

API Documentation u_json

Object oriented JSON in PB

Content

Introduction	3
Code Style	3
How it works	3
Error handling	3
Functions	4
Examples	6
Parse	6
Generate	7
Manipulate	7

Introduction

JSON handling in PowerBuilder is fine to handle relational data, as long as it comes in the format defined by Apeon. But often, we need to work with data from an external interface, that's where `u_json` comes in handy: You can parse, manipulate and/or generate all JSON files with the same object, in a true object oriented way.

Code Style

`u_json` is coded according to the coding guidelines of Informaticon. As PowerBuilder is case insensitive, we only use lower case letters in the snake case practise.

How it works

`u_json` works with it's own JSON structure. When a file is parsed, it will be completely parsed with `jsonparser` and then be available in the object oriented way. When a json string is generated, a `jsongenerator` object will be created from the object structure.

Each array, object or value is considered a node, each node is represented by an `u_json` object. Array and object nodes contain an array of `u_json` objects. With this setup, the JSON information can be easily accessed and manipulated using the dot notation (see examples).

Error handling

Each function in `u_json` can throw an exception. This makes sure every error message is caught. The message can be derived with `exception.getmessage()`.

Functions

Function	Return	Description
of_add_node()	u_json	Adds a new array element
of_add_node(u_json au_json)	u_json	Adds au_json as a new array element
of_add_node(string as_key)	u_json	Adds a new object element
of_add_node(string as_key, u_json au_json)	u_json	Adds au_json as a new object element
of_change_key()	-	Changes the key of the current element
of_delete_node(long al_index)	-	Deletes a node (object or array element) with the index al_index
of_delete_node(long as_key)	-	Deletes an object element with by key
of_get_count()	long	Returns the number of elements in an object or array
of_get_json_string()	string	Returns the generated JSON string, starting from the current node
of_get_key()	string	Returns the key of the current node, if it is an object element
of_get_node(long al_index)	u_json	Returns the node of index al_index (position in an object or array)
of_get_node(string as_key)	u_json	Returns the node of as_key (object key)
of_get_parent()	u_json	Returns the parent node (if the current node isn't root)
of_get_type()	jsonitemtype	Returns the type of the current node
of_get_value_blob()	blob	Returns the value of the current node, if it isn't an array or object.
of_get_value_boolean()	boolean	
of_get_value_date()	date	
of_get_value_datetime()	datetime	
of_get_value_number()	number	
of_get_value_string()	string	
of_get_value_time()	time	
of_key_exists(as_key)	boolean	Returns whether a key with the name as_key exists in the current node

of_load_file(string as_path)	-	Loads a file at as_path and parses it.
of_load_string(string as_json)	-	Loads a JSON string and parses it
of_replace_node(long al_index, u_json au_node)	-	Replaces a node (array, object or value) at al_index with a new one (could be a different type, works recursively)
of_replace_node(string as_key, u_json au_node)	-	Same as above, but finds the node to replace by key instead of index
of_reset()	-	Resets the value of the current node, including all children. The node itself remains as it had been created using of_add_node()
of_save_into_file(string as_filename)	-	Generates the JSON and writes it to a file located in as_filename
of_set_value(boolean abo_value)	-	Sets the value of the current node.
of_set_value(double ado_value)	-	
of_set_value(string as_value)	-	
of_set_value_null()	-	Sets the value of the current node to NULL

Examples

To show how it works, it's easiest to see with a few examples. Let's use an example json file as follows:

```
{
  "file_name": "example.json",
  "number": "5",
  "articles": [
    {
      "name": "watch",
      "price": 208.3,
      "stock": true
    },
    {
      "name": "phone",
      "price": 108.55,
      "stock": false
    }
  ]
}
```

The examples are done without error handling - just wrap the whole code with a try catch statement.

Parse

```
u_json lu_json
```

```
lu_json = create u_json
lu_json.of_load_file("example.json")
lu_json.of_get_node("articles").of_get_count() // returns 2
lu_json.of_get_node("articles").of_get_node(1).of_get_node("name").of_get_value_string() //
returns "watch"
```

```
long ll_i
u_json lu_article
for ll_i = 1 to lu_json.of_get_node("articles").of_get_count()
  lu_article = lu_json.of_get_node("articles").of_get_node(ll_i)
  // loop through articles like that
next

// get a partial JSON string
lu_json.of_get_node("articles").of_get_node(1).of_get_json_string()
```

Generate

This example shows how to create the JSON above.

```
u_json lu_json, lu_articles_node, lu_article
long ll_i
string ls_name[]
double ldo_price[]
boolean lbo_stock[]

ls_name = {"watch", "phone"}
ldo_price = {208.3, 108.55}
lbo_stock = {true, false}

lu_json = create u_json
lu_json.of_add_node("file_name").of_set_value("example.json")
lu_json.of_add_node("number").of_set_value(5)
lu_articles_node = lu_json.of_add_node("articles")

for ll_i = 1 to upperbound(ls_name)
    lu_article = lu_articles_node.of_add_node()
    lu_article.of_add_node("name").of_set_value(ls_name[ll_i])
    lu_article.of_add_node("price").of_set_value(ldo_price[ll_i])
    lu_article.of_add_node("stock").of_set_value(lbo_stock[ll_i])
next

lu_json.of_get_json_string() // returns the example JSON string
```

Manipulate

There is practically no limit in JSON manipulation, here some simple examples.

```
// parse it, change the price of an article and get the JSON String again
lu_json.of_load_string(ls_json)
lu_json.of_get_node("articles").of_get_node(2).of_get_node("price").of_set_value(100.5)
lu_json.of_get_json_string()

// combine all articles from two json files (like the example above)
lu_json1 = of_load_string(ls_json1)
lu_json2 = of_load_string(ls_json2)
for ll_i = 1 to lu_json2.of_get_node("articles").of_get_count()
    lu_json1.of_get_node("articles").of_add_node(lu_json2.of_get_node("articles").of_get_node(ll_i))
next
lu_json1.of_get_json_string() // returns the json with combined articles
```