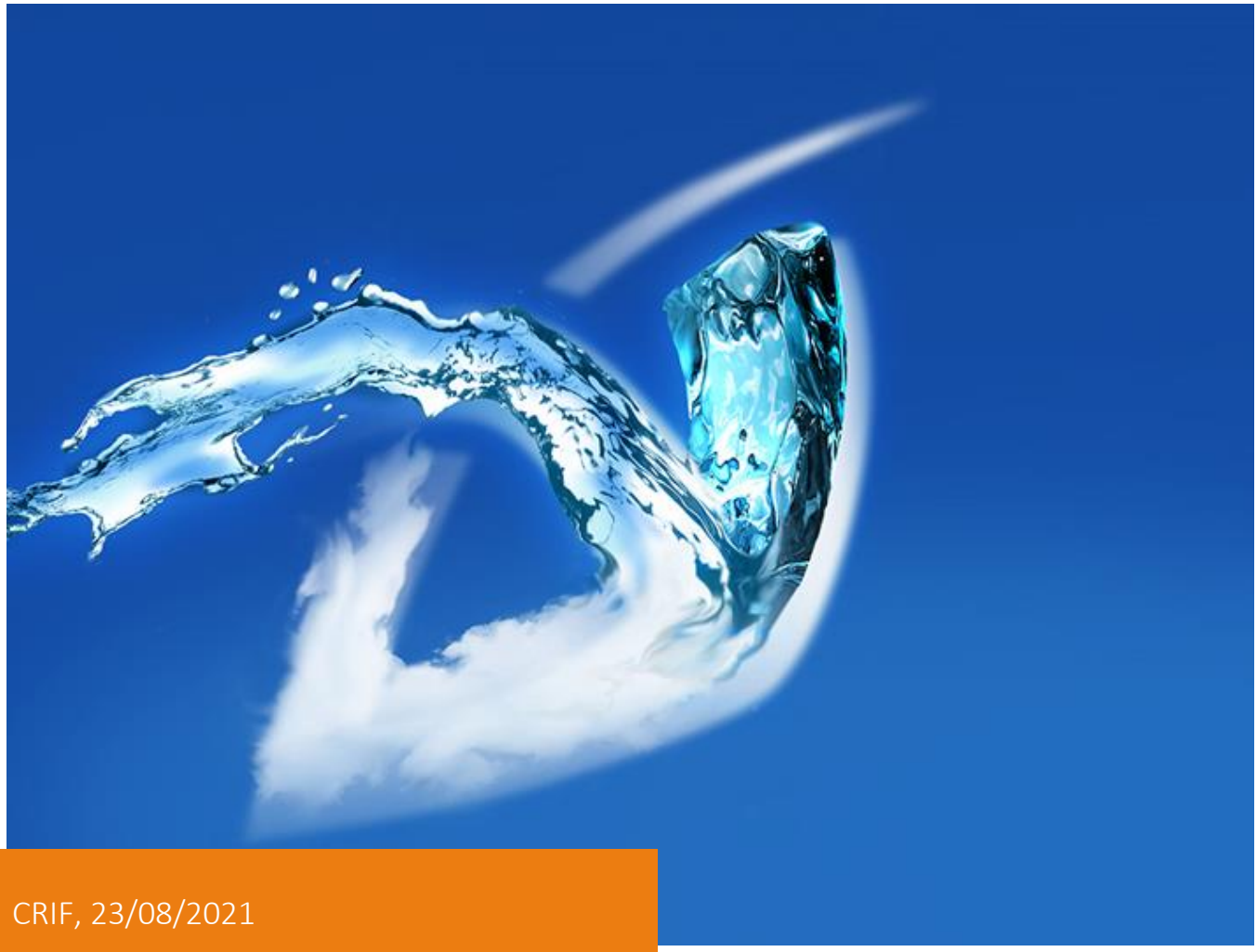
Working with Aspose templates

User Guide v.2.0



CRIF, 23/08/2021

Working with Aspose templates

Contents

1. Introduction

The Aspose library has been supported in the Contact system resulting in users having the opportunity to load templates of printable forms in the format of Word files, as well as files of other text formats. Templates can contain text, variables, tables, conditionals, and so on. After processing a template, the system generates a finished document, substituting the required variables and executing the specified logic. This document is intended for users who will create and customize the templates in the system.

The document covers:

- Description of input data model that is used for working with a template
- Extension of input data model with custom entities
- Typically used cases, including:
  - adding variables
  - using conditionals
  - formation of tables
  - filtering and grouping of tables' data
  - calculating totals in tables
- Adding a ready-made template to the system

2. Syntax and examples

This section covers the most typical cases of template configuration, including:

- Adding variables
- Using conditionals
- Formation of data tables
- Filtering data in a table
- Calculating totals in a table
- Grouping data in a table

2.1 Adding variables

You can output the value of the input data item by using the statement *<<[x]>>*, where the element name is indicated inside the square brackets. If the element is nested, the dot separator is used. The case of characters in the square brackets is not important.

The text outside the *<[[x]]>* statement is static and is displayed as is.

In the template, you can declare your own variable, to which you can assign any value or expression based on other variables.

Example: Adding variables to the text

```
<<var [fio = debt.debtor.lastName + “” + debt.debtor.firstName + “” +debt.debtor.middleName]>>
Debtor: <<[fio]>>
Debt amount: <<[debt.debtAmount]>>rub
Credit issue date: <<[debt.startDate]:”dd MMM yyy”>>
```

Generated result:

```
Debtor: Salnikov Alex Matv
Debt amount: 24537625.977 rub
Loan issue date: 10 Jan 2018
```

2.2 Using conditionals

You can display text in a template when a certain condition is met. The general format of a conditional is as follows:

<< if [condition 1] >>

text for condition 1

<< elseif [condition 2] >>

text for condition 2

<<else>>

text if no conditions are met

<</if>>

The *elseif* and *else* statements are optional.

Example: Using *if* statements

"If the requirement is not met, we will have to go to court!" This text will be displayed if the amount of debt is more than 10000.

```
<<if [debt.debtAmount >= 10000]>> If the requirement is not met, we will have to go to court!<</if>>
<<if [debt.Amount < 10000] Debt amount is less than 1000<<else>>Debt amount is more than 10000<</if>>
```

Generated result:

```
If the requirement is not met, we will have to go to court!
The amount of debt is more than 10000
```

2.3 Formation of data tables

For data passed as a collection (array), you can form duplicate elements by using the *foreach* statement.

<< foreach [in item_containing_collection] >>

Repeating part

<</foreach>>

To build a table, you need to specify in the first cell the beginning of the *<< foreach* ... loop and the output of the first element. The last element is displayed in the last cell of the table and the loop ends with the *<</foreach>>* tag. The elements of the loop are accessed without specifying the parent element in which they are nested.

Example: Formation of a table with payments

Here is a table with the following columns: Payment date, Purpose of payment, Payment amount.

Payment date	Purpose of payment	Payment amount
<<foreach [in debt.payments]>><<[paymentDateTime]:"yyyy-MM-dd">>	<<[purposeName]>>	<<[amount]:"###,###.##">>rub<</foreach>>

Generated result:

Payment date	Purpose of payment	Payment amount
2017-02-01	Principal	8 000 rub
2017-02-01	Principal	500 rub
2017-02-01	Principal	800 rub
2017-01-30	Interest	13 000 rub
2017-01-30	Interest	1 000 rub

2.4 Filtering data in a table

You can use the built-in *where* function to filter the items that you want to display in a table. In this function, you need to specify a condition by which to filter the elements. An object of collection is passed to the function, and you can refer to any element of this object.

The function format is as follows:

where (object_name => condition)

Example: Filtering payments in a table

Here, only payments with an amount more than 1000 are displaying.

Payment date	Purpose of payment	Payment amount
<<foreach [in debt.payments.where(p => p.amount>1000]>><<[paymentDateTime]:"yyyy-MM-dd">>	<<[purposeName]>>	<<[amount]:"###,###.##" >>rub<</foreach>>

Generated result:

Payment date	Purpose of payment	Payment amount
2017-02-01	Principal	8 000 rub
2017-01-30	Interest	13 000 rub

2.5 Grouping data in a table

For grouping elements, you can use the built-in function *groupBy*, which is available for collections (arrays). The function format is as follows:

groupBy (object_name => expression_returning_the_value_of_the_grouping_key).

An object is passed to the body of the function, through which you can refer to any element of the object. The expression must return a value (grouping key) by which the data should be grouped. You can display the value of the grouping key by using the <<*[key]*>> tag.

To calculate the total of the grouped elements, use the *sum* function.

Example: Grouping data by a column

Here, the data has been gouped by the colimn Purpose of payment and the amount of payments has been displayed for each group.

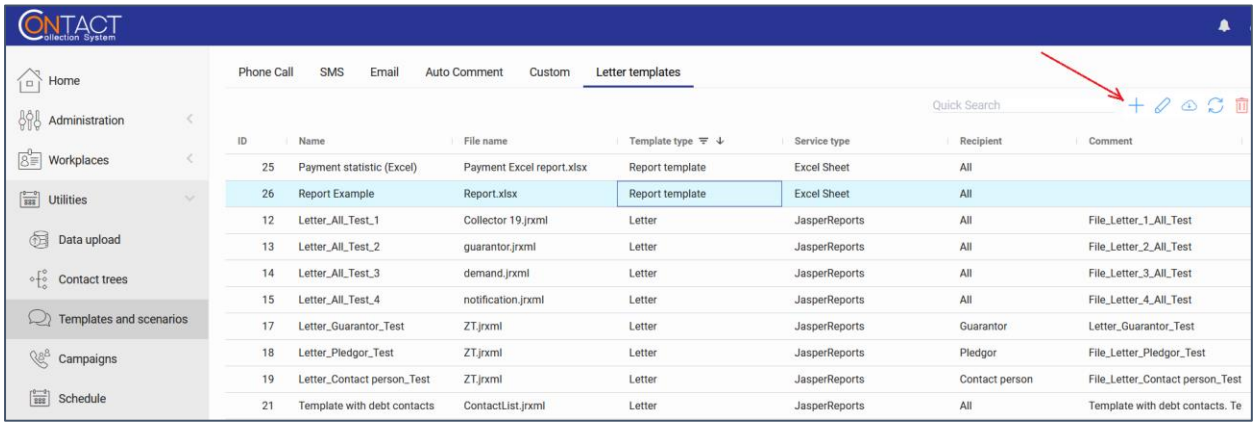
Purpose of payment	Amount of payments
<<foreach [in debt.payments.groupBy(p => p.purposeName)]>><<[key]>>	<<[sum(p=>p.amount)]:"###,###.##">>rub<</foreach>>

Generated result:

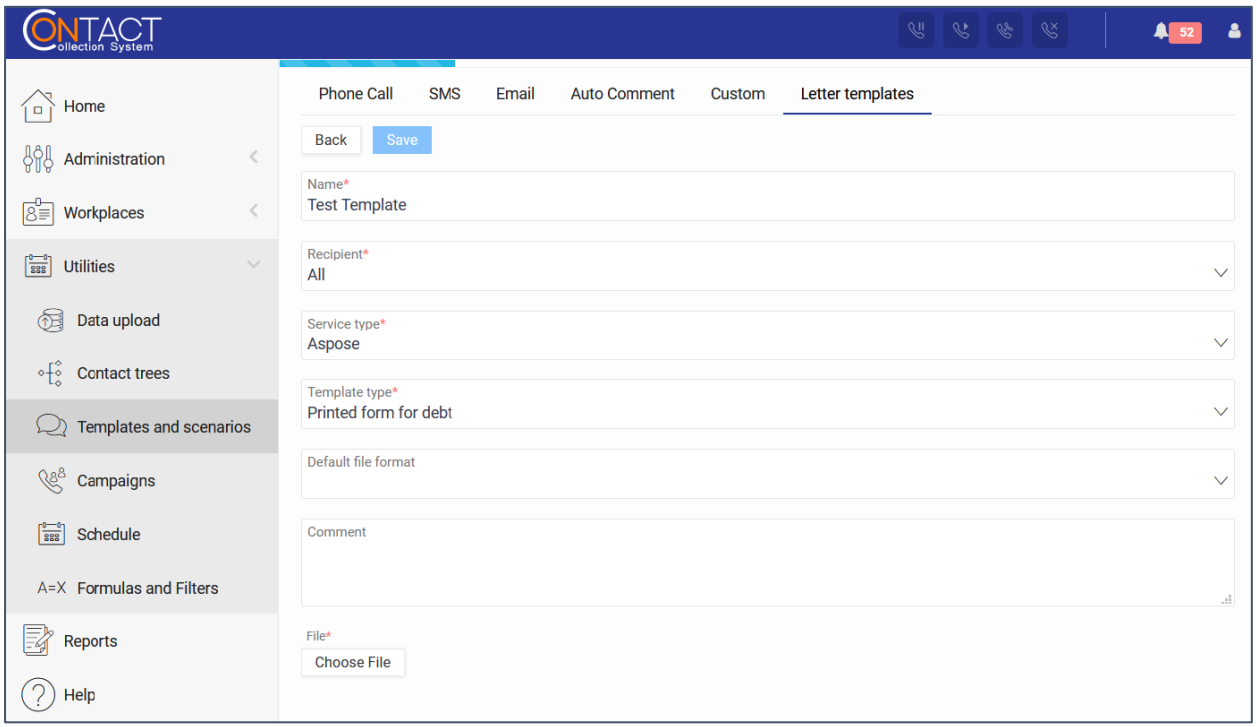
Purpose of payment	Amount of payments
Principal	9 300 rub
Interest	14 000 rub

3. Adding templates to the system

To add a ready-made template into the system, on the main menu, select the item *Utilities -> Templates and scenarios*. On the tab *Letter templates* tab, click the ⊞ button.



Once the form appears, fill in the required parameters:



- **Name**—any name of a template;
- **Recipient**—select the category of recipients who will have access to the template;
- **Service type**—select *Aspose*;
- **Template type**—select *Printed form for debt* or *Letter*;
- **Default file format**—a file format in which the report will be generated (optional);
- **Comment**—a comment (optional);

- **Choose File**—choose a file with the template.

Save the changes. The added template will be displayed in the list of templates.