

- *float x = dc_vars_int_to_float(int value)* – Convert a whole number to floating decimal.

Validation

All validation functions return a Boolean TRUE constant if the validation succeeds or FALSE constant on failure.

- *int x = dc_vars_is_empty(void value)* – TRUE if *value* is NULL() or not set.
- *int x = dc_vars_is_entity(void entity)* – TRUE if *entity* is a valid pointer to a currently existing entity.
- *int x = dc_vars_is_float(float value)* – TRUE if *value* is a valid floating decimal.
- *int x = dc_vars_is_integer(int value)* – TRUE if *value* is a valid whole integer.
- *int x = dc_vars_is_key(void value)* – TRUE if *value* can be used as a variable identifier (valid string, whole integer, or pointer). Note that technically OpenBOR will accept any type of value as an identifier. This function is to enforce the most recommended types.
- *int x = dc_vars_is_numeric(void value)* – TRUE if *value* is a valid number (whole integer or floating decimal).
- *int x = dc_vars_is_pointer(void value)* – TRUE if *value* is a valid pointer.
- *int x = dc_vars_is_string(void value)* – TRUE if *value* is a valid string.

Entity Scope

The following functions operate on entity scope variables and thus require an entity pointer along with a key.

- *int x = dc_vars_clear_entity_var(void entity, void key)* – Destroy a previously set entity scope variable of any type. Returns a Boolean TRUE value if successful, FALSE otherwise. Accepts the following arguments:

    - *entity* - The entity this variable is associated with. Must be a valid entity pointer, but non existing entities are allowed, since it is possible the entity was destroyed after variables had been associated with it.
    - *key* – The variable's identifier. Must be a valid pointer, whole integer, or string.
- *float x = dc_vars_get_entity_float(void entity, void key)* – Get a previously set entity scope floating decimal. If the stored value is not a floating decimal or does not exist at all (NULL or empty), a default floating decimal value will be returned instead. See constants list for the default values. Accepts the following arguments:
    - *entity* - The entity this variable is associated with. Must be a valid entity pointer, but non existing entities are allowed, since it is possible the entity was destroyed after variables had been associated with it.
    - *key* – The variable's identifier. Must be a valid pointer, whole integer, or string. Obviously if you pass an identifier that was not previously set, there will be no associated value.
- *int x = dc_vars_set_entity_float(void entity, void key, float value)* – Set an entity scope floating decimal. Returns a Boolean TRUE value if successful, FALSE otherwise. Accepts the following arguments:
    - *entity* – The entity this variable is associated with. Must be a valid entity pointer, and the entity must currently exist in play.
    - *key* - The variable's identifier. Must be a valid pointer, whole integer, or string.
    - *value* - The value to set. Must be a floating decimal.
- *int x = dc_vars_get_entity_int(void entity, void key)* – See *dc_vars_get_entity_float*, but for whole integers.
- *Int x = dc_vars_set_entity_int(void entity, void key, int value)* – See *dc_vars_set_entity_float* but for whole integers.
- *void x = dc_vars_get_entity_pointer(void entity, void key)* – See *dc_vars_get_entity_float*, but for pointers.
- *int x = dc_vars_set_entity_pointer(void entity, void key, void value)* – See *dc_vars_set_entity_float* but for pointers.
- *void x = dc_vars_get_entity_string(void entity, void key)* – See *dc_vars_get_entity_float*, but for strings.
- *int x = dc_vars_set_entity_string(void entity, void key, void value)* – See *dc_vars_set_entity_float* but for strings.

Global Scope

Operates on global scope vars. See Indexed Scope for details.

- *int x = dc_vars_clear_global_var(void key)*
- *int x = dc_vars_get_global_float(void key)*
- *int x = dc_vars_set_global_float(void key, float value)*
- *int x = dc_vars_get_global_int(void key)*
- *int x = dc_vars_set_global_int(void key, int value)*
- *void x = dc_vars_get_global_pointer(void key)*
- *int x = dc_vars_set_global_pointer(void key, void value)*

- *void x = dc_vars_get_global_string(void key)*
- *int x = dc_vars_set_global_string(void key, void value)*

Local Scope

Operates on local scope vars. See Indexed Scope for details.

- *int x = dc_vars_clear_local_var(void key)*
- *int x = dc_vars_get_local_float(void key)*
- *int x = dc_vars_set_local_float(void key, float value)*
- *int x = dc_vars_get_local_int(void key)*
- *int x = dc_vars_set_local_int(void key, int value)*
- *void x = dc_vars_get_local_pointer(void key)*
- *int x = dc_vars_set_local_pointer(void key, void value)*
- *void x = dc_vars_get_local_string(void key)*
- *int x = dc_vars_set_local_string(void key, void value)*