2018

WATS STANDARD XML FORMAT MANUAL

DESCRIBES THE WSXF FORMAT AND HOW TO USE IT

VIRINCO AS

CONTENTS

GENERAL DETAILS

WATS format name	Wats Standard XML Format (WSXF)
Version	1.0
File example name	WatsStandardXMLFormat.xml
Last modified date	15.Aug.2018 (History at bottom)

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

The WATS Standard XML Format follows the WSXF XML schema.

Each report consists of two main parts; a **Header data** part, a **Step 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 WSXF Converter receives an Xml file with UUT data. It goes through the elements in the file and creates a UUT report in the TDM API with all the tests and measurements added into. The WATS Client can read one or multiple UUT report(s) per file.

XML

WSFX is an XML format. XML is made up of elements. An XML element has an opening tag with a name and attributes, a closing tag, and content.

The opening tag of an element starts with a < character followed by the name of the tag. After the name comes an optional list of attributes for the element. Attributes are defined with the name of the attribute, then the value of the attribute in quotes after an equals sign. The opening tag ends with a > character.

```
<Report type="UUT" Start="2018-01-01T08:30:00" Result="Passed">
```

The closing tag of an element starts with </ followed by the name of the element, and ends with a > character.
WSXF uses <Report> element to organize the data.

```
</Report>
```

Some elements have no content and can be self-closing. In this case the opening tag ends with />

```
<Process Code="10" />
```

Between the opening and closing tags of an element is the contents of the element. The content can be text or other elements. Elements inside other elements are called nested elements and are children of the element they are nested in. All child elements must be closed before the parent can be closed.

```
<Reports>
   <Report type="UUT" Start="2018-01-01T08:30:00" Result="Passed"> ←Child of reports
      <Process Code=”10” />                                          ← Child of report 1
   </Report>
   <Report type="UUT" Start="2018-01-01T08:40:00" Result="Passed"> ← Child of reports
      <Process Code=”10” />                                          ← Child of report 2
   </Report>
</Reports>
```

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 XML 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 . (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.

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.

REPORTS

A WSXF file should start with a <Reports> element. <Reports> can contain one or more <Report> elements, each containing data from one report.

```
<Reports
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://wats.virinco.com/schemas/WATS/Report">
</Reports>
```

HEADER DATA

The Header data part of the file consists of two parts. The first part is three elements containing basic information, such as *SerialNumber* and *PartNumber,* about the UUT. The second part is any number of <MiscInfo> elements containing miscellaneous information allowing you to log any type of information linked to the UUT.

Required information (Header)

```
<Report
      type="UUT"
      Start="1900-01-01T01:00:00.000+01:00"
      SN="FATTest-268" PN="FATPartNo"
      Rev="Rev1" >
   <UUT UserLoginName=”FAToper” />
   <Process Code="60" Name="FST" />
```

Additional information (Miscellaneous)

```
   <MiscInfo Description="Electronics Id">1234</MiscInfo>
   <MiscInfo Description="Electronics Id">1234</MiscInfo>
   <MiscInfo Description="Electronics Id">1234</MiscInfo>
```

REPORT ELEMENT

The <Report> element should have these attributes. It is recommended to fill out as much data as possible as this significantly improves the usability of the final WATS report.

Attribute Name	Description	Data Format	Importance
Type	The type of report.	UUT or UUR	Required
SN (Serial Number)	Serial number of the unit.	String(30)	Required
PN (Part Number)	Part number of the unit.	String(30)	Required
Start	Start time in local time. If missing, calculated based on Start_utc.	Formated according to ISO 8601: YYYY-MM-DDTHH:mm:ss	Recommended
Rev	Revision of the unit.	String(30)	Recommended
Start_utc	Start time in UTC time zone. If missing, calculated based on Start.	Formated according to ISO 8601: YYYY-MM-DDTHH:mm:ss	Optional
Result	Final test status. Will be generated based on values in the steps of the report.	Passed, Failed, Terminated, Error	Optional
Location	Location where the test takes place.	String	Optional
Purpose	Purpose of the test station.	String	Optional
Machine Name	Name of test station. If missing, the computer name will be used.	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 will be generated.	GUID	Optional
Comment	Comment to the UUT	String(500)	Optional

IMPLEMENTATION EXAMPLE

```
<Report type="UUT" Start="1900-01-01T01:00:00.000+01:00" Start_utc="1900-01-01T00:00:00.000Z"
        Result="Failed" SN="FATTest-268" PN="FATPartNo" Rev="Rev1" MachineName="FAT Station"
        Location="FAT Station Location" Purpose="FAT Station Purpose">
```

UUT ELEMENT

If the report is a UUT report, the <Report> element may contain a <UUT> element with these attributes. It is recommended to fill out as much data as possible as this significantly improves the usability of the final WATS report.

The UUT may contain a <Comment> Element.

Attribute Name	Description	Data Format	Importance
UserLoginName	Name of the operator of the test system.	String(30)	Recommended
Execution Time	Total duration of test in seconds. If missing, will be generated from execution time of each step.	Numeric	Recommended
BatchSN	Batch serial number of the unit.	String(20)	Optional
TestSocketIndex	When parallel or batch execution.	Numeric	Optional
FixtureId	ID of the fixture used to perform the tests.	String	Optional
ErrorCode	Error code of any error that occurred during the test, if any.	Numeric	Optional
ErrorMessage	Message of the error that occurred during the test, if any.	String(200)	Optional

IMPLEMENTATION EXAMPLE

```
<UUT UserLoginName="FAToper" BatchSN="BatchSN" TestSocketIndex="999"
    ExecutionTime="123.67" FixtureId="FAT FixtureId" ErrorCode="999"
    ErrorMessage="Error code 999">
  <Comment>This is a Comment</Comment>
</UUT>
```

PROCESS ELEMENT

The <Report> element must contain a <Process> element with these attributes. It is recommended to fill out as much data as possible as this significantly improves the usability of the final WATS report.

Attribute Name	Description	Data Format	Importance
Code	Operation type code for a WATS process.	Numeric	Required / Optional if Name is supplied
Name	Name of the operation type for a WATS process.	String	Required / Optional if Code is supplied

IMPLEMENTATION EXAMPLE

```
<Process Code="10" Name="SW Debug" />
```

MISCINFO ELEMENT

Any further data concerning the UUT or the general test run must be entered as miscellaneous UUT data. It can for example be used for software/firmware versions, or for whatever purpose suits the test. There can be any number of <MiscInfo> elements in a <Report>. String data should the content of the element.

Attribute Name	Description	Data Format	Importance
Description	Name of the miscellaneous info	String	Required
Typedef	Defition of the miscinfo type.	String	Optional
Numeric	If the value of the info is numeric it should be set here.	Numeric	Optional

IMPLEMENTATION EXAMPLE

```
<MiscInfo Typedef="" Description="MiscInfoNumeric" Numeric="100"/>
<MiscInfo Typedef="" Description="MiscInfoString>Hello</MiscInfo>
<MiscInfo Typedef="" Description="InvalidXML" >This is a test with &lt; and &gt;</MiscInfo>
```

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. This allow you track systems and their individual component and provide detailed information about the systems lifecycle.

Register a sub unit by adding a <ReportUnitHierarchy> element to a <Report> with these attributes.

Attribute name	Description	Data (Format)	Importance
PartType	Part type	String(50)	Required
SN	Serial number	String(30)	Required
Rev	Revision	String(30)	Optional
PN	Part number	String(30)	Optional

IMPLEMENTATION EXAMPLE

```
<ReportUnitHierarchy PartType="PCBA" PN="SubPartNo1" SN="SubPartSN1"
                     Rev="Rev2" />
```

STEP DATA

The second part of the file contains the actual test step data. Data is entered in nested step tags following the XML Syntax. The first step must be a step with step type Sequence Call. Each <Step> element should have these attributes.

Any step may contain a <ReportText> element which can contain a text adding additional data to the step.

Attribute Name	Description	Data Format	Importance
Group	Which group the step belongs to.	Setup, Main, Cleanup	Required
Name	Name of the test step.	String(100)	Required & Unique
StepType	The action taken by the step.	See Steptypes	Required
total_time	Duration of the step in seconds.	Numeric	Recommended
StepCausedUUTFailure	Set to 1 if step caused the final test status to fail. Useful when logging multiple steps with “Failed” status.	0, 1	Optional
Status	Status of the step.	See Status Codes	Optional

IMPLEMENTATION EXAMPLE

```
<Step Group="Main" Name="ActionStep" StepType="Action" Status="Passed" total_time="0"
        StepCausedUUTFailure="0" />
```

SEQUENCE STEPS

Sequence steps are called within a Step, to create hierarchy structured data (see below)

```
<Step Group="Main" Name="Pass/Fail tests" StepType="SequenceCall" Status="Failed">
   <SequenceCall Name="Pass/Fail tests" Version="1.0.0" Filename="name"/>
   <Step Group="Main" Name="SinglePass" StepType="ET_PFT" Status="Passed">
     <PassFail Status="Passed" />
   </Step>
   <Step Group="Main" Name="SinglePass1" StepType="ET_PFT" Status="Passed">
     <PassFail Status="Passed" />
   </Step>
   <Step Group="Main" Name="SinglePass2" StepType="ET_PFT" Status="Passed">
     <PassFail Status="Passed" />
   </Step>
</Step>
```

Generates the following structure:

Pass/Fail tests

- SinglePass
- SinglePass1
- SinglePass2

We can also have nested SequenceCall, to support even more detailed tests.

```
<Step Group="Main" Name="Pass/Fail tests" StepType="SequenceCall" Status="Failed">
  <SequenceCall Name="Pass/Fail tests" Version="1.0.0" Filename="name"/>
  <Step Group="Main" Name="SinglePass" StepType="ET_PFT" Status="Passed">
    <PassFail Status="Passed" />
  </Step>
  <Step Group="Main" Name="SinglePass1" StepType="ET_PFT" Status="Passed">
    <SequenceCall Name="Nested Pass/Fail" Version="1.0.0" Filename="name"/>
    <Step Group="Main" Name="SinglePass2.1" StepType="ET_PFT" Status="Passed">
      <PassFail Status="Passed" />
    </Step>
    <Step Group="Main" Name="SinglePass2.2" StepType="ET_PFT" Status="Passed">
      <PassFail Status="Passed" />
    </Step>
  </Step>
</Step>
```

Generates the following structure:

Pass/Fail tests

- Pass/Fail tests
  - Test1.1
  - Nested Pass/Fail
    - Test2.1
    - Test2.2

A step with stepType SequenceCall can and should contain a <SequenceCall> element with these attributes:

Attribute Name	Description	Data Format	Importance
Version	Version of the sequence file	String	Recommend
Filename	Filename of the sequence file	String(200)	Recommend
Filepath	Path to the sequence file	String(200)	Recommend
Name	Name of the sequence. If left blank, it will be set to Partnumber from Report header.	String	Optional

TEST STEPS

A step containing a measurement must have a test step as its step type. There are three test step types:

- NumericLimitTest (Keyword: ET_NLT)
  Used to log numeric test data. the required fields are numericValue, units and CompOperator. Name must be present for multiple tests, but not for a single test.

- StringValueTest (Keyword: ET_SVT)
  Used to log string value test data. The required fields are CompOperator, String Value and Status. Name must be present for multiple tests, but not for a single test.

- PassFailTest (Keyword: ET_PFT)
  Used to log pass/fail test data. The required fields are status field. Name must be present for multiple tests, but not for single test. Step types

A step with one of these types must contain the corresponding measurement element.

Test steps comes in two different forms:

1. Single steps (one measurement) (*recommended*).
2. Multiple steps (two or more measurements):
   a. Using single step syntax (original)
   b. Using keyword including “Multiple”.

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

If Status is left blank the WATS engine 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 Status 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, Done, Terminated and Error)

This table explains the proper keyword to use for single and multiple steps.

Type	Keyword (single)	Keyword (multiple)
MultipleNumericLimitTest	StepType=”ET_NLT”	StepType=”ET_MNLT”
MultipleStringValueTest	StepType=”ET_SVT”	StepType=”ET_MSVT”
MultiplePassFailTest	StepType=”ET_PFT”	StepType=”ET_MPFT”

NUMERIC LIMIT

Attribute Name	Description	Data Format	Importance
NumericValue	The measured value.	Numeric	Required
CompOperator	Which comparison operation to use on the measures value.	See Comparison Operations	Required
Name	Name of the limit test.	String(100)	Optional / Required & Unique if multiple tests in same step.
LowLimit	Limit used to compare the measured value in greater than and equal/not equal comparisons.	Numeric	Optional / Required if comparison operation is Greater Than, Equal, or Not Equal
HighLimit	The high limit used to compare measured value in less than comparisons.	Numeric	Optional / Required if Compare Operator is Less Than
Units	Unit of the measured value.	String(20)	Optional
Status	If the test passed or failed. If missing, generated based on measured value, limits, and comparison operator.	See Status Codes	Optional

IMPLEMENTATION EXAMPLE

```
<Step Group="Main" Name="SingleEQPass" StepType="ET_NLT" Status="Passed" total_time="0">
    <NumericLimit NumericValue="500" LowLimit="500" Units="V" CompOperator="EQ"
Status="Passed" />  </Step>
```

STRING VALUE

Attribute Name	Description	Data Format	Importance
StringValue	The string to compare.	String(100)	Required
CompOperator	Which comparison operation to use on the measures value.	See Comparison Operators	Required
Status	If the test passed or failed. If missing, generated based on measured value, limits, and comparison operator.	See Status Codes	Optional
Name	Name of the stringValue test.	String(100)	Optional / Required & Unique if multiple tests in same step.
StringLimit	The string to compare against.	String(100)	Required / Optional if Comparison Operator is Logging

IMPLEMENTATION EXAMPLE

```
<Step Group="Main" Name="SingleInsiensPass" StepType="ET_SVT" Status="Passed"
total_time="0">
    <StringValue CompOperator="IGNORECASE" StringLimit="CaseSensitive"
StringValue="caseSensitive"
      Status="Passed" />
</Step>
```

PASS FAIL

Attribute Name	Description	Data Format	Importance
Status	If the test passed or failed.	String	Required
Name	Name of the pass-fail test.	String(100)	Optional / Required if multiple tests in same step.

IMPLEMENTATION EXAMPLE

```
<Step Group="Main" Name="SinglePass" StepType="ET_PFT" Status="Passed" total time="0">
        <PassFail Status="Passed" />
 </Step>
```

SINGLE STEPS

A single step is used for one measurement. The tests should not have a Name. The following picture displays the syntax for a single step measurement.

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

- It is recommended to use Passed or Failed to simplify.
- The Step name has to be unique to the UUT report

IMPLEMENTATION EXAMPLE

```
<Step Group="Main" Name="SingleEQPass" StepType="ET_NLT" Status="Passed" total_time="0"
">
      <NumericLimit ……" />
 </Step>
```

As in the example above, the “Status” is provided to Passed. The status will therefore not be calculated in the converter. The “Failed” status will ripple through the levels of SequenceCall and up to main report. WATS will fill out this value if it’s not set, based on low and high limits, but not do anything if the field is already set.

MULTIPLE STEPS

A multiple step is when it is necessary to bundle together two or more measurements. This can be done using Single Step Syntax and include one of the following keywords in the “StepType” attribute. If the attribute is set to “measure” WATS will set the correct attribute based on the multiple step data.

SINGLE (MULTIPLE) STEP

The simplest way to write a multiple step, is to use the same syntax as for a single step, except to include a Name in addition for each test element. Name must be set to different values for the different measurements in the multiple measurement step.

Simple Step Syntax for Multiple Steps have some limitations regarding status. The main step status will be calculated based on the measurements since it is not line specifically assigned to this. If statuses are used for multiple steps with Simple Syntax, Done, Terminated and Error will be translated to Failed, both for the individual measurements and for the main step status. Skipped will show for the individual measurement, but the main step will have Passed even if all measurements are Skipped.

IMPLEMENTATION EXAMPLE

```
<Step “StepType=”ET_MNLT” Status=”Failed”……>
 <NumericLimit Name="T1" NumericValue="1" CompOperator="LOG" Status="Passed"/>
 <NumericLimit Name="T1" NumericValue="2" CompOperator="LOG" Status="Failed"/>
</Step>

<Step “StepType=”ET_MNLT” Status=”Passed”……>
 <NumericLimit Name="T3" NumericValue="1" CompOperator="LOG" Status="Passed"/>
 <NumericLimit Name="T4" NumericValue="2" CompOperator="LOG" Status="Passed"/>
</Step>
```

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

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

ACTION STEPS

Used to log test action data and information. Steps with stepType Action may contain <ReportText> elements.

IMPLEMENTATION EXAMPLE

```
<Step Group="Setup" Name="ActionStep" StepType="Action" Status="Done" total_time="0" >
      <ReportText>Here is a report text </ReportText>
</Step>
```

CHART

The WSXF converter supports chart data. The element is called Chart and has the following attributes: (SteptTypes can include a chart, but not more than one chart for each step)

Attribute name	Description	Data (Format)	Importance
Inx	Index of the chart	string	Optional
ChartType	Type of chart. Line is default.	Line, LineLogXY, LineLogX, LineLogY	Optional
Label	Name of the chart	String	Optional
YUnit	Name of the X axis	String	Optional
YLabel	Unit of the X axis	String	Optional
XUnit	Name of the Y axis	String	Optional
XLabel	Unit of the Y axis	String	Optional

SERIES

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

Attribute	Description	Data (Format)	Importance
YData	All data for y-axis.	Semicolon-separated string.	Required
XData	All data for x-axis.	Semicolon-separated string.	Required. If empty, the x-axis will be added with incrementing elements (1,2,3,4 etc).
Name	Name of plot.	String	Optional. If more than one series, it is recommended to include a name.
DataType	Attribute (only XYG supported)	string	Optional.

Each series must contain one <YData> and one <XData> element. These elements must contain a semicolon-separated list of values to be plotted. If <XData> is empty, the values in <YData> will be plotted at their index on the X axis.

- YData and XData can't have more than 10000 entries each.
- The number of elements in Y and X must match.

IMPLEMENTATION EXAMPLE

```
<Step Group="Main" Name="Fibionacci" StepType="WATS_XYGMNLT" Status="Passed" total_time="0">
    <NumericLimit NumericValue="28.923076" Units="Number" CompOperator="LOG" Status="Passed" />
    <Chart ChartType="Line" Label="Fibonacci numbers" XLabel="X" XUnit="" YLabel="Y" YUnit="">
        <Series Name="Fibionacci" DataType="XYG">
            <ydata>0;1;1;2;3;5;8;13;21;34;55;89;144</ydata>
            <xdata>0;1;2;3;4;5;6;7;8;9;10;11;12</xdata>
        </Series>
        <Series Name="Mean" DataType="XYG">
            <ydata>28.923076;28.923076</ydata>
            <xdata>0;12</xdata>
        </Series>
    </Chart>
</Step>
```

ATTACHMENT

The WSXF supports attachments. Attachments can be included in the WSXF file by using the <Attachment> keyword. WSXF uses Base64 encoding/decoding for handling attachments, and therefore require the attachment to be in the form of a byte-array.

Attribute	Description	Data (Format)	Importance
Name	Name for the attachment	String.	Required
ContentType	Type of attachment	String.	Required.
SizeField		String	Optional.

The byte array is parsed as an element, not as an attribute.

IMPLEMENTATION EXAMPLE

```
<Step Group="Main" Name="ByteArray" StepType="WATSFile" Status="Passed">
        <Attachment Name="WiFiNotification" ContentType="image/png"
        >AAAAAAAAAAAElFTkSuQmCC</Attachment>
</Step>
```

MESSAGE POP UP

Message popup is used when a pop-up message is displayed.

Attribute	Description	Data (Format)	Importance
Button	Code for which button was pressed	Numeric	Required
Response	Message from pop up	String	Required.

IMPLEMENTATION EXAMPLE

```
<Step Group="Main" Name="Message Popup" StepType="MessagePopup" Status="Done" total_time="0">
      <MessagePopup Button="1" Response="Response text" />
</Step>
```

CALL EXECUTABLE

Step which contains a reference to an executable.

Attribute	Description	Data (Format)	Importance
ExitCode	ExitCode from the executable	Numeric	Required

IMPLEMENTATION EXAMPLE

```
<Step Group="Main" Name="CallExecutable" StepType="CallExecutable" Status="Done"
        total_time="0">
   <Callexe ExitCode="0" />
</Step>
```

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
Done	Not used.	Step completed.	Can be used for single steps, should not be used for multiple steps	Cyan
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 or Failed or Skipped (but all Skipped measurements will end with Passed for main step). Done or Error or Terminated will be translated to Failed.

COMPARISON OPERATORS

For a complete list of supported Comparison Codes, see
http://support.virinco.com/entries/25251018

LOG FILE

Two folders are set up in the FilePath that is referred to in the Converters.xml file. These are:

- Done
- Error

When a file with the correct file type (.xml in this case) is added to the FilePath, it is picked up by the WATS Standard XML Converter. After the system has imported the data in the file, it is moved to one of the following folders:

- Done (if the process went ok)
- Error (if it did not)

There are two log-files found within Programdata\Virinco folder.

- wats.log; a file for the client which has data about the files that is picked up by the Service and imported through the converter.
- .error: a file that includes errors that occurred in the import process. The report will not be submitted in this case, and the input xml-file will br added to the Error-folder. The warning file will have the xml-file name before dot.
  - Example: Input file named SampleData.xml, the warning file will be named SampleData.error.

WSXF FILE EXAMPLE

A WSXF example file can be found here:

https://virinco.zendesk.com/hc/en-us/article_attachments/360010145192/WATS_-_WSXF_Example_file.xml