

# Exporting a Test

Document Generator allows you to get the following data from the Xray Test:

- [Test Status](#)
- [Test Repository path](#)
- [Iterating over Test Steps](#)
  - [Manual Test Steps](#)
  - [Modular Test Steps](#)
  - [Modular Test Steps Parameters](#)
  - [Manual Test Steps Custom Fields](#)
- [Pre-Conditions associated with a Test](#)
- [Requirements associated with a Test](#)
- [Test Plans associated with a Test](#)
- [Test Runs associated with a Test](#)
  - [Test Execution from a Test Run associated with Test](#)
  - [Exporting Test Runs custom fields](#)
  - [Exporting Parameterized Tests from a Test Run](#)

## Test Status

To get the status of the Test printed on the document you just need to put the following placeholder in your template:

```
{TestRunStatus}
```

## Test Repository path

To get the path of the Test from the Test Repository Board printed on the document you just need to put the following placeholder in your template:

```
{TestRepositoryPath}
```

## Iterating over Test Steps

### Manual Test Steps

**Definition:**

```
{TestSteps[n].Field}
```

n is the index of Test Step, starting from 0. The field **TestStepsCount** was created in order to give the total number of Test Steps. The fields available for Test Steps are:

- StepNumber
- Action
- Data
- ExpectedResult

Expand to see the example on sample code

```
{for teststeps}
  {TestSteps[n].StepNumber}
  {TestSteps[n].Action}
  {TestSteps[n].Data}
  {TestSteps[n].ExpectedResult}
{end}
```

## Modular Test Steps

You can check if a Step is a Call Test with the mapping **IsCalledTest**.

Call Steps have the following fields available:

- IsCalledTest (which will be true if the step is a called step; false if it's a normal step)
- IsDeletedCalledTest (which will indicate if the called step issue was deleted)
- IsInvalidIssueTypeCalledTest (which will indicate if the called step issue type has changed)
- CalledTestParameters (the Called Test Dataset Parameters in Json format)
- CalledTest.Field (*The issue called. You can export its information*)

Example:

#### Expand to see the example on sample code

```
{for teststeps}
  {if ( ${TestSteps[n].IsCalledTest} )}
    {if ( ${TestSteps[n].IsDeletedCalledTest} )}
      Called test issue not found.
    {end}
    {if ( ${TestSteps[n].IsInvalidIssueTypeCalledTest} )}
      Called test issue type is invalid.
    {end}
    {if ( ${!TestSteps[n].IsDeletedCalledTest} && !${TestSteps[n].IsInvalidIssueTypeCalledTest} )}
      ${TestSteps[n].CalledTestParameters}
      ${TestSteps[n].CalledTest.Key}
      ${TestSteps[n].CalledTest.Summary}
    {end}
  {end}
{end}
```

## Modular Test Steps Parameters

You can iterate over the Test Call Parameters:

#### Expand to see the example on sample code

```
{for teststeps}
  {if ( ${TestSteps[n].IsCalledTest} )}
    {for p=TestSteps[n].CalledTestParametersCount}
      ${TestSteps[n].CalledTestParameters[p].Name}
      ${TestSteps[n].CalledTestParameters[p].Value}
    {end}
  {end}
{end}
```

## Manual Test Steps Attachments

Document Generator allows rendering all Attachments of Manual Test Steps of a Test.

**Definition:**

`${TestSteps[n].Attachments[m].Field}`

n is the index of Test Step and m is the index of the Attachment. The field **AttachmentsCount** was created in order to give the total number of Attachments. The fields available for Test Step Attachments are:

- ID
- Name
- Author
- AuthorFullName
- Created
- Size
- HumanReadableSize
- MimeType

- FileURL

#### Expand to see the example on sample code

```
#{for teststeps}
  #{for m=TestSteps[n].AttachmentsCount}
    ${TestSteps[n].Attachments[m].ID}
    ${TestSteps[n].Attachments[m].Name}
    ${TestSteps[n].Attachments[m].Author}
    ${TestSteps[n].Attachments[m].AuthorFullName}
    ${TestSteps[n].Attachments[m].Created}
    ${TestSteps[n].Attachments[m].Size}
    ${TestSteps[n].Attachments[m].HumanReadableSize}
    ${TestSteps[n].Attachments[m].MimeType}
    ${TestSteps[n].Attachments[m].FileURL}
  #{end}
#{end}
```

## Manual Test Steps Custom Fields

To export Test Steps Custom Fields you just have to define the placeholder with the name of your custom field.

Example: You have a custom field called "Run CF". To get the value printed on your document you just have to use the following placeholder:

```
#Iterating over Test Runs
#{for teststeps}
  Run CF: ${TestSteps[n].Run CF}
#{end}
```



If your custom field type is a Number, Data, or Date Time you can use [formatting functions](#).

## Pre-Conditions associated with a Test

Document Generator allows rendering of all the Pre-Conditions associated with a Test.

#### Definition:

```
${PreConditions[n].Field}
```

**n** is the index of the Pre-Condition, starting from 0. The field **PreConditionsCount** or **PreconditionsCount** was created in order to give the total number of Pre-Conditions.

Since Pre-Condition is a Jira Issue, you can render all the normal mappings which you are used to.

Example:

#### Expand to see the example on sample code

```
#{for preconditions}
  ${PreConditions[n].Key}
  ${PreConditions[n].Summary}
  ${PreConditions[n].Description}
  ${PreConditions[n].Pre-Condition Type}
  ${PreConditions[n].Conditions}
#{end}

or

#{for j=PreConditionsCount}
  ${PreConditions[j].Key}
  ${PreConditions[j].Summary}
#{end}
```

```
    ${PreConditions[j].Description}
    ${PreConditions[j].Pre-Condition Type}
    ${PreConditions[j].Conditions}
#{end}
```

## Requirements associated with a Test

You can print all the Requirements associated with a Test using an [Xray Enhanced querying JQL Function](#), where you input the given Test Key:

### JQL Function:

```
testRequirements('${Key}')
```

In order to give the total of Requirements associated with a given Test, you can use the following JQL Count statement, where you input the given Test Key:

```
$(jqlcount:testRequirements('${Key}'))
```

**n** is the index of the Pre-Condition, starting from 0. The field **PreConditionsCount** was created in order to give the total number of Pre-Conditions.

Since a Requirement is a Jira Issue, you can render all the normal mappings which you are used to.

Example:

### Expand to see the example on sample code

```
#{for k=JQLIssuesCount|clause=key in testRequirements('${Key}')}
  ${JQLIssues[k].Key}
  ${JQLIssues[k].Summary}
  ${JQLIssues[k].Description}
#{end}
```

You can also Iterate over all the Requirements associated with a Test while iterating over all Tests of a Test Set.

### Expand to see the example on sample code

```
#{for tests}
  ${Tests[n].Key}
  ${Tests[n].Summary}
  #{for k=JQLIssuesCount|clause=key in testRequirements('Tests[n].Key')}
    ${Tests[n].JQLIssues[k].Key}
    ${Tests[n].JQLIssues[k].Summary}
    ${Tests[n].JQLIssues[k].Description}
  #{end}
#{end}
```

## Test Plans associated with a Test

### Definition:

```
$(TestPlans[n].Field)
```

**n** is the index of the Test Plan, starting from 0. The field **TestPlansCount** was created in order to give the total number of Test Plans.

Since a Test Plan is a Jira Issue, you can render all the normal mappings which you are used to.

Example:

#### Expand to see the example on sample code

```
#{for testPlans}
    ${TestPlans[n].Key}
    ${TestPlans[n].Summary}
    ${TestPlans[n].Description}
#{end}
```

## Test Runs associated with a Test

**Definition:**

`${TestRuns[n].Field}`

**n** is the index of the Test Run, starting from 0. The field **TestRunsCount** was created in order to give the total number of Test Runs.

Example:

#### Expand to see the example on sample code

```
#Iterating over Test Runs
#{for testruns}
    Execution Status: ${TestRuns[n].Execution Status}
    AssigneeID: ${AssigneeId}
    Rank: ${TestRuns[n].Rank}
    Executed By: ${TestRuns[n].Executed By}
    Started On: ${TestRuns[n].Started On}
    Finished On: ${TestRuns[n].Finished On}
    Comment: ${wiki:TestRuns[n].Comment}
    Execution Defects Count: ${TestRuns[n].ExecutionDefectsCount}
    TestSteps Defects Count: ${TestRuns[n].TestStepsDefectsCount}
    Evidences Count: ${TestRuns[n].ExecutionEvidencesCount}
#{end}
```

## Test Execution from a Test Run associated with Test

It is possible to access Test Execution information by iterating over tests and their Test Runs. The notation is:

#### Expand to see the example on sample code

```
#{for tests}
    ${Tests[n].Key}
    #{for c=Tests[n].TestRunsCount}
        ${Tests[n].TestRuns[c].Executed By}
        ${Tests[n].TestRuns[c].PreCondition.Key}

        ${Tests[n].TestRuns[c].TestExecution.Key}
        ${Tests[n].TestRuns[c].TestExecution.Summary}
        ${Tests[n].TestRuns[c].TestExecution.Description}
        .....
    #{end}
#{end}
```

## Exporting Test Runs custom fields

To export Test Runs Custom Fields you just have to define the placeholder with the name of your custom field.

Example: Image that you have a custom field called "Run CF". To get the value printed on you document you just have to use the following placeholder:

```

#{for testruns}
The Run CF value is: ${TestRuns[n].Run CF}
#{end}

```



If your custom field type is a Number, Data or Date Time you can use [formatting functions](#).

## Exporting Parameterized Tests from a Test Run

To export all this data, we first need to execute the test run. If we change any data, we need to return to the Execution Page and merge it with the new data.

Fields	Description
IsDataDriven	Returns "true" if the current test run is data-driven; otherwise, it returns "false"
Iterations Overall Execution Status.STATUS.Percentage	The percentage of STATUS in the test run iterations
IterationsOverallExecutionStatus	List of all the statuses in the current test run iteration and their percentages
IterationsOverallExecutionStatus.STATUS.Count	The number of STATUS in the test run iterations
Parameters	List of the parameters from each test run iteration
ParametersCount	Total of Parameters declared in Dataset



### Data-driven

The field IsDataDriven is going to return TRUE only if the test run has more than one iteration.

## Exporting Test Runs Parameters from a Test

For each Test Run Parameter you can export the following fields:

- Key
- Value

Below you can find an example of how to iterate over the list of **Test Run Parameters** associated with a **Test**.

### Some mappings we can export from Test Runs Parameters

```

// Iterating each test run
#{for testruns}
  // Iterating over parameters for each test run
  Parameters Total: ${TestRuns[n].ParametersCount}
  #{for m=TestRuns[n].ParametersCount}
    Key: ${TestRuns[n].Parameters[m].Key}
    Value: ${TestRuns[n].Parameters[m].Value}
  #{end}
#{end}

```

## Exporting Test Runs Iterations from a Test

For each Test Run Iteration you can export the following fields:

- Overall Execution Status
- Data from Test Run Iterations
- Parameters from Test Run Iterations
- PreConditions from Test Run Iterations
- Test steps from Test Run Iterations

Below you can find an example of how to iterate over the list of **Test Run Iterations** associated with a **Test**.

### Some mappings we can export from Test Run Iterations

```

// Iterating each test run

```

```

#{for testruns}
  IsDataDriven: ${TestRuns[n].IsDataDriven}

  // Iterations Overall Execution Status (percentage + total of testes per status)
  List of Statuses: ${TestRuns[n].Iterations Overall Execution Status}
  TO DO: ${TestRuns[n].Iterations Overall Execution Status.TODO}% - ${TestRuns[n].Iterations Overall
Execution Status.TODO.Count}
  EXECUTING: ${TestRuns[n].Iterations Overall Execution Status.EXECUTING}% - ${TestRuns[n].Iterations Overall
Execution Status.EXECUTING.Count}
  PASSED: ${TestRuns[n].Iterations Overall Execution Status.PASS}% - ${TestRuns[n].Iterations Overall
Execution Status.PASS.Count}
  FAILED: ${TestRuns[n].Iterations Overall Execution Status.FAIL}% - ${TestRuns[n].Iterations Overall
Execution Status.FAIL.Count}
  ABORTED: ${TestRuns[n].Iterations Overall Execution Status.ABORTED}% - ${TestRuns[n].Iterations Overall
Execution Status.ABORTED.Count}

  // Iterating over test runs iterations
  Total of Iterations from a Test Run: ${TestRuns[n].IterationsCount}
  #{for m=TestRuns[n].IterationsCount}
    Name: ${TestRuns[n].Iterations[m].Name}
    Status: ${TestRuns[n].Iterations[m].Status}
    Parameters: ${TestRuns[n].Iterations[m].Parameters}

    // Iterating over parameters for each test run iteration
    Parameters Total: ${TestRuns[n].Iterations[m].ParametersCount}
    #{for l=TestRuns[n].Iterations[m].ParametersCount}
      Key: ${TestRuns[n].Iterations[m].Parameters[l].Key}
      Value: ${TestRuns[n].Iterations[m].Parameters[l].Value}
    #{end}

    // Iterating over preconditions for each test run iteration
    Preconditions Total: ${TestRuns[n].Iterations[m].PreConditionsCount}
    #{for l=TestRuns[n].Iterations[m].PreConditionsCount}
      Key: ${TestRuns[n].Iterations[m].PreConditions[l].Key}
      Summary: ${TestRuns[n].Iterations[m].PreConditions[l].Summary}
      Definition: ${TestRuns[n].Iterations[m].PreConditions[l].Conditions}
      Type: ${TestRuns[n].Iterations[m].PreConditions[l].Pre-Condition Type}
    #{end}

    //Iterating over test steps for each test run iteration
    #{for i=TestRuns[n].Iterations[m].TestStepsCount}
      Step Number: ${TestRuns[n].Iterations[m].TestSteps[i].StepNumber}
      Action: ${TestRuns[n].Iterations[m].TestSteps[i].Action}
      Data: ${TestRuns[n].Iterations[m].TestSteps[i].Data}
      Expected Result: ${TestRuns[n].Iterations[m].TestSteps[i].ExpectedResult}
      Status: ${TestRuns[n].Iterations[m].TestSteps[i].Status}
      Comment: ${TestRuns[n].Iterations[m].TestSteps[i].Comment}
      Actual Result: ${TestRuns[n].Iterations[m].TestSteps[i].Actual Result}
      // Replace the placeholder text to export any custom field associated with the test step.
      Step Custom Field:${TestRuns[n].Iterations[m].TestSteps[i].<Step Custom Field>}

      // Iteration Test Step Attachments
      #{for l=TestRuns[n].Iterations[m].TestSteps[i].AttachmentsCount}
        Id: ${TestRuns[n].Iterations[m].TestSteps[i].Attachments[l].Id}
        Name: ${TestRuns[n].Iterations[m].TestSteps[i].Attachments[l].Name}
        Image: ${TestRuns[n].Iterations[m].TestSteps[i].Attachments[l].Attachment}
        FileURL: ${TestRuns[n].Iterations[m].TestSteps[i].Attachments[l].FileURL}
      #{end}

      // Iteration Test Step Evidences
      #{for l=TestRuns[n].Iterations[m].TestSteps[i].EvidencesCount}
        Id: ${TestRuns[n].Iterations[m].TestSteps[i].Evidences[l].Id}
        Name: ${TestRuns[n].Iterations[m].TestSteps[i].Evidences[l].Name}
        Evidence: ${TestRuns[n].Iterations[m].TestSteps[i].Evidences[l].Evidence}
      #{end}

      // Iteration Test Step Defects
      #{for l=TestRuns[n].Iterations[m].TestSteps[i].DefectsCount}
        Description: ${TestRuns[n].Iterations[m].TestSteps[i].Defects[l].Description}
        Id: ${TestRuns[n].Iterations[m].TestSteps[i].Defects[l].Id}
        Key: ${TestRuns[n].Iterations[m].TestSteps[i].Defects[l].Key}
        Summary: ${TestRuns[n].Iterations[m].TestSteps[i].Defects[l].Summary}
      #{end}
    #{end}
  #{end}

```

```
    # {end}  
  # {end}  
# {end}
```