Helper Functions

Fields with Wiki Markup

If the fields to be exported are configured to use a Wiki Markup Renderer, you can place the wiki rendering in the result document.

Definition:

```
${wiki:<Field name>}
```

Expand to see the examples on sample code

```
    ${wiki:Description}
${wiki:Custom Free Text}
```

When creating an Excel template document, you should use: **${wiki:Description}**

Inserting Hyperlinks

To create hyperlinks inside of the document you just need to use the following definition.

Definition:

```
@{title=<Link Title>|href=<Link Address>}
```

Generate a link named with issue key, linking to the issue address on the Jira Server

```
    Building the hypoerling using Jira issue data.
@{title=${Key}|href=${BaseURL}/browse/${Key}}

Building the hyperlink using an external URL
@{title=${Key}|href=http://www.google.com}
```

When creating an Excel template document, you should use: **${link:title=${Key},href=${BaseURL}/browse/${Key}}**

Fields with HTML

If the fields to be exported are configured to use an HTML renderer, you can place the HTML rendering in the result document.

Definition:

```
${html:<Field name>}
```

Expand to see the examples on sample code

```
    ${html:Description}
${html:Custom Free Text}
```

JavaScript with Wiki or HTML

Xporter allows applying Wiki or HTML functions in the text rendered based on JavaScript. For example, if the fields to be exported are configured to use the **Default Text Renderer** provided by Jira, you can use JavaScript to format the field using the notation explained here and then apply the Wiki or HTML functions. You can also use JavaScript to format fields that are configured to use the Wiki Markup Renderer or an HTML renderer and then apply the Wiki or HTML functions. To achieve that, the JavaScript function should be defined as a field in the template and the function to be applied to that field.

Definition:

```
${wiki:%{<Javascript>}}

or

${html:%{<Javascript>}}
```

Formatting the first word in field "Description" to "Bold", "Italic" or to have a different text color (red in the examples) using HTML function

```
    ${html:%{var t ='<b>'+'${Description}'.split(' ')[0] +'</b>'+'${Description}'.substr('${Description}'.
indexOf(' ') + 1);t.toString();}}
${html:%{var t ='<i>'+'${Description}'.split(' ')[0] +'</i>'+'${Description}'.substr('${Description}'.indexOf('
') + 1);t.toString();}}
${html:%{var t ='<b>'+'<font color="Red">'+'${Description}'.split(' ')[0] +'</b>'+'</font>'+'${Description}'.
substr('${Description}'.indexOf(' ') + 1);t.toString();}}
```

Formatting the first word in field "Description" to "Bold", "Italic" or to have a different text color (red in the examples) using Wiki function

```
    ${wiki:%{var t ='*'+'${Description}'.split(' ')[0] +'*'+' ${Description}'.substr('${Description}'.indexOf('
') + 1);t.toString();}}
${wiki:%{var t ='_'+'${Description}'.split(' ')[0] +'_'+' ${Description}'.substr('${Description}'.indexOf(' ')
+ 1);t.toString();}}
${wiki:%{var t ='{color:red}'+'${Description}'.split(' ')[0] +'{color}'+' ${Description}'.substr
('${Description}'.indexOf(' ') + 1);t.toString();}}
```

Formatting Date Fields

If you are exporting a date field such as **CreatedDate** and **LastUpdateDate**, you can define how the date is displayed. All the patterns supported by the DateTimeFormatter Java API are supported.

Definition:

```
${dateformat("<Format>"):<Field name>}
```

Expand to see the examples on sample code

```
    ${dateformat("yyyy-MM-dd"):Created}
${dateformat("EEE, MMM d, ''yy"):Created}
${dateformat("EEE, d MMM yyyy HH:mm:ss Z"):Updated}
```

Formatting Number Field

Available in Xporter for Jira 3.3.0 and later.

If you are exporting a Number Field, you can define how the number is displayed. All the patterns supported by the DecimalFormat(String pattern) Java API are supported.

You can also specify the Locale you want to use. All the Locales in JRE 7 Supported Locales page are supported.

Definition:

```
${numberformat( "<pattern>" ):<Field name>}

${numberformat( "<Locale>" , "<pattern>" ):<Field name>}
```

Expand to see the examples on sample code (considering Price as a custom Number Field)

```
    ${numberformat("###,###.###"):Price}
${numberformat("\u20AC ###,###.###"):Price}
${numberformat("de_DE" , "###,###.###"):Price}
${numberformat("##0.#####E0"):Price}
${numberformat("en_UK" , "#0.#####%"):Price}
```

Formatting User Picker Fields

In Xporter for Jira 4 and later, **displayname** and **displayemail** function were replaced by **fullname** and **emailaddress**, respectively.

If you are exporting a User Picker or Multiple Users custom field, you can output the Display Name using the **fullname** function.

Definition:

```
${fullname:<Field name>}
```

Expand to see the example on sample code

```
    ${fullname:UserCustomField}
```

If you are exporting a User Picker or Multiple Users custom field, you can output the email using the **emailaddress** function.

Definition:

```
${emailaddress:<Field name>}
```

Expand to see the example on sample code

```
    ${emailaddress:UserCustomField}
```

Escaping fields

This function allows fields with line breaks or other special characters to be used inside JavaScript.

Definition:

```
${escape:<Field name>}
```

Expand to see the examples on sample code

```
    ${escape:Description}
${escape:Custom Field}

%{('${escape:Description}'.length > 0) ? 'This issue has description': 'This issue does not have description'
```

Creating mappings based on JavaScript using formatted fields

There are times when it may be needed to format a field, execute JavaScript actions on the formatted field, and export the result as HTML or WiKi.
In these cases, all the above formatters are available.

Expand to see the example on sample code

```
    ${html:%{'${escape:Description}'.replace(new RegExp('toReplace','g'),'replacer')}}
```

Formatting Duration time field

If you are exporting a duration time field, you can define the behavior you want in order to get the correspondent value. Let's suppose you want to export the duration of an issue, you can get this value in milliseconds as well as the formatted version of this value.

Furthermore, this field allows you to execute JavaScript action on the duration time field and export the result.

Definition:

```
${durationformat:<Field name>}
```

Expand to see the example on sample code

```
    ${durationformat:Custom Field}

${durationformat:%{var timeValue="<Value>"; timeValue;}}
```

Set

This function allows you to create new variables that will be treated as fields (mappings), and then use it to change the values of those variables.
The new value passed to the Set function can be plain text, an existing field (mapping) or a Helper Functions field.

Definition:

```
${set(<variable name>,<new value>)}
```

Expand to see the examples on sample code

```
    Define a variable "count" with value "0" and then use it as a normal field.

${set(count,0)}
${count}

Define a variable "newKey" with the value of field ${Key} and then use it as a normal field.

${set(newKey,${Key})}
${newKey}

Define a variable "TodayDate" with the value of the current date calculated with Javascript and then use it as
a normal field

${set(TodayDate,%{(new Date()).getDate() + "/" + ((new Date()).getMonth()+1) + "/" + (new Date()).
getFullYear()})}
${TodayDate}
```

Note

The function Set can be also used to change the value of existing fields (i.e., existing mappings), but it has a limitation: all occurrences of that field in the document will be replaced with the value of the latest Set that is applied.

Note

The values of all the variables (mappings) created with the Set function and used on the header/footer of a template will correspond to the last value passed to the Set function.

Prop

This function allows you to define properties that will be used later by Xporter during the document processing.

Usually, integrations require some extra configuration such as security tokens, API keys and others.

Specification

Function Name	prop
First Parameter	Key
Second parameter	Value

Notation

```
    ${prop(Key,Value)}
```

Example

```
    ${prop(my.first.property.key,qwerty12345)}
${prop(service.api.key,84f7970c-5f70-47d9-a4f2-c75151820ba0)}
${prop(service.api.username,userAdmin)}
```

See here the possible props you can define in your document.

Image Loader

This function basically loads an image from a URL and puts that on the document. You can also specify the width and/or height of the images.

Definition:

```
!{<URL>}
```

Below are two examples:

blocked URL

or

To use the code below, check how to iterate here.

blocked URL

After exporting the document, the generated file is going to see this:

blocked URL

QRCode

This function allows you to create a QRCode image on the template. It accepts as content other Xporter mappings.

Definition:

```
${qrcode('<content>',(Optional) <width>, (Optional) <type>)}
```

Expand to see the examples on sample code

```
    Draws a QR Code image with the content Hello World.
${qrcode(‘Hello World’)}


Draws a QR Code image with the Issue Key as content.
${qrcode(‘${Key}’)}


Draws a QR Code image with a Javascript result as content.
${qrcode(‘%{‘${Key}’.substring(0,2)}’)}

Draws a QR Code image with the content noreply@xpand-it.com and the QR Code type email.
${qrcode(‘noreply@xpand-it.com’,email)}


Draws a QR Code image with the content Hello World and width size 200px.
${qrcode(‘Hello World’,200)}


Draws a QR Code image with the content Hello World, width size 200px and the QR Code type email.
${qrcode(‘hugo.freixo@xpand-it.com’,200,email)}
```

QR Code Content

You can use any mapping on the content of the QR Code function, **but you must be careful because Markups won't generate the expected result.**

If you try to export ${qrcode(‘${wiki:Description}’)} or ${qrcode(‘${html:Description}’)}, the content of the QR Code won't be correct.

You can apply format mapping such as dateformat, numberformat, fullname or emailaddress.

QR Code Types

You can learn more about QR Code types here. Note that these types may depend on the QR Code reader and the obtained result may be different than expected.

- **Text** - Plain text. It's the same as not having a type defined.
- **URL** - The content will be read as a URL link.
- **Phone** - The content will be read as a phone number.
- **Geolocation** - The content will be read as GPS coordinates. Content example: "41.714316, -8.811993".
- **Email** - The content will be read as an email.
- **Wifi** - The content will be read as a Wifi connection. Content example: T:WPA;S:NetworkName;P:NetworkPassword;
- **VCard** - The content as a VCard. Please note that not all devices are prepared to understand VCard details. More information here.

QR Code Width

The width of the QR Code must be a value between 0 and 409 (pixels).
If the user introduces a value outside this interval, the width will be set to its default value (177px).

The user must choose a width large enough for the amount of content in the QR Code. Usually, 100px - 200px is enough.

Inserting the number of issues returned by a JQL Query

Xporter for Jira allows you to insert in the document the number of issues returned by the execution of a JQL query.

Definition:

```
${jqlcount:<JQL Query>}
```

Expand to see the example on sample code

```
${jqlcount:Project = DEMO}
```

The example above will put the number of issues of the project DEMO in the document.

FilterJQL & FilterName

This only works when the export is made via the **Export** menu on the Issue Navigator screen.

Xporter for Jira also allows you to render the FilterJQL and the FilterName used to search Issues on the Issue Navigator screen.

Definition:

```
${FilterJQL}

${FilterName}
```

Filter

You can now work with the filters you have defined on your Jira instance.
If you want to get the JQL statement, simply map:

```
${filter:<Filter name> or <Filter ID>}
```

Furthermore, if you want to export the jqlCount, you don't need to put the entire JQL Query, you just need use the filter name or the filter ID to see all the entries you have on this filter:

```
${jqlcount:${filter:<Filter name> or <Filter ID>}}
```

Last but not least, you can iterate issues with the JQL associated to a filter within the clause parameter by passing only the filter name or filter ID:

Expand to see the sample code

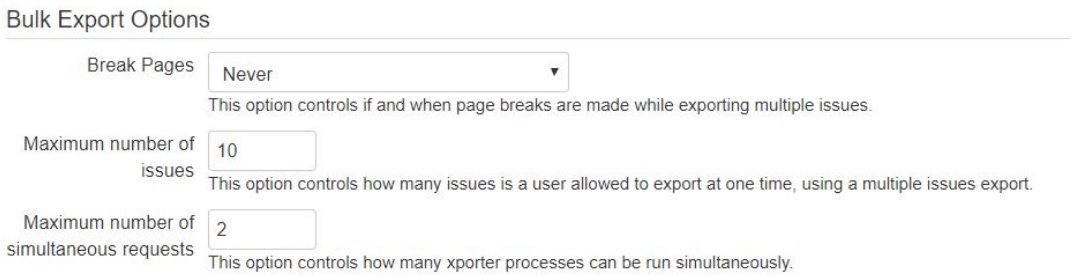
```
    A simple example iterating the details of issues from a filter already defined on your Jira

#{for i=JQLIssuesCount|clause=${filter:<Filter Name> or <FilterId>}}
   ${JQLIssues[i].Key}
   ${JQLIssues[i].Summary}
#{end}
```

JQL limit of issues

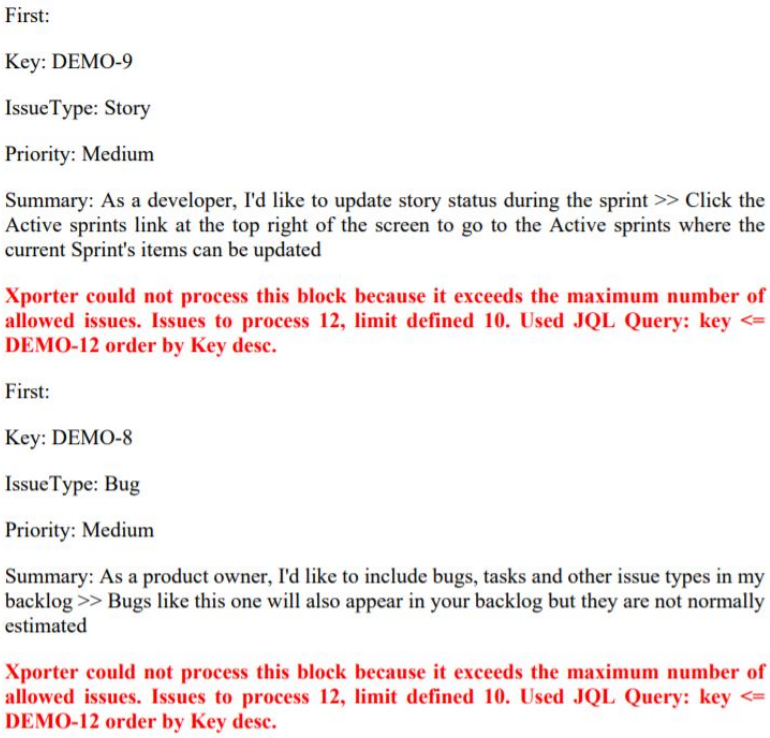
Once the user defines a maximum number of issues on Global Settings, JQL issues processing will replace the iteration that exceeds that limit with a warning message.

This improvement will avoid the JIRA performance decreasing in cases which the number of issues to be export is a large number. This validation will be done in all kinds of exports provided by Xporter.

The maximum number of issues is 10:



The first filter returns fewer issues than the defined limit and the second returns more than 10 issues:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	First:																		
2	Key: DEMO-9																		
3	IssueType: Story																		
4	Priority: Medium																		
5	Summary: As a developer, I'd like to update story status during the sprint >> Click the Active sprints link at the top right of the screen to go to the Active sprints where the current Sprint's items can be updated																		
6	**Xporter could not process this block because it exceeds the maximum number of allowed issues. Issues to process 12, limit defined 10. Used JQL Query: key <= DEMO-12 order by Key desc.**																		
7	First:																		
8	Key: DEMO-8																		
9	IssueType: Bug																		
10	Priority: Medium																		
11	Summary: As a product owner, I'd like to include bugs, tasks and other issue types in my backlog >> Bugs like this one will also appear in your backlog but they are not normally estimated																		
12	**Xporter could not process this block because it exceeds the maximum number of allowed issues. Issues to process 12, limit defined 10. Used JQL Query: key <= DEMO-12 order by Key desc.**																		
13	First:																		
14	Key: DEMO-7																		
15	IssueType: Sub-task																		
16	Priority: Medium																		
17	Summary: This is a sample task. Tasks are used to break down the steps to implement a user story																		
18	**Xporter could not process this block because it exceeds the maximum number of allowed issues. Issues to process 12, limit defined 10. Used JQL Query: key <= DEMO-12 order by Key desc.**																		