Smarten Working with Reports

Statement of Confidentiality, Disclaimer and Copyright

This document contains information that is proprietary and confidential to EMTPL, which shall not be disclosed, transmitted, or duplicated, used in whole or in part for any purpose other than its intended purpose. Any use or disclosure in whole or in part of this information without the express written permission of EMTPL is prohibited.

Any other company and product names mentioned are used for identification purpose only, may be trademarks of their respective owners and are duly acknowledged.

Disclaimer

This document is intended to support administrators, technology managers or developers using and implementing Smarten. The business needs of each organization will vary and this document is expected to provide guidelines and not rules for making any decisions related to Smarten. The overall performance of Smarten depends on many factors, including but not limited to hardware configuration and network throughput.
Contents

1 Introduction .................................................................................................................................. 4

2 Report Creation Workflow ........................................................................................................ 5

3 Creating Report Template ......................................................................................................... 6
   3.1 FreeMarker Template Engine Overview ............................................................................. 6
      3.1.1 FreeMarker Basic Directives ...................................................................................... 6
      3.1.2 Defining and using Variables .................................................................................... 7
      3.1.3 Number Formatting .................................................................................................. 8
      3.1.4 Date Formatting ....................................................................................................... 8
   3.2 Integrating Smarten Object Data into Template ............................................................... 9
      3.2.1 Get Smarten Object Data—Full Data ....................................................................... 9
      3.2.2 Get Smarten Object Data—Based on Row Number ................................................... 10
      3.2.3 Get Smarten Object Data—Based on Row Number and Column Number .............. 11
   3.3 Rendering Different Controls ............................................................................................. 11
      3.3.1 Showing Data in Table ............................................................................................. 11
      3.3.2 Showing Data in Graph ............................................................................................ 12
   3.4 Defining Report Page Formatting ....................................................................................... 14
      3.4.1 Defining Page Break ................................................................................................ 14
      3.4.2 Defining Page Header/Footer .................................................................................. 17
      3.4.3 Defining Page Numbers ........................................................................................... 21

4 Creating Report ........................................................................................................................ 25
   4.1 Upload Report Template ..................................................................................................... 25
   4.2 Associate BI Objects ......................................................................................................... 26
   4.3 Preview ............................................................................................................................. 27
   4.4 Define Page Filters and Retrieval Parameters ................................................................. 28
   4.5 Assign Permissions .......................................................................................................... 29
   4.6 Define Export Parameters ............................................................................................... 29

5 Product and Support Information ............................................................................................. 31
1 Introduction

Smarten provides a reports module for creating custom reports with a complex layout or pixel-perfect reports. Users can create such reports as a credit card statement, invoice, purchase orders, or an annual report. Reports can be exported or published in PDF format through delivery and publishing agent built in Smarten.

The reports framework uses the FreeMarker template engine to define a report template.

Below are the basic steps to create reports:

- Create FreeMarker template file
- Upload template file
- Associate Smarten objects and data to the report template
- Preview report
- Define export parameters
2 Report Creation Workflow

Start

Create static HTML Template for

Create FreeMarker Template from static HTML Template

Upload FreeMarker Template

Associate Smarten Objects

Y

Master-Detail

N

Select Master Object and Column(s)

Map Master Column(s) with column(s) of other objects

Preview

Select Retrieval

Select Page Filter

Report Access

Export Properties

Save Report

End
3 Creating Report Template

3.1 FreeMarker Template Engine Overview

Apache FreeMarker is a template engine, which is a Java library to generate text output, e.g., HTML web pages, e-mails, configuration files, and source code, based on templates and changing data. Templates are written in the FreeMarker Template Language (FTL), which is a simple specialized language. Usually, a general-purpose programming language, such as Java, .NET, and PHP, is used to prepare the data. Then, Apache FreeMarker displays that prepared data using templates. In the template, you are focusing on how to present the data, and outside the template, you are focusing on what data to present.

3.1.1 FreeMarker Basic Directives

The table below describes FreeMarker basic directives with examples.

<table>
<thead>
<tr>
<th>Directive</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
</table>
| if condition| You can use if, elseif, and else directives to conditionally skip a section of the template. The condition(s) must evaluate to a boolean value or else an error will abort template processing. The elseif and else must occur inside if (that is, between the if start-tag and end-tag). The if can contain any number of elseif (including 0) and at the end optionally one else. | <#if condition>
  <#else>
  </#if> |
| counter     | Returns the 1-based index where the iteration (which is identified by the loop variable name) currently stands. | <#list ["a," "b," "c"] as i>
  ${i?counter}: ${i}
</#list> |
| index       | Returns the 0-based index where the iteration (which is identified by the loop variable name) currently stands. | <#list ["a," "b," "c"] as i>
  ${i?index}: ${i}
</#list> |
| is_first    | Tells if the item where the iteration (which is identified by the loop variable name) currently stands is the first item. | <#list ["a," "b," "c"] as i>
  ${i?is_first?c}
</#list> |
| is_last     | Tells if the item where the iteration (which is identified by the loop variable name) currently stands is the last item. | <#list ["a," "b," "c"] as i>
  ${i?is_last?c}
</#list> |
| has_next    | Tells if the item where the iteration (which is identified by the loop variable name) currently stands is not the last item. | <#list ["a," "b," "c"] as i>
  ${i?has_next?c}
</#list> |
| is_even_item| Tells if the item where the iteration (which is identified by the loop variable name) currently stands is an even item. | <#list ["a," "b," "c"] as i>
is identified by the loop variable name) currently stands has an even 1-based index.

\[
\begin{array}{|c|c|}
\hline
\text{is\_odd\_item} & \text{Tells if the item where the iteration (which is identified by the loop variable name) currently stands has an odd 1-based index.} \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
\text{c} & \text{This built-in must be used to print the number.} \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
\text{true} & \text{boolean value “true”} \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
\text{false} & \text{boolean value “false”} \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
\text{gt} & \text{comparison operator “greater than”} \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
\text{gte} & \text{comparison operator “greater than or equivalent”} \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
\text{lt} & \text{comparison operator “less than”} \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
\text{lte} & \text{comparison operator “less than or equivalent”} \\
\hline
\end{array}
\]

### 3.1.2 Defining and using Variables

Two types of variables are defined in a template:

- **Plain variables**: They are accessible from everywhere in the template. You can create and replace these variables with the `assign` directive.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>assign</td>
<td>You can create a new variable or replace an existing variable.</td>
<td>&lt;#assign FLAG=&quot;TRUE&quot;&gt; &lt;#assign Pi=&quot;3.14&quot;&gt;</td>
</tr>
</tbody>
</table>

- **Loop variables**: Loop variables are created automatically by directives, such as `list` (e.g., `x in <#list xs as x>...<#/list>`), and they only exist between the start-tag and end-tag of the directive.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
<td>The simplest form for listing a collection is</td>
<td>&lt;#list users as user&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>${user}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;#/list&gt;</td>
</tr>
</tbody>
</table>
### 3.1.3 Number Formatting

These are the built-in number formatting variables that can be used to format numbers:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>abs</td>
<td>Gives the absolute value of a number.</td>
<td>x?abs, if x is -5, will evaluate to 5</td>
</tr>
<tr>
<td>round</td>
<td>Rounds to the nearest whole number. If the number ends with .5, then it rounds upward, i.e., toward positive infinity.</td>
<td>x?round, if x is 1.5, will evaluate to 2</td>
</tr>
<tr>
<td>floor</td>
<td>Rounds the number downward, i.e., toward negative infinity.</td>
<td>x?floor, if x is 1.5, will evaluate to 1</td>
</tr>
<tr>
<td>ceiling</td>
<td>Rounds the number upward, i.e., toward positive infinity.</td>
<td>x?ceiling, if x is 1.5, will evaluate to 2</td>
</tr>
<tr>
<td>string</td>
<td>Converts a number to a string.</td>
<td>&lt;#assign x = 1.234&gt; ${x?string}</td>
</tr>
<tr>
<td>?string(“#,##0.00”)</td>
<td>Displaying two digits after decimal point.</td>
<td>&lt;#assign total = 3343434&gt; &lt;#assign number = -3343434&gt; ${total?string(“#,##0.00”)} ${number?string(“#,##0.00”)})</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Output: 3,343,434.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-3,343,434.00</td>
</tr>
<tr>
<td>?string(“#,0”)</td>
<td>Comma separator</td>
<td>&lt;#assign total = 123456789&gt; ${total?string(“#,0“})</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Output: 1,234,567,89</td>
</tr>
<tr>
<td>?string(“#,0”)</td>
<td>Comma separator</td>
<td>&lt;#assign total = 123456789&gt; ${total?string(“#,0“})</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Output: 123,456,789</td>
</tr>
</tbody>
</table>

### 3.1.4 Date Formatting

These are the built-in date formatting variables that can be used to format date values:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>.now</td>
<td>Returns the current date-time</td>
<td>${.now}</td>
</tr>
</tbody>
</table>
For more details, you can refer to FreeMarker template framework documentation from:

https://freemarker.apache.org/docs/index.html

3.2 Integrating Smarten Object Data into Template

This section explains how to integrate Smarten object data into the FreeMarker template.

3.2.1 Get Smarten Object Data—Full Data

This section shows how to use Smarten object data in the FreeMarker template. We have used “Yearly Sales” tabular report shown below in this example.

<table>
<thead>
<tr>
<th>Year</th>
<th>GrossSales</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015</td>
<td>39407504.10</td>
</tr>
<tr>
<td>2016</td>
<td>56141853.47</td>
</tr>
<tr>
<td>2017</td>
<td>59748470.08</td>
</tr>
<tr>
<td>2018</td>
<td>63890412.68</td>
</tr>
</tbody>
</table>

TABULAR REPORT
The snippet code below iterates object data above and displays it.

```freemarker
<#list r170611e29cf as innerList>
<#list innerList as obj>
${obj}
</list>
<br>
</list>
```

In the example above,

r170611e29cf—This is the object id of the Smarten object that we want to integrate into the template. The user can get this object ID from the Smarten administration panel >> Repository section. They must use object ID without extension in template script. In the example above, the actual object ID of the tabular report is “r170611e29cf.rpt.” In FreeMarker script, the user should use “r170611e29cf” (without extension).

innerList—This is the user-defined list variable that is used to iterate loop in the template. This variable contains each row of Smarten object data passed to the template.

Output

```
<table>
<thead>
<tr>
<th>Year</th>
<th>GrossSales</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015</td>
<td>39497594.10</td>
</tr>
<tr>
<td>2016</td>
<td>56141853.47</td>
</tr>
<tr>
<td>2017</td>
<td>59748479.98</td>
</tr>
<tr>
<td>2018</td>
<td>63890412.68</td>
</tr>
</tbody>
</table>
```

### 3.2.2 Get Smarten Object Data—Based on Row Number

This section shows how to get data of a specific row from Smarten object data. We have used the same object as in the example above.

The example below shows how to read second row (Year 2015 row) data from object data.

```freemarker
<#list r170611e29cf as innerList>
<#if innerList?counter == 2><#-- Displaying content of second row only -->
<#list innerList as obj>
${obj}
</list>
</#if>
<br>
</#list>
```
3.2.3 Get Smarten object Data—Based on Row Number and Column Number

This section shows how to get data of a specific row and column cell from Smarten object data. We have used the same object as in the example above.

The example below shows how to read fifth row and second column (Year 2018 row and Grosssales) data from object data.

```html
<html>
<body>
<table>
<tbody>
<#list r170611e29cf as innerList>
<#list innerList as obj>
<if innerList?counter == 5><!-- Displaying content of fifth row -->
    <#if obj?counter == 2><!-- Displaying content of second column -->
        ${obj}
    </if>
</if>
</if>
</tbody>
</table>
</body>
</html>
```

Output

63890412.68

3.3 Rendering Different Controls

This section shows how to render different controls in the FreeMarker template.

3.3.1 Showing Data in Table

This section shows how to render HTML table control using Smarten object data. We have used the same object as in the example above.

Below is the example script for showing data in table:

```html
<html>
<body>
<table>
<tbody>
<#list r170611e29cf as innerList>
<#list innerList as obj>
<if innerList?counter == 5><!-- Displaying content of fifth row -->
    <#if obj?counter == 2><!-- Displaying content of second column -->
        ${obj}
    </if>
</if>
</if>
</tbody>
</table>
</body>
</html>
```
3.3.2 Showing Data in Graph

This section shows how to render graph control using Smarten object data. We have used the same object as in the example above.

Below is the example script for showing data in a bar graph:

```html
<html>
<head>
<script type="text/javascript" src="../js/am/amcharts.js"></script>
<script type="text/javascript" src="../js/am/serial.js"></script>
<script>
    var chart = AmCharts.makeChart("chartdiv", {
        "type": "serial",
        "theme": "none",
        "rotate": true,
        "dataProvider": [
            <#list a175ffac5d97 as outerList>
                <#if outerList?is_last>
                    {
                        <#list outerList as obj>
                            <#if obj?is_first>
                                "Team": "${obj}"
                            </#if>
                            <#if obj?is_last>
                                "GrossSales": "${obj}"
                            </#if>
                        </#list>
                    }<#if outerList?is_last>
                    ...
                </#if>
            </#list>
        ]
    });
</script>
</head>
</html>
```

Output

<table>
<thead>
<tr>
<th>Year</th>
<th>GrossSales</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015</td>
<td>39497594.10</td>
</tr>
<tr>
<td>2016</td>
<td>56141853.47</td>
</tr>
<tr>
<td>2017</td>
<td>59748479.98</td>
</tr>
<tr>
<td>2018</td>
<td>63890412.68</td>
</tr>
</tbody>
</table>
"value": ${obj}
</#if>
</#list>

</#else>
{
<#list outerList as obj>

<#if obj?is_first>

"Team": "${obj}'",

</#if>

<#if obj?is_last>

"value": ${obj}
</#if>
</#list>

</#if>
</#list>

,"startDuration": 0,
"graphs": [{$
"labelText": "[[value]]",
"balloonText": "<b>[[category]]: [[value]]</b>'",
"fillColorsField": "color",
"fillAlphas": 0.7,
"lineAlpha": 0.9,
"lineColor": "#0D52D1",
"type": "column",
"valueField": "value"
}],
"categoryField": "Team",
"borderAlpha": 0,
"categoryAxis": {
"gridPosition": "start",
"gridAlpha": 0,
"labelRotation": 45
},
"valueAxes": [{$
"gridAlpha": 0,
"title": ""
}
]};
3.4 Defining Report Page Formatting

This section explains how to define page formatting, such as page break, page header/footer, and page numbers.

3.4.1 Defining Page Break

This section shows an example for how to define page break in a report template.

The “page-break-before” property in CSS is used to achieve page break in PDF. This property helps to define page break in an HTML document when it is printed or outputted in PDF format. In the example below, we present a scenario of showing 48 records per page. After every 48 records, the next record will be printed on a new page. A page break code snippet is highlighted in bold in the template code below.

Example:

```html
<html>
<head>
  <title>Ledger Account Demo</title>
  <style>
    table { page-break-inside:auto }
    tr { page-break-inside:avoid; page-break-after:auto }
    thead { display:table-header-group }
  </style>
</head>
</html>
```
<table>
<thead>
<tr>
<th>Column 1</th>
<th>Column 2</th>
<th>Column 3</th>
<th>Column 4</th>
<th>Column 5</th>
<th>Column 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column 1</td>
<td>Column 2</td>
<td>Column 3</td>
<td>Column 4</td>
<td>Column 5</td>
<td>Column 6</td>
</tr>
</tbody>
</table>

```
<tr><td align="center">$innerList[0] </td>
<td align="left">Cr</td>
<td><strong>$innerList[2] </strong></td>
<td align="left"><strong>$innerList[3] </strong></td>
<td align="center">$innerList[5] </td>
<td align="right" style="padding-right:55px">$innerList[7] &nbsp;</td>
<td align="left"></td>
</tr>
```

```html
<#assign indexCounter = indexCounter + 1>
<#if indexCounter == counter>
<#assign counter = counter + 48>
<tr><td align="center">$innerList[0] </td>
<td align="left">Cr</td>
<td><strong>$innerList[2] </strong></td>
<td align="left"><strong>$innerList[3] </strong></td>
<td align="center">$innerList[5] </td>
<td align="right" style="padding-right:55px">$innerList[7] &nbsp;</td>
<td align="left"></td>
</tr>
```

© 2021, Smarten
Visit us at www.smarten.com
<table>
<thead>
<tr>
<th>TimDate</th>
<th>ProductName</th>
<th>Type_Sale</th>
<th>Qty_Defect</th>
<th>Amount_NetBeforeDisc</th>
<th>Amount_Net</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

# Page Break code start -->

</table>
</div>

# Page Break code end -->

<tr>
<td colspan=7>&nbsp;</td>
</tr>

</#list>
</table>
</div>
3.4.2 Defining Page Header/Footer

This section explains how to include a page header and a footer in a report. The example below shows a template code for defining a page header and footer. The code snippet is highlighted in bold.

**Example:**

```html
<html>
<head>
  <title>Ledger Account Demo</title>
  <style>
    table { page-break-inside:auto }
    tr { page-break-inside:avoid; page-break-after:auto }
    thead { display:table-header-group }
  </style>
</head>
<body>
<div style="max-width: 1000px; width: 100%; margin:0px auto; padding: 10px !important;">
  <div style="text-align: center; width: 100%;">
    <table width="100%" border="0" cellspacing="0" cellpadding="0">
      <tbody>
        <tr>
          <td align="center" width="30%" style="font-family:Arial, 'sans-serif'; font-size:12px; line-height: 20px;">
            b>ABC Industries Ltd.</b><br>
            <u>Ahmedabad</u>
          </td>
        </tr>
      </tbody>
    </table>
  </div>
</div>
</body>
</html>
```
XYZ Enterprises
Ledger Account
Maninagar, Ahmedabad

1-Apr-2019 to 12-Feb-2020

<table>
<thead>
<tr>
<th>Description</th>
<th>Amount</th>
<th>Date</th>
<th>Notes</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Amount</td>
<td>Date</td>
<td>Notes</td>
<td>Reference</td>
</tr>
</tbody>
</table>

Page 1/1
<table width="100%" border="0" cellspacing="0" cellpadding="2" style="font-family: Arial, 'sans-serif'; font-size: 8px;">
<thead style="display: table-row-group;">
<tr>
<th colspan="2" align="center" valign="middle" bgcolor="#FFFFFF" style="border-top: 1px solid; border-bottom: 1px solid;">TimDate</th>
<th align="left" valign="middle" bgcolor="#FFFFFF" style="border-top: 1px solid; border-bottom: 1px solid;">ProductName</th>
<th align="left" valign="middle" bgcolor="#FFFFFF" style="border-top: 1px solid; border-bottom: 1px solid;">Type_Sale</th>
<th align="left" valign="middle" bgcolor="#FFFFFF" style="border-top: 1px solid; border-bottom: 1px solid;">Qty_Defect</th>
<th align="center" valign="middle" bgcolor="#FFFFFF" style="border-top: 1px solid; border-bottom: 1px solid; padding-left:42px;">Amount_NetBeforeDisc</th>
<th align="center" valign="middle" bgcolor="#FFFFFF" style="border-top: 1px solid; border-bottom: 1px solid;">Amount_Net</th>
</tr>
</thead>
</table>
### 3.4.3 Defining Page Numbers

This section shows how to define page numbers in a page header or a footer of a report.

In the example below, we show the page number in the page header of a report. The page number is shown in “Page n / Total pages” format. We need to follow the steps mentioned below to include the page number.

1. Define “pageNum” variable in the FreeMarker template as shown below.

   ```html
   <#assign pageNum = 1>
   ```

2. Then in the template code where you want to display the total number of pages, you need to insert a span tag as shown below.

   ```html
   <span class="spId"></span>
   ```

3. After that please add the JavaScript snippet below in the template code. This script assigns the total number of pages value dynamically to the span tag.

   ```html
   <script type="text/javascript">
   var cols = document.getElementsByClassName('spId');
   for(i = 0; i < cols.length; i++) {
     cols[i].innerText = ${pageNum};
   }
   </script>
   ```

In the template example below, we have highlighted the code snippet explained in the steps above.

**Example:**

```html
<html> <head> <title>Ledger Account Demo</title> <style>
  table { page-break-inside:auto }
  tr { page-break-inside:avoid; page-break-after:auto }
  thead { display:table-header-group }
</style> </head> <body> <div style="max-width: 1000px; width: 100%; margin:0px auto; padding: 10px !important;"> 
  <table width="100%" border="0" cellspacing="0" cellpadding="0">
    <tbody>
      <tr>
        <td align="center" width="30%" style="font-family:Arial, 'sans-serif'; font-size:12px; line-height: 20px;"><b>ABC Industries Ltd.</b><br>
          <u>Ahmedabad</u><br>
            <b> XYZ Enterprises</b>
          </td>
    </tr>
  </tbody>
</table>
</div>
</body>
</html>
```
Ledger Account <br>
Maninagar, Ahmedabad <br>
<p style="font-size:10px">1-Apr-2019 to 12-Feb-2020</p>
<table>
<thead>
<tr>
<th>TimDate</th>
<th>ProductName</th>
<th>Type_Sale</th>
<th>Qty_Defect</th>
<th>Amount_NetBeforeDisc</th>
<th>Amount_Net</th>
</tr>
</thead>
</table>

```javascript
var cols = document.getElementsByClassName('spId');
for(i = 0; i < cols.length; i++) {
    cols[i].innerText = ${pageNum};
}
```
4 Creating Report

Smarten provides a step-by-step wizard for creating a print report. The following steps are involved to create a print report:

1. Upload the FreeMarker report template, which is created as explained in the sections above.
2. Associate BI objects
3. Preview report output
4. Define page filters and retrieval parameters
5. Assign permissions
6. Define export parameters

4.1 Upload Report Template

The user can create a FreeMarker template as explained in the sections above. Once the template is ready, the user can create a new report from the Smarten administrator panel. In the first step, the user needs to upload a created report template.
4.2 Associate BI Objects

Once the report template is uploaded, the user needs to select Smarten BI objects, which are used in the FreeMarker template script to fetch data. Based on the report template, the user can select and associate one or more BI objects to the report. The system fetches data from these objects and generates the report based on the report template.

Note:
Currently, Smarten allows the user to associate Crosstab and Tabular BI objects only.

It is also recommended that associated BI object’s metadata or column sequence should not be changed after creating a print report. This may lead to the wrong reference of object data and column sequence and may generate a wrong print report. It is also recommended to give view only permission for associated BI objects to users to avoid any user modifying the report.

For more than one object, the system provides a mechanism to set a relationship between two objects for a master-detail data kind of scenario. For example, let us take a credit card statement report scenario. For this report, two objects will be required. One is CustomerDetails, and the other is TransactionDetails. The CustomerDetails object will have customer-specific master details, and TransactionDetails will have transactions done by customers. Now, a credit card statement should be generated for each customer, and there should be a separate statement for each customer. For such a scenario, we need to specify a master-detail relationship between the two objects, and the system can fetch data accordingly to generate a report. So here, you should select the CustomerDetails object as the master object and CustID as the master ID column. TransactionDetail will be considered as the detail object, and you need to map its CustID column with the master ID column. So, based on this definition, the system will fetch data of each customer from the master object, and the detail object and will generate the report.
Note:
Currently, Smarten allows mapping of a single key column. If you have a multiple keys column scenario, i.e., combined keys, you should combine key columns into one column in your dataset using SSDP.

For more than one object, if you don’t want to define a master-detail relationship between objects, you just need to select NONE as the master object. So, there will not be any relationship between objects, and the system fetches data accordingly and will generate a report. For example, let’s take a company annual report in which you want to show, for example, a P/L statement, balance sheet, and performance of a key product segment. You can associate required objects to the report. Here, there is no relationship between objects, so you need not specify the master object and mapping column. The system will take individual object data accordingly and generate a report as per template logic.

4.3 Preview
Once object association with template is done, the system will generate an actual report preview using limited data. Users can verify their report output and confirm whether it is proper or not.
4.4 Define Page Filters and Retrieval Parameters

Users can define appropriate page filters or retrieval parameters for filtering the data.
4.5  Assign Permissions

The user can assign permissions to appropriate users. Permitted users will be able to see the report in the object repository along with other BI objects. They can open, filter, and export a report.

4.6  Define Export Parameters

The user can define export parameters, such as PDF page size, orientation, export file naming pattern, and export in a single file or separate files.

For a master object scenario, Smarten provides two options to generate an export file:

**Generate a separate file for each record**—The system will generate a separate file for each master object record. For a credit card statement example, it will generate a separate file for each customer.

**Generate single file for all records**—The system will generate a single file for all records. In a master object scenario, each master record will start from a new page. For a credit card example, statements for all customers will be exported in a single PDF file, and each customer statement will start from a new page.

For a non-master object scenario, only this option will be applicable. For a company annual report example, it will be exported in a single file only.

In case of a large single file, the system will break it into small chunks. For a ledger report example, if there are a large number of transactions and it creates thousands of pages, the system will generate multiple export files by breaking it based on the number of records per file.

**File name pattern predefined tags**

The following tags can be used to create export file name pattern:

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ReportName$</td>
<td>Returns the name of the report</td>
</tr>
<tr>
<td>Tag</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>$DD$</td>
<td>Returns the day part of current system date. If system date is 01-Mar-2021, returns 01</td>
</tr>
<tr>
<td>$MM$</td>
<td>Returns the month part of current system date. If system date is 01-Mar-2021, returns 03</td>
</tr>
<tr>
<td>$MMM$</td>
<td>Returns the short form of month name of current system date. If system date is 01-Mar-2021, returns Mar</td>
</tr>
<tr>
<td>$YY$</td>
<td>Returns the 2-digit year part of current system date. If system date is 03-Mar-2021, returns 21</td>
</tr>
<tr>
<td>$YYYY$</td>
<td>Returns the 4-digit year part of current system date. If system date is 03-Mar-2021, returns 2021</td>
</tr>
<tr>
<td>$HH$</td>
<td>Returns the hour part of system date. If system date is 01-Mar-2021 10:30:50, returns 10</td>
</tr>
<tr>
<td>$mm$</td>
<td>Returns the minutes part of system date. If system date is 01-Mar-2021 10:30:50, returns 30</td>
</tr>
<tr>
<td>$ss$</td>
<td>Returns the seconds part of system date. If system date is 01-Mar-2021 10:30:50, returns 50</td>
</tr>
<tr>
<td>$master-object-column-name$</td>
<td>These tags are available only when a Master object is selected in the report definition. All master objects' dimensions are available as tags. For example, if it is a CustomerDetails object and custID, custName as dimensions, $custID$ and $custName$ will be the tags.</td>
</tr>
</tbody>
</table>

For example,

1. For a credit card statement example, if you want to generate an export file name as shown in the pattern below:

   ABC Bank-Card Statement-0001001-20210304
   Where 0001001 is customer id and 20210304 is statement generation date
   The pattern below should be specified in the report:
   
   ABC Bank-$ReportName$-$CustomerID$-$YYYY$-$MM$-$DD$

2. For an annual report example, if you want to generate an export file name as shown below:

   ABC Pharma-Annual Report-2021
   The pattern will be:
   
   ABC Pharma-$ReportName$-$YYYY$

3. For an outstanding customer report, if you want to generate an export file pattern as shown below:

   XYZ LTD-Outstanding Report-Feb2020
   The pattern will be:
   
   $CustomerName$-$ReportName$-$MM$-$YYYY$
5 Product and Support Information

Find more information about Smarten and its features at www.smarten.com
Support: support@smarten.com
Sales: sales@smarten.com
Feedback & Suggestions: support@smarten.com
Support & Knowledgebase Portal: support.smarten.com