



# White Paper

## Fabasoft Integration for JSON

Fabasoft Folio 2019

Copyright © Fabasoft R&D GmbH, Linz, Austria, 2019.

All rights reserved. All hardware and software names used are registered trade names and/or registered trademarks of the respective manufacturers.

No rights to our software or our professional services, or results of our professional services, or other protected rights can be based on the handing over and presentation of these documents.

## Contents

|  |          |
|--|----------|
| <b>1 Introduction</b>                                  | <b>4</b> |
| <b>2 Software Requirements</b>                         | <b>4</b> |
| <b>3 Convert JSON to Value</b>                         | <b>4</b> |
| <b>4 Convert Value to JSON</b>                         | <b>4</b> |
| <b>5 JSON Web Services</b>                             | <b>5</b> |
| 5.1 Building JSON Web Services Using Fabasoft app.ducx | 5        |
| 5.2 Accessing JSON Web Services                        | 5        |
| 5.2.1 HTTP GET Request                                 | 6        |
| 5.2.2 HTTP POST Request                                | 6        |
| 5.2.3 HTTP Response                                    | 6        |
| <b>6 JSON Parser Boundary Conditions</b>               | <b>6</b> |
| 6.1 Complex Arrays                                     | 6        |
| 6.2 Null Values  | 7        |
| 6.3 Invalid UTF-8 Characters                           | 7        |

## 1 Introduction

JSON, or JavaScript Object Notation, is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript scripting language for representing simple data structures and associative arrays, called objects.

Fabasoft Folio allows to parse a JSON string and use the data as values represented in Fabasoft Folio data types. Values can also be converted into a JSON string and therefore used in a JavaScript environment or as transfer format when consuming a JSON aware web service.

For detailed information about JSON, please consult the introduction at <http://www.json.org>.

## 2 Software Requirements

**System environment:** All information contained in this document implicitly assumes a Microsoft Windows environment or a Linux environment.

**Supported platforms:** For detailed information on supported operating systems and software see the software product information on the Fabasoft distribution media.

## 3 Convert JSON to Value

The use case *Convert JSON to value* (FSCEXPEXT@1.1001:JSON2Value) is used to convert a JSON string to Fabasoft Folio values.

### Example

app.ducx Use Case Language

```
import FSCEXPEXT@1.1001;
usecase SampleConvertValueToJSON() {
  variant Object {
    impl = expression {
      string json = "{\"key\":\"value\"}";
      jsonval = coouser.JSON2Value(json);
      coort.Trace("value", jsonval);K
    }
  }
}

// trace result
value[0/1]: DICTIONARY = 1 entries
-- key[0/1]: STRING = value
```

The use case defines the following parameters:

- json  
Input parameter that takes the JSON string as `STRING` type.
- value  
Output parameter (return value) that returns the converted value. The type depends on the JSON string (complex JSON strings are returned as `DICTIONARY`, whereas a simple JSON integer array is returned as `INTEGERLIST`)

## 4 Convert Value to JSON

The use case *Convert value to JSON* (FSCEXPEXT@1.1001:Value2JSON) is used to convert Fabasoft Folio values into a JSON string.

## Example

app.ducx Use Case Language

```
import FSCEXPEXT@1.1001;
usecase SampleConvertJSONToValue() {
  variant Object {
    impl = expression {
      string json = coouser.Value2JSON(coouser);
    }
  }
}

// resulting JSON string
{
  " ": "User",
  "objaddress": "COO.1.1065.1.15",
  "objname": "Administrator, System"
}
```

## 5 JSON Web Services

### 5.1 Building JSON Web Services Using Fabasoft app.ducx

To create web services that use JSON, the specific use cases have to be encapsulated within a *Web Service Definition* object (FSCOWS@1.1001:WebServiceDefinition).

In the following example, a web service `contactaddress` is used to fetch address data from a contact person who is identified via first name and surname.

## Example

app.ducx Object Model Language

```
instance WebServiceDefinition JSONWebSvc {
  webserviceactions<webserviceoperation, webserviceaction> = {
    {
      "contactaddress",
      GetContactAddress
    }
  }
}
```

app.ducx Use Case Language

```
usecase GetContactAddress(string firstname, string surname, out Address address) {
  variant Object {
    impl = expression {
      if (firstname && surname) {
        string query = "LIMIT 1 SELECT objname FROM FSCFOLIO@1.1001:ContactPerson
          where .userfirstname = '" + firstname +
            "' and .usersurname = '" + surname + "'";
        ContactPerson contact = coort.SearchObjects3(coortx, query);
        address = contact.address[0];
      }
    }
  }
}
```

### 5.2 Accessing JSON Web Services

JSON web services can be accessed via a HTTP GET or HTTP POST method call via the “wsjson” Friendly URL.

### 5.2.1 HTTP GET Request

```
http://localhost/fsc/wsjson/JSONSAMPLE_1_1065_JSONWebSvc/contactaddress?firstname=Leo&surname=Mayr
```

The „wsjson“ Friendly URL takes two parameters:

- The reference of the *Web Service Definition* object containing the web service action.
- The operation name of the web service action.

In case of HTTP GET the parameters are transmitted as URL parameters.

### 5.2.2 HTTP POST Request

```
http://localhost/fsc/wsjson/JSONSAMPLE_1_1065_JSONWebSvc/contactaddress
```

In case of HTTP POST, the parameters are transmitted in the http request body as JSON string.

```
{
  "firstname": "Cadence",
  "surname": "Jones"
}
```

### 5.2.3 HTTP Response

In both cases the result is sent in the HTTP response body.

**Note:** The key “address” within the JSON string corresponds to the outgoing parameter `address` of the `GetContactAddress` use case.

```
{
  "address": {
    "_": "Address",
    "addrstreet": "Pfarrgasse 5",
    "addrzipcode": "5409",
    "addrcity": "Hallein",
    "addrcountry": "Österreich",
    "addrtopic": {
      "_": "TermComponentObject",
      "objaddress": "COO.1.1001.1.182477",
      "objname": "Business"
    },
    "addrsource": {
      "_": "ContactPerson",
      "objaddress": "COO.1.1065.3.402",
      "objname": "Jones Cadence"
    }
  }
}
```

## 6 JSON Parser Boundary Conditions

The following boundary conditions exist in the current JSON parser implementation:

### 6.1 Complex Arrays

Fabasoft Folio does not support multi-typed arrays, for instance integers and strings within one array.

**Example:**

```
[
  1,
  2,
  3,
  4,
  "a",
  "b",
  "c",
  {
    "foo": "bar",
    "core": "dump"
  },
  true,
  false,
  true,
  true,
  null,
  false
]
```

**Result:**

Error: Type mismatch.

## 6.2 Null Values

A JSON string containing `null` values or empty arrays is always represented as an invalid Fabasoft Folio value.

**Example:**

- `[null]` or `[""]`  
**Result:** `[]`
- `[{}]`  
**Result:** `{}`

## 6.3 Invalid UTF-8 Characters

A JSON string containing invalid UTF-8 characters is not parsed.

**Example:**

- `["â,-Ã¼Ä+Ä"É"ÄÿÄ° some utf-8 Ä,Ê'Ä-Ä<ÂµÃ¥Ã¸Ã¸ø?,,ž"]`  
Syntax error: invalid utf-8 character
- `["\uD834\uDD1E surrogate, four-byte UTF-8"]`  
Syntax error: invalid utf-8 character