Rocket software

Rocket Uniface Library 10.4

structToJson

Convert a Struct to a JSON text, as an object or an array.

`structToJson` {`/bmp`} {`/whitespace`} *JsonTarget*, *StructSource*

Example: `structToJson /whitespace vOutput, vStruct`

Qualifiers

Table: Qualifiers

Qualifier	Description
`/bmp`	Specify that all characters outside of the Unicode BMP range are escaped in the JSON way: `/u`*XXXX* `/u`*YYYY* , where *XXXX* and *YYYY* are the hexadecimal values of the surrogate pair for the Unicode character that is outside the BMP range.
`/whitespace`	Write JSON string with additional whitespace for readability, using indentation to indicate levels of nesting. By default, the JSON string is written without extra whitespace; it is included only when inside double quotes.

Parameters

Table: Parameters

Parameter	Data Type	Description
JsonTarget	`string`	Variable, parameter, or field to hold the returned JSON data
StructSource	`struct` or `any`	Variable or parameter that references the Struct to be transformed

Return Values

If the conversion was successful, the JSON text is returned in *JsonTarget*. This can then be written to a file using `filedump`.

Table: Values returned in `$status`

Value	Meaning
0	Conversion was successful. However, the JSON produced may not be what is expected if non-fatal errors occurred during conversion. Warnings about such conditions are made available in `$procReturnContext`
<0	Conversion failed. `$procerror` contains the exact error.

Common Errors

Table: Values Commonly Returned by `$procerror`

Error Number	Error Constant	Meaning
`-1905`	`STRUCTERR_INPUT`	Input struct data is not valid. For example, the `struct` variable may have been

Error Number	Error Constant	Meaning
		declared, but not initialized.

Return Context

`structToJson` sets `$procReturnContext` when it encounters non-fatal errors encountered when transforming the Struct to JSON, such as:

- Duplicate names of JSON object members
- Elements of an array that have names. The names are ignored when writing the JSON text.
- Members of an object that have no name. An empty string is given as a name.
- jsonClass holds an unknown value.
- Non-scalar Structs that are annotated with **jsonDataType**.
- A scalar Struct that is tagged as number but cannot be converted to a JSON number. In this case, `structToJson` falls back to the Uniface data type.

Use

Allowed in all component types.

Description

Use `structToJson` to convert a Struct to a JSON text. The *JsonTarget* must be a variable, parameter, of field, so use `filedump` or `lfiledump` to write it to a file.

The top-level Struct must represent a JSON object or array.

If the root Struct has jsonClass annotation, the value determines whether `structToJson` transforms the root Struct to a JSON object or an array. It then recursively processes all members of the Struct.

- For a JSON object, any member without a name is given the name `""`, and a warning is placed in `$procreturncontext`).
- For a JSON array, any member names are ignored and a warning is placed in `$procreturncontext`).

If the root Struct does not have a jsonClass tag, the presence of names is used to determine how members at the root level are handled:

- If all the members have names, a JSON object is written and the named members are recursively processes.
- If at least one of the members at this level does not have a name, a JSON array is written, and the members are recursively processed; the names of members that have a name is ignored.
- If there are no members, the `null` keyword is written. However, if the root Struct has no members (that is, the Struct is empty), an empty object {} is written because `null` is not a valid JSON text.

Converting Structs to JSON

When preparing a Struct for conversion to JSON, keep the following in mind:

- You can use `$tags` to set the jsonClass annotation for each Struct member. Valid values are `object`, `array`, and

value. For more information, see Structs for JSON Data.

- If jsonClass holds an unknown value, the Struct member is ignored and a warning is returned in `$procReturnContext`.
- `structToJson` produces JSON encoding sequences for GOLD and text formatting characters (bold, italic, underline, and their combinations), and for the double quote mark ", backslash /, and slash /. For example, GOLD ; is encoded as `/u001B` , and GOLD ! is encoded as `/u0015`.
- The name of a JSON object member is a string, and a string is defined as being Unicode characters and a number of escaping mechanisms enclosed within double quotes. Therefore, object names are treated the same as the values—in the generated JSON text they are enclosed in double quotes, and escaping is applied for the characters ", /, /, Uniface text formatting characters, and GOLD characters.
- Unicode characters outside the BMP region (higher than x10000) do not need to be escaped but 3rd party tools may be limited to UCS-2 (=BMP) and require these characters to be escaped. In this case, you can use the `/bmp` flag to ensure that all characters outside of the BMP range are correctly escaped
- The generated JSON is formatted without whitespace and indentation. For example:

```
{"My JSON text":[1,{"color":"blue"}]}
```

To generate extra whitespace to make the JSON more readable, use the `/whitespace` qualifier to produce:

```
{
 "My JSON text" :
 [
 1,
 {
 "color" : "blue"
 }
 ]
}
```

Related concepts

Structs for JSON Data
jsonToStruct
Transforming Complex Data Using Structs
filedump