Iterations

Iterating Issue Links

Because it is not known in advance how many linked issues exist for an issue, you can iterate a section over all the linked issues of an issue. This allows you to create a table that dynamically grows according to the number of existing linked issues.

All fields listed here are available on Links[n] because they represent an issue. In addition, there are two new fields at the u can also use a Filter Name or a Fi[n] level:

Field	Description
AppType	Returns the Application Type. The values can be:
LinkType	Returns the Link Type.

Application Value	Description
JIRA	Link from the same Jira Instance
External Jira	Link from the another Jira Instance
Confluence	Link from a Confluence page
External	External link

Expand to see the sample code

```
#{for links}
   ${Links[n].AppType}
   ${Links[n].LinkType}
   ${Links[n].Key}
   ${Links[n].Summary}
   ${Links[n].URL}
#{end}

or

#{for <VariableName>=LinksCount}
   Content and Linked Issue Mappings. Example: ${Links[VariableName].Field}
#{end}
```

Note: When the link you are iterating is of AppTypes **External Jira** or **Confluence,** the name is obtained using the Summary property.

The documents below demonstrate examples both in Word and Excel template that iterates over linked issues.

Iterating_Issue_Links.docx

Iterating_Issue_Links.xlsx

Iterating Issue Comments

Because it is not known in advance how many comments exist for an issue, you can iterate a section over all the comments on an issue. This allows you to create a table that dynamically grows according to the number of existing comments. The notation is:

Comments Fields	Description
Author	The author of the comment
AuthorFullName	The full name of the author of the comment
Body	The comment body WIKI
Created	The date the comment was posted
CreatedDate	The date the comment was posted
CreatedDateTime	The date the comment was posted
GroupLevel	The group level of the comment
Internal	The comment is internal or public

Expand to see the sample code

```
#{for comments}
   ${Comments[n].Author}
   ${Comments[n].AuthorFullName}
   ${Comments[n].Body}
   ${dateformat("dd-MM-yyyy HH:mm:ss"):Comments[n].Created}
   ${dateformat("dd-MM-yyyy HH:mm:ss"):Comments[n].CreatedDate}
   ${dateformat("dd-MM-yyyy HH:mm:ss"):Comments[n].CreatedDateTime}
   ${Comments[n].GroupLevel}
   ${Comments[n].Internal}
#{end}

or

#{for <VariableName>=CommentsCount}
   Content and Issue Mappings. Example: ${Comments[VariableName].Field}
#{end}
```

The documents below demonstrate examples both in Word and Excel template that iterates over all the issue comments.

Iterating_Issue_Comments.docx

Iterating_Issue_Comments.xlsx

Iterating Issue Worklogs

Because it is not known in advance how many worklogs exist for an issue, you can iterate a section over all the worklogs of an issue. This allow you to create a table that dynamically grows according to the number of existing worklogs. The notation is:

Worklogs Fields	Description
Author	The author of the worklog
AuthorFullName	The full name of the author of the worklog

Comment	The comment of the worklog WIKI
Created	The worklog's creation date.
CreatedDate	The worklog's creation date.
CreatedDateTime	The worklog's creation date.
Date Started	The date the worklog was started
StartDate	The date the worklog was started
StartDateTime	The date the worklog was started
TimeSpent	The time spent in seconds
Time Spent	The time spent in seconds
TimeSpentFormatted	The time spent as displayed on Jira
BillableSeconds	The billable seconds (**Belongs to Tempo Timesheets plugin**)

Expand to see the sample code

```
#{for worklogs}
   ${Worklogs[n].Author}
   ${Worklogs[n].AuthorFullName}
   ${Worklogs[n].Comment}
   ${dateformat("dd-MM-yyyy HH:mm:ss"):Worklogs[n].Created}
   ${dateformat("dd-MM-yyyy HH:mm:ss"):Worklogs[n].CreatedDate}
   ${dateformat("dd-MM-yyyy HH:mm:ss"):Worklogs[n].CreatedDateTime}
   ${dateformat("dd-MM-yyyy HH:mm:ss"):Worklogs[n].StartDate}
   ${dateformat("dd-MM-yyyy HH:mm:ss"):Worklogs[n].Date Started}
   ${dateformat("dd-MM-yyyy HH:mm:ss"):Worklogs[n].StartDateTime}
   ${Worklogs[n].Time Spent}
   ${Worklogs[n].TimeSpent}
   ${Worklogs[n].TimeSpentFormatted}
   ${Worklogs[n].BillableSeconds}
#{end}

or

#{for <VariableName>=WorklogsCount}
   Content and Worklog Mappings. Example: ${Worklogs[VariableName].Field}
#{end}
```

The documents below demonstrate examples both in Word and Excel template that iterates over the issue worklogs.

Iterating_Issue_Worklogs.docx

Iterating_Issue_Worklogs.xlsx

Iterating Issue Subtasks

All fields listed here are available on Subtasks[n] because they represent an issue.

Because it is not known in advance how many subtasks exist for an issue, you can iterate a section over all the subtasks of an issue. This allows you to create a table that dynamically grows according to the number of existing subtasks. The notation is:

Subtasks Fields	**Description**
Key	The key of the subtasks
Summary	The summary of the subtasks
AssigneeUserDisplayName	The assignee user of the subtasks

ParentIssueKey	The issue parent key

Expand to see the sample code

```
#{for subtasks}
   ${Subtasks[n].Key}
   ${Subtasks[n].Summary}
   ${Subtasks[n].AssigneeUserDisplayName}
   ${Subtasks[n].ParentIssueKey}
#{end}

or

#{for <VariableName>=SubtasksCount}
   Content and Issue Mappings. Example: ${Subtasks[VariableName].Field}
#{end}
```

The documents below demonstrate examples both in Word and Excel template that iterates over the issue subtasks.

Iterating_Issue_Subtasks.docx

Iterating_Issue_Subtasks.xlsx

Iterating Issue Components

Because it is not known in advance how many components exist for an issue, you can iterate a section over all the components of an issue. This allows you to create a table that dynamically grows according to the number of existing components. The notation is:

Components Fields	Description
Name	The name of the component
Description	The description of the component
Lead	The name of the component lead
Id	The ID of the component
ProjectId	The project ID of the component
AssigneeType	The assignee type of the component

Expand to see the sample code

```
#{for components}
   ${Components[n].Name}
   ${Components[n].Description}
   ${fullname:Components[n].Lead}
   ${Components[n].Id}
   ${Components[n].ProjectId}
   ${Components[n].AssigneeType}
#{end}

#{for <VariableName>=ComponentsCount}
   Content and Components Mappings. Example: ${Components[VariableName].Field}
#{end}
```

The documents below demonstrate examples both in Word and Excel template that iterates over the issue components.

Iterating_Issue_Components.docx

Iterating_Issue_Components.xlsx

Iterating Issue Status Transitions

Because it is not known in advance how many Status Transitions exist for an issue, you can iterate a section over all the Status Transitions of an issue. This allows you to create a table that dynamically grows according to the number of existing status transitions. The notation is:

Status Transitions Fields	Description
Author	The author of the status transition
Created	The date the status transition was performed
CreatedDate	The date the status transition was performed
CreatedDateTime	The date the status transition was performed
OldStatus	The old status of the status transition
NewStatus	The new status of the status transition

Expand to see the sample code

```
#{for statusTransitions}
   ${StatusTransitions[n].Author}
   ${dateformat("dd-MM-yyyy HH:mm:ss"):StatusTransitions[n].Created}
   ${dateformat("dd-MM-yyyy HH:mm:ss"):StatusTransitions[n].CreatedDate}
   ${dateformat("dd-MM-yyyy HH:mm:ss"):StatusTransitions[n].CreatedDateTime}
   ${StatusTransitions[n].OldStatus}
   ${StatusTransitions[n].NewStatus}
#{end}

or

#{for <VariableName>=StatusTransitionsCount}
   Content and StatusTransitions Mappings. Example: ${StatusTransitions[VariableName].Field}
#{end}
```

The documents below demonstrate examples both in Word and Excel template that iterates over the issue status transitions.

Iterating_Issue_StatusTransitions.docx

Iterating_Issue_StatusTransitions.xlsx

Iterating Issue Attached Images

Because it is not known in advance how many Images can exist for an issue (as an attachment), you can iterate a section over all the attached images of an issue to get some metadata about them. This allows you to create a table that dynamically grows according to the number of existing images. The notation is:

Attachments Images Fields	Description
ID	The ID of the attached image
Image	The image of the attached image
Name	The name of the attached image
Size	The size of the attached image
HumanReadableSize	The size of the attached image
Author	The author (ID) of the attached image
Created	The date the attached image was created
CreatedDate	The date the attached image was created

CreatedDateTime	The date the attached image was created
MimeType	The type of the attached image
ThumbnailURL	The URL to the thumbnail of the image

Expand to see the sample code

```
#{for images}
   ${Images[n].Image|maxwidth=150|maxheight=150}
   ${Images[n].Name}
   ${Images[n].ID}
   ${Images[n].Size}
   ${Images[n].HumanReadableSize}
   ${Images[n].Author}
   ${dateformat("dd-MM-yyyy HH:mm:ss"):Images[n].Created}
   ${dateformat("dd-MM-yyyy HH:mm:ss"):Images[n].CreatedDate}
   ${dateformat("dd-MM-yyyy HH:mm:ss"):Images[n].CreatedDateTime}
   ${Images[n].MimeType}
   ${Images[n].ThumbnailURL}
 #{end}

or

#{for <VariableName>=ImagesCount}
   Content and Images Mappings. Example: ${Images[VariableName].Field}
#{end}
```

The documents below demonstrate examples both in Word and Excel template that iterates over the attached images for each issue.

Iterating_Issue_Images.docx

Iterating_Issue_Images.xlsx

Xporter will automatically read the EXIF orientation property of an image and rotate it to its correct orientation. You can turn this off by adding thi s property to your template.

You can use the mappings width and height to define the exact width and height of the printed image.

Expand to see the sample code

```
#{for images}
   ${Images[n].Image|width=150|height=150}
 #{end}
```

These values are in pixels and if you only define one of them the image will be rescaled.

Note that, if you use both maxWidth and width mappings, only the max value will be read. The same behavior happens with height and maxHeight.

Iterating Issue Attachments

Because it is not known in advance how many attachments exist in an issue, you can iterate a section over all the attachments of an issue. This allows you to create a table that dynamically grows according to the number of existing attachments. The notation is:

Attachments Fields	Description
ID	The ID of the attachment

Id	The ID of the attachment
Name	The name of the attachment
Author	The author of the attachment
AuthorFullName	The full name of the author of the attachment
Created	The date the attachment was created
CreatedDate	The date the attachment was created
CreatedDateTime	The date the attachment was created
Size	The size of the attachment
HumanReadableSize	The formatted size of the attachment
MimeType	The type of the attachment

Expand to see the sample code

```
#{for attachments}
   ${Attachments[n].ID}
   ${Attachments[n].Name}
   ${Attachments[n].Author}
   ${Attachments[n].AuthorFullName}
   ${dateformat("dd-MM-yyyy HH:mm:ss"):Attachments[n].Created}
   ${dateformat("dd-MM-yyyy HH:mm:ss"):Attachments[n].CreatedDate}
   ${dateformat("dd-MM-yyyy HH:mm:ss"):Attachments[n].CreatedDateTime}
   ${Attachments[n].Size}
   ${Attachments[n].HumanReadableSize}
   ${Attachments[n].MimeType}
#{end}

or

#{for <VariableName>=AttachmentsCount}
   Content and Issue Mappings. Example: ${Attachments[VariableName].Field}
#{end}
```

The documents below demonstrate examples both in Word and Excel template that iterates over the issue's attachments.

Iterating_Issue_Attachments.docx

Iterating_Issue_Attachments.xlsx

Iterating Issue Labels

Because it is not known in advance how many labels exist in an issue, you can iterate a section over all the labels of an issue. The notation is:

Attachments Fields	Description
Name	The name of the label

Expand to see the sample code

```
#{for labels}
   ${Labels[n].Name}
#{end}

or

#{for <VariableName>=LabelsCount}
   Content and Versions Issue Mappings. Example: ${Labels[VariableName].Field}
#{end}
```

The documents below demonstrate examples both in Word and Excel template that iterates over the issue's labels.

Iterating_Issue_Labels.docx

Iterating_Issue_Labels.xlsx

Iterating Fix Versions of an Issue

You can iterate over all fix versions to which the issue belong to. The notation is:

Versions Fields	Description
Name	The version name
Description	The version description
Start date	Starting date of the version
Release date	Release date of the version
Archived	Boolean that indicates if the version is archived or not
Released	Boolean that indicates if the version is released or not

Expand to see the sample code

```
#{for FixVersions}
        ${FixVersions[n].Name}
        ${FixVersions[n].Description}
        ${dateformat(“dd-MM-yyyy”):FixVersions[n].Start date}
        ${dateformat(“dd-MM-yyyy”):FixVersions[n].Release date}
        ${FixVersions[n].Archived}
        ${FixVersions[n].Released}
#{end}

or

#{for <VariableName>=FixVersionsCount}
   Content and Versions Issue Mappings. Example: ${FixVersions[VariableName].Field}
#{end}
```

The documents below demonstrate examples both in Word and Excel template that iterates over the issue's fix versions.

Iterating_Issue_FixVersions.docx

Iterating_Issue_FixVersions.xlsx

Iterating Affected Versions of an Issue

You can iterate over all affected versions to which the issue belongs to. The notation is:

Versions Fields	Description
Name	The version name
Description	The version description
Start date	Starting date of the version
Release date	Release date of the version
Archived	Boolean that indicates if the version is archived or not
Released	Boolean that indicates if the version is released or not

Expand to see the sample code

```
#{for AffectedVersions}
        ${AffectedVersions[n].Name}
        ${AffectedVersions[n].Description}
        ${dateformat(“dd-MM-yyyy”):AffectedVersions[n].Start date}
        ${dateformat(“dd-MM-yyyy”):AffectedVersions[n].Release date}
        ${AffectedVersions[n].Archived}
        ${AffectedVersions[n].Released}
#{end}

or

#{for <VariableName>=AffectedVersionsCount}
   Content and Versions Issue Mappings. Example: ${AffectedVersions[VariableName].Field}
#{end}
```

The documents below demonstrate examples both in Word and Excel template that iterates over the issue's affected versions.

Iterating_Issue_AffectedVersions.docx

Iterating_Issue_AffectedVersions.xlsx

Iterating Project Versions

You can iterate over all project versions to which the issue belongs to. The notation is:

Project Versions Fields	Description
Name	The version name
Description	The version description
Start date	Starting date of the version
Release date	Release date of the version
Archived	Boolean that indicates if the version is archived or not
Released	Boolean that indicates if the version is released or not

Expand to see the smaple code

```
#{for projectVersions}
        ${ProjectVersions[n].Name}
        ${ProjectVersions[n].Description}
        ${dateformat(“dd-MM-yyyy”):ProjectVersions[n].Start date}
        ${dateformat(“dd-MM-yyyy”):ProjectVersions[n].Release date}
        ${ProjectVersions[n].Archived}
        ${ProjectVersions[n].Released}
#{end}

or

#{for <VariableName>=ProjectVersionsCount}
   Content and Project Versions Mappings. Example: ${ProjectVersions[VariableName].Field}
#{end}
```

The documents below demonstrate examples both in Word and Excel template that iterates over the issue's project versions.

Iterating_Issue_ProjectVersions.docx

Iterating_Issue_ProjectVersions.xlsx

Iterating Sprints

You can iterate over all sprints to which the issue belongs. The notation is:

Project Versions Fields	Description
Name	The sprint name
Status	The sprint status

Expand to see the smaple code

```
#{for sprints}
        ${Sprints[n].Name}
        ${Sprints[n].Status}
#{end}

or

#{for <VariableName>=SprintsCount}
   Content and Sprints Mappings. Example: ${Sprints[VariableName].Field}
#{end}
```

The documents below demonstrate examples both in Word and Excel template that iterates over the issue's project versions.

Iterating_Issue_Sprints.docx

Iterating_Issue_Sprints.xlsx

Iterating Issue History Entries

You can iterate over all issue's changelogs. The notation is:

Project Versions Fields	Description

Author	The user who made the change
Created	Date of the change
CreatedDate	Date of the change
CreatedDateTime	Date of the change
ChangedItemsCount	Number of items changed

Expand to see the smaple code

```
#{for historyEntries}
        ${HistoryEntries[n].Author}
        ${HistoryEntries[n].Created}
        ${HistoryEntries[n].CreatedDate}
        ${HistoryEntries[n].CreatedDateTime}
        ${HistoryEntries[n].ChangedItemsCount}
        #{for i=HistoryEntries[n].ChangedItemsCount}
                ${HistoryEntries[n].ChangedItems[i].Field}
                ${HistoryEntries[n].ChangedItems[i].From}
                ${HistoryEntries[n].ChangedItems[i].To}
        #{end}
#{end}

or

#{for <VariableName>=HistoryEntriesCount}
   Content and History Entries Mappings. Example: ${HistoryEntries[VariableName].Field}
#{end}
```

The documents below demonstrate examples both in Word and Excel template that iterates over the issue's changelogs.

Iterating_Issue_Histories.docx

Iterating_Issue_Histories.xlsx

Iterating Project Components

You can iterate over all project components. The notation is:

Expand to see the sample code

```
#{for ProjectComponents}
   ${ProjectComponents[n].Name}
   ${ProjectComponents[n].Description}
   ${fullname:ProjectComponents[n].Lead}
   ${ProjectComponents[n].Id}
   ${ProjectComponents[n].ProjectId}
   ${ProjectComponents[n].AssigneeType}
#{end}

#{for <VariableName>=ProjectComponentsCount}
   Content and Components Mappings. Example: ${ProjectComponents[VariableName].Field}
#{end}
```

The documents below demonstrate examples both in Word and Excel template that iterates over the project components.

Iterating_Issue_ProjectComponents.docx

Iterating_Issue_ProjectComponents.xlsx

Iterating Issues In Epic

All fields listed here are available on IssuesInEpic[n] because they represent an issue.

Because it is not known in advance how many issues exist for an epic, you can iterate a section over all the issues of an epic issue. This allows you to create a table that dynamically grows according to the number of existing issues. The notation is:

Expand to see the sample code

```
#{for IssuesInEpic}
   ${IssuesInEpic[n].Key}
   ${IssuesInEpic[n].Summary}
   ${IssuesInEpic[n].Description}
   ${IssuesInEpic[n].Epic Link.Key}
#{end}

or

#{for <VariableName>=IssuesInEpicCount}
   Content and Issue Mappings. Example: ${IssuesInEpic[VariableName].Field}
#{end}
```

The documents below demonstrate examples both in Word and Excel template that iterates over the issues in epic.

Iterating_Issues_In_Epic.docx

Iterating_Issues_In_Epic.xlsx

Iterating JQL Queries

You can iterate issues that are the result of a JQL Query. The syntax is similar to the other iterations, but there is a **clause** parameter that will receive the JQL Query. A few examples are provided below.

Expand to see the sample code

```
a simple example iterating the details of issues from a specified Project:

#{for i=JQLIssuesCount|clause=project = DEMO}
   ${JQLIssues[i].Key}
   ${JQLIssues[i].Summary}
#{end}

or a more advanced example iterating the details of the Parent issue from the current Subtask:

#{for i=JQLIssuesCount|clause=issuekey = ${ParentIssueKey}}
        ${JQLIssues[i].Key}
        ${JQLIssues[i].Id}
        ${JQLIssues[i].Description}
#{end}
```

The documents below demonstrate examples both in Word and Excel template with JQL examples.

JQL.docx

JQL.xlsx

You can also use a Filter Name or a Filter Id as a clause. For more info, check [http://confluence.xpand-addons.com/display/public/XPORTER/JQL]

Applying filters to Iterations

If you want to take the previous iterations over comments, subtasks and issue links to another level of control, you can use a JavaScript filter to define over which issues the iteration will be made. This can be useful in the following scenarios:

- Iterating over linked issues that are only of a specific issue type
- Iterating over subtasks of a specific issue type
- Iterating over linked issues with a specific priority
- Iterating over comments created by a specific user

The notation for applying filters to the iterations is:

Expand to see the sample code

```
#{for <VariableName>=<LinksCount|SubtasksCount|CommentsCount|WorklogsCount>|filter=%{<Javascript>}}
   Content here
#{end}
```

- **VariableName:** is the name of the variable to use as the iteration index.
- **LinksCount|SubtasksCount|CommentsCount:** indicates over which type of entities you want to iterate.
- **Filter:** indicates the filter to be applied in the iteration.

Notice that the filter is evaluated as a JavaScript expression, which provides flexibility in the definition of the conditions. You can use and (&&), or (||) and other logical operators supported by the JavaScript language.

It is also possible to format fields inside iteration filters. For more information on formatters, see Iterations.

The document below demonstrates an example of a template that iterates over issue links and comments with filters being applied.

Links_with_Filter_and_HighPriority.docx

Nested Iterations

You can have multiple levels of iterations inside other iterations. This can be useful if you want to iterate the comments of each linked issue or the attached images for each subtask.

Expand to see the sample code

```
#{for <VariableName1>=LinksCount}
          #{for <VariableName2>=Iteration[n].Count}
                Content here
        #{end}
#{end}
```

The document below demonstrates multiple scenarios where nested iterations can be useful.

Nested_iterations.docx

Iterating in the same line of the document

You can also possible to iterate values in the same line of the document. This can be useful if you want to display a list of Subtasks on Linked Issues in the same line, separated by commas or spaces.

Expand to see the sample code

```
Users that added comments to this issue: #{for comments}${Comments[n].Author} #{end}

Subtasks of this issue: #{for j=SubtasksCount}${Subtasks[j].Key};#{end}

Linked issues this issue duplicates: #{for j=LinksCount|filter=%{'${Links[j].LinkType}'.equals
('duplicates')}}${Links[j].Key} #{end}
```

Iterating in the same cell in an Excel document

You can also iterate values in the same cell in an Excel document. You can achieve this by simply making your Iteration inside the same cell.

You can use all the Iterations that you are used to and construct them in the exact same way, the difference being that you only use one cell to do them.

Expand to see the sample code

```
Issue iteration as a demonstration.
Copy this iteration below and paste it into a cell.

&{for issues} ${Key} &{end}
```

Iterating with the BREAK or CONTINUE statement

You can iterate anything, set up a Conditional expression and then utilize the BREAK and CONTINUE statements.

The way to do this is by doing a normal Conditional expression and using the mapping #{break} or #{continue} inside it.

Expand to see the sample code

```
Imagine that you have a Jira Issue that contains these comments:
- Hello
- World
- Greetings
- Hi

For the Break functionality, lets say that you want to stop the iteration if the current comment is "World".
Here is the template for that:
#{for comments}
Current Comment: ${Comments[n].Body}
#{if (%{'${Comments[n].Body}'.equals('World')})}
#{break}
#{end}
Current Comment Author: ${Comments[n].Author}
#{end}
In this case, Xporter for Jira will print the comment "Hello" and it´s author. Next it will print the comment
Body "World" but since the Conditional expression is true, it will stop the iteration all together and not
print anything else.
Note: Anything after the #{break} mapping will not be printed in the exported document.

For the Continue functionality, lets say that you want to skip to the next iteration if the current comment is
"World", bypassing the Author mapping for this iteration. Here is the template for that:
#{for comments}
Current Comment: ${Comments[n].Body}
#{if (%{'${Comments[n].Body}'.equals('World')})}
#{continue}
#{end}
Current Comment Author: ${Comments[n].Author}
#{end}
In this case, Xporter for Jira will print the comment "Hello" and it´s author. Next, it will print the comment
Body "World" but since the Conditional expression is true, it will continue to the next iteration, not printing
the Author of the "World" comment.
```

Sorting iterations

Imagine that you have an iteration and want to sort it by any field that it can export normally. This will be the header for such an iteration:

```
#{for comments|sortby=<Iteration mapping>}
```

NOTE: The mapping after the "sortby" must be equal to the supported mappings for each Iteration.

Example:

Expand to see the sample code

```
This iteration will be sorted by the Body of all the comments in the issue.

#{for comments|sortby=Body}
${Comments[n].Author}
${Comments[n].Body}
#{end}
```

Sort By Bulk export

The sortby can also be used to sort a &{for issues} iteration on a Bulk Export.

Expand to see the sample code

```
&{for issues|sortby=IssueTypeName}
${Key} - ${IssueTypeName}
&{end}
```

Sorting Criteria

asc and **desc** can be defined in order to define how do you want to sort your data. The default value is **asc**.

WIKI indicates that the field supports wiki format. More about here.