2024

WATS STANDARD JSON FORMAT MANUAL UUR

DESCRIBES THE WSJF FORMAT AND HOW TO USE IT

VIRINCO AS

CONTENTS

GENERAL DETAILS

WATS format name	Wats Standard JSON Format (WSJF)
Version	1.2
File example name	WatsStandardJSONFormat.json
Last modified date	2024-Apr-18

The WATS Standard JSON Format is a standardised format for importing or exporting WATS reports. The WATS Standard JSON Format adheres to the WSJF JSON schema.

This document describes WSJF for use with the built-in WATS Client converter, WSJF-Importer, for repair reports.

JSON

WSJF is a JSON format. JSON defines objects with properties, which have values. Property values can be one of the following.

STRING

Strings are quoted using “. They may have a format specified, for example date-time.

```
"property": "string value"
```

NUMBER

Numbers are not quoted. The decimal separator is ‘.’, so PI should be written as 3.14159. Do not use thousand separators. Only digits and decimal point is allowed.

```
"property": 3.14159
```

INTEGER

Integers are not quoted. Do not use thousand separators.

```
"property": 100
```

BOOLEAN

Boolean values are not quoted. They can be either “true” or “false”.

```
"property": false
```

ARRAY

Arrays contain multiple of another value type.

```
"property": ["value 1", "value 2", "value 3"]
```

OBJECT

Objects have properties.

```
"property": {
     "property1": "some value 1",
    "property2": "some value 2",
}
```

NULL

Null is an empty value. The property then has no value.

```
"property": null
```

IMPORTANCE

Required properties must be defined. If a property has value type null, it can have no value, even if the property is required. If a property is not required, it can be omitted. For export, some properties are not included when their value is null.

NUMERIC AND DATE FORMATS

Numeric: By default, '.' (period) is used as decimal separator, so PI should be written as 3.14159. Do not use thousand separators. Only digits and decimal point is allowed.

Date and time: Use the ISO 8601 format, e.g. 2019-09-12T14:26:16.977+02:00

WATS CLIENT PARAMETERS

Some fields in WSJF can be configured as parameters in the WATS Client Configurator. These parameters are used if the corresponding field is not included in the WSJF report.

machineName, location, and purpose fields will use the computer name, location, and purpose values from the WATS Client if the fields are not included.

start will use the current time when the converter is running if start is not included.

REPORT

WSJF must begin with an object. The root object describes the report.

Property Name	Description	Data Format	Importance
type	The type of report.	String (1) (R for a UUR report)	Required
pn	The part number of the unit.	String (100)	Required, parameter
rev	Revision of the unit.	String (100)	Required, parameter
sn	The serial number of the unit.	String (100)	Required, parameter
processCode	The operation type code for a WATS process.	Integer (16)	Required, parameter
location	The location where the test takes place.	String (100)	Required, parameter
purpose	The purpose of the test machine.	String (100)	Required, parameter
machineName	The name of the test station.	String (100)	Required, parameter
start	The start date and time in local time.	String (date-time)	Required, parameter
startUTC	The start date and time in UTC time.	String (date-time)	Optional
processName	The operation type name for a WATS process	String (100)	Optional
id	A Globally Unique ID of the report. A report submitted with the same ID as another will overwrite the report. If missing it will be generated.	GUID	Optional
miscInfos	Searchable miscellaneous information about the report.	Array of MiscInfo objects	Optional
subUnits	Information about sub units of the unit.	Array of SubUnit objects	Optional
uur	The header data for a UUR report.	UUR object or null	Optional (Not included if null)
binaryData	A list of attachments not tied to any unit.	BinaryData object or null	Optional (Not included if null)

IMPLEMENTATION EXAMPLE

```json
{
  "type": "T",
  "id": "bf5e5f36-8d25-4140-9ca9-dd1dea24154f",
  "pn": "WATS FAT",
  "sn": "1894.212031",
  "rev": "FAT",
  "productName": null,
  "processCode": 10,
  "processName": "SW Debug",
  "result": "F",
  "machineName": "VIC-OEF-TEST2",
  "location": "VM",
  "purpose": "apptest",
  "origin": null,
  "start": "2019-10-15T11:22:26.57+02:00",
  "startUTC": "2019-10-15T09:22:26.57Z",
  "uur": null,
  "miscInfos": [],
  "subUnits": [],
  "binaryData": []
}
```

UUR

The UUR object contains header data specific to a UUR report. It is recommended to fill out as much data as possible as this significantly improves the usability of the final WATS report.

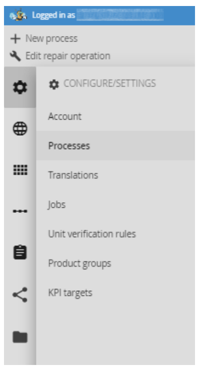
Property Name	Description	Data Format	Importance
processCode	The process code of the referenced UUT report.	Integer (16)	Required, parameter
user	The name of the operator of the test system.	String (100)	Required, parameter
execTime	The total duration of the test in seconds.	Number or null	Optional
processName	The process name of the referenced UUT report.	String (100)	Optional
refUUT	The id of the UUT report where the failure(s) were uncovered.	String (100)	Optional
confirmDate	The date and time with timezone when the repair was confirmed.	String (date-time)	Optional
finalizeDate	The date and time with timezone when the repair was finalized.	String (date-time)	Optional
comment	A comment about the report.	String (5000)	Optional

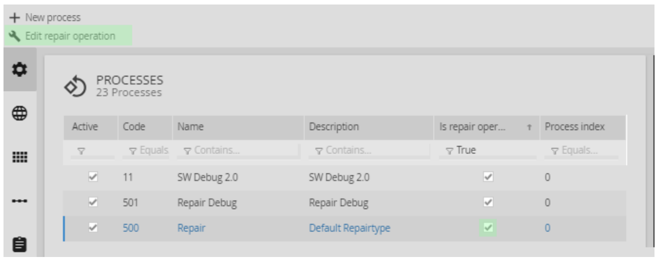
IMPLEMENTATION EXAMPLE

```
"uur": {
    "comment": "A comment",
    "user": "admin",
    "processCode": 10,
    "processName": "SW Debug",
    "refUUT": "eecc69b6-973f-4171-ae39-3aea7f6e9d98",
    "confirmDate": "2018-11-13T13:02:10.173",
    "finalizeDate": "2018-11-13T13:23:41.283",
    "execTime": 6193.2
}
```

MISCINFO

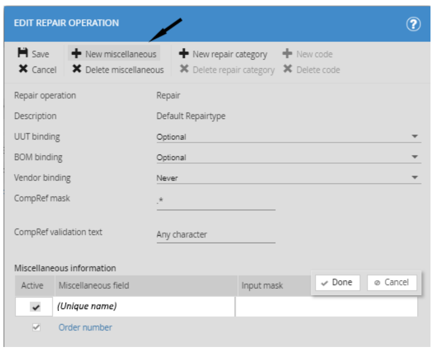
Any further data related to the UUR must be entered as miscellaneous UUR data. Miscellaneous info must be defined in a repair type before they can be used. Each miscellaneous description can only be used ***once*** in a UUR report. To register a new miscellaneous element, make sure you have permissions to access the Control panel -> Configure settings -> Processes settings. Make sure you have a registered repair operation. To edit a repair type, click on the operation to mark it, then click 'Edit repair operation'. You can now add 'new miscellaneous'. Each misc. object must have a unique name.





PROCESSES
23 Processes

Active	Code	Name	Description	Is repair oper...	Process index
	Equals	Contains...	Contains...	True	Equals...
✓	11	SW Debug 2.0	SW Debug 2.0	✓	0
✓	501	Repair Debug	Repair Debug	✓	0
✓	500	Repair	Default Repairtype	✓	0



Repair operation	Repair
Description	Default Repairtype
UUT binding	Optional
BOM binding	Optional
Vendor binding	Never
CompRef mask	.*
CompRef validation text	Any character

Miscellaneous information

Active	Miscellaneous field	Input mask
✓	*(Unique name)*	
✓	Order number	

Property Name	Description	Data Format	Importance
description	The name of the misc info	String (100)	Required
typedef	The type defition of the misc info.	String (30)	Optional
numeric	The numeric value of the misc info	Integer (32)	Optional
text	The text value of the misc ifno	String (100)	Optional

IMPLEMENTATION EXAMPLE

```
"miscInfos": [
    {
      "description": "Misc info 1",
      "typedef": null,
      "text": "Misc string 1",
      "numeric": 1
    },
    {
      "description": "Misc info 2",
      "typedef": null,
      "text": "Misc string 2",
      "numeric": 2
    }
  ]
```

SUB UNITS

In a UUR report, you add the unit and its sub units to the SubUnits property on the root object. These units are the units that have failures that needs repairs. The unit with index 0 is considered the main unit.

Property name	Description	Data Format	Importance
pn	The partnumber of the sub unit.	String (100)	Required
sn	The serial number of the sub unit.	String (100)	Required
rev	The revision of the sub unit.	String (100)	Required
partType	The type of sub unit.	String (50)	Required
idx	The index of the sub unit.	Integer (32) or null	Optional (Not included if null)
parentIdx	The index of the parent sub unit.	Integer (32) or null	Optional (Not included if null)
position	The position of the unit.	Integer (32) or null	Optional (Not included if null)
replacedIdx	The index of the sub unit that replaced this unit.	Integer (32) or null	Optional (Not included if null)
failures	A list of failures on this sub unit.	Failure objects or null	Optional (Not included if null)

IMPLEMENTATION EXAMPLE

```
"subUnits": [
  {
    "partType": null,
    "pn": "UURTest-PN1",
    "rev": "Rev1",
    "sn": "UURTest-SN1",
    "idx": 2,
    "parentIdx": 0,
    "position": 1,
    "failures": []

  }
]
```

FAILURE

Failures contain information about what needed to be repaired. To register a failure, you will need to have a fail category and/or a failure code. Failure types are defined within a repair operation, at the same location where you add misc. data.

Property Name	Description	Data Format	Importance
code	Repair operation category repair code. Must be unique for each category.	String (200)	Required
category	Repair operation category name.	String (200)	Required
refStepId	The id of the step from the reference UUT report that uncovered the failure.	Integer (32)	Required
comRef	Electric Component reference.	String (50)	Optional
funcBlock	A group of electric components.	String (100)	Optional
artNumber	Electric component articlenumber.	String (100)	Optional
artRevision	Electric component revision.	String (100)	Optional
artVendor	Electric component vendor. Used to compare yield based on vendor.	String (100)	Optional
artDescription	Electric component description.	String (100)	Optional
comment	A comment about the failure or repair.	String (5000)	Optional
attachments	A list of attachments on this failure	Array of BinaryData objects or null	Optional (Not included if null)

IMPLEMENTATION EXAMPLE

```
"failures": [
      {
        "artNumber": null,
        "artRev": null,
        "artVendor": null,
        "artDescription": null,
        "category": "Assembly Process",
        "code": "Missing component",
        "comment": "g44",
        "comRef": "A4",
        "funcBlock": null,
        "refStepId": 9,
        "refStepName": "ET_PFT_1",
        "attachments": null
      }
    ]
```

BINARY DATA

Attachments can be included in a step by adding a BinaryData object to a failure. WSJF uses Base64 encoded contents to attach a file.

Property name	Description	Data Format	Importance
name	The name for the attachment.	String (256)	Required
contentType	The Mime-type of the attachment.	String (100)	Required
data	A base64 encoded string of the contents of the file.	String (100)	Required

IMPLEMENTATION EXAMPLE

```
"attachments": [
  {
    "contentType": "image/png",
    "data": "iVBORw0KGgoAAAANSUhE8Dq+XaD/wlfAgwALvKJdMi++N4AAAAASUVORK5CYII=",
    "name": "warning.png"
  }
]
```