JSON

JSON Operations

classToJSON

Syntax	`set aString = anObject.classToJSON()` `set aString = anObject.classToJSON(options)`
Semantics	The operation `classToJSON()` takes any object (`anObject`) or array of objects and tries to map it to a JSON string `aString`. If this is not possible, an error is raised. JSON types and xUML types are mapped as listed on JSON Type Mapping. `classToJSON()` does not support cyclic structures. When a cyclic structure is detected during serializing the given object, an exception will be thrown.
Substitutables	`anObject` — Any object. `aString` — Target JSON document as **String**.

On this Page:

JSONAdapter_Example

Click the icon to download a simple example model that shows the usage of the JSON adapter (classToJSON) in **Sc heer PAS** *Designer*.

Related Pages:

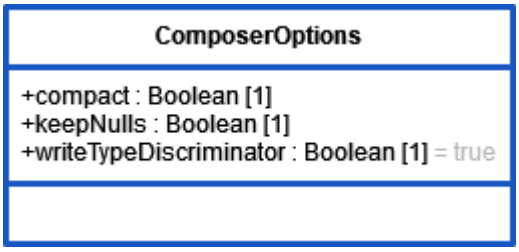
- classToJSON() Operation
- classToExtendedJSON() Operation
- jsonToClass() Operation
- JSON Type Mapping
- NULL Values

Related Documentation:

- JSON Homepage
- RFC 4627
- MongoDB Extended JSON

options

The optional parameter is an object of type **ComposerOptions**.



Its attribute is:

Attribute	Type	Description	Allowed Values	
keepNulls	Boolean	When **keepNulls** is true, attributes of anObject with NULL values will be rendered to the JSON object, otherwise they will be left out completely (see also chapter NULL Values). The default behavior of `classToJson()` (i.e. when called without options) is equivalent to **keepNulls**=false.	true	Render NULL attributes to the JSON object.
			false	Leave out NULL attributes (default).
compact	Boolean	When **compact** is true, the JSON composer will generate compact JSON, otherwise it will generate pretty JSON. The default behavior of `classToJson()` (i.e. when called without options) is equivalent to **compact**=false.	true	Generate compact JSON.
			false	Generate pretty JSON (default).
writeTypeDiscriminator	Boolean	Use **writeTypeDiscriminator** to suppress the generation of xUML type properties ("e2e:type") to the generated JSON. If this option is true, the Runtime will write the original xUML type to the generated JSON in form of `"e2e:type": "<name of the original xUML type>"` if the type being serialized does not match the expected metadata. This is necessary if you want to convert the generated JSON back to an xUML class using jsonToClass(). Runtime versions before 2021.6 will ignore the value.	**true**	Write xUML type discriminator (default).
			false	Do not write xUML type discriminator.

Examples

```
set addressAsJSONDocument = myAddress.classToJSON();
```

JSONAdapter_Example

Click the icon to download a simple example model that shows the usage of the JSON adapter (classToJSON) in **Scheer PAS** *Designer*.

classToExtendedJSON

Syntax

```
set aString = anObject.classToExtendedJson()
set aString = anObject.classToExtendedJson(format)
set aString = anObject.classToExtendedJson(options)
```

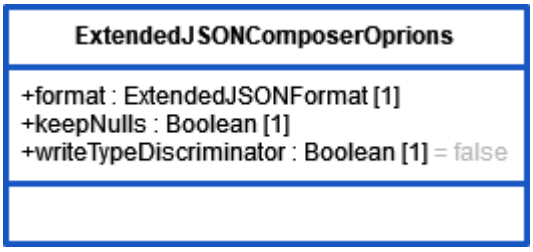
Semantics	The operation `classToExtendedJson()` takes any object (`anObject`) and tries to map it to an Extended JSON string `aString`. In opposition to classToJSON() Operation, `classToExtendedJson()` preserves the order of attributes of the converted class. JSON types and xUML types are mapped as listed in chapter jsonToClass() Operation. These details also refer to Extended JSON. If the conversion is not possible, an error is raised. `classToExtendedJSON()` does not support cyclic structures. When a cyclic structure is detected during serializing the given object, an exception will be thrown.

Substitutables

`anObject`	Any object.
`aString`	Target Extended JSON document as **String**.
`format`	String that specifies the format of the Extended JSON that should be produced (more details see `options` below).

options

This optional parameter is an object of type **ExtendedJSONComposerOptions**.



Its attributes are:

Attribute	Type	Description	Allowed Values	
format	String	Use **format** to specify which kind of Extended JSON should be produced: canonical or relaxed Extended JSON.	**canonical**	Produce canonical Extended JSON (default).
			relaxed	Produce relaxed Extended JSON.
keepNulls	Boolean	When **keepNulls** is true, attributes of anObject with NULL values will be rendered to the Extended JSON object, otherwise they will be left out completely (see also chapter NULL Values). The default behavior of `classToJson()` (i.e. when called without options) is equivalent to **keepNulls**=false.	true	Render NULL attributes to the Extended JSON string.
			false	Leave out NULL attributes (default).
writeTypeDiscriminator	Boolean	Use **writeTypeDiscriminator** to suppress the generation of xUML type properties ("e2e:type") to the generated JSON. If this option is true, the Runtime will write the original xUML type to the generated JSON in form of `"e2e:type": "<name of the original xUML type>"` if the type being serialized does not match the expected metadata. This is necessary if you want to convert the generated Extended JSON back to an xUML class using extendedJSONToClass() Operation. In contrast to classToJSON() Operation, **writeTypeDiscriminator** defaults to **false** for `classToExtendedJson()`.	true	Write xUML type discriminator.
			false	Do not write xUML type discriminator (default).

Examples

```
set aMongoDBdocument = myAddress.classToExtendedJson();set aMongoDBdocument = myAddress.
classToExtendedJson("relaxed");
```

For more information on Extended JSON refer to MongoDB Extended JSON.

jsonToClass

Syntax

```
set anObject = aString.jsonToClass()
set anObject = jsonToClass(literal)
```

Semantics	The operation `jsonToClass()` takes an JSON string (`aString`) and tries to map it to `anObject`. If this is not possible, an error is raised. By default, the following mapping rules apply: • JSON attributes are mapped to class attributes having the same name. • Attributes not defined in the target class are ignored. For more mapping details refer to JSON Type Mapping.	
Substitutables	`aString`	A JSON document as **String**.
	`anObject`	Target object, can be any object.
	`literal`	String literal.

Examples

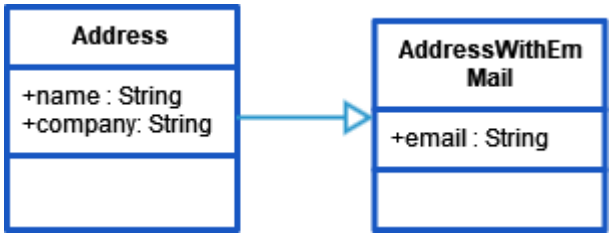
```
set myAddress = addressAsJSONDocument.jsonToClass();
```

The action script below creates an object of type **Address**. An output object flow state named **myAddress** of type **Address** needs to be defined in the activity diagram.

```
create myAddress;
set myAddress = addressAsJSONDocument.jsonToClass();
```

The example below shows how to specify the xUML type at runtime, e.g. if using a derived class.

```
{
  "e2e:type":"urn:json.e2e.ch.AddressWithEmail",
  "name":"John Snow",
  "company":"Winter & Partners",
  "mailaddress":"john.snow@winter.com"
}
```



The xUML type is declared by the attribute "`e2e:type`". The type has to be specified like `<xml_namespace>.<classname>`

For more information on JSON refer to the JSON Homepage or to RFC 4627.