Rocket Uniface Library 10.4

Structs for JSON Data

You can use the `jsonToStruct` and `structToJson` ProcScript statements to convert complex data from and to JSON.

The JavaScript Object Notation format (JSON) provides an alternative to XML for exchanging information. It is often used for serializing and transmitting structured data over a network connection, especially between a server and web application.

A JSON text always starts with a brace `{` or square bracket `[`, and ends with `}` or `]`, to form a JSON object or array.

- Braces `{}` denote a JSON object, and contain a comma-delimited list of zero or more members, consisting of a name and value separated by a colon. A member can have a value that is an object or an array.

  Struct members representing a JSON object are addressable using the name, for example: `vStruct->MemberName`.

- Brackets (`[]` denote a JSON array, which contains zero or more values separated by commas. A value can be an object or an array. The values have only an index number, not a name, so they are converted to nameless top-level members of the Struct.

  Struct members representing JSON array elements are addressable by their index number, for example: `vStruct->*{1}`.

Struct Annotations for JSON

When converting between Uniface components and Structs using `jsontToStruct` and `structToJson`, you can set and retrieve annotations that provide information about the type of JSON construct and data types.

Table: Struct Annotations for JSON

Tag	Values	Comments
jsonClass	`object` | `array`	Type of JSON construct
jsonDataType	`boolean` | `string` | `number` | `null`	Data type of the Struct member

JSON to Struct Mapping

When converting from JSON using `jsonToStruct` both the carriage return (`/r`) and linefeed (`/n`) characters are converted to `/r` characters. It is not possible to determine whether characters were originally linefeeds.

When converting to JSON using `structToJson`, the CR characters are written as `/n`. Any carriage return character that is preceded or followed by a linefeed character is ignored. Thus, a Struct string member `"/n/r/n/r"` or `"/r/n/r/n"` is converted to `"/n/n"` in JSON.

Table: JSON to Struct Mapping

JSON Value	Struct Value	Struct Annotation
{ *Data*	Named member containing the object defined, with sub-members representing the *Name* **:** *Value* pairs of the object.	`jsonClass=`

JSON Value	Struct Value	Struct Annotation
`}`		`object`
`[` *Data* `]`	Struct member containing nameless sub-members representing the values.	`jsonClass= array`
`True` | `False`	Member with `Boolean` data type and value `"T"` or `"F"`	`jsonDataType= boolean`
`null`	Member with no value (an empty string). The `jsonDataType` annotation indicates that it represents the `null` JSON keyword.	`jsonDataType= null`
`""`	Member with no value. The `jsonDataType` indicates that it represents an empty string.	`jsonDataType= string`
`"` *String* `"`	Member with `String` data type and value will be the string. Any sequence representing a Unicode character (`/u`*XXXX*) is translated, and UTF-16 substitution characters are translated to the Uniface internal character (UTF32). The following escape sequences are also translated so that the resulting string contains the characters represented by these sequences. `/" // // /b /f /n /r /t`	`jsonDataType= string`
Number `-` *Number* `+` *Number*	Member with data type `Numeric`. The number can start with a leading `0`.	`jsonDataType= number`

Example: JSON Object to Struct Mapping

The following JSON stream defines an object that consists of name-value pairs.

1. The third pair has a name of `Date of birth` and a value that is an object itself.
2. The fifth pair has a name of `Children` and the value is an array of two objects.

JSON Stream for an Object

```
{
    "First Name" : "Barbara",
    "Last Name" :  "Singh",
    "Date of birth" :   { "year" : 1955, "month" : 1, "day" : 23 }, [1]
    "Married" : true,
```

JSON Stream for an Object

```
    "Children" :  [2]
    [
        {  "Name" : "Martin", "Year of birth" :  1980 },
        {  "Name" : "Margaret", "Year of birth" : 1983 }
    ],
    "Mobile phone" : null
}
```

The following example shows the corresponding Struct, as returned by `$dbgStringPlain`:

Struct for a JSON Object

```
[]
 [First Name] = "Barbara"
 [Last Name] = "Singh"
 [Date of birth]
 [year] = 1732
 [month] = 2
 [day] = 22
 [Married] = "T"
 [Children]
 []
 [Name] = "Martin"
 [Year of birth] = 1980
 []
 [Name] = "Margaret"
 [Year of birth] = 1983
 [Mobile phone]
```

The following example shows the corresponding Struct with annotations, as returned by `$dbgString`:

Struct for JSON Object, Showing $tags Structs

```
[]
 [$tags]
 [jsonClass] = object
 [First Name] = "Barbara"
 [$tags]
 [jsonDataType] = string
 [Last Name] = "Singh"
 [$tags]
 [jsonDataType] = string
```

Struct for JSON Object, Showing $tags Structs

```
[Date of birth]
[$tags]
[jsonClass] = object
[year] = 1955
[$tags]
[jsonDataType] = number
[month] = 1
[$tags]
[jsonDataType] = number
[day] = 23
[$tags]
[jsonDataType] = number
[Married] = T
[$tags]
[jsonDataType] = boolean
[Children]
[$tags]
[jsonClass] = array
[]
[$tags]
[jsonClass] = object
[Name] = "Martin"
[$tags]
[jsonDataType] = string
[Year of birth] = 1980
[$tags]
[jsonDataType] = number
[]
[$tags]
[jsonClass] = object
[Name] = "Margaret"
[$tags]
[jsonDataType] = string
[Year of birth] = 1983
[$tags]
[jsonDataType] = number
[Mobile phone] = ""
[$tags]
[jsonDataType] = null
```

Example: JSON Array to Struct Mapping

The following JSON text defines an array of three values:

1. The first is a simple numeric value (2).
2. The second value is an object.
3. The third value is an array. This embedded array also contains four values:
    - The first and third are numeric.

- the second value is an array of two values.
- The last value is a string of Unicode character codes, followed by a new line (`/n`) and the word `Four`.

JSON Text for an Array

```
[
    2, [1]
    {  "First Name" : "Margaret",
       "Last Name" :  "Singh",
       "Married" : false,
       "Date of birth" :   {  "year" : 1983, "month" : 11, "day" : 24 },
       "Mobile phone" : null
    }, [2]
    [ 1, [2e-3,2], 3, "\u05D0\u05E8\u05D1\u05E2\nFour" ] [3]
 ]
```

The following example shows the corresponding Struct, as returned by `$dbgStringPlain`:

Struct for a JSON Array

```
[]
 2
 []
 [First Name] = "Margaret"
 [Last Name] = "Singh"
 [Married] = "F"
 [Date of birth]
 [year] = 1983
 [month] = 11
 [day] = 24
 [Mobile phone]=""
 []
 [1]
 []
 0.002
 [2]
 [3]
 �?רבע
Four
```

Related concepts

[JSON]

[jsonToStruct]

[structToJson]