SERAM Excel Merge

Sustainable Enterprise Reporting And Management

by Sirius Technologies AG

Notices

Disclaimer
The information in this manual is provided on an “as is” basis, without warranty. While every effort has been taken by the author in the preparation of this manual, the author shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this manual.

This manual may contains links to third-party web sites that are not under the control of the author. The author is not responsible for the content of any linked site. Inclusion of a link in this manual does not imply that the author endorses or accepts any responsibility for the content of that third-party site.

Trademarks
All terms mentioned in this manual that are known to be trademarks or service marks have been capitalized as appropriate. Use of a term in this manual should not be regarded as affecting the validity of any trademark or service mark.

Versions
Software release: Banker (Build 2472)
Document revision: 495

Contents

Chapter 1

Excel Merge

Excel Merge is a SERAM companion desktop application which allows users to extract, process and compute data from Excel files and feed the values into SERAM by the means of the SetValues REST API.

Key Features

- Can process any number of Excel files as one batch
- The mappings can reference any sheet, cell and range
- Computations can be done when mapping, such as aggregating cell ranges or looking up names
- Additional files (for instance containing some lookup data) can be loaded and used
- Can not only set the value, but also comments etc.
- Detailed log provides feedback on progress, operations performed and errors

Editing a mapping

The mapping defines how the files processed in the application shall be converted to values.

The mappings are defined in the `ExcelMerge.exe.config` configuration file. Excel Merge does not have a built-in editor. Please use your favorite XML editor to open the file and edit the mapping.

Chapter 2

Mapping Language

Excel Merge uses a domain-specific mapping language for the mapping definitions.

Grammar

The Excel Merge mapping language uses a well-defined grammar, with a lexical syntax inspired by Excel.

Overview

- All *identifiers* and *keywords* are not case-sensitive
- *Statements* are separated by `;`, typically written at the end of the statement line
- *Line comments* (e.g. comments which implicitly end at the end of the text line) are started with `#`
- *Block comments* are same as in XML: `<!--` ... `-->`
- *Identifiers* start with a letter and continue with any number of alphanumeric characters
- *Strings* are enclosed in `""` (`"` is escaped by doubling it)

Tip: newlines etc. are allowed in strings

- *Numbers* are integers or floating point with optional exponent (e.g. `1`, `1.5`, `3.65e-2` are all valid numbers)
- *Worksheet references* are enclosed in single quotes `''` (just like in Excel)
- *Variable names* are identifiers prefixed with a `$` sign

Expressions

Expressions generate a value. They can be a simple mathematical formula, but also access object properties or call methods of objects. They always produce a result value.

A special global function is `IMPORT(<Expression>)`. The import will load an additional Excel workbook, relative to the current directory. This workbook, or a part of it, may then for instance be assigned to a variable for later use, for instance as lookup table.

Lamdas

Some methods of collections accept a lambda as argument. Lamdas can have one or more arguments, and they return a value computed based on the passed arguments.

```
DEFINE $factor := 10;
DEFINE $values1 := 'Sheet 1'!A1:A10.Select($cell => $cell *
 $factor);
```

This would generate a collection of numbers, where each number is based on one of the cell `A1..A10` values multiplied by the variable `$factor`.

```
DEFINE $valies2 := 'Sheet 1'!B1:B10.SelectIndex($cell|$index
 => $cell * $index);
```

This would generate a collection of numbers, where each number is based on one of the cell `B1..B10` values multiplied by the index in the collection (starts with 0), e.g. the first value would be 0.

Parameters

Parameters are used for specifying structures, indicators, periods and time indexes when setting values. The expressions for a parameter can either return a single value or a collection of values (such as a cell range). When a collection is returned, the value expression will be evaluated once per item in the collection. A variable name can be specified which is then available with the current value of the enumeration when the value expression is being evaluated.

```
["Site" | "Something" | $year: 'Years'!A1:A5.Select($cell =>
 $cell.ToNumber()) | $index: 'Months'!A1:A12.Select($date =>
 $date.Month.Index)] := $year*$index
```

This would take the range `A1` to `A5` in the worksheet `'Years'` as year values, and `A1` to `A12` of the worksheet `'Months'` as dates where the month index is extracted, which would cause 60 values (e.g. 5 * 12) to be processed.

Statements

The Excel Merge mapping language uses a small set of statements for defining the mapping to be performed.

SHEET *<WorksheetName>* ALIAS *<Epxression>*

When referencing the sheet <*WorksheetName*>, also consider a sheet with the name produced by <*Expression*>. This can be used for multilingual workbooks, or when different versions with diverging sheet names are used.

The worksheet references in the mapping will therefore remain constant, even if the actual sheets have different names in different files being processed.

MESSAGE *<Expression>*

Write a message to the log. Useful for providing status information or when debugging a mapping.

DEFINE *<VariableName>* := *<Expression>*

Evaluate the <*Expression*> and store its result in the specified variable for later reference.

[*<StructureParameter>* | *<IndicatorParameter>* | *<YearParameter>* | *<IndexParameter>*] := *<Expression>* (optionally followed by additional data dimensions)

Compute and set the value(s) with the given parameters.

Additional value dimensions can optionally be added after the value expression:

- `COMMENT` *<Expression>*
- `FORECASTED` *<Expression>*
- `DATAQUALITY` *<Expression>*
- `STATUS` *<Expression>*
- `DATECHANGED` *<Expression>*

Objects

Object (base for all objects)

Often implicitly convertible to Text

Property IsNull: Boolean

Function ToBoolean(): Boolean

Function ToText(): Text

Function ToNumber(): Number

Boolean: Object (equatable)

Implicitly convertible to Number (1 for "True", 0 otherwise)

Operators `&& || ^ !`

Cell: Object

Implicitly convertible to Number (by using the cell value)

Implicitly convertible to Date (by using the cell value)

Implicitly convertible to Boolean (by using the cell value)

Property Column: Range

Property Row: Range

Property ColumnIndex: Number

Property RowIndex: Number

Property Position: Text (Excel cell reference)

Property Value: Object

Property Worksheet: Worksheet

Function Relative(Number horizontalOffset, Number verticalOffset): Cell

Date: Object (comparable)

Implicitly convertible to Number (as Excel date numeric)

Property Millisecond: Number

Property Second: Number

Property Minute: Number

Property Hour: Number

Property Day: Number

Property DayOfWeek: Number (Sunday=0, Monday=1 etc.)

Property DayOfYear: Number

Property Month: DateMonth

Property Quarter: DateQuarter

Property Year: DateYear

DateMonth: Object (comparable)

Implicitly convertible to Number

Property Index: Number (SERAM index 5..16)

DateQuarter: Object (comparable)

Implicitly convertible to Number

Property Index: Number (SERAM index 1..4)

DateYear: Object (comparable)

Implicitly convertible to Number

Property Index: Number (SERAM index 0)

Enumeration<?>: Object (? can be any Object class)

Property Count: Number

Function First(): ? (first element)

Function Last(): ? (last element)

Function Join(Text separator): Text (concatenate as Text with given separator in between items)

Function Average(): Number (average, no value => empty)

Function Sum(): Number (sum, no value => empty)

Function SumWith(Number number): Number (sum, starting with number)

Function Min(): ? (smallest item)

Function Max(): ? (largest item)

Function Skip(Number count): Enumeration<?>

Function Take(Number count): Enumeration<?>

Function NumbersWithValue(): Enumeration<?>

Function Select(lambda with ? returning !): Enumeration<!>

Function SelectIndex(lambda with ? | Number returning !): Enumeration<!>

Function SkipWhile(lambda with ? returning Boolean): Enumeration<?>

Function TakeWhile(lambda with ? returning Boolean): Enumeration<?>

Function Where(lambda with ? returning Boolean): Enumeration<?>

File: Object (comparable)

Property FullName: Text

Property Name: Text

Property Extension: Text

Property DirectoryName: Text

Property Exists: Boolean

Function ReadAllLines: Enumeration<Text>

Function ReadAllText: Text

Number: Object (comparable)

Implicitly convertible to Date

Implicitly convertible to Boolean

Property HasValue: Boolean

Operators `+ - / * mod % ??` (with same rules as SERAM: + and - with empty values have no effect)

Range: Enumeration<Cell>

Property Worksheet: Worksheet

Property Rows: Enumeration<Range>

Property Columns: Enumeration<Range>

Property CellCount: Number

Property RowCount: Number

Property ColumnCount: Number

Property FirstColumnIndex: Number

Property LastColumnIndex: Number

Property FirstRowIndex: Number

Property LastRowIndex: Number

RegexMatch: Object

Property Count: Number

Property Index: Number

Property Success: Boolean

Function Format(Text format): Text (format with format pattern)

Function NextMatch(): RegexMatch

Text: Object (comparable)

Implicitly convertible to Number (by using the cell value)

Implicitly convertible to Date (by using the cell value)

Implicitly convertible to Boolean (by using the cell value)

Property Length: Number

Function Append(Text other): Text (concatenate two texts)

Function Left(Number count): Text (beginning of string)

Function Right(Number count): Text (end of string)

Function Match(Text pattern): RegexMatch

Function Replace(Text pattern, Text replacement): Text

Function Split(Text pattern): Enumeration<Text>

Function Substring(Number index, Number count): Text

Function RemoveAccents(): Text

Workbook: Object

Property ActiveSheet: Worksheet

Property File: File

Function Sheet(Text name): Worksheet

Worksheet: Object

Property Exists: Boolean

Property Name: Text

Property Workbook: Workbook

Function Cell(Text position): Cell

Function Range(Text startPosition, Text endPosition): Range