# Page Numbering

Page Numbers and Page Counts can be inserted into the report output using the special variable substitutions *${#} (current page number)* and *${##} (total page count)* respectively.

*Note that you cannot use these substitutions in scripts because the values are only substituted at the end of rendering. It is not possible to determine these values during rendering as a) the report might be embedded as a subreport, b) there may be multiple sections c) the total page count cannot be known until all pages have been rendered. To perform operations based on the current page, consider using **OnLayout** instead.

## Modifying Page Numbering Behavior

Not happy with *${#}* and *${##}*? Then do it yourself with scripts...

To know the total page count you need to wait until all pages have been rendered.

You then need to go back and insert the values by building a list of callback functions and then executing it at the end of rendering.

```
Report:On Render Begin

=Code Snippet=

pageCount = 0;

pageFooters = new java.util.ArrayList();



Field():On Render End // this is the page count literal field

=Code Snippet=

function fn(t,pageNo,pageTotal)

{

  t.setText("Page "+pageNo+" of "+pageTotal);

}

fn.target = result.getLogicalElement(0);
```

```
pageFooters.add(fn);

++pageCount;



Report:On Render End

=Code Snippet=

i = 1;

it = pageFooters.iterator();

while (it.hasNext())

{

  fn = it.next();

  fn(fn.target,i,pageCount);

  ++i;

}
```

At the beginning (**Report onRenderBegin**) we initialize the pageCount to 0 and create an empty list for holding the functions. Each time we hit the field (in the page footer in this case) that should show the page info, we define a function to do the job and store it in the list. We then increment the pageCount (this would be in a different script if the field were not in the page footer).

Finally, when we get to the end (**Report onRenderEnd**), we iterate through the list, getting each function, and calling it with the appropriate parameters. Of course, the text fields must be wide enough to hold the text (We suggest putting a literal like "Page 999 of 999" as the default value, because if you leave it empty there is nothing to render and you will not get a text element in the result to write over) because the wrapping and pagination has already been done before this substitution.

Note that **fn.target** is not a special name, you can add attributes on the fly to any JavaScript object, including functions.

Now you can do any kind of formatting of the numbers because the whole thing is in JavaScript.

Of course you can generalize this mechanism to any sort of post-processing - "reverse hide duplicates" which hides all but the last value, putting page totals in page headers, and many others.

We are naming this concept "temporal scripts" - scripts whose execution can be delayed to arbitrary points - eg. OnGroupEnd, OnPageEnd, OnSectionEnd, OnReportEnd.

## Applying Locale Formatting

1. Report On Render Begin:

```
pageCount=1
```

2. Page Footer On Render End:

```
++pageCount
```

3. Field in Page Footer:

```
"Page "+ Format.formatNumber(pageCount,0,"th_TH_TH")
```