Apparo Fast Edit

Data quality

Version 3.2.2

Table of content

1 Introduction 3

2 Data row validation 4

2.1 Basics 4

2.2 Activating the feature 4

2.3 Help menu 5

2.3.1 Data row validation help 5

2.3.2 JavaScript help 5

2.4 Usage example 6

2.4.1 You need to know 7

2.4.2 The code behind the example 8

2.4.3 Check #1 in detail 9

2.5 Useful Apparo JavaScript methods 10

2.5.1 Example for calling a Java class with return value 10

2.5.2 Example for calling a Java class with arguments and return value 10

2.5.3 Example for working with date widget variables 10

2.5.4 Example for executing a SQL query 11

2.5.5 Example for executing a SQL select 11

2.5.6 Example for executing a SQL query with parameters 11

3 Validation on widget level 12

3.1 Limits 12

3.2 Java classes 13

3.2.1 Java class for testing 14

3.3 Regex – Regular expressions for input fields and text areas 15

4 Lookup widgets 16

1 Introduction

Apparo offers several ways to ensure data quality.

Data checking is performed regardless of the method of data entry, whether manually inserted or imported from Excel.

Basically, there are two ways to check data:

- Validation on row level, which can compare values or check them directly.
- Validation on widget level, checking one entry only.

2 Data row validation

2.1 Basics

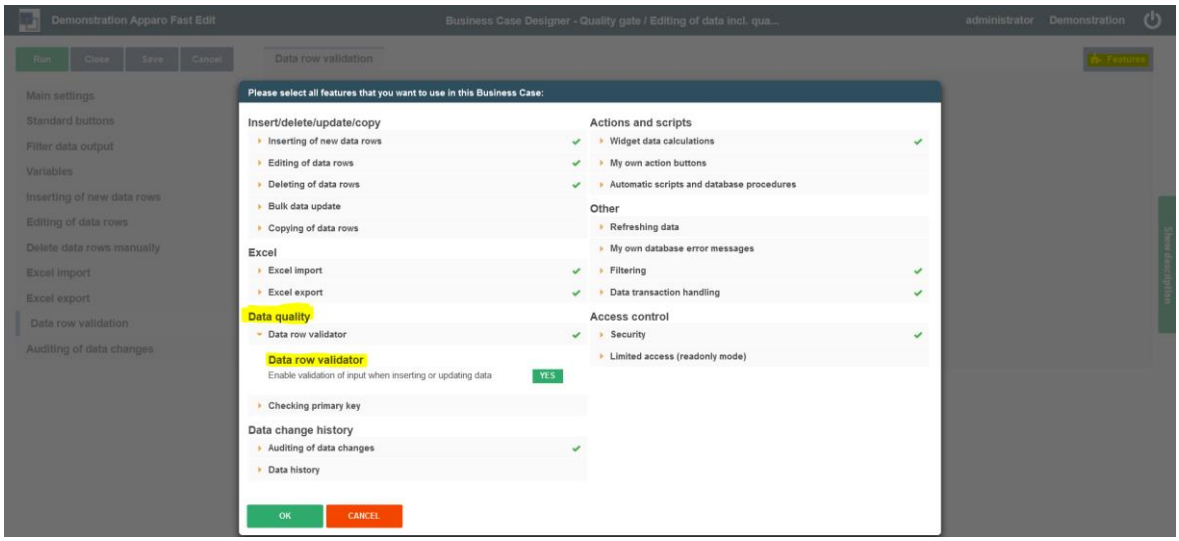
Data row validation uses simple JavaScript that is running server side to check the data values of a data row in relation to each other.

It is also possible to check only one or more entries independently using the full spectrum of JavaScript or by calling external Java classes.

This feature is used for manual data entry using web forms and Excel data imports (all Excel formats).

2.2 Activating the feature

The data row validation feature must be enabled first:

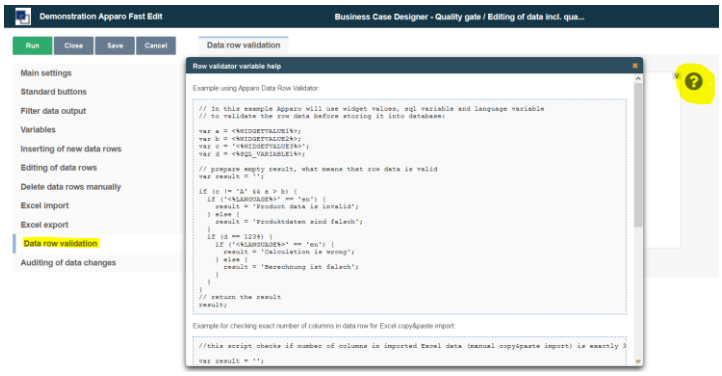


Go to the features menu and switch the Data row validator under Data quality to ‘Yes’ (enabled)

2.3 Help menu

2.3.1 Data row validation help

The data row validation help can be called by clicking the grey question mark in the data row validation menu:

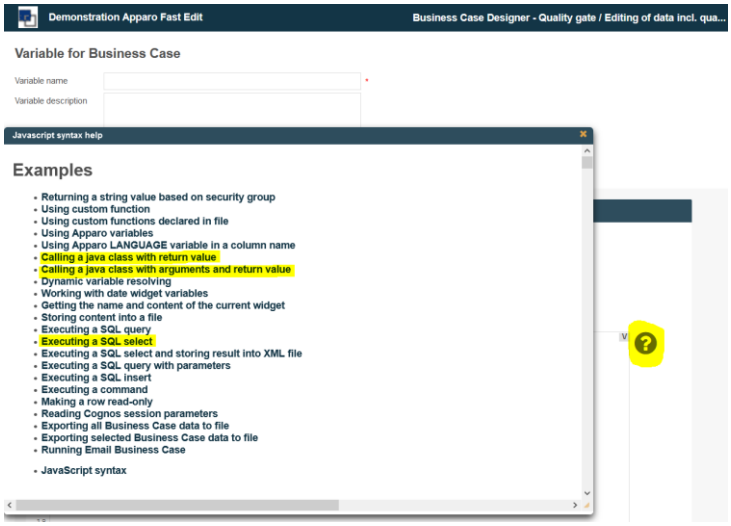


The help contains three examples:

- Example using Apparo Data Row Validator
- Example for checking exact number of columns in data row for Excel copy&paste import
- Example of returning warning message instead of error result of validation. Warning message is displayed to user, but row is processed normally

2.3.2 JavaScript help

You will find the complete JavaScript help when creating a new JavaScript variable in the Variables menu of the Business Case:



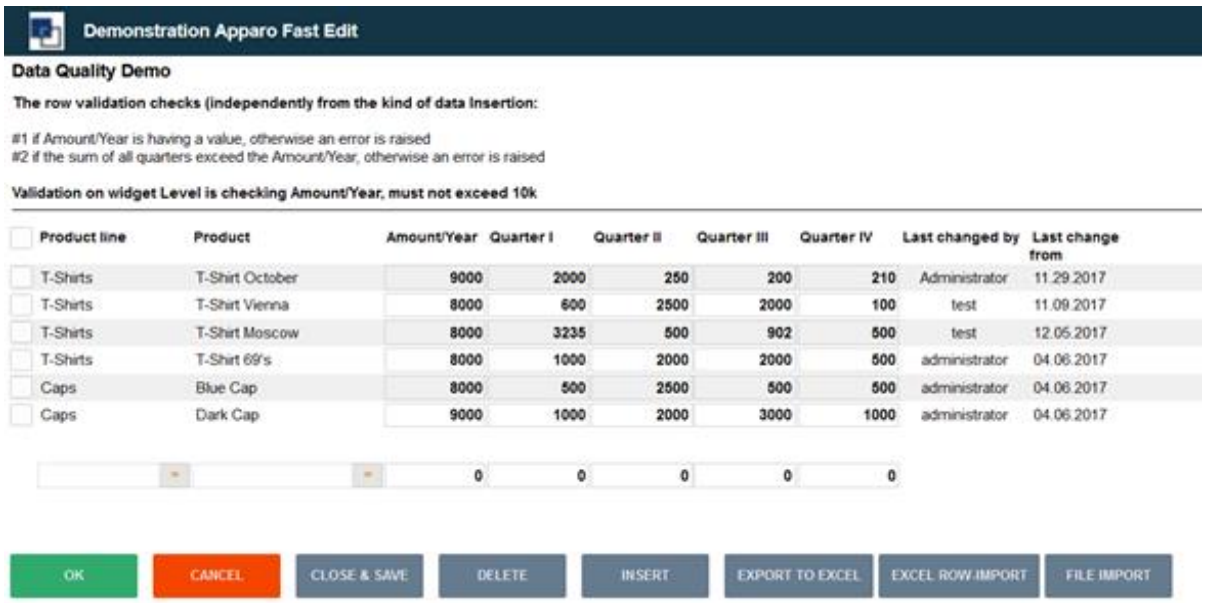
The marked examples may be helpful in the data row validator too and will be explained later in this guide

The help contains Apparos custom JavaScript methods, helpful examples and the complete JavaScript syntax.

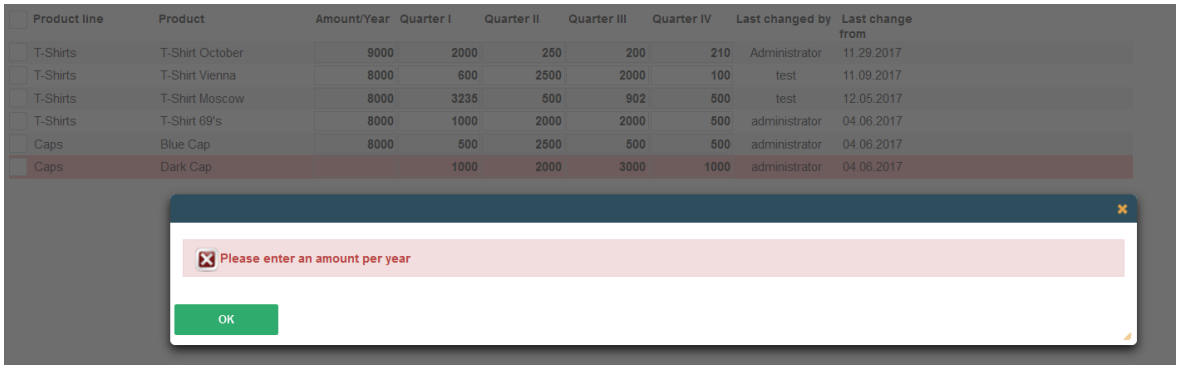
The description of all custom Apparo JavaScript methods is also available in the user guide.
The user guide is contained in the Apparo installation package, linked in the Training menu in the Apparo Designer or online under https://apparo.solutions/en/docs-en.

2.4 Usage example

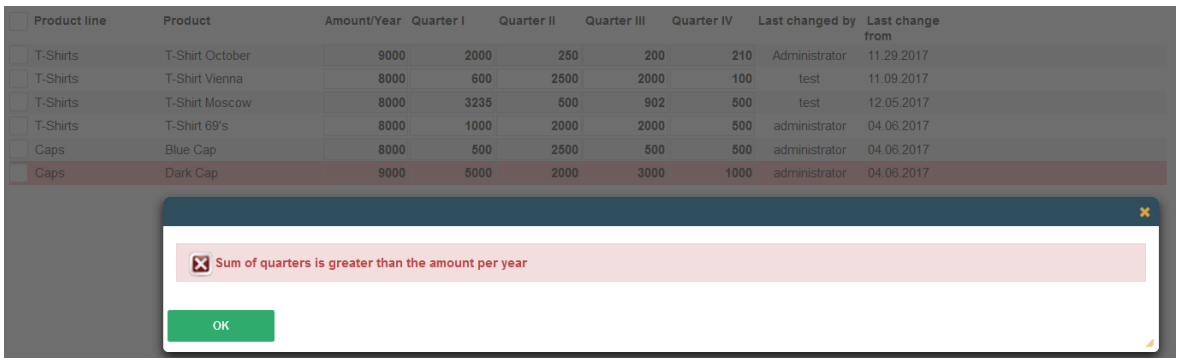
Our example Business Case contains two checks. The checks are working in insert and edit area.



Check #1 if Amount/Year is having a value, otherwise an error is raised



Check #2 if the sum of all quarters exceeds the Amount/Year, then an error is raised



2.4.1 You need to know

The code consists essentially only of the declaration of variables, their assignment and the use of if-conditions.

Variables will be declared by using 'var' followed by the variable names and the closing sign:

var variables_name;

You can assign values directly in this step, also Apparo variables or JavaScript methods like

var variables_name = <%Apparo_variable%>;

You can use all types of Apparo variables here.

var productsArray = afe.executeSqlSelect('select id, name, price from MySchema.MyProduct');

Strings must be enclosed in single quotes, also variables containing string values

var txt = 'text';

var txt = '<% Apparo_String_Variable%>';

In case of errors, an error message must be returned

In the following examples a JavaScript variable named "result" is used to return an error.
It contains an empty string (var result = '';) which means = no errors.
If the check is hitting then fill the variable with the error message (result = 'Error message';)
Error messages always prevent users from inserting the values.

Warnings

It is possible to return a warning instead of an error.
Warnings create a message window returning the the text of the warning, but the user is still able to insert and store the value.

Usage: *(result = 'APPARO-WARNING:Warning message text';)*

2.4.2 The code behind the example

```
var a = afe.resolveVariable('FORECAST');      //This method resolves table column names as variables
var b = <%FORECAST2%>;                        //This a widget reference variable, containing its value
var c = <%FORECAST3%>;                        // Both methodes are equal, returning the widget's value
var d = <%FORECAST4%>;
var e = <%FORECAST5%>;
var result = '';                              // prepare empty result, what means that row data is valid
 if (a < (b+c+d+e)) {                         // This if compares the variables for check #2
   if ('<%LANGUAGE%>' == 'en') {              // This if is a language switcher return errors in En and De
     result = 'Sum of quarters is greater than the amount per year';  }
else { result = 'Summe der Quartale ist größer als das Jahr';}}
if (a == null || a==0) {                      // This if is the check for #1
 if ('<%LANGUAGE%>' == 'en') {                // Another language switcher
   result = 'Please enter an amount per year';  }
else { result = 'Bitte geben Sie die Anzahl je Jahr an'; }}
result;                                       // return the result, which means raise an error or not
```

2.4.3 Check #1 in detail

if (a == null || a==0)

An if clause in JavaScript is followed by the condition check enclosed in parentheses.
'a' is here the widget reference variable mapped to the JavaScript variable named 'a'.
To compare if a= null (db null, which means empty) we have to write a==null.
One single equal sign = would assign the value null to the JavaScript variable 'a'.

{ if ('<%LANGUAGE%>' == 'en')

If the condition of the if clause is met, the code in the braces is executed.
In this case another if clause is started asking if the users language is EN.
If yes, it executes the following code:

{ result = 'Please enter an amount per year'; }

This error message replaces the empty result and will be output to the user.
If the users language is not EN, then the next condition after else is executed.

else { result = 'Bitte geben Sie die Anzahl je Jahr an'; }}

Any other user language than EN will create this German error message. Which is ok for our demo, it uses EN und DE only.

2.5 Useful Apparo JavaScript methods

The JavaScript language has additional Apparo-methods for executing SQL, other Java classes, reading widget values etc.

2.5.1 Example for calling a Java class with return value

```
// In this example Fast Edit creates an instance of 'MyCustomClass' class and executes the
'myCustomMethod' method
var result = afe.callClassMethod('MyCustomClass', 'myCustomMethod');
result;
```

2.5.2 Example for calling a Java class with arguments and return value

```
// In this example Fast Edit creates an instance of 'MyCustomClass' class and executes the
'myCustomMethod' method
var args = [];        // create new array
args[0] = "stringValue";
args[1] = 256;       // passed to Java as Java.lang.Double
args[2] = (new Date()).getTime();    // passed to Java as Java.lang.Double

var result = afe.callClassMethod('MyCustomClass', 'myCustomMethod', args);
result;
```

2.5.3 Example for working with date widget variables

```
// In this example we will compare current Date with widget Date
var my_date_widget = afe.resolveVariable('DATE_WIDGET');
var current_date = new Date();

//for explicit date usage, e.g. December 24, 2016 at 6:30pm use format: Date(year, month-1, day, hour,
minute, second, millisecond)
//var date = new Date(2016,11,24,18,30,0,0);

if (my_date_widget == null) {
 result = 'The date widget is empty';
}
else if (my_date_widget.getTime() > current_date.getTime()) {
 result = 'The date widget value is after current date';
}
else if (my_date_widget.getTime() < current_date.getTime()) {
 result = 'The date widget value is before current date';
}
else {
 result = 'The dates are equal';
}

result;
```

2.5.4 Example for executing a SQL query

```
// In this example Fast Edit executes SQL query to retrieve 'user_id' value of 'John Smith' in table 'MyTable'.
var user_id = afe.executeSql("select id from MySchema.MyTable where sales_name='John Smith'");
```

2.5.5 Example for executing a SQL select

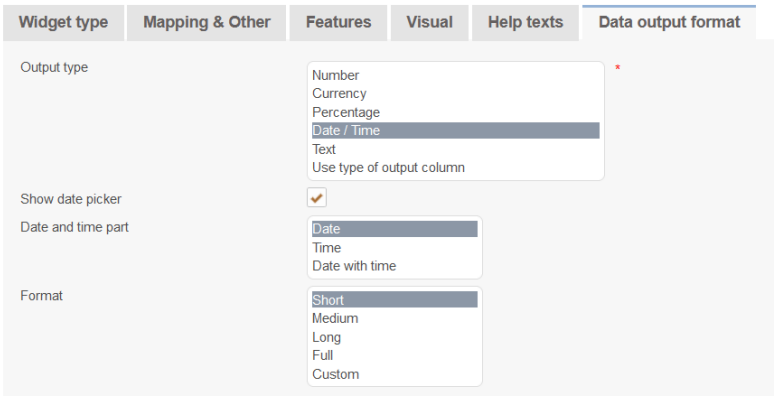
```
// In this example Fast Edit executes SQL select to retrieve 'id', 'name' and 'price' values of all products in
table 'MyProduct'.
var productsArray = afe.executeSqlSelect('select id, name, price from MySchema.MyProduct');
var i;
var rowData;
var id;
var name;
var price;

for(i = 0; i < productsArray.length; i++) {
  rowData = productsArray[i];
  id = rowData[0];
  name = rowData[1];
  price = rowData[2];
}
```

2.5.6 Example for executing a SQL query with parameters

```
// In this example Fast Edit executes SQL query with parameters.
var params = [];   // create new Array
params[0] = 'John Smith';
params[1] = 'Germany';
var user_id = afe.executeSql('select id from MySchema.MyTable where sales_name=? and country=?',
params);
```

3 Validation on widget level

In the Widget settings -> Data output format you can define the desired data formats and also enable e.g. datepickers, which are preventing wrong dates.

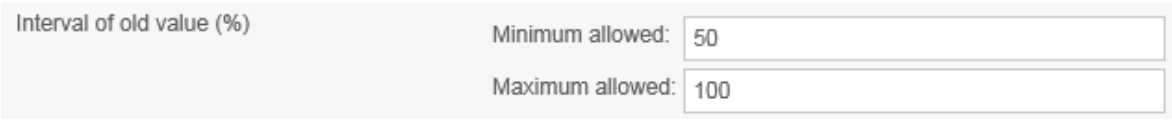


3.1 Limits

Limits can be defined as validity interval or as validity interval in percent of the old value.
Interval limits can be set as number (fix value) or dynamically with variables with e.g. calculations.

Variables can be used too. These variables can use widget reference variables (values of the current row for checking) too.

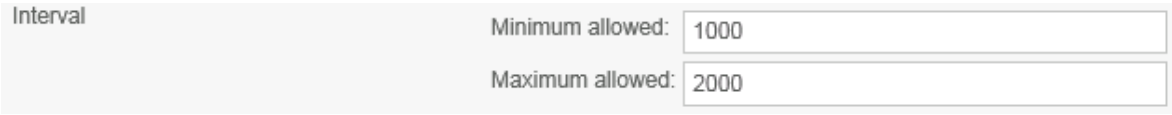
Interval of old value (%)



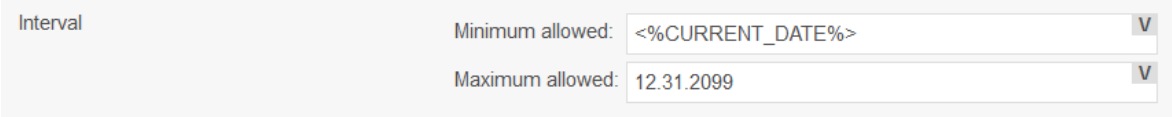
Hereby you limit the validity of the entered values based on the existing values.

Example: In the widget, the value is 100. In this case, users may only enter values between 50% and 100% of the original value, so values between 50 and 100. Otherwise, the user receives an error message.

Interval



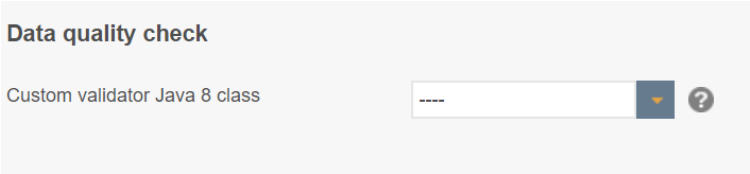
Limits the validity of entries based on an absolute interval. Permissible are values only from 1000 to 2000.



Restricts the validity of date entries between today and the end of the century.

3.2 Java classes

Custom validator Java 8 class



Optional. A Java 8 class that is testing the input value. The file directory of this file is defined in the Apparo Configuration Manager. This class is called automatically before Apparo Fast Edit is updating or inserting this row.

By clicking the grey question mark a help pops up containing three code examples:

#1 Definition of the Java validator interface

Contains the code

#2 Example for a string validator

This example shows a simple Java validator that checks if the value of the widget starts with "Apparo".

#3 Example for a date validator

This example shows a Java validator that checks if the value of the widget is a date between 12/01/2020 and 12/31/2025. Additionally, a warning is displayed if the value is between 01/01/2025 and 12/31/2025.

3.2.1 Java class for testing

Examples for a user exit – test if a value of a widget is valid or not:

- TesterPK.Java
- TesterNUMBER_VALUE.Java
- DateInRangeValidator.Java

Both are stored in *[APPARO HOME]\FastEdit\samples*

You need an installed Java 8 JDK because you need **Javac** for compiling.

Java version 8 must be used.

Please open a command shell (cmd/sh) and go into the file directory *[APPARO HOME]\FastEdit\samples*

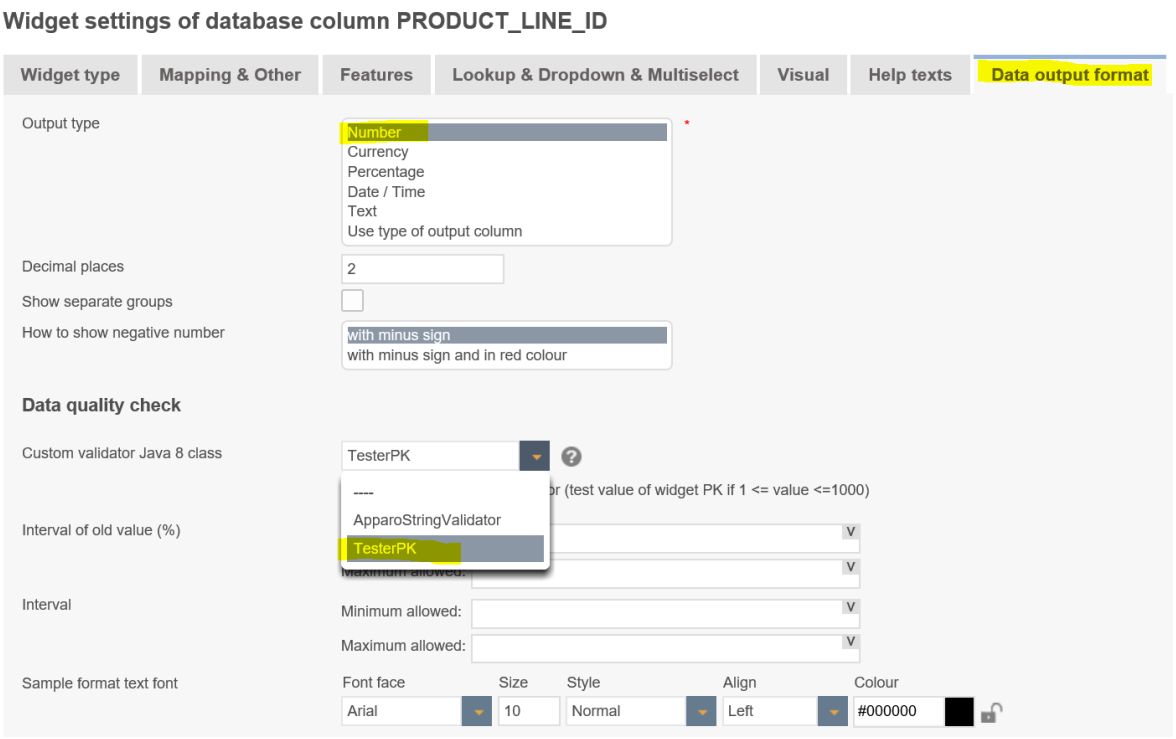
Enter now:

Javac TestValidator.Java TesterPK.Java

The result is the file **TesterPK.class** in the same file directory.

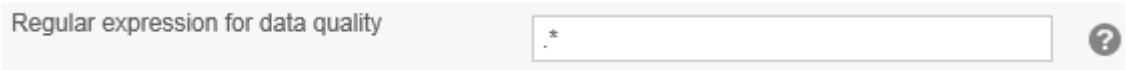
Please copy the **TesterPK.class** into the file directory *[APPARO HOME]\FastEdit\user_scripts*
The path may be different, depending on the settings in the Apparo Configuration Manager.

Then (without restarting Apparo) you can use this file in the Apparo Designer (Business Cases must be reopened eventually):



3.3 Regex – Regular expressions for input fields and text areas

Regular Expression (Only for type ‚Text')



Using a regular expression is helpful to define more complex input rules.

Click the '?' icon to see the detail instructions.

Characters		
Character	**Description**	**Example**
Any character except `[\^$.|?*+()`	All characters except the listed special characters match a single instance of themselves.	`a` matches a
`\` (backslash) followed by any of `[\^$.|?*+()`	A backslash escapes special characters to suppress their special meaning.	`\+` matches +
`\xFF` where FF are 2 hexadecimal digits	Matches the character with the specified ASCII/ANSI value, which depends on the code page used. Can be used in character classes.	`\xA9` matches © when using the Latin-1 code page.
`\n`, `\r` and `\t`	Match an LF character, CR character and a tab character respectively. Can be	`\r\n` matches a

Example

You want to validate international phone numbers. The numbers should start with a plus sign, followed by the country code and national number +49 55512345678

Regex

^\+(?:[0-9] ?){6,14}[0-9]$

Explanation in detail

```
^          # Assert position at the beginning of the string.
\+         # Match a literal "+" character.
(?:        # Group but don't capture:
 [0-9]     #   Match a digit.
 \x20      #   Match a space character (Or just use space " " character)
  ?        #    between zero and one time.
)          # End the noncapturing group.
 {6,14}    #   Repeat the group between 6 and 14 times.
[0-9]      # Match a digit.
$          # Assert position at the end of the string.
```

4 Lookup widgets

A lookup widget is accepting a value only if it is member of a value list.
Wrong data format entries are not possible.

There are 3 types of lookup widgets you can use:

Simple dropdown widget (for target table only)

Creates a dropdown list with all values that are already available in the target table

Lookup dropdown (for all tables)

Creates a dropdown list with values read from a lookup table.
In most cases such tables contain a lookup id and another column with the lookup output text

Lookup multi select

Creates a multi select widget which is not directly connected to the target table of the Business Case. Requires a lookup source table, containing lookup id and lookup output text and also requires a lookup target table, where all entries are stored. The primary keys of the Business Case are serving as foreign keys to map the entries.