| | |
|---|---|
| WATS format name | Wats Standard Text Format (WSTF) |
| Version | 2.1.2 |
| File example name | WatsStandardTextFormat.txt |
| Last modified date | 2023-Mar-10 (History at bottom) |

# General details

This format is a standardised format that the WATS Client will automatically read and import into WATS (was first release in 2014).

The format of the Wats Standard Text Format is a tab delimited ASCII file (*.txt). Each line in the file represents an entry containing any number of data fields.

The file consists of three main parts; a **Header data** part, a **Step data** part and a **Footer data** part. The Header data part is further split into two sub parts; a list of predefined headers followed by an optional section of miscellaneous UUT data.

The WATS Client can read one UUT report per file or multiple UUT reports in one file. Use the Data Markers to organize the data.

# Converter Method

The WATS API supports two methods for importing data: Active and Import. When operating in Active mode, the result of a step, sequence and test is set automatically using the measure, compare operator and the limit(s). In Import mode, the status (pass/fail) must be set manually.

The Wats Standard Text Format converter use a combination of Active and Import. If status is given in the data; it will be used. If not, limits are used. See below for rules and recommendation regarding this.

# Numeric and Date Formats

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

DateTime: Use the ISO 8601 format YYYY-MM-DDTHH:mm:ss e.g. 2013-07-17T15:09:04

In a later version on request, we will support to have custom numeric / date formats.

# Header data

The Header data part of the file consists of two parts. The first part is a list of predefined headers such as *SerialNumber* and *PartNumber* containing basic information about the UUT.
The second part is miscellaneous allowing you to log any type of information linked to the UUT.
For any data use the format “EntryName<tab>data” e.g. SerialNumber 423664753.

Start the report with the Data Marker:

```
--Header-Start--
```

## Predefined headers

It is recommended to fill out as much data as possible as this significantly improves the usability of the final WATS report. The predefined headers mentioned below are all reserved words and cannot be used as miscellaneous data fields. For any recurrent headers the last occurrence is used.

Use the format “EntryName<tab>data”

| EntryName | Description | Data (Format) | Importance |
|---|---|---|---|
| SerialNumber | UUT Serial Number | String(100) | Required |
| PartNumber | UUT Part Number | String(100) | Required |
| Revision | UUT HW revision | String(100) | Recommended |
| BatchSerialNumber | Batch number of the UUT | String(100) | Optional |
| OperationTypeCode | WATS Process. Consult the WATS documentation for further information on the WATS Processes and Test Operations. | Numeric. Look up in the Control Panel (Configuration/ Settings ->Processes) | Required (Either OperationTypeName or OperationTypeCode is required, not both) |
| OperationTypeName | WATS Process. Consult the WATS documentation for further information on the WATS Processes and Test Operations. | String. Look up in the Control Panel (Configuration/ Settings ->Processes) | Required (Either OperationTypeName or OperationTypeCode is required, not both) |
| UUTStatus | Final test status (UUT status) | Passed, Failed, Error, Terminated. | Optional (If missing, the WATS Client will calculate the status) |
| ErrorCode | Error code in case of any error during the test | Numeric | Optional |
| ErrorMessage | Error message in case of any error during the test | String(500) | Optional |
| StartDateTime | Start time in local time zone | Time must be formatted according to ISO 8601 i.e. YYYY-MM-DDTHH:mm:ss e.g. 2013-07-17T15:09:42 | Recommended (if missing, the StartDateTime will be calculated using the UTCStartDateTime) See also UTSStartDateTime |
| UTCStartDateTime | Start time in UTC time zone. | Time must be formatted according to ISO 8601 i.e. YYYY-MM- | Optional (if missing, the WATS Client will calculate UTC based on StartDateTime). |

| | | | |
|---|---|---|---|
| | | DDTHH:mm:ss e.g. 2013-07-17T13:09:42 | If UTCStartDateTime or StartDateTime is missing, the current time on the client machine will be used. |
| ExecutionTime | Duration of test in seconds | Seconds (Use decimal point for milliseconds e.g 0.39) | Recommended |
| StationName | Name of test station. | String(100) | Optional (if missing, the WATS Client will use the Computer name) |
| TestSocketIndex | When parallel or batch execution | Numeric | Optional |
| FixtureId | Id of fixture | String(100) | Optional |
| OperatorName | Name of the operator of the test system | String(100) | Recommended |
| SoftwareName | Name of the test sequence e.g. file name (full path is supported) | String(200) | Recommended |
| SoftwareVersion | Version number of the test sequence | Major.<br>Minor.<br>Revision.<br>Build<br>(1.0.0.0) | Recommended |
| Comment | Comment to the UUT | String(5000) | Optional |

## Miscellaneous Headers

Any further data concerning the UUT or the general test run must be entered as miscellaneous UUT data. The Entry Name will be used as the description in the UUT report.

| EntryName | Description | Data (Format) | Importance |
|---|---|---|---|
| Any Name (except reserved words from UUT header) | Add any name or description (e.g. Temperature or uCSwVer | String(100) | Optional |

## Sub units

You can register sub units as a part of the test report. It will appear in the report header. For each unit you want to add, specify description, serial number, part number and revision.

| Column Name | Description | Data (Format) | Importance |
|---|---|---|---|
| Subunit | The first column must contain “Subunit” | String | Required |
| **Description** | Description of the subunit | String(50) | Required |

| | | | |
|---|---|---|---|
| PartNumber | Part number | String(100) | Can be blank |
| SerialNumber | Serial number | String(100) | Required |
| Revision | Revision | String(100) | Can be blank |

## Step data

The second and usually main part of the file contains the actual test step data. Data is entered in a tab-delimited pattern with multiple columns.

Start the step data section with the Data Marker:

```
--Step-Data-Start--
```

**NOTE**: All columns must be available in the file. They can be empty. The importance (Recommended/ Optional) applies for the data in the column.

### Columns

| Column Name | Description | Data (Format) | Importance |
|---|---|---|---|
| StepType | All standard WATS step types are supported, except the XY-Graph step type. See details below | String | Required |
| StepName | Name of the test step | String(100) | Required (recommended to be unique per sequence call) |
| MeasureName | Name of the measurement. Used to distinguish between different measurements of a multi-step type. For a single-step type this field is left blank. | String(100) | Optional |
| Value | The measurement value of the step | Numeric/String(100)/ Boolean | Required |
| LowLimit | Low limit for the test step. For single sided test limits only this limit is used, even for a “less than” comparison. | Numeric/String(100) | Required |
| HighLimit | High limit for the test step. This field is not used in case of a single- | Numeric | Required/Optional (depending on type of comparison operator) |

| | | | |
|---|---|---|---|
| | sided comparison operator. | | |
| CompOperator | Comparison operator for the test step. | See Comparison Operators | Required |
| Unit | Unit of the measured value (e.g. Volt) (leave blank if empty) | String(20) | Required for numeric limit tests (redundant for others). |
| StepStatus | Status of test step. If this field is left blank the WATS engine will use the limits and comparison operator to decide the step status. | See Status Codes | Optional |
| StepExecutionTime | Duration of the step in seconds | Seconds (Use decimal point for milliseconds e.g 0.39) | Recommended |
| StepReportText | Add any description or additional text | String(5000) | Optional |
| StepErrorCode | Used to report any step error code | Numeric | Optional |
| StepErrorMessage | Used to report any step error message | String(200) | Optional |
| CausedSequenceFailure | Set True if the step caused the final test status (UUT status) to fail. Useful when logging multiple steps with “Failed” status | Boolean | Optional |

## Step Types

Below is a list of supported step types.

### Test steps

Test steps comes in two different forms:

1. Single steps (one measurement) (*recommended*).
2. Multiple steps (two or more measurements)

Test steps is recommended to have StepStatus set to Passed, Failed, or Skipped. For single steps it is also possible to use Terminated and Error.

If StepStatus is left blank WATS will use the limits and comparison operator to decide the step status (to Passed or Failed). This value will also ripple up through levels of SequenceCall. If StepStatus has a status, the limits and comparison operator will not be used to validate it, and the status will be kept as it is in the input file (Passed, Failed, Skipped, Terminated, or Error).

Types of step (these can be used as a single or as multiple):

- **NumericLimitTest**
  Used to log numeric test data
- **StringValueTest**
  Used to log string value test data
- **PassFailTest**
  Used to log pass/fail test data

## Single steps

A single step is used for one measurement. They must not have a MeasureName.

| StepType | StepName | MeasureName |
|---|---|---|
| NumericLimitTest | Single test 1 | |
| NumericLimitTest | Single test 2 | |
| NumericLimitTest | Single test 3 | |

A single step can have Status set to Passed, Failed, Skipped, Terminated and Error.

It is recommended to use Passed or Failed to simplify.

The Step name has to be unique to the UUT report.

```
StringValueTest->Name1->->same->same->->EQ->->Passed->5->strings are the same
StringValueTest->Name2->->same->not->->EQ->->Failed->10->strings are not the same

NumericLimitTest->Name3->->5->0->10->GELE->V->->2->between limits
NumericLimitTest->Name2->->15->0->10->GELE->V->->10->not between limits
```

As in the example of the two StringValueTest lines above, the Status is provided to Passed for the first line and Failed for the second line. The status will therefore not be calculated in the converter.

As in the example of the two NumericLimitTest lines above, the Status is not provided for the lines. They will therefore be calculated by the converter, to Passed for the first line and Failed for the second.

For both cases, the Failed status will ripple through the levels of SequenceCall and up to main report.

## Multiple steps

A multiple step is when it is necessary to group together two or more measurements.

A step is made into a multiple step by adding the same step multiple times (the same StepName), and by including a MeasureName. The MeasureName must be unique in the multiple step and each measurement must have the same StepType.

The StepName in a multiple step can either be set on each line of the multiple step, or only on the first and empty on the rest.

| StepType | StepName | MeasureName |
|---|---|---|
| NumericLimitTest | Test with multiple meas | Meas1 |
| NumericLimitTest | | Meas2 |
| NumericLimitTest | | Meas3 |

| StepType | StepName | MeasureName |
|---|---|---|
| NumericLimitTest | Test with multiple meas | Meas1 |
| NumericLimitTest | Test with multiple meas | Meas2 |
| NumericLimitTest | Test with multiple meas | Meas3 |

Multiple steps have some limitations regarding status. The step status will be calculated based on the measurements since there is no line specifically assigned for status.

If statuses are used for multiple steps, Terminated, and Error will be translated to Failed, both for the individual measurements and for the step status. Skipped will show for the individual measurement, but the step will have Passed even if all measurements are Skipped.

```
NumericLimitTest->Name1->meas1->5->0->10->GELE->V->->2->between limits
NumericLimitTest->Name1->meas2->15->0->10->GELE->V->->10->not between limits
```

```
NumericLimitTest->Name2->meas1->5->0->10->GELE->V->Passed->2->between limits
NumericLimitTest->Name2->meas2->15->0->10->GELE->V->Passed->10->not between limits
```

```
NumericLimitTest->Name3->meas1->5->0->10->GELE->V->Skipped->2->between limits
NumericLimitTest->Name3->meas2->15->0->10->GELE->V->Skipped->10->not between limits
```

In the first example above, Status will be calculated for both lines (to Passed and Failed). The step status will be set based on these (to Failed).

In the second example above, Status will NOT be calculated for the lines (both set to Passed). The step status will be set based on these (to Passed).

In the third exampled above, Status will NOT be calculated for the lines (both set to Skipped). The step status will be set based on these, but in this case translated to Passed (since multiple steps only have Passed or Failed).

If it is important to set Status more specifically than these rules, it is recommended to use single steps.

### Sequence steps

Only StepType, StepName, StepStatus, StepTime, StepReportText, StepErrorCode and StepErrorMessage used for the sequence step types.

MeasureName, Value, LowLimit, HighLimit, CompOperator, Units are not used.

- **SequenceCall**
  Used in order to log sub sequences combined with the EndSequenceCall step type.
  Is not possible to set Status through. Use EndSequenceCall.
- **EndSequenceCall**
  Used to mark the end of a sub sequence. Offers the possibility to set the StepStatus, StepTime, StepReportText, StepErrorCode and StepErrorMessage used for the sequence step types entered in SequenceCall.

EndSequenceCall can either stand alone or with a name corresponding to correct SequenceCall, and/or also set one or more of the above listed fields.

Status for Sequence call will be calculated from the steps below in the hierarchy. If it is necessary to set this from the file, put it in EndSequenceCall (see ex table below):

| StepType | StepName | MN | V | LL | HL | CO | U | Status | Time | Text |
|---|---|---|---|---|---|---|---|---|---|---|
| SequenceCall | SubSeq1 | | | | | | | optional (not used) | 10 | Seq call info |
| NumericLimitTest | Test1.1 | | 2 | 1 | 5 | GELE | V | Passed | 2 | Test info |
| NumericLimitTest | Test1.2 | | 3 | 1 | 5 | GELE | V | Passed | 8 | Test info |
| EndSequenceCall | SubSeq1 (optional) | | | | | | | **Passed** | Optional (will override | Optional( will override) |

```
SequenceCall->SubSeq1->->->->->->->->->10->seq call info->->->
NumericLimitTest->Test1.1->->2->1->5->GELE->V->Passed->2->Test info->->->
NumericLimitTest->Test1.2->->3->1->5->GELE->V->Passed->8->Test info->->->
EndSequenceCall->->->->->->->->->Passed->->->->->->
```

(arrows for tabs in example above)

Sub sequences are incorporated by adding a SequenceCall step. Multiple levels of sub sequences are accomplished by adding a new SequenceCall step prior to adding the EndSequenceCall step. Examples of this can be seen in the two following figures.

| StepType | StepName | MeasureName |
|---|---|---|
| SequenceCall | SubSeq1 | |
| NumericLimitTest | Test1.1 | |
| NumericLimitTest | Test1.2 | |
| EndSequenceCall | | |
| SequenceCall | SubSeq2 | |
| NumericLimitTest | Test2.1 | |
| NumericLimitTest | Test2.2 | |
| EndSequenceCall | | |

Result:

- SubSeq1
  - Test1.1
  - Test1.2
- SubSeq2
  - Test2.1
  - Test2.2

| StepType | StepName | MeasureName |
|---|---|---|
| SequenceCall | SubSeq1 | |
| NumericLimitTest | Test1.1 | |
| NumericLimitTest | Test1.2 | |
| SequenceCall | SubSeq2 | |
| NumericLimitTest | Test2.1 | |
| NumericLimitTest | Test2.2 | |
| EndSequenceCall | | |
| EndSequenceCall | | |

Result:

- SubSeq1
  - Test1.1
  - Test1.2
  - SubSeq2
    - Test2.1
    - Test2.2

**Action steps**

Used to log test action data and information. These are always simple steps, never multiple.

Only StepType, StepName, StepStatus, StepTime, StepReportText, StepErrorCode and StepErrorMessage used for action steps.

MeasureName, Value, LowLimit, HighLimit, CompOperator, Units are not used.

| StepType | StepName | MN | V | LL | HL | CO | U | Status | Time | Text |
|---|---|---|---|---|---|---|---|---|---|---|
| ActionStep | Name | | | | | | | Passed | 5 | Info |

```
ActionStep->Name1->->->->->->->->->Passed->5->This action is cool
ActionStep->Name2->->->->->->->->->Passed->10->This action is less cool
```

(arrows for tabs in example above)

## Chart

A chart belongs to a Step. All steptypes can include a chart, but not more than one chart for each.

| Column Name | Description | Data (Format) | Importance |
|---|---|---|---|
| Chart | The first column must contain “Chart” | String | Required |
| ChartType | Type of chart.<br>Line, LineLogXY, LineLogX or LineLogY. | String | Can be blank.<br>If not recognized or blank field, Line will be used. |
| Label | Label of chart | String(100) | Can be blank |
| YUnit | Y Unit | String(20) | Can be blank |
| YLabel | Y Label | String(50) | Can be blank |
| XUnit | X Unit | String(20) | Can be blank |
| XLabel | X Label | String(50) | Can be blank |

Example:

| Chart | ChartType | Label | YUnit | YLabel | XUnit | XLabel |
|---|---|---|---|---|---|---|
| Chart | Line | GraphLabel | Time | Seconds | Weight | Kilos |

### Series

A chart can have one or more series (plots). It can’t have more than 10 series.

| Column Name | Description | Data (Format) | Importance |
|---|---|---|---|
| Series | The first column must contain “Series” | String | Required |
| PlotName | Name of plot. | String(100) | Can be blank. If more than one series, it is recommended to include a name. |
| DataType | Type of data. | String(10) | Required |
| YData | All data for y-axis. | Semicolon-separated string. | Required |
| XData | All data for x-axis. | Semicolon-separated string. | Can be blank. If blank, the x-axis will be added. |

The DataType is required. Currently, only XYG is a valid input.

YData and XData can’t have more than 10000 entries each.

The number of elements in Y and X must match.

Examples:

| Series | PlotName | DataType | YData | XData |
|---|---|---|---|---|
| Series | Plotname0 | XYG | 1;2;3;4;5;6;7;8;9 | 1;2;3;4;5;6;7;8;9 |
| Series | Plotname1 | XYG | 1;2;3;4;5;6;7;8;9 | |
| Series | Plotname3 | XYG | 1.1;2.02;3.003;4.0004 | 100;200;300;400 |
| Series | Plotname4 | XYG | 1e-53;-1;1e-23;-2 | -26,2856;26,2557;25,6985;25,7882 |

Example for a chart with one series belonging to a Numeric Limit Test (arrows for tabs):

```
NumericLimitTest->test1->meas0->8->7->10->GELE->Volt->Passed->5
Chart->Line->Label->kilos->weight->seconds->time
Series->Plotname0->XYG->1;2;3->100;200;300
```

## Footer data

The Footer data part is required when logging multiple UUT reports in one file.
End the step data section with the Data Marker:

```
--Step-Data-End--
```

The predefined footers mentioned below are all reserved words. Use the format “EntryName<tab>data”.

| EntryName | Data (Format) | Importance |
|---|---|---|

| | | |
|---|---|---|
| UUTStatus | Passed, Failed, Error, Terminated. | Optional |
| ErrorCode | Number (int) | Optional |
| ErrorMessage | String (200) | Optional |
| ExecutionTime | Number (double) | Optional |

They are present to be able to override the same header data, in case this is information is only available at the end of a test.

```
--Step-Data-End--
UUTStatus->Failed
ErrorCode->2
ErrorMessage->Terrible error
ExecutionTime->15
-- UUT-End--
```

End the entire UUT data section with the Data Marker:

```
--UUT-End--
```

## Step Status Codes

Below is a list of supported status codes for a step.

| Status | UUT description | Step / measurement description | Additional information | Color code |
|---|---|---|---|---|
| Passed | UUT passed | Step passed | | Green |
| Failed | UUT failed | Step failed | | Red |
| Skipped | Not used. | Step execution skipped | Can be used for single steps, with caution for multiple steps. | Yellow |
| Error | An error occurred during test execution | An error occurred during step execution | Can be used for single steps, should not be used for multiple steps. | Orange |
| Terminated | Operator terminated test execution | Operator terminated step execution | Can be used for single steps, should not be used for multiple steps. | Blue |

The UUT report itself can be Passed, Failed, Error or Terminated.

A single step can have all statuses.

A single measurement (in a multiple step) can only be Passed, Failed, or Skipped (but all Skipped measurements will end with Passed for the step). Error or Terminated will be translated to Failed.

## Comparison Operators

For a complete list of supported Comparison Codes, see https://virinco.zendesk.com/hc/en-us/articles/207424473-Comparison-Operators

## Example

```
--Header-Start--

SerialNumber  423664753

PartNumber    100200

Revision      A

OperationTypeName    SW Debug

UUTStatus     Failed

ErrorCode     0

ErrorMessage

StartDateTime 2014-07-04T12:09:04

ExecutionTime 10

BatchSerialNumber    B888
StationName   VIC-WS-ML

OperatorName  Martin Lentz

SoftwareName  c:\test.seq

SoftwareVersion      1.0.0.0

Subunit       Main board    Partno 1 SN 6654433  Rev 1
Subunit       Controller Card      Partno 2 SN 5555     Rev B
uCSwVer       1.0

--Step-Data-Start--

StepType      StepName      MeasureName   Value  LowLimit      HighLimit
       CompOperator  Unit   StepStatus    StepExecutionTime    StepReportText
       StepErrorCode StepErrorMessage     CausedSequenceFailure
```

```
SequenceCall  SubSeq1                                                Passed 10
       This is a seq call report text
NumericLimitTest    test1.1             7      7      10     GELE   Volt   Passed 5
       This is a test step report text
NumericLimitTest    test1.2             8      7      10     GELE   Volt   Passed 5

EndSequenceCall

SequenceCall  SubSeq2                                                Passed 10

NumericLimitTest    test2.2       meas1  9      7      10     GELE   Volt   Passed 5

NumericLimitTest    test2.2       meas2  10     7      10     GELE   Volt   Passed 5

EndSequenceCall

NumericLimitTest    Test3  meas1  8      10            EQ     Volt   Failed 1
                    True
NumericLimitTest           meas2  6      5             EQ     Volt   Failed

NumericLimitTest           meas3  10     10            EQ     Volt   Passed

PassFailTest  test4        TRUE                                Passed 2

PassFailTest  test5  meas1  FALSE                               Failed 3

PassFailTest         meas2  TRUE                                Passed

PassFailTest         meas3  TRUE                                Passed

PassFailTest         meas4  FALSE                               Failed

StringValueTest      Test6         The quick brown fox jump over lazy dogs  the
quick brown fox jump over lazy dogs             EQ             Failed 4

Chart  Line   Label  kilos  weight seconds       time

Series Plotname0     XYG    1e-53;1,0001;1e-53;3.2323;1e-53;5.0005;1e-53;1.0001;1e-
53;3.2323;1e-53;5.0005;1e-53;1.0001;1e-53;3.2323;1e-53;5.0005

Series Plotname1     XYG    1e-53;1.0001;1e-53;3.2323;1e-53;5.0005;1e-53;1.0001;1e-
53;3.2323;1e-53;5.0005;1e-53;1.0001;1e-53;3.2323;1e-53;5.0005
       0;2;4;6;8;10;12;14;16;18;20;22;24;26;28;30;32;34

Series Plotname2     XYG    1e-53;-1.0001;2.222202;3.2323;4.4;1e-53;1e-
53;1.0001;2.222202;3.2323;4.4;1e-53;1e-53;1.0001;2.222202;3.2323;4.4;1e-53
       0;1;2;3;4;5;6;7;8;9;10;11;12;13;14;15;16;17

Series Plotname3     XYG    1e-53;1.0001;1e-53;3.2323;1e-53;5.0005;1e-53;1.0001;1e-
53;3.2323;1e-53;5.0005;1e-53;1.0001;1e-53;3.2323;1e-53;5.0005
       0;1;2;3;4;5;6;7;8;9;10;11;12;13;14;15;16;17

ActionStep    Action                                             Passed 5

SequenceCall  SubSeq3                                                    Passed 20
       Test Multiple with same stepName
NumericLimitTest    SameStepName  meas1  7      7      10     GELE   Volt   Passed 5
       This is a test step report text
NumericLimitTest    SameStepName  meas2  8      7      10     GELE   Volt   Passed 5
```

```
EndSequenceCall
--Step-Data-End--
UUTStatus     Failed
--UUT-End--
```

Change history:

2023-Mar-10 Specified that StepName should be unique per sequence call.

2022-Sep-01 Specified that measurements in multiple steps must have same StepType.

2020-Sep-03 Added FixtureID to header.

2020-Feb-02 Removed the section about Multiple keyword for Multiple Steps.

2020-Jan-14 Corrected string lengths.

2018-Oct-30 Moved Log File section to article instead.

2017-Apr-07 Added Comment to UUT header, added max length in documentation.

2015-Nov-28 Added ConverterMode Header keyword (is not finished implemented yet, specifically Import for MultiplePassFailTest and MultipleStringValueTest).

2016-Oct-01 Changed documentation for Subunit (Partnumber/Serialnumber order).

2016-Sep-30 Changed the validation of the multiple numeric limit steps (not finished with stringvalue and passfail).

2016-Sep-08 Updated wstf documentation for no-validation.

2016-Feb-25 Added several points for Status.

2016-Feb-04 Added info for startdatetime and endsequencecall.

2015-Dec-15 Changed warning file info.