Configuration Guide
Message Transformation Guidelines - JSONDataTypeConvertor
January 2026
English

Message Transformation Guidelines - JSONDataTypeConvertor

Content

1 Introduction

In SAP Cloud Integration there is an integration object called ‘XML to JSON Converter’ that can be used to convert an XML payload to JSON format. However, this conversion transforms all the field into String type, not being possible to configure any other types, except arrays. In SAP PI/PO, these conversions are done using the adapter module ‘XMLtoJSONConverter’, where it is possible to configure the expect data type for each field.

This document presents the groovy script ‘JSONDataTypeConvertor’, an alternative solution created to convert JSON properties in a payload from String to Integer, Decimal, or Boolean or vise-versa.

This groovy script is not a replacement for the integration object ‘XML to JSON Converter’, it is meant to be used after the XML to JSON conversion takes place to enforce the expected data types.

The document also describes in detail the required and available properties/configurations and how to use them, followed by an example for each scenario.

1.1 Properties

- **json.dataTypeConvertor.wrongType:** This property indicates if the process should stop or ignore the fields in which a conversion is not possible. If the value is ‘ignore’ the process does not stop. The default value is ‘error’. This field is optional.
- **json.dataTypeConvertor.integer:** This property indicates the JSON fields to be converted from String to Integer. If the String “2.5” is to be converted to Integer, the final value will be 2, cutting out the decimal cases. This field is optional.
- **json.dataTypeConvertor.decimal:** This property indicates the JSON fields to be converted from String to Decimal. This field is optional.
- **json.dataTypeConvertor.boolean:** This property indicates the JSON fields to be converted from String to Boolean. This field is optional.
- **json.dataTypeConvertor.string:** This property indicates the JSON fields to be converted from any of the other types to String. This field is optional.
- **json.dataTypeConvertor.null:** This property indicates the JSON fields to be converted from the null format of the standard converter to the expected format. This field is optional.
    - (ex: from "foo": {"@nil" : "true"} to "foo": null)
- **json.dataTypeConvertor.dynamic:** This field indicates the JSON fields to be converted to a different type. For String fields, the values are converted to Integer, Decimal or Boolean depending on their value. Ex: ‘true’ is converted to Boolean, ‘2’ is converted to Integer and ‘2.0’ is converted to Decimal. For not String fields, the values are converted to String. It also applies the null conversion. This field is optional.
- **json.dataTypeConvertor.escapeInvalidStart:** This field indicates if the start of the JSON field names are to be escaped, if required, to allow a correct JSON to XML conversion. If the value is ‘true’, the invalid XML start characters will be escaped. This field is optional.
- **json.dataTypeConvertor.mangleInvalidChars:** This field indicates if the JSON field names are to be mangled, in case there are invalid XML characters, to allow a correct JSON to XML conversion. If the value is ‘true’, the invalid XML characters will be mangled with the pattern ‘u%04X’, representing the Unicode of the char. This field is optional.
- **json.dataTypeConvertor.escapeChar:** This field represents the escape character to be used in the escape and mangle. This field is mandatory if **json.dataTypeConvertor.escapeInvalidStart** or **json.dataTypeConvertor.mangleInvalidChars** values are ‘true’.

Multiple JSON fields can be set in each property, using a semicolon (;) as the field delimiter. To convert fields from nested objects, the path to the field must be added like an XPATH, using a slash (/) as the path delimiter.

Ex Payload:

```
{
  "college": {
    "collegeDetails": {
      "contactEmail": "info@abcuniversity.edu",
      "established": "1965",
      "location": "New York, USA",
      "name": "ABC University",
      "number": "10",
      "bool": "true"
    }
}
```

json.dataTypeConvertor.integer=college/collegeDetails/established;college/collegeDetails/number

json.dataTypeConvertor.boolean=college/collegeDetails/bool

1.2 Examples

1.2.1 Full dynamic conversion

1. Input Properties:

json.dataTypeConvertor.dynamic=users/id;users/isActive;users/accountBalance;users/profile/age;users/profile/height;users/profile/preferences/email;users/foo

2. Input Payload:

```
{
  "users": [
    {
      "accountBalance": "2489.75",
      "email": "alice.johnson@example.com",
      "id": "1023",
      "isActive": "true",
      "name": "Alice Johnson",
       "foo": {
           "@nil": "true"
          },
      "profile": {
        "age": "34",
        "height": "5.7",
        "preferences": {
          "email": "false",
          "theme": "dark"
        }
      }
    },
    {
      "accountBalance": "172.40",
      "email": "brian.lee@example.net",
      "id": "2045",
      "isActive": "false",
      "name": "Brian Lee",
      "profile": {
        "age": "29",
        "height": "6.1",
        "preferences": {
          "email": "true",
          "theme": "light"
        }
      }
    },
    {
      "accountBalance": "9300.00",
      "email": "cynthia.gomez@example.org",
      "id": "3087",
      "isActive": "true",
      "name": "Cynthia Gomez",
      "profile": {
        "age": "41",
        "height": "5.4",
        "preferences": {
          "email": "true",
          "theme": "auto"
        }
      }
    }
  ]
}
```

3. Output Payload:

```
{
  "users": [
    {
      "accountBalance": 2489.75,
      "email": "alice.johnson@example.com",
      "id": 1023,
      "isActive": true,
      "name": "Alice Johnson",
      "foo": null,
      "profile": {
        "age": 34,
        "height": 5.7,
        "preferences": {
          "email": false,
          "theme": "dark"
        }
      }
    },
    {
      "accountBalance": 172.40,
      "email": "brian.lee@example.net",
      "id": 2045,
      "isActive": false,
      "name": "Brian Lee",
      "profile": {
        "age": 29,
        "height": 6.1,
        "preferences": {
          "email": true,
          "theme": "light"
        }
      }
    },
    {
      "accountBalance": 9300.00,
      "email": "cynthia.gomez@example.org",
      "id": 3087,
      "isActive": true,
      "name": "Cynthia Gomez",
      "profile": {
        "age": 41,
        "height": 5.4,
        "preferences": {
          "email": true,
          "theme": "auto"
        }
      }
    }
  ]
}
```

1.2.2 Integer, Decimal and Boolean coversion

1. Input Properties:
   json.dataTypeConvertor.integer= users/accountBalance;users/profile/height
   json.dataTypeConvertor.decimal= users/profile/age
   json.dataTypeConvertor.boolean= users/isActive;users/profile/preferences/email

2. Input Payload:

➔ Same as the **INPUT** of the previous sample.

3. Output Payload:

```
{
  "users": [
    {
      "accountBalance": 2489.75,
      "email": "alice.johnson@example.com",
      "id": "1023",
      "isActive": true,
      "name": "Alice Johnson",
      "profile": {
        "age": 34,
        "height": 5.7,
        "preferences": {
          "email": false,
          "theme": "dark"
        }
      }
    },
    {
      "accountBalance": 172.4,
      "email": "brian.lee@example.net",
      "id": "2045",
      "isActive": false,
      "name": "Brian Lee",
      "profile": {
        "age": 29,
        "height": 6.1,
        "preferences": {
          "email": true,
          "theme": "light"
        }
      }
    },
    {
      "accountBalance": 9300.0,
      "email": "cynthia.gomez@example.org",
      "id": "3087",
      "isActive": true,
      "name": "Cynthia Gomez",
      "profile": {
        "age": 41,
        "height": 5.4,
        "preferences": {
          "email": true,
          "theme": "auto"
        }
      }
    }
  ]
}
{
 "users": [
  {
   "accountBalance": "2489.75",
   "_9email": "alice.johnson@example.com",
   "id1": "1023",
   "isActive": "true",
   "na_x0026_me": "Alice Johnson",
   "foo": {
    "_x0040_nil": "true"
   },
   "profile": {
    "age": "34",
    "height": "5.7",
    "preferences": {
     "email": "false",
     "theme": "dark"
    }
   }
  }
 ]
}
```

1.2.3 Escape and Mangle of invalid XML chars

1. Input Properties:
   json.dataTypeConvertor.escapeInvalidStart = ‘true’
   json.dataTypeConvertor.mangleInvalidChars = ‘true’
   json.dataTypeConvertor.escapeChar = ‘_’

2. Input Payload:

```
{
  "users": [
    {
      "accountBalance": "2489.75",
      "9email": "alice.johnson@example.com",
      "id1": "1023",
      "isActive": "true",
      "na&me": "Alice Johnson",
      "foo": {
        "@nil": "true"
      },
      "profile": {
        "age": "34",
        "height": "5.7",
        "preferences": {
          "email": "false",
          "theme": "dark"
        }
      }
    }
  ]
}
```

3. Output Payload:

```
{
  "users": [
    {
      "accountBalance": "2489.75",
      "_9email": "alice.johnson@example.com",
      "id1": "1023",
      "isActive": "true",
      "na_u0026me": "Alice Johnson",
      "foo": {
        "_u0040nil": "true"
      },
      "profile": {
        "age": "34",
        "height": "5.7",
        "preferences": {
          "email": "false",
          "theme": "dark"
        }
      }
    }
  ]
}
```