Contents

Overview of the new ProjectFirma Report Templating Engine

ProjectFirma now supports a report templating system. Within the system, users create Microsoft Word (.docx) templates that can be used to generate reports on Projects within a given ProjectFirma tenant's instance. Each user's goals are different and their reporting needs potentially broad; the report templating paradigm was chosen because it is flexible and gives users freedom in how and what they report on within ProjectFirma.

The following are examples of variety desired within a multi-tenant application:

1. Tenant specific text, branding, or colors
2. Reporting on various aspects of projects that are particularly important to a tenant
3. Adding images that *do not* exist in ProjectFirma
4. Formatting that is project-specific, or project stage-specific

Report generation involves writing some simple C# code within a template document, selecting projects to include into the report, and then generating a compiled report with a chosen report template.

How users add report templates and generate reports

SITKA ADMINS can now enable a "Report Center" for tenants within ProjectFirma. If you would like to enable the "Report Center," please reach out to Sitka.

ADMINS can create and manage a *report template* by doing the following:

1. Access the "**REPORT CENTER**" item in the main menu of the application
2. Click on "**MANAGE REPORT TEMPLATES**"
3. Above and to the right of the grid, click on "**+ Create a new Report Template**"
4. Enter a Title for the report (e.g., Project Status Report)
5. Enter an optional description
6. Select a model that the report template will be used for (currently "Project" is the only option)
7. Upload a .docx report template
8. Save

ADMINS will be able to run reports on selected projects by doing the following:

1. Access the "REPORT CENTER" item in the main menu of the application
2. Click on "PROJECTS"
3. (Optional) Filter and sort the grid to select the projects that are intended to be in the final generated report.
    a. **NOTE:** If a previously selected checkbox is hidden with a filter, it will not be included in the generated report.
4. Check the checkboxes at the left of the grid that are intended to be in the generated report
5. Click on "**Generate Reports**" at the top right of the grid
6. Select a "**Selected Report Template**" that will be used as the template for the report
7. Click "Generate"

Introduction to the “templating language”

ProjectFirma *report templates* are formatted Word documents that contain “templating language” – instructions that the ProjectFirma application evaluates, replaces with current data, and renders into a downloadable Word document.

The templating language consists of variables and/or logic that is placed between “delimiters.” The delimiters are:

- <% … %>   for expressions (such as loops, if statements, etc…)
- <%= … %>   for outputting the results of an expression or a variable

The templating language does not display in the final report document. The language within the <%= … %> delimiter is replaced with ProjectFirma data that is formatted in the same style as the templating script.

For example, here is a short *report template*. It includes templating language to access selected projects in ProjectFirma, loop through them and display their name and description.

```
<% var projects = Model.ReportModel; %>

<% foreach(var project in projects) { %>

Project: <%= project.ProjectName %>

Project Description

<%= project.ProjectDescription %>

<% } %>
```

Note: For clarity, report template examples in this User Guide will display in a gray box. In the documents you create, you don’t need to use a gray box – unless you want to!

The following output is the report document generated from the above template if a user had selected two projects to run the report on:

Project: Bird Track Springs

Project Description

The United States Forest Service (USFS) and the Bonneville Power Administration (BPA) are proposing to restore habitat for spring/summer Chinook salmon, Steelhead, and bull trout within a 2-mile reach of the Grande Ronde River (GRR). The Bird Track Springs Fish Enhancement Project (project) would re-establish natural river-floodplain connections and processes. Natural processes within this reach of the GRR include multiple channel networks usually created through forcing mechanisms of large wood, ice, beaver, and rock. These actions are proposed to be implemented on the La Grande Ranger District of the Wallowa-Whitman National Forest (WWNF) and some adjacent private lands.

Project: Buffalo Flats

Project Description

Buffalo Flats Description would go here if it existed in ProjectFirma. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent libero justo, luctus at magna a, lobortis lobortis mi. Etiam ante orci, eleifend convallis luctus at, scelerisque non nisl. Ut ipsum nibh, dignissim sed sollicitudin in, eleifend non lorem. Nullam ut est ullamcorper nunc scelerisque facilisis vitae vel metus.

Note: For clarity, report template output in this User Guide will display in a yellow box. In the documents you create, there will be no yellow box – unless you embed the templating language within one!

The Name and Description of a project are not the only attributes you can display within the document templates. While many of the project attributes are currently provided within the report template, we are looking forward to working with users of these reports to implement more features, models and types of data within the report templating engine. A separate document outlines all the attributes currently available to the Project model.

Basics with examples

Variables

```
At any point in the text you can insert C# statements. Like right <% var i = 1; %>here.
```

The result looks like this:

At any point in the text you can insert C# statements. Like right here.

The Write method

```
If you want to display the value of i, you can use the Write method. Right now, i is <% Write(i); %>.
```

Which will result in:

```
If you want to display the value of i, you can use the Write method. Right now, i is 1.
```

Shorthand Write method

```
If you want to display the value of i with the shorthand method: i is still <%= i %>.
```

Note that the shorthand statement above doesn’t require a semi-colon “;” at the end.

Which will result in:

```
If you want to display the value of i with the shorthand method: i is still 1.
```

Conditional content

```
<% var projects = Model.ReportModel; %>

<% foreach(var project in projects) { %>

<% if (project.ProjectStage == "Implementation") { %>

Project is in the Implementation stage

<% } %>

<% if (project.ProjectStage == "Planning/Design") { %>

Project is in the Planning/Design stage

<% } %>

<% } %>
```

If the above template was used on two projects, the first in the “Planning/Design” stage and the second in the “Implementation” stage it would result in the following:

```
Project is in the Planning/Design stage

Project is in the Implementation stage
```

NOTE: If you type Planning/Design within quotes directly into the document, Word automatically formats the quotes to be left and right quotes: “Planning/Design”. Unfortunately, the template language cannot read these quotes! The quotes must be “Unicode” quotes, i.e. straight like this – "Planning/Design" – to be readable. There are several ways to ensure that your quotes are readable:

- Copy and paste from the samples in this guide or the Template Language column in the Model documentation spreadsheet.
- After typing a quote “ press undo or Ctrl+z once to turn it into a "straight" quote
- Type "Planning/Design" in Notepad, and the copy it into the document
- Use the Insert / Symbol function in Word to insert Unicode quotes (character code: 0022, from: Unicode hex)
- Turn off the auto-formatting in Word that changes "straight" quotes to “curly” quotes.

Loops

You can add repeating text blocks to a report template like this:

```
<% for (i = 0; i < 5; i++) { %>

The value of i is <%= i %>

<% } %>
```

This will result in:

```
The value of i is 0

The value of i is 1

The value of i is 2

The value of i is 3

The value of i is 4
```

Nested Loops

You can also make nested loops within the report templating engine.

Loops, tables and the AppendRow method

Sometimes you want to loop through a list of models, but you don’t want a repeating text block. For example, you want to add Organization rows to a table that lists the Organizations for a ProjectFirma project, but do not want to repeat the table itself. In this case use {! Instead of {.

```
<% var projects = Model.ReportModel; %>

<% foreach(var project in projects) { %>

<% var projectOrganizations = project.GetProjectOrganizations(); %>

<% if(projectOrganizations.Any()) { %>

<% foreach(var projectOrganization in projectOrganizations) {! %>
```

Below is a table of the organizations belonging to the <%= project.ProjectName %> project.

Organization Name	Organization Short Name
<%= projectOrganization.OrganizationName %>	<%= projectOrganization.OrganizationShortName %><% AppendRow(); %>

```
<% } %>

<% } %>

<% } %>
```

Note the <% AppendRow(); %> method at the end of the repeating row.

If the above template was used on one project, that had two organizations belonging to it, it would result in the following:

Below is a table of the organizations belonging to the Bird Track Springs project.

Organization Name	Organization Short Name
Sitka Technology Group	Sitka
U.S. Bureau of Reclamation	USBR

Syntax and spelling:

Accuracy is important in the templating language. While ProjectFirma provides some help and notifications for errors in templates, it cannot detect every use case. Here are some things to keep in mind:

- Every code block must have a clear beginning and end.

  - A code block started with “<%” or “<%=” must have a matching “%>” at the end of it
- Loops and if statements require a clear beginning and end.
  - Every “<% if(true) { %>” requires a closing “<% } %>” at the end of the statement
  - Every “<% foreach(var item in items) { %>” requires a closing “<% } %>” at the end of the loop
- Spaces between the characters defining the delimiters such as “<%” or “<%=” can cause errors when uploading or compiling reports.
  - Accepted:
    - <% var projectStage = project.ProjectStage; %>
  - Not Accepted:
    - < % var projectStage = project.ProjectStage; %>
    - <% = projectStage %>
- When declaring variables or executing a statement, a semi-colon is required at the end.
  - Accepted:
    - <% var projectContacts = project.GetProjectContacts(); %>
  - Not Accepted:
    - <% var projectContacts = project.GetProjectContacts() %>
- As mentioned earlier, when providing string (text) values to a method (function) in the model, the type of quotes used matters.
  - Accepted:
    - <% var projectManagers = project.GetProjectContactsByType("Project Manager"); %>
  - Not Accepted:
    - <% var projectManagers = project.GetProjectContactsByType(“Project Manager”); %>
  - **Word automatically formats the quotes to be left and right quotes: “Project Manager”. Unfortunately, the template language cannot read these quotes! The quotes must be “Unicode” quotes, i.e. straight like this – "Project Manager" – to be readable. There are several ways to ensure that your quotes are readable:**
    - Copy and paste from the samples in this guide or the Template Language column in the Model documentation spreadsheet.
    - After typing a quote “ press undo or Ctrl+z once to turn it into a "straight" quote
    - Type "Project Manager" in Notepad, and the copy it into the document
    - Use the Insert / Symbol function in Word to insert Unicode quotes (character code: 0022, from: Unicode hex)
    - Turn off the auto-formatting in Word that changes "straight" quotes to “curly” quotes.