PLATINUMSOFTWARE

Grape SQL Function Reference

Table of Contents

1. GRAPE SQL FUNCTIONS

1.1 API result functions

This functions deal with the creation of standardized API results (in JSON format) to be sent back to the API call. They can be found in api_result_json.sql

NAME	PARAMETERS	DESCRIPTION
api_result_error	*message* TEXT *code* INTEGER *info* JSON	Returns a standardized JSON error object with stats as "ERROR" and the other fields populated. Example: `{"status":"ERROR", "message":"Message", "code": -2, "error": {} }`
api_error	*message* TEXT *code* INTEGER *info* JSON	Overload for api_result_error
api_error		With no arguments, an "Unknown error" message will be generated
api_error_invalid_input		Similar to calling `api_result_error("Invalid input", -2)`
api_success	*keys* TEXT[] *values* TEXT[] *types* TEXT[]	This function will construct a JSON object containing at least one field, "status" with the value "OK". The 3 input parameters should be arrays containing additional keys, values and the associated types (n/i/number/integer, j/json or nothing for text).
api_success	*keys* TEXT[] *values* TEXT[]	
api_success	*key* TEXT *value* INTEGER	
api_success	*key1* TEXT *value1* INTEGER *key2* TEXT *value2* INTEGER	Create an API result success JSON object with two integer fields added.
api_success	*key* TEXT *value* JSON	Create an API result success JSON object with a JSON field merged into the result.
api_success		Returns a API result object with a status field set to "OK".

1.2 Data importing functions

NAME	PARAMETERS	DESCRIPTION
data_import_insert		
data_upload_done		
data_import_row_insert		

1.3 JSON helpers

1.3.1 json2xml(_data JSON, _root TEXT)

Filename: json2xml.sql

1.3.2 json_diff (_old JSONB, _new JSONB)

Filename: json_diff.sql

1.3.3 json_diff (_old JSON, _new JSON)

Filename: json_diff.sql

1.3.4 json_object_diff (_old JSONB, _new JSONB)

Filename: json_diff.sql

1.3.5 json_array_diff (_old JSONB, _new JSONB)

Filename: json_diff.sql

1.3.6 json_to_composite_type_text(target_schema TEXT, target_type TEXT, data JSON)

1.3.7 json_to_composite_type(target_schema TEXT, target_type TEXT, data JSON)

This function will populate a custom type from a JSON object. What sets it apart from the functions available in PostgreSQL, is the fact that it supports complicated multi-level nested objects. **Filename:** json_to_composite_type_text.sql

1.3.8 cast_json_array_to_int_array (JSON)

Provides an implicit cast from JSON to INT[] (cast_json_array_to_int_array.sql).

```
# select cast_json_array_to_int_array('[1,2,3]'::JSON);
 cast_json_array_to_int_array
 ------------------------------
  {1,2,3}
```

1.3.9 cast_json_array_to_text_array (JSON)

Provides an implicit cast from JSON to TEXT[] (cast_json_array_to_text_array.sql).

1.4 List query

Grape's list_query call provides an easy way to retrieve rows from a table. Before the contents of a table can be retrieved this way it needs to be added to a whitelist. This functions can be found in list_query.sql

1.4.1 list_query(JSON)

This function returns row from a database table. The following input fields are recognized:

- tablename
- schema (optional) TEXT
- sortfield (optional) TEXT
- sortorder (optional) TEXT DESC
- limit (optional) INTEGER default 50
- offset (optional) INTEGER default 0
- filter (optional) array of fields:
- field TEXT
- operand TEXT of '=', '#x003E;', '#x003C;', '#x003E;=', '#x003C;=', 'LIKE', 'ILIKE', 'IS_NULL', 'IS_NOT_NULL', 'IN'
- value text

1.4.2 list_query_whitelist_add(_schema text, _tables TEXT[], _roles TEXT[])

Adds tables to the whitelist for use in grape list_query. Users must be in _roles to be able to access the data in the table

1.4.3 list_query_whitelist_delete(_schema TEXT, _tablename TEXT)

Removes a table from the whitelist.

1.5 Reports

This functions can be found in reports.sql.

1.5.1 save_report ( )

1.5.2 save_report ( )

1.5.3 save_report (JSON)

1.5.4 execute_report (_report_id INTEGER, _parameters JSON)

1.5.5 execute_report (JSON)

1.6 User and session related functions

1.6.1 toggle_user (JSON)

Filename: user.sql

1.6.2 user_save (JSON)

Save user field. Also used to add a new user to the system. **API Call:** POST /grape/user/save
Filename: user.sql

1.6.3 user_save_password (JSON)

Filename: user.sql

1.6.4 username (_user_id INTEGER)

Returns a TEXT field containing the username matching the user ID provided
Filename: user.sql

1.6.5 user_id_from_name (_username TEXT)

Returns the user ID (or NULL if not found) for the user matching the username provided.
Filename: user.sql

1.6.6 user_id_from_fullnames(_fullnames TEXT)

Returns an integer containing the user ID matching the full names provided
Filename: user.sql

1.6.7 username_from_fullnames(_fullnames TEXT)

Returns the username for the user matching the fullnames provided
Filename: user.sql

1.6.8 hash_user_password (_user_id INTEGER)

Hashes a password for user and updates the user table afterwards.

- If the hash length is the same as the password length and the password starts with a '$' sign, it is assumed that the password is already hashed and the update is ignored (-1 is returned)
- If grape.setting passwords_hashed isn't true, nothing is done (return -2)

- On success 0 is returned

1.6.9 current_user_roles()

Returns a list of all roles the current user belongs to **Filename:** user.sql
Filename: user.sql

1.6.10 hash_user_password (_username TEXT)

Overload function for ***hash_user_password (_user_id INTEGER)***, taking a username instead of a user ID as input.
Filename: user.sql

1.6.11 current_user_id()

Returns the integer value of the current session's "grape.user_id" setting. This is typically set with grape before any API call is called.
Filename: current_user_id.sql

1.6.12 check_session_access (_session_id TEXT, _check_path TEXT, _check_method TEXT)

This function performs access control on an API call (based on the path and session ID). It is automatically called by the express app before any API call is performed:

1. Check that the path has access control on it. If it cannot be found, the grape setting ***default_access_allowed*** is checked, and if true, access will be granted. If not, it will be denied and code 9 will be returned
2. If the path has a role 'guest' granted access to it, everyone will be allowed (even if the session is invalid)
3. If the session is invalid, access will be denied and code 1 returned
4. If the path has a role 'all', only, and all, valid sessions will be granted access
5. If the user has access granted to the access path's role, access is granted
6. If all the above fails, access is denied with code 2

Filename: session_access_path.sql

1.6.13 set_session_user_id (JSON)

Sets the current database session's ***grape.user_id*** config variable for use in stored procedures to identify the current user.
Filename: session_access_path.sql
Returns: JSON

1.6.14 session_insert (JSON)

This function inserts a new session for a valid username and password provided.
Filename: session.sql

1.6.15 logout (JSON)

Filename: session.sql

1.7 Other utility functions

NAME	PARAMETERS	DESCRIPTION
month_diff	*_d1* DATE *_d2* DATE	Returns an integer containing the number of months between the two dates provided. If the first parameter is after the second (higher date), the return value will be negative.
set_value	*_name* TEXT *_value* TEXT	Sets the value (insert if new, replace if exist) in the key-value pair table grape.setting returning _value.

NAME	PARAMETERS	DESCRIPTION
get_value	***_name*** TEXT ***_default_value*** TEXT	Gets the value for setting ***_name***, and if not found it will return ***_default_value***. Defined in setting.sql
generate_uuid		Generates a unique UUID (for example b1086d35-e973-4356-3adc-2eeb6f4963e2). Defined in uuid.sql
array_lowercase	TEXT[]	
clean_telephone_number	***_tel*** TEXT	
random_string	***length*** INTEGER	Generates a random string of ***length*** length. Defined in random_string.sql