

User Guide

Apparo Fast Edit

for

IBM Cognos Analytics

Version 3.2.2



## Content

<b>1</b>	<b>Premises .....</b>	<b>7</b>
<b>2</b>	<b>Designing Business Cases.....</b>	<b>8</b>
2.1	Start screen with a list of all Business Cases .....	9
2.1.1	Definition .....	9
2.1.2	Buttons and Sorting .....	9
2.1.3	Folder.....	10
<b>3</b>	<b>Description and normal mode .....</b>	<b>11</b>
<b>4</b>	<b>Database &amp; E-Mail connections .....</b>	<b>12</b>
4.1	Database connections .....	12
4.1.1	Supported databases .....	12
4.1.2	Buttons .....	12
4.1.3	Creating a new database connection .....	13
4.2	Email connections .....	18
4.2.1	List of e-mail connections .....	18
4.2.2	Supported protocols.....	18
4.2.3	Buttons .....	18
4.2.4	Creating a new e-mail connection .....	19
<b>5</b>	<b>Database transaction handling using OK/STORE/CLOSE/CANCEL buttons.....</b>	<b>21</b>
5.1	OK-Button behaviour without the CLOSE-Button .....	21
5.2	OK-Button and CLOSE-Button behaviour.....	22
5.3	Behaviour of the CANCEL-Buttons .....	22
5.4	CLOSE-Button in an embedded Business Case .....	22
<b>6</b>	<b>Table Business Cases (Table BC).....</b>	<b>23</b>
6.1	Definition .....	23
6.2	Areas of a Table Business Case .....	24
6.3	Create a new Business Case.....	25
6.4	Business Case features .....	26
6.4.1	Features areas and features .....	27
6.5	Edit view of the Business Case .....	28
6.6	Business Cases Settings .....	29
6.6.1	Main settings .....	29
6.6.2	Widgets.....	30
6.6.3	Edit view .....	30
6.6.4	Widget types and areas.....	31
6.6.5	Widgets in the Edit Area .....	33
6.6.6	Special functions in the widget settings .....	34
6.6.7	Widget settings for the example ,Input field' .....	37
6.6.8	Special settings other widget types.....	48
6.7	Business Case Functions .....	63
6.7.1	Standard Buttons.....	63
6.7.2	Own action buttons .....	65
6.7.3	Filter data output.....	71

6.7.4	Filter widgets .....	72
6.7.5	Combine Widgets with AND/OR.....	72
6.7.6	Variables .....	73
6.7.7	Inserting of new data rows .....	93
6.7.8	Editing of data rows.....	94
6.7.9	Deleting of data rows .....	95
6.7.10	Bulk data update.....	96
6.7.11	Excel Import.....	97
6.7.12	Verify that all the files were imported.....	107
6.7.13	Excel export .....	108
6.7.14	Copying of data rows .....	111
6.7.15	Checking primary key .....	112
6.7.16	Data row validator .....	113
6.7.17	Data transaction handling .....	114
6.7.18	Automatic scripts and database procedures .....	115
6.7.19	Auditing of data changes .....	120
6.7.20	Data History .....	122
6.7.21	Security .....	126
6.7.22	Limited Access .....	127
6.7.23	My own database error messages.....	128
<b>7</b>	<b>Single Business Cases (SBC) .....</b>	<b>129</b>
7.1	Structure of the SBC.....	129
7.2	Arrangement of the widgets in the SBC .....	130
7.3	Visual .....	132
7.4	Starting the Single Business Case in Insert mode using URL parameter .....	133
<b>8</b>	<b>Business Case Sets (Set).....</b>	<b>134</b>
8.1	Selection and positioning of Business Cases in the set (Set) .....	134
8.2	Colors .....	135
8.3	Tab Widths .....	135
8.4	Global Set filters .....	136
<b>9</b>	<b>E-Mail Import Business Case (EIBC).....</b>	<b>137</b>
9.1	Creating a new Business Case of type 'Email import' .....	139
9.1.1	New Business Case - Main Settings .....	140
9.2	Overview of all possible settings .....	141
9.3	Main Settings.....	142
9.4	Importing Groups.....	142
9.5	Importing group settings .....	143
9.5.1	Main group settings.....	143
9.5.2	Business Cases .....	144
9.5.3	Email texts .....	147
9.5.4	Security .....	148
9.6	eMails .....	149
9.7	Logging.....	150
9.8	Variables .....	151
<b>10</b>	<b>E-mail Business Cases (EBC) .....</b>	<b>152</b>

10.1	Creating an EBC .....	153
10.2	Header and Footer .....	154
10.3	E-mail properties .....	155
10.4	E-Mail body .....	156
10.5	Email attachments .....	156
10.6	Button titles .....	157
11	<b>Action Business Cases (ABC) .....</b>	<b>158</b>
11.1	Possible Actions .....	158
11.2	Use Cases .....	159
11.3	Creating a new Action Business Case .....	160
11.4	Defining the main settings.....	160
11.5	Header & Footer .....	161
11.6	Visual settings.....	161
11.7	<b>Actions .....</b>	<b>162</b>
11.7.1	Javascript commands.....	162
11.7.2	SQL commands .....	162
11.7.3	Database stored procedures and functions .....	162
11.7.4	Exceuting anonymous block .....	164
11.7.5	Calling a script or batch file on the server .....	164
11.8	<b>Buttons.....</b>	<b>165</b>
11.9	<b>Variables .....</b>	<b>166</b>
11.10	<b>Link into IBM Cognos Analytics.....</b>	<b>167</b>
11.11	<b>Action Business Case with user interface output .....</b>	<b>167</b>
11.12	<b>Action Business Case without user interface output.....</b>	<b>168</b>
11.12.1	Placing a button into a Cognos report .....	168
11.12.2	List with checkbox column.....	172
11.12.3	Adapting the message window .....	175
12	<b>Business logic serverside .....</b>	<b>176</b>
12.1	<b>Example of a script variable.....</b>	<b>176</b>
12.2	<b>Example of a JavaScript script file .....</b>	<b>177</b>
12.3	<b>Example of the Row-Validator .....</b>	<b>177</b>
12.4	<b>Additional Apparo Fast Edit Methods .....</b>	<b>178</b>
12.5	<b>Example for the usage of Apparo Fast Edit Methods in Javascript.....</b>	<b>179</b>
12.5.1	Custom script example returning a string value based on security group .....	179
12.5.2	Refresh another embedded Business Case in an IBM Cognos Analytic report .....	179
12.5.3	Custom function example returning a string value .....	179
12.5.4	Example using custom functions declared in file .....	180
12.5.5	Example using Fast Edit variables .....	180
12.5.6	Example for calling a java class with return value .....	180
12.5.7	Example for calling a java class with arguments and return value .....	180
12.5.8	Example for dynamic variable resolving .....	181
12.5.9	Example for working with date widget variables .....	181
12.5.10	Example for getting the name and content of the current widget.....	181

12.5.11	Example for storing content into a file .....	182
12.5.12	Example for executing a SQL query .....	182
12.5.13	Example for executing a SQL select .....	182
12.5.14	Example for executing a SQL select and storing result into XML file .....	182
12.5.15	Example for executing a SQL query with parameters .....	183
12.5.16	Example for executing a command .....	183
12.5.17	Example for making a row read-only .....	183
12.5.18	Example for reading Cognos session parameters .....	184
12.5.19	Example for exporting all Business Case data to file .....	184
12.5.20	Example for exporting selected Business Case data to file .....	184
12.5.21	Example for running Email Business Case .....	185
<b>13</b>	<b>Business logic &amp; widget control in the web browser .....</b>	<b>186</b>
<b>13.1</b>	<b>JavaScript Selektor ID .....</b>	<b>187</b>
13.1.1	Structure of the ID .....	187
<b>13.2</b>	<b>Use in a Table Business Case .....</b>	<b>188</b>
13.2.1	Activating the feature .....	188
13.2.2	Available JavaScript methods .....	189
<b>13.3</b>	<b>Use in a Single Business Case .....</b>	<b>190</b>
13.3.1	Activating the Feature .....	190
13.3.2	Available JavaScript methods .....	190
<b>13.4</b>	<b>Read/Write Widget Values .....</b>	<b>191</b>
13.4.1	Reading widget values .....	191
13.4.2	Writing widget values .....	191
13.4.3	Example function .....	191
13.4.4	In detail .....	191
13.4.5	Use in Apparo .....	192
<b>13.5</b>	<b>Possibilities of a Checkbox .....</b>	<b>193</b>
<b>13.6</b>	<b>Disabling Row Selection Checkbox .....</b>	<b>194</b>
13.6.1	Application example .....	194
<b>13.7</b>	<b>Show and hide a widget .....</b>	<b>196</b>
<b>13.8</b>	<b>Possibilities of Lookup Widgets .....</b>	<b>197</b>
13.8.1	Lookup key values .....	197
13.8.2	Lookup output values (Label) .....	197
<b>13.9</b>	<b>Aggregate all values of a column in a table Business Case .....</b>	<b>198</b>
13.9.1	Example for sum over one column .....	198
<b>13.10</b>	<b>Use of variables .....</b>	<b>199</b>
<b>13.11</b>	<b>Using server side variables .....</b>	<b>200</b>
<b>13.12</b>	<b>Use of larger JavaScript programmes .....</b>	<b>202</b>
<b>13.13</b>	<b>Enter key for calling the JavaScript routine .....</b>	<b>202</b>
<b>13.14</b>	<b>Updating the entered numbers to match the existing number format .....</b>	<b>203</b>
<b>13.15</b>	<b>Example of a Table Business Case for planning .....</b>	<b>204</b>
<b>14</b>	<b>Business logic in a Cognos report .....</b>	<b>208</b>
<b>15</b>	<b>Primary Keys &amp; Not Null columns .....</b>	<b>209</b>
<b>16</b>	<b>Adding Business Cases into Portal, Cognos reports or Dashboards .....</b>	<b>210</b>
16.1	New entry in Cognos portal .....	210

16.2	Linking Business Case into a report.....	211
16.3	Adding a Business Case link.....	212
17	Usage of Cognos Prompts for filtering in Business Cases .....	215
18	Ouput of multiple Business Cases in one report including synchronizing .....	217
18.1	Usage of the JavaScript-Method .....	218
18.1.1	Usage in a Script-Variable .....	218
18.1.2	Usage in a Javascript-file.....	218
18.2	Usage of HTML hyperlink .....	218
19	Apparo database repository .....	219
20	About Apparo .....	220
21	Addendum .....	221
21.1	Java class for testing .....	221
21.2	Creating an encrypted password .....	222
21.3	DB-Session Handling .....	223
21.4	Usage of external web javascript frameworks like jQuery .....	224
21.5	Hidding of the top header of Business Case output .....	225

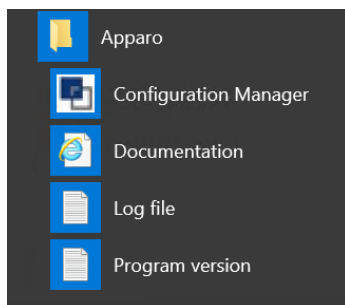
## 1 Premises

Apparo Fast Edit must be installed successfully and the Apparo Designer entry must be already created in the IBM Cognos portal.

It is advisable to first read the document

- **“Training Guide”**

**Windows:** You can find the complete documentation in Windows Start / Programs / Apparo / Documentation



**Unix/Linux:** See [APPARO\_HOME]/FastEdit/doc

## 2 Designing Business Cases

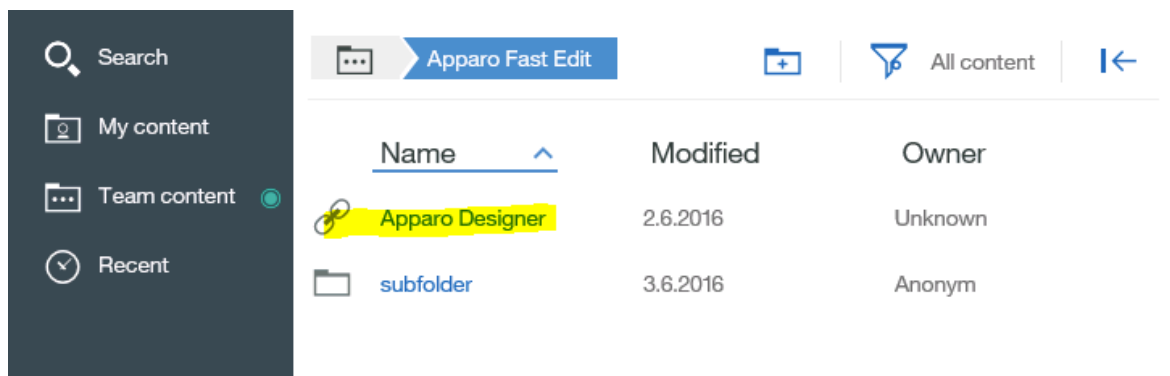
Open the Apparo Fast Edit Designer by clicking the **Apparo Designer entry** in the IBM Cognos portal.

The designer link will be defined by the Administrator during installation.

If you do not know where this link is located then ask the administrator who installed Apparo Fast Edit.

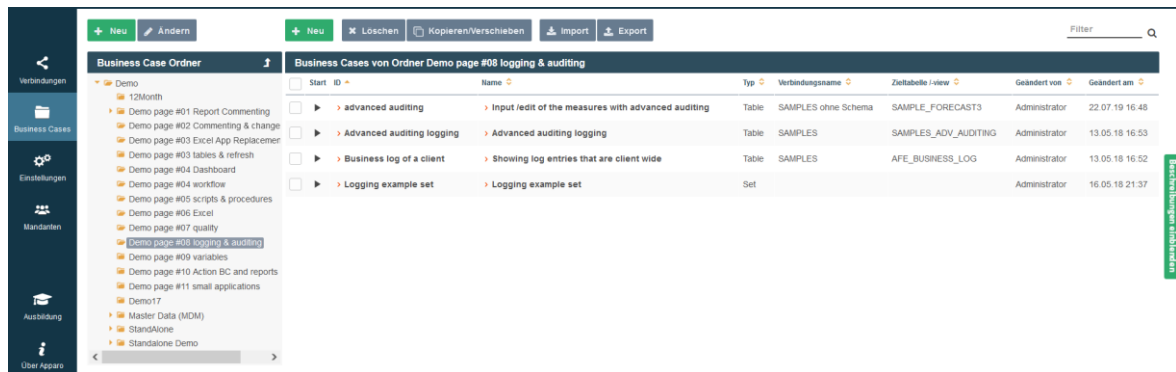
If you see the message “**Insufficient privileges**” then the current IBM Cognos user is not member of the group Apparo Designers (that’s the default name for this group).

Please inform your Cognos Administrator that he must add you into this group first or change the expected security group name in the Apparo Configuration Manager.





## 2.1 Start screen with a list of all Business Cases



At the start of the Apparo Designer, you will see a list of all business cases that are stored in the Apparo Repository. If the Apparo Repository does not contain any definition, this list will have no entries.

### 2.1.1 Definition

Each business case is an own application that can be called separately.

Business cases can be linked together, so that from the user's perspective, a business case can also consist of several masks.

All business cases are stored in the Apparo Repository, which is a separate database.

With **Close Designer** you can leave the Apparo Designer and turn back to the IBM Cognos portal.

### 2.1.2 Buttons and Sorting



The following buttons are at your disposal:

- New - creates a new Business Case
- Delete - deletes all selected Business Cases
- Copy - copies all selected Business Cases
- Import - imports Business Cases from a file
- Export - exports selected Business Cases into a file
- Filter - filters all business cases from the input string by its ID

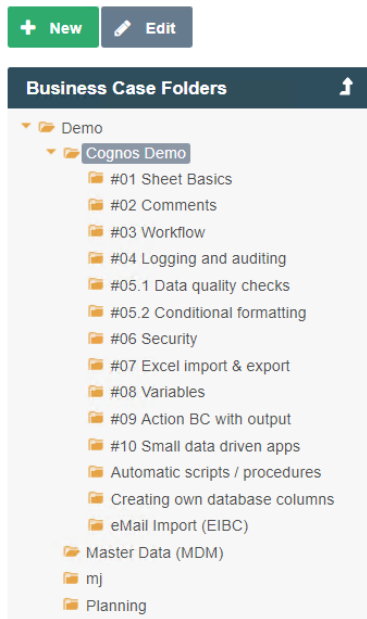
The sort can be changed by clicking the orange arrows:



### 2.1.3 Folder

Business cases can be grouped in folders. Inside the folder you can create subfolders.

For the Folders panel, there are three buttons (delete icon is hidden in this screenshot):



Depending on the given rights, the user can:

- Create new folders and subfolders
- Delete folder and its contents (subfolders, business cases)
- Change the properties of the folder

#### Folder properties:

The following properties can be changed:

- Name of the folder
- The necessary security group to open the folder
- The necessary security group to edit the folder
- The necessary security group to execute containing Business Cases
- The necessary security group to execute containing Business Cases with limited access (**read only**)

### 3 Description and normal mode

In the description mode most of the settings are explained briefly while the normal mode lacks these descriptions.

Example of the description mode:

Widget settings of database column OFFICE\_ID

Widget type Mapping & Other Features Visual Help texts Data output format

**Hiding**

☐ Hide this widget in the editing area  
If enabled then the user will not see this widget in the editing area. If you use a constant value then it will be used even if the widget is hidden.

☒ Hide this widget in the inserting area  
If enabled then the user will not see this widget in the inserting area. If you use a constant value then it will be used even if the widget is hidden.

☒ Hide this widget in edit and inserting area for all users  
The data field is to be used, but not shown in insert and editing area, optional security group based. That means this widget can be hidden for certain user groups only.

**Read-only**

☐ Read-only in edit and inserting area for all users  
The data field cannot be altered in editing and inserting area but it is still visible with another background colour, optional security group based.

☒ Read-only in edit area for all users  
The data field cannot be altered in editing area, optional security group based. Read-only widgets have an own background colour.

☐ Read-only in inserting area all users  
The data field cannot be altered in inserting area, optional security group based. Read-only widgets have an own background colour.

☒ Show content as a label and don't display the widget frame and read-only colour

The same page in normal mode:

Widget settings of database column OFFICE\_ID

Widget type Mapping & Other Features Visual Help texts Data output format

**Hiding**

☐ Hide this widget in the editing area

☒ Hide this widget in the inserting area

☒ Hide this widget in edit and inserting area for all users

**Read-only**

☐ Read-only in edit and inserting area for all users

☒ Read-only in edit area for all users

☐ Read-only in inserting area all users

☒ Show content as a label and don't display the widget frame and read-only colour

Show description

The Designer is switching the mode by clicking the button on the right side of the screen.

button on the

## 4 Database & E-Mail connections

### 4.1 Database connections

#### 4.1.1 Supported databases

Apparo Fast Edit can read data from the following databases:

- Oracle
- Oracle Client
- IBM DB/2
- IBM DB/2 Client
- IBM DB/2 i (iSeries, AS/400)
- IBM DB/2 z via IBM DB/2 Client
- MS SQL Server (optionally using Windows Authentication)
- MySQL
- Teradata from Version V2R6
- Exasol
- Informix
- SAP HANA
- SAP Sybase ASE and IQ
- Snowflake
- Greenplum
- PostgreSQL

As a technical access path JDBC type 4 is used or JDBC Type 2 for IBM DB/2 Client / Oracle Client.

#### 4.1.2 Buttons



The following buttons are at your disposal:

- |                       |   |
|-----------------------|---|
| • New                 | - creates a new database connection               |
| • Test DB-Connections | - is testing all selected DB connections          |
| • Import              | - imports DB connections from a file              |
| • Export              | - exports all selected DB connections into a file |

#### 4.1.3 Creating a new database connection

Click on the button 

##### 4.1.3.1 Settings of the tab ,Main‘

Main	Advanced	Variables	Security
Connection name	<input type="text"/> * <small>The database connection name is the name that will be used in all Business Cases.</small>		
Database type	<div> <div></div> <ul style="list-style-type: none"> <li>Exasol</li> <li>IBM DB2</li> <li>IBM DB2 Client</li> <li>IBM DB2 i</li> <li>IBM dashDB</li> <li>IBM Netezza</li> <li>Informix</li> <li>MS SQL Server 2008-2019</li> <li>Oracle</li> <li>Oracle (using service name)</li> <li>Oracle Client</li> <li>PostgreSQL</li> <li>SAP HANA</li> <li>SAP Sybase ASE</li> <li>SAP Sybase IQ</li> <li>SAP Sybase SQL Anywhere</li> <li>Teradata</li> <li>MySQL</li> </ul> </div>		
Database host	<input type="text"/> V * <small>Host name or host_name\instance_name for SQL Server named instance.</small>		
TCP/IP port	<input type="text" value="1433"/> V <small>The TCP/IP port for communication:</small> <ul style="list-style-type: none"> <li>• Oracle: default value 1521</li> <li>• MS SQL Server: default 1433 (Please note: If you want to use named instances then please see at SQL Server Configuration Manager / SQL Server Network Configuration / Protocols for INSTANCE NAME / TCP/IP / IPAddresses for the correct port number or let this field empty if the port number is changing dynamically)</li> <li>• IBM DB2: default: 6789 or 50000</li> </ul>		
Database name	<input type="text"/> V * <small>The database instance name (SID for Oracle) or database name (for MS SQL Server):</small>		
Windows authentication	<input type="checkbox"/> ? <small>In Windows authentication mode the current Windows user that is used for the Windows service of Apparo will be used for authentication for the MS SQL Server. MS SQL Server, the optional BI system and Apparo must be in the same Active Directory domain. In this mode, the database connection pooling will be automatically disabled. Please see the description text of the ?-icon for detail information about Windows authentication.</small>		

- Connection name - Freely selectable unique identifier for the connection
- Database type - select from list your database type
- Database host - IP address or host name of the database
- TCP/IP Port - listening port of the database
- Database name - Name of the DB
- Database user - user name of the login
- Password - password of the login

#### 4.1.3.2 Settings of the tab ‚Advanced‘

Main	Advanced	Variables	Security
Working schema	<input type="text"/> <span>V</span> The database schema that is used automatically. If left empty, the system will ask for it. In most cases, the schema name is expected in uppercase letters.		
Additional optional connect parameters	<input type="text"/> <span>V</span> You can add here additional parameters for the JDBC connection. Format depends on selected Database type. Use format required by database vendor. e.g. encrypt=true; trustServerCertificate=false; or ?serverTimezone=UTC These parameters are used for opening the database connection.		
SQL command	<input type="text"/> <span>V</span> This SQL command is executed directly after opening a database session and is helpful to define session settings like encrypting.		
Optimize concurrent access	<input type="checkbox"/> If enabled then Apparo is using additional techniques to prevent data overwriting between multiple parallel database user sessions. Using this feature is helpful if there are parallel user sessions that are working with the same data at same time. Before a data change is made an extra SQL select is made, to check if the data wasn't changed by other user in the meantime. If the data was changed in another session then the user is getting a warning message. This feature is working for non-primary key widgets only.		
Data row locking	<input type="checkbox"/> If enabled then Apparo is using row locking between multiple parallel user sessions. Using this feature is helpful if there are parallel user sessions that are working with the same data at same time. If an user is trying to update a row that is updated at same time in another user session then the user will see immediately an error message.  behavior for Oracle: The user is seeing all data rows but the locked data rows are not editable and they have another background colour. behavior for MS SQL Server: if a data row is locked by another session then the user is seeing an error message and not all data rows.  Without using this feature the Business Case is completely locked ("frozen") and the user can just close the window. The best solution in this case is to use auto-commit (see tab "Features"). Using this feature the lock possibility is decreased.		
Use connection pooling	<input type="checkbox"/> The connection pool is helpful for improving general performance. If opening a database connection need much time then it's helpful to generate a pool of connections automatically for Apparo and the database connections are reused again. If you are using Script Variables in the database connection then pooling is disabled automatically.		

OK
CANCEL

The following settings can be made in the tab advanced:

#### **Working schema**

If entered then this schema will be used only and it can't be changed at Business Case design time.

#### **SQL command**

This SQL command is executed directly after opening a database session and is helpful to define session settings like encrypting.

#### **Optimize concurrent access**

If enabled then Apparo Fast Edit is using additional techniques to prevent data overwriting between multiple parallel database user sessions. Using this feature is helpful if there are parallel user sessions that are working with the same data at same time.

Before a data change is made an extra SQL select is made, to check if the data wasn't changed by other user in the meantime. If the data was changed in another session then the user is getting a warning message.

#### **Use connection pooling**

The connection pool is helpful for improving general performance. If opening a database connection need much time then it's better to generate a pool of connections and the database connections will be re-used again.

If you are using Script Variables in the database connection then pooling is disabled automatically.

#### **Minimum pool size**

Any positive value can be used. If zero is used then the size of connection pool is unlimited.

#### **Maximum pool size**

Any positive value can be used. If zero is used then the size of connection pool is unlimited.

#### **Reconnect idle database session after (sec)**

If this is a number greater than 0, pooling system will test all idle connections in the pool, every this number of seconds. Setting a fairly long value (hours), is an excellent, high-performance approach.

The testing is done by executing a metadata select into the database, therefore low values may slow down the application performance.

#### **Discard idle database sessions after (sec)**

Seconds a connection can remain pooled but unused before being discarded.

Zero means idle connections never expire.

If this number greater than 0, pooling system will close and remove from pool all connections that are idle for this number of seconds.

Low values may slow the application performance. Normally this value should be set to several hours.

#### **Max idle time of excess connections (sec)**

Some users want their pools to release quickly unnecessary connections after a spike in usage that forced a large pool size.

You can achieve this by setting here a value much shorter than above, forcing connections over your set minimum size to be released if they sit idle for more than a short period of time.

#### **Database session increment**

This number must be greater than 1, determines how many connections at a time the pooling system tries to acquire when the pool is exhausted.

#### **Maximum number of total cached statements**

Defines the total number of prepared statements a database connection will cache.

The cache will destroy the least-recently-used prepared SQL statement when it hits this limit.

#### **Maximum number of cached statements**

Defines how many statements each pooled connection is allowed to own.

#### 4.1.3.3 Variables

Main
Advanced
Variables
Security

Variables are helpful for defining flexible database connections. Every Business Case that is using this database connection can use the variables of the database connection too. Script variables can be used in following properties:

- Database Host
- Database Name
- Database Port
- Database User
- Password

The connection pool will be automatically disabled on connections that use variables.

+ Add
✖ Delete

User defined variables

☐ Variable name
Variable type

No user defined variables found

Internal variables ready for use

Variable name	Variable description
<%CURRENT_DATE%>	Current date and time
<%TIME_MS%>	The number of milliseconds since 1.1.1970 (UNIX timestamp)
<%USER_NAME%>	Name of currently logged user
<%USER_LOGIN%>	Unique login name of currently logged user
<%SERVER_NAME%>	Name of application server where Apparo is running
<%LANGUAGE%>	Identifier of language in which user interface is displayed

OK
CANCEL

You can make database connections dynamically, using variables.

You have the ability to create your own JavaScript-based variables and have access to a selection of predefined variables.

#### 4.1.3.4 Automatic tables/columns creation

Main
Advanced
Variables
Automatic tables/columns creation
Security

Here you can define the type of this database connection:  
Type 1: It can read and write data only  
Type 2: It combines type 1 and 3 and is used to create DB-tables or DB-columns in the Designer. Needs a predefined type 3 connection.  
Type 3: It can create new database tables and columns.

[Click here to watch the video guide.](#)

☒ This database connection type is useable for creating/editing Business Cases (reading and writing of data in existing tables) only

☐ This database connection type combines both types and should be used if users shall be able to auto create tables/columns (needs a predefined connection for creating new tables/columns)

☐ This database connection type is used for creating new database tables/columns only (cannot access data in Business Cases)

OK
CANCEL

If for the current client this function is activated (see client settings), you will see these options.

A connection for creating new database columns or tables, it requires to have the CREATE right granted. This database connection will be no longer used during the runtime of Business Cases.

Basically there are three types of DB connections:

- Database connections for reading and writing data
- Database connections for creating tables and columns
- Database connection that combines both types



#### 4.1.3.5 Security

If not everybody should be able to use this database connection then it is possible to restrict access to specific designer users.

Add all security groups that shall be able to use these database connections.

If these settings remain empty, then everybody can use this database connection.

Main	Advanced	Variables	Automatic tables/columns creation	Security
<p>If not everybody should be able to use this database connection then it is possible to restrict access to specific designer users.            Add all security groups that must be able to use this database connections.            If this settings remains empty then everybody can use this database connection.</p> <p>Security groups <input type="text"/></p>				
OK		CANCEL		

## 4.2 Email connections

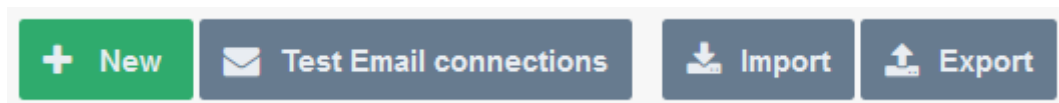
### 4.2.1 List of e-mail connections



### 4.2.2 Supported protocols

Incoming and outgoing e-mail connections use basically the POP or SMTP protocol.

### 4.2.3 Buttons



You can use the following buttons:

- New - creates a new e-mail connection
- Test Email connections - tests the selected email connection
- Import - imports e-mail connections from a file
- Export - exports e-mail connections into a file

## 4.2.4 Creating a new e-mail connection



Click on the button:

### 4.2.4.1 Configuration

Basic settings

Connection name

The email connection name is the name that will be used in all email Import Business Cases.

Email address

Only emails sent to this address will be processed and the address will be used for information messages sent to the original email sender.

Trusted email servers

Comma separated list of SMTP server domain names or IP addresses. If you don't leave this field blank, then only emails received from the specified will be processed. To know what server name you want to trust, send an email to this address and look at its source code. Values of 'Received' headers say what servers the email came from (top most value is the latest). The application will search this list of trusted servers for the latest non-localhost server from an email headers.

POP3 settings for fetching emails

Email server

Host name or IP address of the email server

Port

Port number of email server. It is usually different for secured (SSL) and not secured connections. Default for secured: 443, default for unsecured: 25

Use secured connection (SSL)

☐

When checked, secured connection (SSL) to email server will be established. The email server must be configured to support such connections.

User name

The login name of the email account.

Password

The password for given email account.

SMTP settings for sending emails

SMTP server

Host name or IP address of an SMTP server

SMTP Port

Port number of the SMTP server. There use to be different ports for secured (SSL) and plain connections.

Use secured connection (SSL) for SMTP

☐

If checked, secured connection (SSL) to the SMTP server will be used. The email server must be configured to support such connections.

SMTP user name

Authentication name for sending emails.

SMTP password

Password for sending emails.

### Basic settings

<b>Connection name</b>	The name that will be used in all E-mail Import Business Cases.
<b>Email address</b>	Only emails sent to this address will be processed and the address will be used for information messages sent back to the original email sender.
<b>Trusted email servers</b>	Comma separated list of SMTP server domain names or IP addresses. If you don't leave this field blank, then only emails received from the specified will be processed. To know what server name you want to trust, send an email to this address and look at its source code. Values of 'Received' headers say what servers the email came from (top most value is the latest). The application will search this list of trusted servers for the latest non-localhost server.

### POP3 settings for fetching emails

<b>Email server</b>	Host name or IP address of the email server
<b>Port</b>	Port the of e-mail server. It is usually different for secured (SSL) connections.
<b>Use secured connection</b>	When checked, secured connection (SSL) to email server will be established. The mail server must be configured to support such connections.
<b>User name</b>	The login name of the email account.

**Password**                      The password for given email account.

***SMTP settings for sending emails***

**SMTP server**                      Host name or IP address of an SMTP server

**SMTP Port**                      Port number of the SMTP server

**Use secured connection**      If checked, a secured connection (SSL) to the SMTP server will be used. The email server must be configured to support such connections.

**SMTP user name**                Authentication name for sending emails.

**SMTP password**                Password

## 5 Database transaction handling using OK/STORE/CLOSE/CANCEL buttons

All database changes are done in a database transaction.

You can define the transaction behaviour for a Business Case with following setting:

Please select all features that you want to use in this Business Case:

Feature Category	Feature	Status
Insert/delete/update/copy	Inserting of new data rows	✓
	Editing of data rows	✓
	Deleting of data rows	✓
	Copying of data rows	✓
Excel	Excel import	✓
	Excel export	✓
Data quality	Data row validator	✓
	Checking primary key	✓
Data change history	Auditing of data changes	✓
	Data history	✓
Actions and scripts	Widget data calculations	✓
	My own action buttons	✓
	Automatic scripts and database procedures	✓
Other	Refreshing data	✓
	My own database error messages	✓
	Filtering	✓
Data transaction handling	Data transaction handling	YES
	Auto-commit: Store all data changes immediately into the database	YES
Access control	Security	✓
	Limited access (readonly mode)	✓

OK CANCEL

If auto-commit is **enabled** then the data changes (inserts, update, deletes) are committed as soon as possible. That means database locks will disappear as soon as possible and the data is readable using different database session too. It is not possible to make a session rollback.

If auto-commit is **disabled** then there will be no automatic commit.

If the user wants to commit explicitly the data changes, then it is possible to enable a **STORE** button that is making a commit.

Main settings

- Standard buttons
- Filter data output
- Variables
- Inserting of new data rows
- Editing of data rows
- Excel import
- Excel export
- Data history

Button type	Button label	Enabled	Order
> OK	OK	✓	↓ ↑
> Store	Store	✓	↓ ↑
> Cancel	Cancel	✓	↓ ↑
> Close	Close	✓	↓ ↑
> Excel export	Export to Excel	✓	↓ ↑
> Import copy&paste	Excel Row-Import	✓	↓ ↑
> Help	Help	✓	↓ ↑

### 5.1 OK-Button behaviour without the CLOSE-Button

Using auto-commit:

With pressing the **OK**-button all data changes in the Business Cases are stored into the database table including commit. The Business Case is closed too.

Using **no** auto-commit:

With pressing the **OK**-button all data changes in the Business Cases are stored into the database table without commit. Without a **CLOSE**-Button the Business Case is making a commit and will be closed too.

## 5.2 OK-Button and CLOSE-Button behaviour

Using auto-commit:

With pressing the **OK**-button all data changes are stored in the database table including a **COMMIT**.

With pressing the **CLOSE**-button all data changes are stored in the database table with a following **COMMIT**. After that the Business Case is closed.

Using **no** auto-commit:

With pressing the **OK**-button all data changes are stored in the database table.

With pressing the **CLOSE**-buttons there is a **COMMIT**. After that the Business Case is closed.

## 5.3 Behaviour of the CANCEL-Buttons

Using auto-commit:

With pressing of the **CANCEL**-button the Business Case will be closed.

If there is no **CLOSE/OK**-button then the Business Case is closed after that.

Using **no** auto-commit:

With pressing of the **CANCEL**-button the database transaction is **roll backed**.

That means all changes are dropped and the old values are still in the database table.

If there is no **CLOSE/OK**-button then the Business Case is closed after that.

## 5.4 CLOSE-Button in an embedded Business Case

If a Business Case is **embedded** in a report or dashboard then the **CLOSE**-button is **hidden** automatically and the **CANCEL**-button is **not closing** the Business Case.

## 6 Table Business Cases (Table BC)

### 6.1 Definition

- In a table BC all records of the target table are displayed in the browser window.
- All individual elements on the form are so called widgets e.g. Input fields, check boxes, buttons, etc
- The navigation buttons can be used to scroll through the records page by page.
- This representation makes it possible to effectively carry out changes within a database table.

#### Example for a Table BC:

**Data Quality Demo**

The row validation checks (independently from the kind of data insertion:  
 #1 if Amount/Year is having a value, otherwise an error is raised  
 #2 if the sum of all quarters exceed the Amount/Year, otherwise an error is raised

Validation on widget Level is checking Amount/Year, must not exceed 10k

Product line	Product	Accept yn	Amount/Year	Quarter I	Quarter II	Quarter III	Quarter IV	Last changed by	date from	valid to	Last change from
<input type="checkbox"/> T-Shirts	T-Shirt Vienna	<input type="checkbox"/>	5100	1000	600	2000	1200	administrator	Jul 9, 2021 3:27:20 PM		09.07.2021
<input type="checkbox"/> T-Shirts	T-Shirt October	<input type="checkbox"/>	12000	1000	600	2000	1200	Anonymous	Jul 13, 2021 2:51:13 PM		13.07.2021
<input type="checkbox"/> T-Shirts	T-Shirt blue	<input type="checkbox"/>	600	111	111	111	111	Anonymous	Jul 13, 2021 2:51:14 PM		13.07.2021
<input type="checkbox"/> T-Shirts	T-Shirt Vienna3	<input type="checkbox"/>	1100	100	800	100	100	Anonymous	Jul 13, 2021 2:51:14 PM		13.07.2021
<input type="checkbox"/> T-Shirts	T-Shirt Moscow	<input type="checkbox"/>	1100	100	800	100	100	administrator	Jul 9, 2021 3:28:42 PM		09.07.2021
<input type="checkbox"/> T-Shirts	T-Shirt 69's	<input type="checkbox"/>	600	111	111	111	111	administrator	Jul 9, 2021 3:28:15 PM		09.07.2021

0 0 0 0 0

OK CANCEL CLOSE & SAVE DELETE INSERT EXPORT TO EXCEL EXCEL ROW-IMPORT FILE IMPORT EXECUTABLE BUTTON APPROVE ALL DATA FOR CONTROLLING

## 6.2 Areas of a Table Business Case

A Table Business Case consists of different (partially optional) areas

The screenshot displays the 'Apparo Fast Edit' application interface. At the top, the 'Application Header' includes the app name and user/role information. Below it, the 'Header area' contains a title and description. The 'Filter area' features search and filter controls. The 'Bulk update area' includes a dropdown for selecting update types. The 'Edit area' is a table for managing data, with columns for widget, product, status, and various metrics. The 'Insert area' is a row at the bottom of the table for adding new records. The 'Calculation area' shows a calculated value. The 'Navigation area' contains buttons for OK, CANCEL, CLOSE & SAVE, DELETE, INSERT, COPY, EXPORT TO EXCEL, EXCEL ROW IMPORT, and FILE IMPORT. The 'Button area' is a red button labeled 'BUTTON AREA'. The 'Footer area' contains a description of the footer.

Widget in edit area	Product	My status	Amount/Year	Quarter I	Quarter II	Quarter III	Quarter IV	Last changed by	Comment	Last change from
<input type="checkbox"/> T-Shirts	T-Shirt Vienna	Open	5100	1000	600	2000	1200	administrator		09.07.2021
<input type="checkbox"/> T-Shirts	T-Shirt October	Open	5100	1000	600	2000	1200	Anonymous		13.07.2021
<input type="checkbox"/> T-Shirts	T-Shirt blue	Open	600	111	111	111	111	Anonymous		13.07.2021
<input type="checkbox"/> T-Shirts	T-Shirt Vienna3	Open	1100	100	800	100	100	Anonymous		13.07.2021
<input type="checkbox"/> T-Shirts	T-Shirt Moscow	Open	1100	100	800	100	100	administrator		09.07.2021
<input type="checkbox"/> T-Shirts	T-Shirt 69's	Open	600	111	111	111	111	administrator		09.07.2021

### Application Header

- optional, contains application name, user name and client name

### Header area

- includes the title and description

### Filter area

- for example, contains filter widgets to filter the data output

### Bulk update area

- mass update panel

### Edit area

- to modify existing data

### Insert area

- for adding new records

### Calculation area

- used to display information, such as text or calculations of variables

### Navigation area

- includes page counter, navigation and buttons for resizing

### Button area

- contains buttons

### Footer area

- comparable to the header area



### 6.3 Create a new Business Case



Click on the button:

Now select the entry ,Table‘

Please select type of Business Case you want to create now

**Table**

A table Business Case is showing many data rows on the same page. The user can filter the data, edit, import from Excel, export to Excel and so on.

**Single**

A single Business Case is showing just one data row only.

**Set**

A grouping of multiple Business Cases (table/single) for more comfortable usage. You can define global filters that are filtering all Business Cases automatically too.

**Email Import**

Importing Excel data directly by email - send Excel sheets using email attachments and Apparo will import the Excel data directly into the database including file uploads. No web browser is necessary, just an email.

**Email**

An email Business Case is a definition of an email text including usage behavior and can be used in another Business Cases of type 'table' or 'single' only. In these Business Cases it is possible to define buttons that can use this email Business Case.

**Action**

Purpose of Action Business Case is to execute scripts or database procedures that can be called from a report/HTML page. Usage of AJAX and Javascript for automatically executing in the background is possible too.

SHOW DESCRIPTION

CANCEL

Business Case selection

Following, the general settings for the Business Case :

Please provide a unique short name (ID), a name and select the target table.

The description is optional and can contain declarations, release notes, or other information.

If multiple database connections are set up, this selection is automatically extended by the points 'database connection' and 'database schema'.

#### Main database Business Case settings

Identifier / Short name

bc short name \*

Business Case name

bc long name \*

Target database table/view

Select table \*

Notes

NEXT

CANCEL

Main settings

## 6.4 Business Case features

The features of a Business Case open automatically after creating a Business Case .

If the Business Case is opened later for editing again, you can open the feature selection with the button on

top right corner:



Please select all features that you want to use in this Business Case:

<b>Insert/delete/update/copy</b> <ul style="list-style-type: none"> <li>Inserting of new data rows ✓</li> <li>Editing of data rows ✓</li> <li>Deleting of data rows</li> <li>Bulk data update</li> <li>Copying of data rows</li> </ul>	<b>Actions and scripts</b> <ul style="list-style-type: none"> <li>Widget data calculations</li> <li>My own action buttons</li> <li>Automatic scripts and database procedures</li> </ul>
<b>Excel</b> <ul style="list-style-type: none"> <li>Excel import ✓</li> <li>Excel export ✓</li> </ul>	<b>Other</b> <ul style="list-style-type: none"> <li>Refreshing data</li> <li>My own database error messages</li> <li>Filtering</li> <li>Data transaction handling ✓</li> </ul>
<b>Data quality</b> <ul style="list-style-type: none"> <li>Data row validator</li> <li>Checking primary key</li> </ul>	<b>Access control</b> <ul style="list-style-type: none"> <li>Security</li> <li>Limited access (readonly mode)</li> </ul>
<b>Data change history</b> <ul style="list-style-type: none"> <li>Auditing of data changes</li> <li>Data history</li> </ul>	

OK CANCEL

The features are divided into seven sections. The various features can be enabled or disabled as needed. If a feature is activated, then the selection menu in the Business Case edit view will be extended accordingly.

The advantage of this activation is obvious, if the Business Case is opened for editing again after some time, then you can see with one look at the menu, which features are used in this Business Case.

#### 6.4.1 Features areas and features

- **Insert/delete/update/copy**
  - Inserting of new data rows
  - Deleting of data rows
  - Bulk data update
  - Copying of data rows
- **Excel**
  - Excel Import
  - Excel Export
- **Data quality**
  - Data row validator
  - Checking primary key
- **Data change history**
  - Auditing of data changes
  - Data history
- **Actions and scripts**
  - Widget data calculations
  - My own action buttons
  - Automatic scripts and database procedures
- **Other**
  - Reloading reports
  - My own database error messages
  - Filtering
  - Data transaction handling
- **Access control**
  - Security
  - Limited access

A detailed description of the functions can be found in the section Business Case features.

## 6.5 Edit view of the Business Case

The edit screen is divided into two areas:

**Menu bar**, the buttons on the controller and all activated functions as menu items contains.

Buttons:

- **Start** - saves all changes and starts the Business Case
- **Close** - saves all changes and closes the edit view
- **Save** - saves all changes
- **Cancel** - discards any unsaved changes and closes the Business Case

**Settings area**, contains settings for the various functions and optionally divided again into tabs.

The screenshot displays the APPARO edit view interface. At the top, there is a menu bar with buttons: Run, Close, Save, and Cancel. Below the menu bar is a settings area with a sidebar on the left containing various menu items like 'Main settings', 'Standard buttons', 'My own action buttons', 'Filter data output', 'Variables', 'Inserting of new data rows', 'Editing of data rows', 'Delete data rows manually', 'Bulk data update', 'Excel import', 'Excel export', 'Copying of data rows', 'Data row validation', 'Auditing of data changes', 'Security', and 'My own database error messages'. The main settings area has tabs: Target table, Header, Footer, Visual, Colours, Widgets, Row ordering, and Link into Portal. The 'Widgets' tab is currently selected. It contains three sections: 'Filtering widgets', 'Bulk update widgets', and 'Editing widgets'. Each section has a table with columns for Row, Column, Column name, Widget type, Title, and checkboxes for PK, RO, H, and NN. The 'Filtering widgets' table has one row with Column 1, Column name PRODUCT\_ID, Widget type Input field, and Title Filter area. The 'Bulk update widgets' table has one row with Column 1, Column name STATUS\_ID, Widget type Lookup dropdown (for all tables), and Title Bulk update widgets. The 'Editing widgets' table has 12 rows with various columns and widget types like Input field, Lookup dropdown, and Checkbox. A 'Show description' button is visible on the right side of the editing widgets table.

## 6.6 Business Cases Settings

### 6.6.1 Main settings

The main settings are divided into several tabs and are containing the settings of the data source and the optics of the Business Case. Above all, it contains the widget settings.

Widgets are the actual control and output elements of a Business Case. This can be a filter, input or selection fields, buttons and more.

Because of their importance are Widgets the first tab you see when you open the Main settings.

The screenshot shows the 'Main settings' window with the 'Widgets' tab selected. The interface includes a sidebar with various settings categories and a main table area with three sections: Filtering widgets, Bulk update widgets, and Editing widgets. Each section contains a table with columns for Row, Column, Column name, Widget type, Title, and checkboxes for PK, RO, H, and NN. A 'Show description' button is located on the right side of the main table area.

Row	Column	Column name	Widget type	Title	PK	RO	H	NN
1	1	PRODUCT_ID	Input field	Filter area				
1	STATUS_ID	Lookup dropdown (for all tables)	Bulk update widgets					
1	OFFICE_ID	Input field	Office					
2	PRODUCT_LINE_ID	Lookup dropdown (for all tables)	Widget in edit area					
3	PRODUCT_ID	Lookup dropdown (for all tables)	Product					
4	accept_yn	Checkbox	Accept yn					
5	SALES	Input field	Sales					
6	STATUS_ID	Lookup dropdown (for all tables)	My status					
7	STATE_REVISION_ID	Lookup dropdown (for all tables)	Revision status					
8	FORECAST	Input field	Amount/Year					
9	FORECAST2	Input field	Quarter I					
10	FORECAST3	Input field	Quarter II					
11	FORECAST4	Input field	Quarter III					
12	FORECAST5	Input field	Quarter IV					

Main settings, Widgets

## 6.6.2 Widgets

This chapter covers the central area of a Business Case.

Here you can have different widgets that are normally connected with the target table, positioned in different areas.

Each widget has its own individual settings.

## 6.6.3 Edit view

You can open the settings of an existing widget, by clicking on the column name or widget type:

Editing widgets							
<input type="checkbox"/>	Column	Column name	Widget type	Title	PK	RO	H NN
<input type="checkbox"/>	1	> OFFICE_ID	> Lookup dropdown (for all tables)	> Office	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/>	2	> PRODUCT_LINE_ID	> Lookup dropdown (for all tables)	> Product line	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Depending on the type of widget has the edit view different options, divided into tabs.

### Widget settings of database column OFFICE\_ID

Widget type

Mapping & Other

Features

Visual

Help texts

Data output format

☒ Input field  
☐ Text area  
☐ Checkbox  
☐ Simple dropdown (target table only)  
☐ Lookup dropdown (for all tables)  
☐ Multi select lookup  
☐ Label  
☐ Label with variables  
☐ Spacer & Title  
☐ Business Case link  
☐ File upload/download

Input field: Simple input field with one line.

Internal description

JavaScript selector ID

jsID\_E\_0\_0

OK

CANCEL

Show description

Widget edit view for the type ,Input field‘

### Widget settings of database column OFFICE\_ID

Widget type

Mapping & Other

Features

Lookup & Dropdown & Multiselect

Visual

Help texts

Data output format

☐ Input field  
☐ Text area  
☐ Checkbox  
☐ Simple dropdown (target table only)  
☒ Lookup dropdown (for all tables)  
☐ Multi select lookup  
☐ Label  
☐ Label with variables  
☐ Spacer & Title  
☐ Business Case link  
☐ File upload/download

Lookup Dropdown (for all tables): Combo-Box with values of another free definable lookup table. It is possible to filter, sorting etc.

Internal description

JavaScript selector ID

jsID\_E\_0\_0

OK

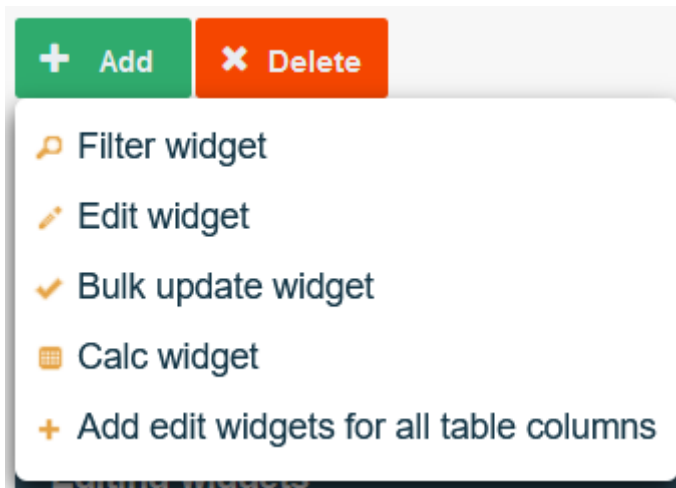
CANCEL

Show description

Widget edit view for the type ,Lookup dropdown (for all tables)‘

#### 6.6.4 Widget types and areas

When creating a new widget, you will first be asked for which area it is intended:



The option to add widgets for all database columns 'Add edit widgets for all table columns', is adding an 'input field' widget for each existing database column, if no widget is existing for this database column.

The corresponding areas for the widget types are:

- **Filter area**
- **Edit area**
- **Bulk update area**
- **Calculation area**

Based on the area for which the widget is thought, is there a selection of different widget types:

- Input field** - A standard entry field which allows the input of alphanumeric data
- Text area** - A multiline entry area that allows formatted text
- Checkbox** - Allows exactly two values, checked or not checked
- Simple dropdown** (for target table only) - Based on data in the target table
- Lookup dropdown** (for all tables) - Replaces numerical values with plain text from a 2nd table
- Simple multiselect** - Select multiple values
- Lookup multiselect** - Multiselect based on a lookup table
- Label** - Enables you to output text
- Label with variables** - Enables the output of text and values of variables
- Spacer & Title** - To set up void spaces between individual widgets
- Business Case Link** - To call e.g. detail BCs, data values are passed here
- File Upload/Download** - To attach files to data rows

Business Case Link and File Upload/Download can only be used in the edit area.

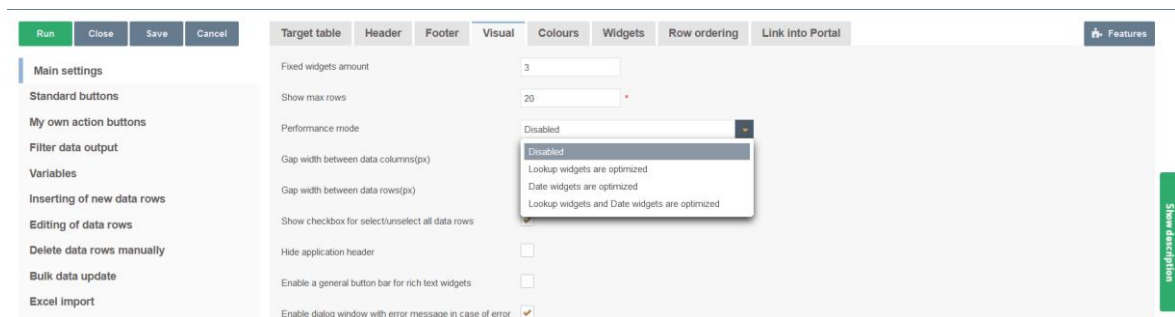
## Performance improvements necessary?

If your Business Case is using many lookup widgets and many data rows (20 data rows and more) in the editing area for a page then performance troubles can occur. That means the loading of the page need much time, the behavior of the browser is slow.

Solution:

See in tab "Visual" the setting "Performance mode".

If this mode is active then all lookup widgets in the editing area are displayed as labels with a pen icon. This output is much faster.





## 6.6.5 Widgets in the Edit Area

The edit area in a Table Business Case (TBC) is mainly used for displaying data in list form and gives users the ability to edit the data.

Editing widgets									
<input type="checkbox"/>	Column	Column name	Widget type	Title	PK	RO	H	NN	
<input type="checkbox"/>	1	> OFFICE_ID	> Input field	> Office	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	2	> PRODUCT_LINE_ID	> Lookup dropdown (for all tables)	> Widget in edit area	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	3	> PRODUCT_ID	> Lookup dropdown (for all tables)	> Product	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	4	> accept_yn	> Checkbox	> Accept yn	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	5	> SALES	> Input field	> Sales	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	6	> STATUS_ID	> Lookup dropdown (for all tables)	> My status	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	7	> STATE_REVISION_ID	> Lookup dropdown (for all tables)	> Revision status	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	8	> FORECAST	> Input field	> Amount/Year	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	9	> FORECAST2	> Input field	> Quarter I	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	10	> FORECAST3	> Input field	> Quarter II	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	11	> FORECAST4	> Input field	> Quarter III	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	12	> FORECAST5	> Input field	> Quarter IV	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	13	> FORECAST6	> Input field	> Plan6	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
<input type="checkbox"/>	14	> FORECAST7	> Input field	> Plan7	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	

Widgets of an area are grouped.

Example for edit widgets in the user view:

Header area with the headline and logos											
Description											
APPARO <sup>®</sup> Group											
Filter area											
SEARCH RESET FILTERS											
Bulk update widgets											
Open											
UPDATE											
Widget in edit area											
<input type="checkbox"/>	T-Shirts	T-Shirt Vienna	Open	▼	\$100	1000	600	2000	1200	administrator	09.07.2021
<input type="checkbox"/>	T-Shirts	T-Shirt October	Open	▼	\$100	1000	600	2000	1200	Anonymous	13.07.2021
<input type="checkbox"/>	T-Shirts	T-Shirt blue	Open	▼	600	111	111	111	111	Anonymous	13.07.2021
<input type="checkbox"/>	T-Shirts	T-Shirt Vienna3	Open	▼	1100	100	800	100	100	Anonymous	13.07.2021
<input type="checkbox"/>	T-Shirts	T-Shirt Moscow	Open	▼	1100	100	800	100	100	administrator	09.07.2021
<input type="checkbox"/>	T-Shirts	T-Shirt 69's	Open	▼	600	111	111	111	111	administrator	09.07.2021

In the edit area you can choose between these types of widgets:

- **Input field** - A standard entry field which allows the input of alphanumeric data
- **Text area** - A multiline entry area that allows formatted text
- **Checkbox** - Allows exactly two values, checked or not checked
- **Simple dropdown** (for target table only) - Based on data in the target table
- **Lookup dropdown** (for all tables) - Replaces numerical values with plain text from a 2nd table
- **Label** - Enables you to output text
- **Label with variables** - Enables the output of text and values of variables
- **Spacer & Title** - To set up void spaces between individual widgets
- **Business Case Link** - To call e.g. detail BCs, data values are passed here
- **File Upload/Download** - To attach files to data rows

### 6.6.6 Special functions in the widget settings

There are some special functions in the widget settings:

#### 6.6.6.1 Reading and writing expressions

Reading and writing expressions allow the usage of SQL to manipulate data before it is shown to users or stored to the database.

Widget settings of database column OFFICE\_ID

Widget type	Mapping & Other	Features	Visual	Help texts	Data output format
Column name	OFFICE_ID				
Enable expressions	<input checked="" type="checkbox"/>				
Read expression	<div></div>				
Write expression	<div></div>				
<input checked="" type="checkbox"/> The used read and write expressions having an inverse behaviour					

Show description

Variables are allowed here.

Common examples for expressions are:

- |         |                                      |
|---------|--------------------------------------|
| TRIM()  | - Removes spaces from strings        |
| UPPER() | - Turns all letters into upper cases |
| LOWER() | - Turns all letters into lower cases |

### 6.6.6.2 Conditional options

Many functions can be controlled with reference to conditions.

Thus, there are e.g. for the function 'Hidden', which hides a widget for the user when activated, several options.

Widget settings of database column OFFICE\_ID

Widget type Mapping & Other Features Visual Help texts Data output format

**Hiding**

☐ Hide this widget in the editing area

☒ Hide this widget in the inserting area

☒ Hide this widget in edit and inserting area for all users

**Read-only**

☐ Read-only in edit and inserting area for all users

☒ Read-only in edit area for all users

☐ Read-only in inserting area all users

☒ Show content as a label and don't display the widget frame and read-only colour

all users  
selected security groups  
specific value  
if variable returns true

Show description

#### For all users

This option is set by default. It hides the widget for all users.

#### For selected security groups

This hides the widget, but only for users who are member of one of the entered user groups. Other users can see the widget.

**Hiding**

☐ Hide this widget in the editing area

☐ Hide this widget in the inserting area

☒ Hide this widget in edit and inserting area for selected security groups > Specify settings for security group

Hide the widget for the selected user groups

#### Widget security groups definition - Hidden

**Existing security groups**

Security group

Group\_A

**Add new security group**

Security group

Group\_B ADD SECURITY GROUP

BACK TO WIDGET EDITOR

Security group editor

### For a specific value

The option applies here as soon as the value of one of the columns in the target table in the corresponding data row matches with the stored value.

In our example, the widget would be hidden once in a data row in the product line column the value '3' appears.

**Hiding**

☐ Hide this widget in the editing area

☐ Hide this widget in the inserting area

☒ Hide this widget in edit and inserting area for     has value

The values can also be configured dynamically by using variables. <%VARIABLE%>

## 6.6.7 Widget settings for the example ,input field'

Each widget type has its own settings. The following settings using the example of the widget type ,input field' can be found in almost all widget types. Distinctive features of each widget type are described in the following chapter ,More widget settings'

Widget settings of database column OFFICE\_ID

### 6.6.7.1 Mapping & Other

Widget settings of database column OFFICE\_ID

#### Column name

Here you can select the database column that is connected to this widget. The widget is reading the content of the column and is writing to this column. It is possible to use a variable in this column name too.

#### Default value

If you define a default value (using a variable is possible too) then the user will see this value in this input field in the inserting area. It is possible to define different default values for different user roles/groups. Use English format to define numeric or date values. Default value of lookup widget must be the lookup key value.

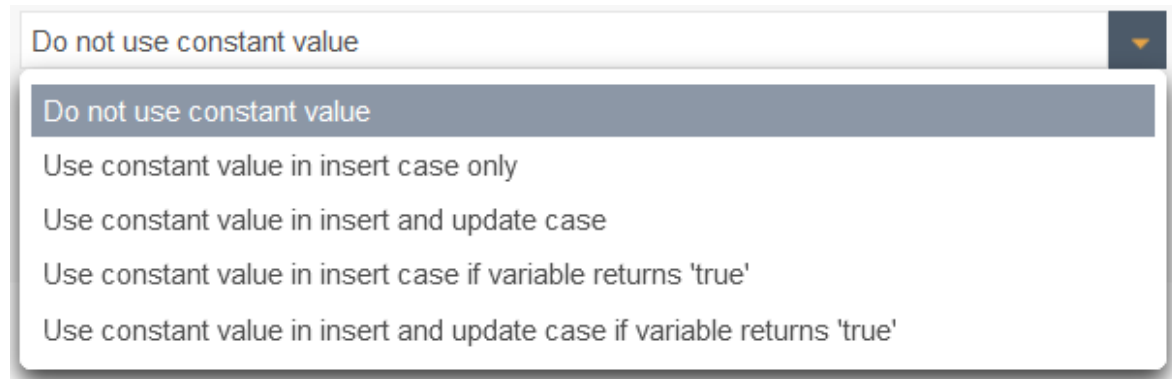
The function ,constant value' has the following options:

You can set different default values for different user groups.

### Constant value

A constant value is a value that will be always used for this widget. Even if the widget is hidden, read-only or if the user is inserting a value, then the constant value will be used. It is possible to define different constant values for different user roles/groups.

The function ,constant value' has the following options:



### Variable for using content in detail BC

If this Business Case has a widget of type 'Business Case Link' for opening a detail Business Case , it is possible to define a variable that contains the value of the current widget.

The detail Business Case can use this report variable with the current value of this widget for example for output.

Important: In the detail Business Case you must define this report variable in tab "Variables" too.

### 6.6.7.2 Features

In the tab Features you can control the behavior of the widget in detail.

Widget settings of database column OFFICE\_ID

Widget type Mapping & Other Features Visual Help texts Data output format

**Hiding**

☐ Hide this widget in the editing area

☒ Hide this widget in the inserting area

☒ Hide this widget in edit and inserting area for all users

**Read-only**

☐ Read-only in edit and inserting area for all users

☒ Read-only in edit area for all users

☐ Read-only in inserting area for all users

☒ Show content as a label and don't display the widget frame and read-only colour

**Other**

☐ Database column is the primary key or a part of it

☐ Database column is computed by database (for example using a database trigger or auto-increment feature)

☐ Show a small icon for easier deleting of the complete content of this widget

☐ Value is mandatory (not null)

☐ Remove all spaces at the begin and at the end automatically

☐ Hide output in password style

☐ Store value in upper case

☐ Store value in lower case

OK CANCEL

#### Hiding-Group

Includes options for hiding widgets.

**Hiding**

☐ Hide this widget in the editing area

☐ Hide this widget in the inserting area

☒ Hide this widget in edit and inserting area for all users

**Read-only**

all users  
selected security groups  
specific value  
if variable returns true

#### Hide this widget in the inserting area

If enabled, the user will not see this widget in the inserting area. If you use a constant value then it will be used no matter if the widget is hidden or not.

### Hide this widget in edit and inserting area for

The data field is to be used, but not shown in insert and editing area, optional security group based. That means this widget can be hidden for certain user groups only.

### Read only group

Includes options to disable the entering or changing of values in widgets

#### Read-only

☐ Read-only in edit and inserting area for

☒ Read-only in edit area for

☐ Read-only in inserting area

☒ Show content as a label and don't display the widget frame and read-only colour

### Read-only in edit and inserting area for

The data field cannot be altered in editing and inserting area but it is still visible with another background color, optional security group based.

Options:

☒

☒

☐

☒

### Read-only in edit area for

The data field cannot be altered in editing area, optional security group based. Read-only widgets have an own background color.

Options:

☒

☒

☐

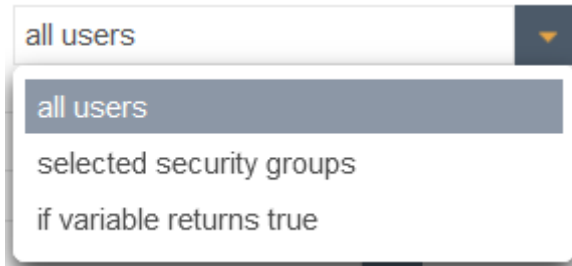
☒



**Read-only in inserting area**

The data field cannot be altered in inserting area, optional security group based.

Options:



## Other-Group

Contains all other settings

**Other**

☐ Database column is the primary key or a part of it
 ☐ Database column is computed by database (for example using a database trigger or auto-increment feature)
 ☐ Show a small icon for easier deleting of the complete content of this widget
 ☐ Value is mandatory (not null)
 ☐ Remove all spaces at the begin and at the end automatically
 ☐ Hide output in password style
 ☐ Store value in upper case
 ☐ Store value in lower case

### Database column is the primary key or a part of it

The widget is the primary key of the underlying table or is an part of the key (with combined keys). This definition is independent of the primary key definition in the database and at least one column must be defined as primary key. A primary key is identifying an unique data row of the target table/view.

### Database column is computed by database (for example using a database trigger or auto-increment feature)

The database table column value is filled automatically by the database (e.g. with triggers, auto-increment field). Apparo Fast Edit is not changing this value in the target table.

### Show a small icon for easier deleting of the complete content of this widget

Showing a small delete icon for deleting the widget content.

### Value is mandatory (not null)

If a widget value is mandatory, the user must enter a value in this widget (or use a default or constant value). The definition of this behavior is independent of the definition of the target table column in the database.

If a filtering widget is mandatory, it is a good idea to define a default value for it as well. You will avoid some error messages when starting the business case.

### Remove all spaces at the begin and at the end automatically

If enabled then all spaces at the begin and end are removed automatically before storing into database table

### Hide output in password style

Entered characters are shown as dots only

### Store value in upper case

If enabled then all characters are changed to upper case before storing into database table

### Store value in lower case

If enabled then all characters are changed to lower case before storing into database table

### 6.6.7.3 Visual

In the tab ,Visual' you will find the header (column heading), and settings for the layout, and settings to limit the maximum allowed input length in this widget.

By default the maximum entry length is defined by the database column definition, for example Varchar(20) allows a maximum of 20 alphanumeric characters. This can be further limited by the input of an own value.

Widget settings of database column OFFICE\_ID

Widget type Mapping & Other Features **Visual** Help texts Data output format

Column label

Language	Column label	
Spanish		✕
German	Filiale	✕
English	Office	✕

Visual settings

Label style

Font face	Size	Style	Align	Colour
Arial	11	bold	Left	#000000

Background colour for label

No background colour is defined

Widget align

Left

Column width (px)

200

Border colour

#000000

Background colour

No background colour is defined

Font

Font face	Size	Style	Align	Colour
Arial	12	normal	Left	#000000

Maximum input length

Defined by database column definition

Show data preview

The layout can also be controlled by CSS.

To overwright global visual settings of the client settings, the lock symbol must be closed.

#### 6.6.7.4 Visual help texts

Contains the settings for description and hint text

### Widget settings of database column OFFICE\_ID

Widget type	Mapping & Other	Features	Visual	Help texts	Data output format								
<b>Description text</b> <table border="1"> <thead> <tr> <th>Language</th> <th>Description text</th> </tr> </thead> <tbody> <tr> <td>Spanish</td> <td><input type="text"/></td> </tr> <tr> <td>German</td> <td>Üblicherweise werden solche Werte versteckt, zu Demozwecken sichtbar</td> </tr> <tr> <td>English</td> <td>Normally hidden, for demonstrating purposes in this case visible</td> </tr> </tbody> </table>						Language	Description text	Spanish	<input type="text"/>	German	Üblicherweise werden solche Werte versteckt, zu Demozwecken sichtbar	English	Normally hidden, for demonstrating purposes in this case visible
Language	Description text												
Spanish	<input type="text"/>												
German	Üblicherweise werden solche Werte versteckt, zu Demozwecken sichtbar												
English	Normally hidden, for demonstrating purposes in this case visible												
<b>Hint text</b> <table border="1"> <thead> <tr> <th>Language</th> <th>Hint text</th> </tr> </thead> <tbody> <tr> <td>Spanish</td> <td><input type="text"/></td> </tr> <tr> <td>German</td> <td><input type="text"/></td> </tr> <tr> <td>English</td> <td><input type="text"/></td> </tr> </tbody> </table>						Language	Hint text	Spanish	<input type="text"/>	German	<input type="text"/>	English	<input type="text"/>
Language	Hint text												
Spanish	<input type="text"/>												
German	<input type="text"/>												
English	<input type="text"/>												

#### Description text

This text can describe the widget and can be helpful for the user. You can add a more detail description text for each installed language.

The user is seeing this text if he is pointing to the label of this widget.

#### Hint text

The hint text is displayed only if the widget has no value.

Is displayed in the input area in gray text, e.g. 'Enter date in the format: dd.MM.yy'

#### 6.6.7.5 Data output format

Under data output format you will find several options for testing the validity of the data. The default setting 'Use type of output column data' and provides no further adjustments. With this option, the definition in the database of the associated database column determines which data type is used.

Widget settings of database column OFFICE\_ID

Widget type Mapping & Other Features Visual Help texts Data output format

Output type

- Number
- Currency
- Percentage
- Date / Time
- Text
- Use type of output column

OK CANCEL

#### Output types:

- **Number** - Requires a number
- **Currency** - Shows number values with currency symbol
- **Percentage** - Percentages, e.g. 12,34%
- **Date / Time** - Requires a date / time
- **Text** - To enter text, as a special validation option, there are regular expressions

Widget settings of database column OFFICE\_ID

Widget type Mapping & Other Features Visual Help texts Data output format

Output type

- Number
- Currency
- Percentage
- Date / Time
- Text
- Use type of output column

Decimal places

2

Show separate groups

How to show negative number

- with minus sign
- with minus sign and in red colour

Data quality check

Custom validator Java 8 class

Interval of old value (%)

Minimum allowed:

Maximum allowed:

Interval

Minimum allowed:

Maximum allowed:

OK CANCEL

- Decimal places** - You can set the number of decimal places displayed
- Show separate groups** - Serves for better readability of large numbers e.g. 1,000,000,000
- How to show negative number** - Negative numbers can only be viewed by a minus or colored red

#### Output type 'Currency'

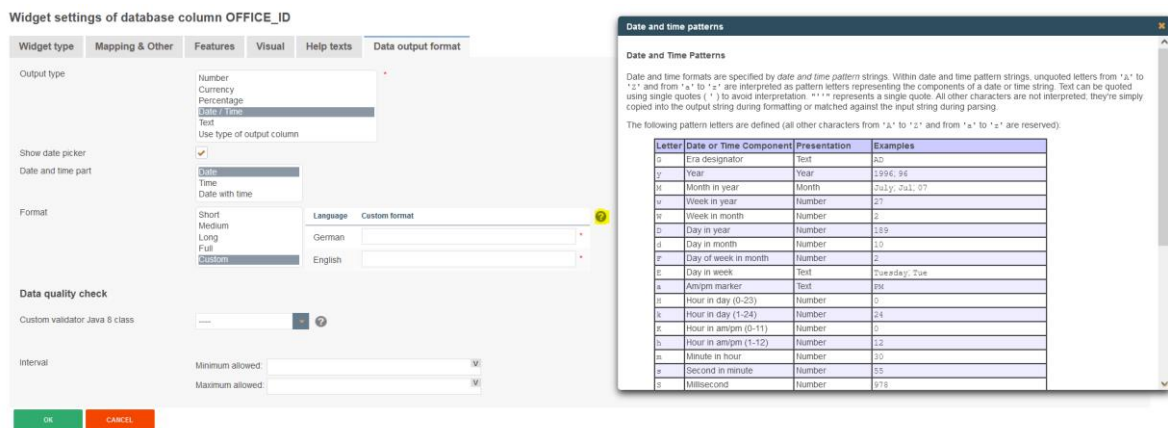
Is identical to the output type 'number', but contains as a further option the setting for a currency symbol

Currency symbol

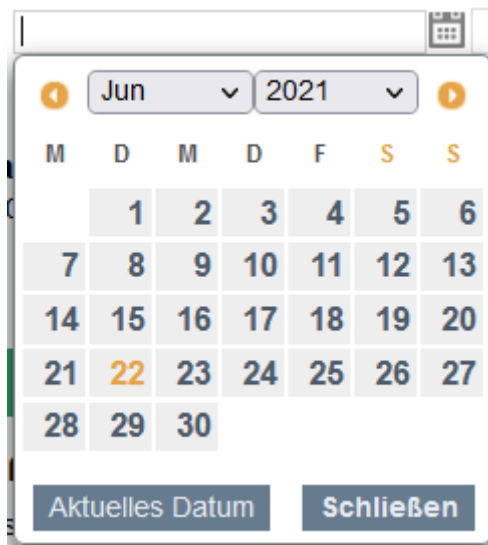
€

## Output type ,Date and Time'

The grey question mark hides the help for custom date/time patterns.



With ,Show date picker' (default) users can easily pick a date.





The date picker prevents entry mistakes.

#### 6.6.7.5.1 Data Quality check

##### Custom validator Java 8 class

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.

##### Interval of old value (%)

Interval of old value (%)

Minimum allowed:

Maximum allowed:

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

Interval


Minimum allowed:

Maximum allowed:

Limits the validity of entries based on an absolute interval. Permissible are values only from 1000 to 2000. Interval limits can be set dynamically with variables.

##### Regular Expression (Only for type ,Text')

Regular expression for data quality



Using a regular expression is useful to define more complex input rules. For example, you can specify that the first character must be an 'A' and then only numbers are allowed. Click the '?' icon to see the detailed instructions.

Characters		
Character	Description	Example
Any character except [ <code>\^\$. ?*+()</code> ]	All characters except the listed special characters match a single instance of themselves.	<code>a</code> matches <code>a</code>
<code>\</code> (backslash) followed by any of [ <code>\^\$. ?*+()</code> ]	A backslash escapes special characters to suppress their special meaning.	<code>\+</code> matches <code>+</code>
<code>\xFF</code> 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.	<code>\xA9</code> matches © when using the Latin-1 code page.
<code>\n</code> , <code>\r</code> , and <code>\t</code>	Match an LF character, CR character and a tab character respectively. Can be	<code>\n\n</code> matches a

## 6.6.8 Special settings other widget types

### 6.6.8.1 Widget type Text area

For the type text area you will find an extra block of settings in the tab 'Visual'

**Rich text edit**

☒ Use HTML tags for flexible text design like bold, underline, etc.

☐ Show just bold, italic, underline and colour icons in HTML editor

Store as plain text into column COMMENT\_TEXT

#### Use HTML tags for flexible text design like bold, underline, etc.

With this feature the user can change the text style (for example bold, italic, colors underline,...).

In this case the input text is stored including HTML tags.

#### Show just bold, italic, underline and color icons in HTML editor

If enabled then just the most important buttons for text style are displayed

#### Store as plain text into column

Optional it is possible to save the input text without HTML tags in another column of the same target table.

### 6.6.8.2 Widget type Checkbox

The additional options for this widget type, you will find in the tab 'Other':

- Value, if the checkbox is activated
- Value, if the checkbox is deactivated

Apparo Fast Edit Business Case Designer - BC\_Areas / Editing of data incl. quality...

**Widget settings of database column**

Widget type Mapping & Other Features Visual Help texts Data output format

Column name

Enable expressions ☐

Default value  for all users

If you define a default value (using a variable is possible too) then the user will see this value in this input field in the inserting area. It is possible to define different default values for different user roles/groups. Default value of lookup widget must be the lookup key value.

- Use English format to define numeric values (e.g. 1000.13)
- For date use format YYYY-MM-DD e.g. 2020-12-31
- For timestamp use format YYYY-MM-DD Hours:Minutes:Seconds e.g. 2020-12-31 23:59:59

Constant value  Do not use constant value

Checked value  1

Unchecked value  0

Variable for using content in detail BC

OK CANCEL



### 6.6.8.3 Widget type Simple dropdown (target table only)

A simple Combo-Box with values. The values are loaded from the target table only. It is possible to filter and to sort etc.

Widget settings of database column OFFICE\_ID

Widget type Mapping & Other Features **Lookup & Dropdown & Multiselect** Visual Help texts Data output format

Multivalue 'Order By' Ascending

SQL where condition

Maximum number of options to display 1000 (0 = unlimited)

OK CANCEL

SYNTAX CHECK

#### Tab Lookup & Dropdown & Multiselect

##### Multivalue 'Order By'

The widget will display a list of values. With this setting the sorting order can be defined.

Items in multivalue will be sorted by:

- None - no sorting for values (use default sorting order taken from database)
- Ascending - Ascending Value Sorting
- Descending - Descending Value Sorting

##### SQL where condition

You can filter the output of this widget using this setting. Usage of variables is possible. It is possible to filter the values of a lookup widget depending on the value of another widget.

##### Maximum number of options to display

This option determines number of dropdown options to be displayed.  
0 = unlimited.

#### 6.6.8.4 Lookup dropdown

The lookup dropdown behaves identically to 'simple dropdown', but the plaintext comes from another table

The lookup dropdown, compared with the simple dropdown provides other options:

##### Database connection

With the database connection you can define how the Lookup Dropdown (for all tables) can read the selectable values. All database connections are stored in "Database Connections".

##### Lookup table

The lookup table is a database table that has a mapping, e.g. 1=white, 2=blue, 3=yellow. In this case the number is the lookup key and the colour name is the lookup value. All possible and selectable values are stored in this database table/view.

##### Lookup table key column for comparing

The key column of the lookup table will be stored in the target table. This key column will be compared with the "Column Name" (see Mapping & Other). Be sure that both have the same data type. If you must compare both columns in a more flexible way then you can use the "Reading Expression"

##### Lookup table value column for output

The value column of the lookup table is used for displaying only. It will be not stored in the target table. If you want to change the output then it is possible to use "Reading Expression". Variables, like <%LANGUAGE%> are allowed. This would output the Label text in the users language by automatically choosing the right column (e.g. PRODUCT\_LINE\_NAME\_EN or PRODUCT\_LINE\_NAME\_DE)

##### Multivalue 'Order By'

The widget displays a list of values. With this setting, the sorting order can be set. Elements in the widget are sorted:

- None - No sorting for values (use default sort order from database).
- Ascending - Ascending value sort order
- Descending - Descending value sorting

### **Lookup table sort column**

Normally the output is sorted alphabetically, but it is also possible to use another lookup column for sorting.

### **SQL where condition**

With this setting you can filter the output of this widget. The use of variables is possible. It is also possible to filter the values of a lookup widget depending on the value of another widget.

### **Security group based filtering**

Allows different SQL where conditions for different user groups

### **Accept the Lookup Key Value as import value too**

Lookups accept both values: lookup key value and lookup output value

### **Make values distinct**

Make duplicate values unique:

If there are many output values with different filter values, all filter values are used for filtering when the user selects the unique output designation.

### **Maximum number of options to display**

This option determines the number of drop-down values to be displayed. For all values, use 0.

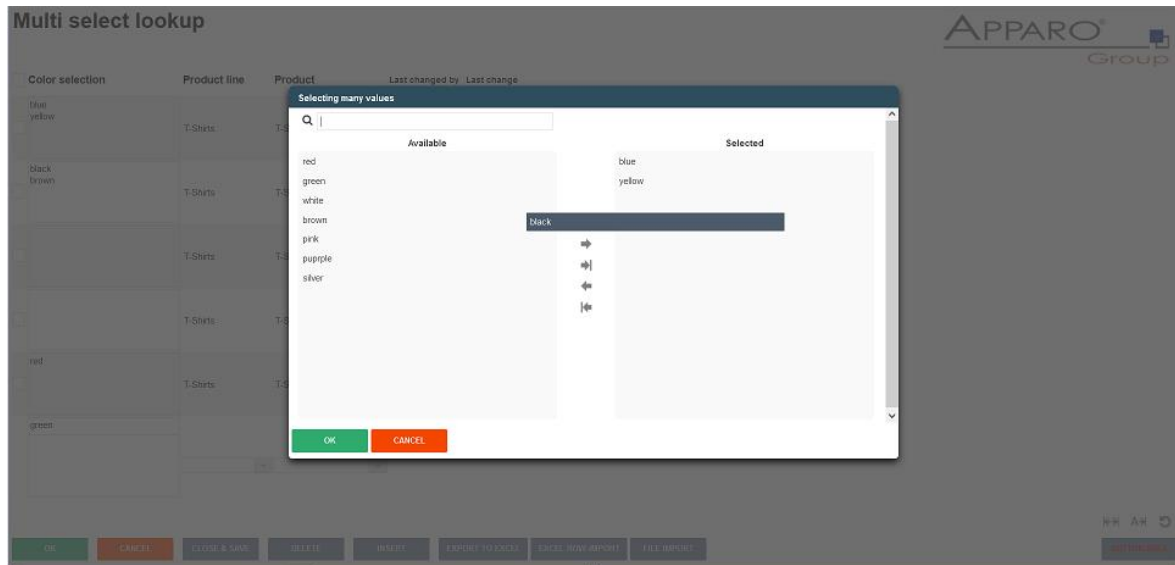
### **Show an already selected value which isn't in the selectable list of values**

Display a value that has already been selected, even if it is not in the current list of values (due to filter settings).

#### 6.6.8.5 Widget type “Multi select lookup”

This widget type exists only in the edit area.

You can select many values which are stored in a detail database table.



This widget does not need an own column in the target table of the Business Case.

The lookup values are stored in the lookup table, while the used values are stored in a third table.

### 6.6.8.5.1 Tables & Mapping

#### Widget settings

Widget type

Tables & Mapping

Default & Constant Values

Widget behaviour

Visual

Help texts

Database connection settings

Database connection

SAMPLES

With the database connection you can define how the Lookup Dropdown (for all tables) can read the selectable values. All database connections are stored in "Database Connections".

Lookup source table

Source lookup table name

SAMPLE\_CAR\_COLOR

The source lookup table contains all possible values that can be selected

Source lookup table key-column

PRODUCT\_COLOR\_ID

The IDkey column that identify a value

Source lookup table value-column

PRODUCT\_COLOR\_NAME

The column that contains the value that must be displayed

Source lookup table description column

The column that contains an optional description text

Source lookup table value sorting-column

PRODUCT\_COLOR\_ID

The column that is used for ordering the list of values

Multivalue "Order By"

Ascending

The widget will display a list of values. With this setting the sorting order can be defined. Items in multivalue will be sorted

None - no sorting for values (use default sorting order taken from database)

Ascending - Ascending value Sorting

Descending - Descending value Sorting

SQL where condition

SYNTAX CHECK

Security group based filtering

Lookup target table

Lookup table

SAMPLE\_CAR\_COLOR\_STORING

All selected values of this multi select widget are stored into this lookup target table. The lookup target table contains for each row the selected value (or key) and the primary key of the target table (that is the master table).

Join lookup table key(s) to target table

PK columns of this DC

Mapped column of target lookup table

PRODUCT\_LINE\_ID

PRODUCT\_LINE\_ID

PRODUCT\_ID

PRODUCT\_ID

Foreign key to lookup source table

COLOUR\_ID

Please select the column of the Lookup target table that must contain the key or value of the selected value. For each selected value a new data row will be stored.

OK

CANCEL

Show description

Show description

### 6.6.8.5.2 Database connection settings

#### Database connection

You can freely specify any database connection that will be used for this widget.

### 6.6.8.5.3 Lookup source table

Is used to translate stored IDs into clear text output.

A color lookup table could look this way:

ID	Text
1	green
2	red
3	silver

#### Settings

##### Source lookup table name

The source lookup table contains all possible values that can be selected

#### Source lookup table key-column

The ID/key column that identify a value

#### Source lookup table value-column

The column that contains the value that must be displayed

#### Source lookup table description column

The column that contains an optional description text

#### Source lookup table value sorting-column

The column that is used for ordering the list of values

#### Multivalue 'Order By'

The widget will display a list of values. With this setting the sorting order can be defined.

#### Items in multivalue will be sorted

- None - no sorting for values (use default sorting order taken from database)
- Ascending - Ascending Value Sorting
- Descending - Descending Value Sorting

#### SQL where condition

An optional filter for lookup values

#### 6.6.8.5.4 Lookup target table

This table stores the set values and could look like this:

Product line id	Product id	Colour id
2,00	60,00	4,00
2,00	60,00	2,00
2,00	70,00	6,00
2,00	70,00	7,00
2,00	440,00	1,00
2,00	480,00	10,00
2,00	480,00	8,00
2,00	480,00	3,00
2,00	480,00	9,00

#### Settings

##### Lookup table

All selected values of this multi select widget are stored into this lookup target table.

The lookup target table contains for each row: the selected value (or key) and the primary key of the target table (that is the master table).

## Join lookup table key(s) to target table

PK columns of this BC

## Foreign key to lookup source table

This is the ID of the lookup table and will be stored along with the primary keys of the target table.  
For each selected value a new data row will be stored.

### 6.6.8.5.5 Widget behavior

Contains special settings only for this widget type.

**Value filtering**

Max size of result set	<input type="text" value="1000"/>	<small>The max amount of the visible result set. This setting is helpful if you are working with a large list of values.</small>
Make values distinct	<input checked="" type="checkbox"/>	<small>Making double values distinct. If there are many output values with different filter values then all filter values are used for filtering if the user is selecting the unique output label.</small>
List of values is very large, use own dialog	<input checked="" type="checkbox"/>	<small>If you have a large list of values then it is possible to use an own window for better overview.</small>
User can filter values	<input checked="" type="checkbox"/>	<small>If enabled then the user is seeing a filter possibility. This is helpful if there are many values.</small>
Minimum text length to filter values	<input type="text" value="2"/>	
Load data automatically when dialog is shown	<input checked="" type="checkbox"/>	<small>This feature is helpful if your list of values is very large</small>
Show checkboxes for easier selection	<input type="checkbox"/>	<small>Show checkboxes for easier selection</small>
Dialog window width (px)	<input type="text" value="600"/>	
Dialog window height (px)	<input type="text" value="400"/>	

## Value filtering

### Max size of result set

The max amount of the visible result set. This setting is helpful if you are working with a large list of values.

### Make values distinct

Making double values distinct:

If there are many output values with different filter values then all filter values are used for filtering if the user is selecting the unique output label.

### List of values is very large, use own dialog

If you have a large list of values then it is possible to use an own window for better overview.

### User can filter values

If enabled then the user is seeing a filter possibility. This is helpful if there are many values.

### Minimum text length to filter values

### Load data automatically when dialog is shown

This feature is helpful if your list of values is very large

### Show checkboxes for easier selection

### Dialog window width (px)

### Dialog window height (px)

#### 6.6.8.6 Widget type ,Simple & Lookup multiselect'

This widget type exists only in the filter area.  
User can select multiple values simultaneously.

Usually, the settings are identical to those of the widget ,dropdown'

*The specific settings for this widget types can be found in ,Visual'*

Number of visible rows  \*

#### Visual Settings

##### Number of visible rows

Here you can set the number of displayed choices that appear without scroll bar. The default is a widget size of 8 lines.

*And in Lookup & Dropdown & Multiselect*

Show only choices used in target table ☐  
If enabled then only values that are used in target table will be shown as selectable dropdown options.

##### Show only choices used in target table

If enabled then only values that are used in target table will be shown as selectable dropdown options.



### 6.6.8.7 Widget type 'Label with variables'

This type of widget provides no direct way to assign a database column.

Widget settings

Widget type Mapping & Other Features Visual Help texts Data output format

Label value

Label text: <%city%>

Hide value of this widget if used variable is empty ☐

OK CANCEL

#### Special options for label with variables:

##### Label value

Usually contains Text and variables. HTML and all variables can be used in this field.

##### Hide value of this widget if used variable is empty

If one variable does not return a value, then the output is completely blocked

#### 6.6.8.8 Widget type Business Case Link

This type of widget is used to link multiple Business Cases. With a hyperlink in each row, you can e.g. display details in another Business Case.

### Mapping & Other

Widget settings

Widget type Mapping & Other Features Visual Help texts

Select the detail Business Case that must be linked to the current master Business Case: BC\_Areas2

Primary key mapping

All widgets with enabled 'primary key' property of the detail Business Case	Widgets of master Business Case
PRODUCT_ID	PRODUCT_ID

OK CANCEL

In 'Mapping & Other' you can choose a Business Case and assign a primary key. The primary key mapping is used to filter the called Business Case. Not assigned data is displayed unfiltered.

### Features

Other

☐ Display the detail Business Case in the same window like the current Business Case

In Features you will find another option. The default is to open the called Business Cases in a separate browser window. If this function is enabled, the called Business Case opened in the same browser window. If the called Business Case is closed, then the calling Business Case is opened again.

### Visual

Visual settings

Label style

Font face Arial Size 11 Style Bold Align Left Colour #000000

Background colour for label

No background colour is defined

Add

Widget align

Left

Column width (px)

200

Font

Font face Arial Size 12 Style Normal Align Left Colour #000000

Window title

Language	Window title name
German	Anderungshistorie
English	History of changes

Hyperlink title

Language	Link name
German	Anderungshistorie
English	History of changes

Detail window width (px)

Math.floor(screen.width \* 0.75)

Detail window height (px)

Math.floor(screen.height \* 0.75)

Detail window left (px)

Math.floor(screen.width \* 0.125)

Detail window top (px)

Math.floor(screen.height \* 0.125)

In addition to the general options there are further options for this widget type in tab 'Visual'

**Window title**

All variables can be used in this input field  
Contains the window title name for all defined languages.

**Hyperlink title**

All variables can be used in this input field  
Contains the name of the shown hyperlink

**Detail window width (px) & Detail window height (px)**

This property is used to setup the size of the browser window, which shows the called Business Case .  
You can use JavaScript language to resolve window parameter.  
If you use number, quote it with single quote mark e.g. '540'

**Detail window left (px) & Detail window top (px)**

Defines the location of the browser window (in pixel) from the left or the top of the screen.

### 6.6.8.9 Widget type File Upload/Download

This type of widget is used to attach files to data rows.  
It is possible to execute scripts and forwarding the file to an existing DMS.

Special Variables of this type are:

<%UPLOADED_FILE_NAME%>	Name of the uploaded file
<%DISPLAY_FILE_NAME%>	Name of file as displayed
<%DELETED_FILE_NAME%>	Name of the deleted file

**There are 2 ways to store files:**

File storage type                      This can be selected with this radio button (Database or File system)

**Database**                      -                      **Files will be stored in the database.**  
(Supported: Oracle, IBM DB2, IBM dashDB, MS SQL Server, PostgreSQL, MySQL and and SAP HANA)

The unique setting for storing in the DB is:

#### Column for storing file content

Database column where file will be stored. Supported types of the column are:

- ORACLE: BLOB
- IBM DB2: BLOB
- MS SQL SERVER: BINARY, VARBINARY
- PostgreSQL: BYTEA

**File system**                      -                      **Files will be stored on the server filesystem:**

#### Mapping & Other

Widget settings of database column OFFICE\_ID

Widget type	Mapping & Other	Features	Actions	Visual	Help texts
<p>File storage type</p> <p><input checked="" type="radio"/> File system    - Files will be stored on the server filesystem</p> <p><input type="radio"/> Database    - Files will be stored in the database. (Supported databases: Oracle, IBM DB2, IBM dashDB, MS SQL Server, PostgreSQL, MySQL and and SAP HANA)</p>					
<p>Column for storing file name</p> <p>OFFICE_ID</p>					
<p>Enable expressions</p> <p><input type="checkbox"/></p>					
<p>Column for storing display name</p> <p>Here you can specify the database column where the display name will be stored. If not specified then display name will be the same as the real file name.</p>					
<p>File name template</p> <p>&lt;%UPLOADED_FILE_NAME%&gt;</p> <p>The name of the uploaded file can be changed using the file name template.</p>					
<p>Display name template</p> <p>&lt;%UPLOADED_FILE_NAME%&gt;</p> <p>The display name of the uploaded file can be changed using the display name template.</p>					
<p>Copy file when data row is copied</p> <p><input type="checkbox"/></p> <p>If selected then file will be copied too when data row is copied.</p>					
<p>File name mask</p> <p>*</p> <p>Using this mask the user can upload files with defined name parts only: e.g. *.doc = files with .doc extension only or *.jpg*.img means .jpg and .img files are allowed only</p>					
<p>File directory path</p> <p></p> <p>The file directory path for storing the uploaded files. Apparo must have readwrite rights for this file directory. Every "File UploadDownload widget" shall have unique directory path.</p>					
<p>Maximum allowed file size</p> <p>0</p> <p>Using the maximum allowed file size (MB) you can limit the file size for uploading. 0 = all file sizes are allowed.</p>					
<p>OK    CANCEL</p>					

#### Column for storing file name

DB column for storing the file name as set in the file name mask

#### Column for storing display name

DB column for storing the display name as set in the display name mask, usually the uploaded file name

**File name template**

The name of the uploaded file can be changed using the file name template.

**Display name template**

The name of the uploaded file can be changed using the display name template.

**File name mask**

Using this mask the user can upload files with defined name parts only, e.g. \*.docx = files with .docx extension only. .excel allows all Excel types(csv, xls, xlsx)  
Variables and wildcards (% , \* , ?) are allowed.

**File directory path**

Path where the uploaded files are stored.

Apparo Fast Edit must have read/write rights for this file directory.

Every "File Upload/Download widget" should have a unique directory path.

Network pathes are allowed [\\Server\Path\To\Folder](#), access rights are necessary.

**Maximum allowed file size**

Using the maximum allowed file size (MB) you can limit the file size for uploading.

0 = all file sizes are allowed.

## Tab ,Actions‘

After uploading the file, it is possible to start an action. This is useful when using a document management system.

### Options:

- Enable action after file upload
- Enable action before file download
- Enable action after file delete action

### Actions

- Executing JavaScript commands
- Executing SQL commands
- Running a DB procedure
- Executing an anonymous block (acts like a stored procedures)
- Running a script (batch file, SQL or JavaScript file)

Widget settings of database column OFFICE\_ID

Widget type Mapping & Other Features **Actions** Visual Help texts

☒ Enable action after file upload

Automatic execution of JavaScript commands JavaScript definition

JavaScript commands  
SQL commands  
Database procedure  
Anonymous database block  
Script on server (batch file, SQL file or JavaScript file)

☐ Enable action before file download

☐ Enable action after file delete action

OK CANCEL

Variables are allowed here

## 6.7 Business Case Functions

This chapter covers in detail all Business Case functions

### 6.7.1 Standard Buttons

#### Main

All the preset by the system buttons can be enabled or disabled using checkboxes.

Information about the transaction handling of the buttons can be read in the chapter [transaction handling](#)

The screenshot shows the APPARO interface with the 'Main' settings tab selected. The 'Button titles' sub-tab is active, displaying a table of available standard buttons. The table has columns for 'Button type', 'Button label', 'Enabled', and 'Order'. The 'Enabled' column contains checkboxes, and the 'Order' column contains up/down arrows for reordering. Below the table, there are two checkboxes for additional settings: 'Show "Widget auto resizing" buttons for right-sizing of the widget' and 'Show "<<" and ">>" buttons for easier navigation'.

Button type	Button label	Enabled	Order
> OK	OK	<input checked="" type="checkbox"/>	⬇ ⬆
> Store	store	<input type="checkbox"/>	⬇ ⬆
> Cancel	Cancel	<input checked="" type="checkbox"/>	⬇ ⬆
> Close	Close & Save	<input checked="" type="checkbox"/>	⬇ ⬆
> Delete	Delete	<input checked="" type="checkbox"/>	⬇ ⬆
> Insert	Insert	<input checked="" type="checkbox"/>	⬇ ⬆
> Copy	Copy	<input checked="" type="checkbox"/>	⬇ ⬆
> Reload	Reload	<input type="checkbox"/>	⬇ ⬆
> Excel export	Export to Excel	<input checked="" type="checkbox"/>	⬇ ⬆
> Import copy&paste	Excel Row-Import	<input checked="" type="checkbox"/>	⬇ ⬆
> Import file	File Import	<input checked="" type="checkbox"/>	⬇ ⬆
> Help	Help	<input type="checkbox"/>	⬇ ⬆
> Delete all		<input type="checkbox"/>	⬇ ⬆

☒ Show "Widget auto resizing" buttons for right-sizing of the widget  
☒ Show "<<" and ">>" buttons for easier navigation

You can change the button title for all defined languages by clicking on the name of the type:

Cancel button settings

Label	Language	Label
	German	Abbrechen <span>V</span>
	English	Cancel <span>V</span>

Gap (right)

OK

CANCEL

Gap defines the space to the next button (in pixel) to the right.

### Button title

You can change the label of different standard buttons here:

Main

Button titles

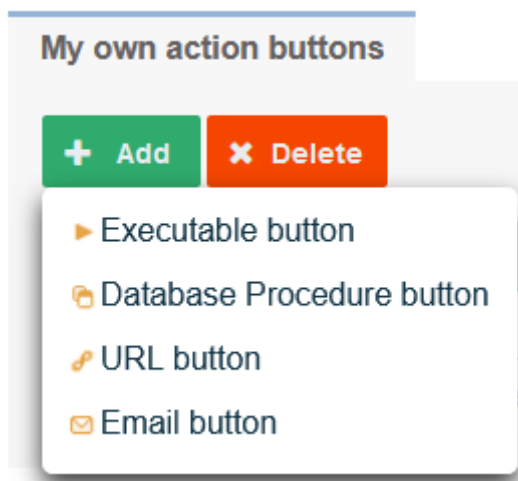
Buttons	German	English
Bulk update	Ändern <span>V</span>	Update <span>V</span>
Filter	Suchen <span>V</span>	Search <span>V</span>
OK (Filter page)	OK <span>V</span>	OK <span>V</span>
Cancel (Filter page)	Abbrechen <span>V</span>	Cancel <span>V</span>
OK (Import)	Importieren <span>V</span>	Start Import <span>V</span>
Cancel (Import)	Import beenden <span>V</span>	Finish Import <span>V</span>



## 6.7.2 Own action buttons

Action buttons can call executables files or scripts, database procedures, URLs and eMail Business Cases.

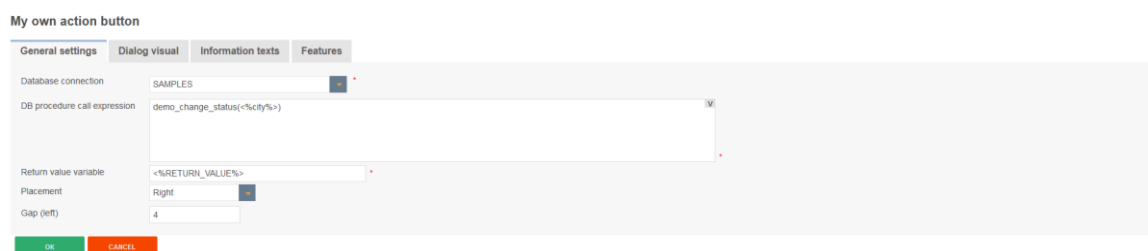
It is possible to specify different behavior patterns. For example, single call or a call for each selected row of data, etc.



### 6.7.2.1 Executable button

With Apparo Fast Edit you have several options for further processing of the data. With "Executable Button" you can add a button for processing (like .bat, .cmd, .sh, .sql). All files that are to be called up must be stored in the script file directory which was defined in the Apparo Configuration Manager. Using the Apparo Configuration Manager it is possible to change the used file directory.

#### 6.7.2.1.1 General Settings



#### Executable filename

Here you can select the batch processing file (or sql, JavaScript file), which will be executed by this button.

#### Arguments

Optional you can use arguments (variables are allowed too) that will be delivered to the script or database procedure.

### Return value variable

In this variable is the return value of the function/script stored.

### Placement

Arrangement of buttons on the screen.

### Gap

Space to the next button in pixel

#### 6.7.2.1.2 Dialog visual

Here you will find settings for the layout and behavior of the message window.

### My own action button

General settings	Dialog visual	Information texts	Features
"Please wait" font	Font face: Arial Size: 13 Style: Bold Align: Left Colour: #000000		
Output message font	Font face: Arial Size: 13 Style: Bold Align: Left Colour: #000000		
Finish message font	Font face: Arial Size: 13 Style: Bold Align: Left Colour: #000000		
Background colour	#FFFFFF		
Logo URL			
Dialog window size	Width: 440 Height: 220		
Automatically close dialog window	<input type="checkbox"/>		

OK CANCEL

#### 6.7.2.1.3 Information texts

At this point you can define a label for your button and individual texts for waiting and finishing.

### My own action button

General settings	Dialog visual	Information texts	Features
Language	Button label	"Please wait" message	Finish message
German	Button für ausführbare Datei	Bitte warten...	Das Skript wurde beendet
English	Executable Button	Please wait...	The script has finished

OK CANCEL

#### 6.7.2.1.4 Features

Here you can define the behavior of the button.

#### My own action button

General settings	Dialog visual	Information texts	Features
Refresh Business Case data after finish	<input checked="" type="checkbox"/>		
Show process output	<input type="checkbox"/>		
Hidden for	> Specify settings for security group		
Execution mode	Execute the script/procedure/email once		
Send eMail after execution	<input type="checkbox"/>		

OK CANCEL

#### Refresh Business Case data after finish

If enabled, the Business Case is reloading the database data again after execution of the script/procedure. This is helpful if your script/procedure is changing data that must be displayed in the Business Case too.

#### Show process output

If enabled, the user will see the script output in a small window.

#### Hidden for

Hides the button for defined user groups

#### Execution mode

Execute the script/procedure/email once

Here you can define the exact behavior of this button. Your script/procedure can be called for each row, selected row or just once.

Execute the script/procedure/email once

- Execute the script/procedure/email once
- Execute the script/procedure/email for all rows of the current page only
- Execute the script/procedure/email for each row of all pages
- Execute the script/procedure/email for each already selected row only

#### Send eMail after execution

After execution of a script or database procedure it is possible to send automatically an eMail. This eMail Business Case has access to all widget values of this Business Case .

That means that the eMail body can contain values of this current Business Case .

### 6.7.2.2 Database procedure button

My own action button

General settings	Dialog visual	Information texts	Features
Database connection	SAMPLES		
DB procedure call expression	demo_change_status(<%city%>)		
Return value variable	<%RETURN_VALUE%>		
Placement	Right		
Gap (left)	4		

OK CANCEL

#### Database connection

Here you can select the database connection on which the button will be proceeded.

#### DB procedure call expression

How to call a database function or procedure:

*[Calling convention] procedure/function\_name ( argument1, argument2, ..., argumentN )*

Please use same character cases for schema and procedure/function like defined in your database.

If the database connection of this procedure/function is same as the one for Business Case than the procedure/function is executed within the same database transaction.

The procedure must not commit or rollback the existing transaction, but is allowed to start its own inner (named) transaction (if supported by database) or use savepoint's.

For character or string argument use ' character to enclose argument.  
Use at least one space between [Calling convention] and procedure name.

Your parameters may contain Apparo Fast Edit variables, for example: <%USER\_NAME%>  
Do not enclose Apparo Fast Edit variables with apostrophes or quotes.

*Oracle or IBM DB2 database:*

**return** - For calling a stored function that returns a value

*MS SQL Server database:*

Calling functions on SQL Server is not supported. It is possible to have a return value from procedure but [Calling convention] must be empty in this case.

Please use in your SQL Server procedure at the begin "SET NOCOUNT ON;" Then it is possible to use SQL commands in your procedure without having impact to the return value.

*Sybase database:*

**select** - For calling a stored function that returns a value

*Teradata database:*

**return macro** - For calling a Teradata macro that returns a value

**macro** - For calling Teradata macro that does not return a value

**return** - For calling a stored functions that returns a value

**Return value variable**

In this variable is the return of the function/script stored.

**Placement**

Here you can decide about the arrangement of buttons on the screen.

**Gap**

Space to the next button in pixel

### 6.7.2.3 URL buttons

With these buttons you can call any URL:

- Web Sites & Portals
- Reports & Dashboards
- Business Cases

My own action button

General settings | Dialog visual | Information texts | Features

URL:

Placement:  [v]

Gap (right):

OK CANCEL

### 6.7.2.4 E-mail Buttons

With these buttons you can send e-mails.

My own action button

General settings | Dialog visual | Information texts | Features

Email settings:  [v]  
Superstore - Detail - SendMailas

Placement:  [v]

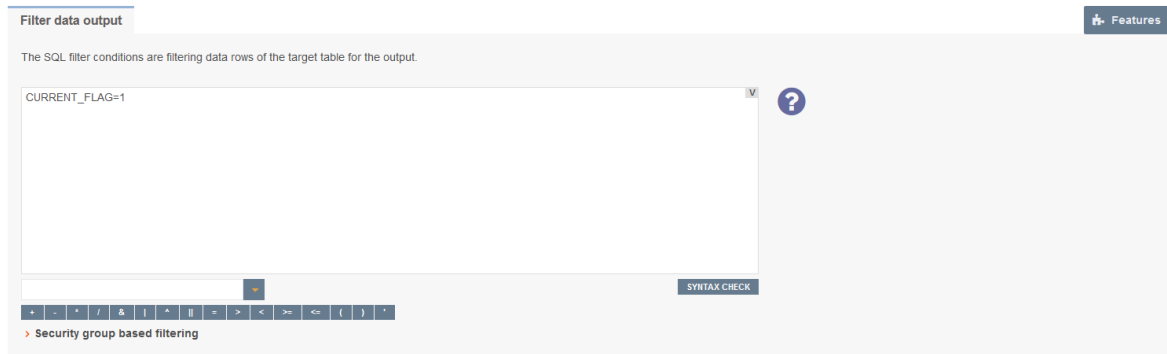
Gap (right):

OK CANCEL

The settings for the e-mail you make in the selected e-mail Business Case .  
All variables of the calling Business Case can be used.

### 6.7.3 Filter data output

The function filter data output represents the global filter of the Business Case. Additional filters can be added through filter widgets.



You can create different filters for different security groups. If a user is a member of the security group, only the security group based filter is used instead of the global BC filter.

Variables are allowed.

#### Syntax

In the filter, you can use native SQL. It represents the Where clause of the SQL query and filters the output of the target table.

#### Example

`SELECT * FROM target table WHERE [=data output filter]`

#### 6.7.4 Filter widgets

Contains the optical settings for the additional filter page and the settings for combining filter widgets

#### 6.7.5 Combine Widgets with AND/OR

The function extends the filtering possibilities with filter widgets.



Standard type of searching is combining all used searching widgets with logical 'AND' operator. If you want to combine them differently then you must use 'Advanced Type Of Searching'. In 'Search Expression' you can define your own combination of searching widgets. You can combine them with operators 'AND' and 'OR' and you can also use brackets '(' and ')'

**Each searching widget must be used exactly once in the search expression.**

The following examples contain combinations of these four filters widgets

#### Examples

**One of the set criteria is met:**

<%SEARCH\_VALUE\_PRODUCT\_ID%> or <%SEARCH\_VALUE\_PRODUCT\_COLOUR%> or  
<%SEARCH\_VALUE\_PRODUCT\_SIZE%> or <%SEARCH\_VALUE\_PRODUCT\_LINE\_ID%>

**The product line and ONE of the other filter criteria is met:**

<%SEARCH\_VALUE\_PRODUCT\_LINE\_ID%> and (<%SEARCH\_VALUE\_PRODUCT\_ID%> or  
<%SEARCH\_VALUE\_PRODUCT\_COLOUR%> or <%SEARCH\_VALUE\_PRODUCT\_SIZE%>)

**The product line or the combination of ALL other filter criteria are met:**

<%SEARCH\_VALUE\_PRODUCT\_LINE\_ID%> or (<%SEARCH\_VALUE\_PRODUCT\_ID%> and  
<%SEARCH\_VALUE\_PRODUCT\_COLOUR%> and <%SEARCH\_VALUE\_PRODUCT\_SIZE%>)



## 6.7.6 Variables

**Syntax:** <%Variable\_name%>

User defined variables | Internal variables | Features

+ Add - Delete

Variable name	Variable type
<%bulk_text%>	Script variable
<%city%>	Script variable
<%NextID%>	SQL variable (target table)

Variable name	Variable description
<%SEARCH_VALUE_PRODUCT_ID%>	Value of filter widget mapped to PRODUCT_ID column name

Basically, there are user-defined variables and internal variables.

**Apparo Fast Edit supports 6 different types of variables:**

- Internal pre-defined variables
- Operating system environment variables
- Script variables
- SQL variables
- Report variables
- Widget reference variables

Variables can be used in practically all settings and other variables

**In Business Cases, you can create these types of variables:**

Select the type of new variable

	> Script variable	You can use JavaScript to compute advanced calculations and the result can be used in Apparo as any other variable. The execution is done server side only.
	> Report variable	Report variables: They are used to deliver content to a Business Case using the URL e.g. from a report or to deliver the content of a widget from one Business Case to another one.
	> SQL variable (for all tables)	SQL variable to execute commands on all tables. Every time the SQL variable is used then the defined SQL is executed. The variable content is the first column of the first row of the executed SQL. You can use the variable (e.g. <%current_year%> in many input fields of the Business Case definitions, e.g. in header text, default value, constant value and so on.
	> SQL variable (for target table only)	SQL variable for Business Case target table only.

CANCEL

#### 6.7.6.1 Use of variables in the Designer

Many widget settings can be made dynamic with variables.

##### Examples:

##### Variables in lookup definitions

Lookup column for join with target column	PRODUCT_LINE_ID	<input type="checkbox"/> Use read expression
Lookup table value column for output	PRODUCT_LINE_NAME_<%LANGUAGE%>	<input type="checkbox"/> Use read expression

The associated database column is composed of, Name\_ 'and the return value of the language used. German users are assigned to the column NAME\_DE and English users to the NAME\_EN column

##### Variables in labels, hint texts, the header and footer

Column label	
Language	Column label
German	<%LABEL_DE%>
English	<%LABEL_EN%>

In this example, the heading of the column is output by variables

##### Variables in filter definitions

SQL where condition	PRODUCT_LINE_ID = <%PRODUCT_LINE_ID%>
---------------------	---------------------------------------

Dynamic SQL filter

## Variables in variables and in the data row validator

An example for the use of variables in the data row validation:

Data row validation

Data row validator

```

1  var a = afe.resolveVariable('FORECAST');
2  var b = <%FORECAST2%>;
3  var c = <%FORECAST3%>;
4  var d = <%FORECAST4%>;
5  var e = <%FORECAST5%>;
6  // prepare empty result, what means that row data is valid
7  var result = '';
8  if (a < (b+c+d+e)) {
9    if ('<%LANGUAGE%>' == 'en') {result = 'Sum of quarters is greater than the amount per year';
10   } else {
11     result = 'Summe der Quartale ist größer als das Jahr'; }}
12 if (a == null || a==0) {
13   if ('<%LANGUAGE%>' == 'en') {
14     result = 'Please enter an amount per year'; } else {
15       result = 'Bitte geben Sie die Anzahl je Jahr an'; }}
16 // return the result
17 result;
18
19
20
21

```

SYNTAX CHECK

In this example widget reference variables, SQL variables and internal variables have been used

```

1  var result;
2  if(<%LANGUAGE%>=='en')
3  {result = 'Message text';}
4  else
5  {result = 'Nachrichtentext' ;}
6  result;
7

```

In this example, an internal variable is used in a JavaScript variable

Variable value

Data output format

SQL expression

```

select OFFICE_ID where PRODUCT_ID = <%PRODUCT_ID%>|

```

Widget reference variables are often used in SQL variables. JavaScript variables are also possible.

### An example, for the use of dynamic variables as interval:

In a widget of type "input field", the permissible range of values is restricted:

Interval	Minimum allowed:	<%MIN_INTERVAL%>
	Maximum allowed:	<%MAX_INTERVAL%>

Example of dynamic intervals that restrict the values input by calculations.

*Dynamic values are realized via variable:*

Our SQL variable is of type SQL variable (target table only). This has the advantage that automatically all user-group-dependent filters are used.

The current line is identified by the value in the widget PRODUCT\_ID. That PRODUCT\_ID is a primary key.

The following sample SQL for SQL variable would be possible:

**SELECT min\_value FROM target\_table WHERE product\_id = <%PRODUCT\_ID%>**

In this case, <%PRODUCT\_ID%> refers to the widget PRODUCT\_ID in the Business Case and returns the current value.

The SELECT returns the value min\_value of the current line and stores it in the new SQL variable "VAR\_MIN\_CALC".

The SQL is executed every time when accessing the variable "VAR\_MIN\_CALC".

### 6.7.6.2 Internal variables

The following variables are predefined and can be used immediately:

Variable name	Variable description
<%AFE_HOME_DIR%>	Folder on the server which contains AFE settings
<%AFE_CLIENT_ID%>	Contains the client ID of the current client
<%AFE_BC_NAME%>	Name of currently opened Business Case
<%AFE_BC_ID%>	ID of currently opened Business Case
<%AFE_BC_FOLDER%>	The Designer path of this Business Case
<%SERVER_NAME%>	Name of server where Apparo Fast Edit is running
<%USER_NAME%>	Name of currently logged user
<%USER_LOGIN%>	Unique login name of currently logged user
<%USER_EMAIL%>	Email address (in upper case) of currently logged user
<%LANGUAGE%>	Identifier of language in which user interface is displayed
<%NEW_UNIQUE_VALUE%>	Unique value (everytime variable is resolved, its value will be unique)
<%FILE_CONTENT%>	Returns content of the file as a string. Syntax is: <%FILE_CONTENT(C:\myFile.txt)%>
<%CURRENT_DATE%>	Current date and time
<%CURRENT_DATE_STANDARD%>	Current date in the format "yyyy-MM-dd 00:00:00.000" e.g. "2021-02-28 00:00:00.000". This variable can be used for comparing with widget reference variables of type date.
<%CURRENT_TIMESTAMP_STANDARD%>	Current date and time in the format "yyyy-MM-dd HH:mm:ss.SSS" e.g. "2021-02-28 14:51:45.456". This variable can be used for comparing with widget reference variables of type date+time.
<%DATE%>	Current date
<%DATE_ISO%>	Current date formatted according 'yyyy-MM-dd' format
<%DATE_TIMESTAMP_SHORT%>	Current timestamp for file names etc.
<%TIMESTAMP%>	Current date and time
<%TIME_MS%>	The number of milliseconds since 1.1.1970 (UNIX timestamp)
<%CURRENT_TARGET_TABLE_NAME%>	Name of the current target table
<%PRIMARY_KEY%>	The primary key of current row
<%PRIMARY_KEYS%>	Comma delimited list of the used primary keys
<%ROW_EDIT_TYPE%>	Type of data modification. Output is of type string
<%SELECTED_ROWS_COUNT%>	This variable is helpful for output e.g. "Are you sure you want to delete X rows?"
<%ROWS%>	Count of current visible rows
<%BULK_UPDATED_ROWS%>	Count of all updated rows
<%INSERTED_ROWS%>	Count of all inserted rows during Excel import
<%UPDATED_ROWS%>	Count of all updated rows during Excel import
<%IMPORTED_ROWS%>	Count of all imported rows during Excel import
<%EXCEL_IMPORT_COLUMNS_COUNT%>	Contains current count of Excel columns of current Excel row of Excel import with copy & paste. This count is calculated for every Excel data row.
<%IMPORTED_FILE_NAME%>	Name of the currently imported Excel file
<%EXCEL_IMPORT_ID%>	Universally unique identifier (UUID) of type String of each Excel import
<%EXPECTED_COLUMNS%>	List of expected columns for Excel import

<%LINE%>	This variable is helpful for display error during import e.g. "Import error in line X:"
<%UPLOADED_FILE_NAME%>	Name of the uploaded file (file upload/download widget)
<%DISPLAY_FILE_NAME%>	Display name of the uploaded file (file upload/download widget)
<%DELETED_FILE_NAME%>	Name of the deleted file (file upload/download widget)
<%IMPORT_TICKET_ID%>	Ticket number of the current import process when importing data using Email Import Business Case
<%ORIG_EMAIL_SUBJECT%>	Subject of the users original email when importing data using Email Import Business Case
<%RETURN_VALUE%>	In this variable the return code of the function/script is stored.
<%CURRENT_WIDGET_NAME%>	The current widget name. Can be used for defining background colours, default values or constant values for many similar widgets
<%ACTIVE_COPY_WINDOW%>	Variable has value 'true' if the current window is the window for copying rows
<%COPY_ROW_MODE%>	Variable has value 'true' if the copy feature is applied. It is helpful if you want to define a special widget behaviour in copy case.

If the Business Case uses search fields, e.g. a filter lookup, then the matching variables are automatically defined for each search widget:

<%SEARCH_KEY_COLOR%>	Key-Value of the Lookup widget, mapped to column 'COLOR'
<%SEARCH_VALUE_COLOR%>	Value of the Lookup widgets, mapped to column 'COLOR'

### 6.7.6.3 Report variables

They are used to deliver content to a Business Case using the URL e.g. from a report, web portal or any place where hyperlinks can be used or simply to deliver the content of a widget from one Business Case to another one.

The content of a report variable can be defined in a report in a column of a query.  
Using a hyperlink in the report, the value can be transported to the connected Business Case .  
A report variable in an URL has the syntax FE\_name.

## Variable for Business Case

Variable name	<input type="text" value="Report_Var_Name"/>	*
Variable description	<div></div>	

Variable value	Data output format
Default value	<input type="text" value="999"/>

The default value is used only if the report does not provide a value for this variable.

## Variable for Business Case

Variable name	<input type="text" value="Report_Var_Name"/>	*
Variable description	<div></div>	

Variable value	Data output format
<p>Defines the type of the variable, the data output format and the expected data input format.</p> <p>If you want to define the value of this variable at Business Case start time then you must care about the <b>expected data input format</b>. Expected format for date is MM.DD.YYYY (for example: 12.31.2018) or YYYY-MM-DD (for example: 2018-12-31), for time is HH:mm:ss (for example: 23:59:59). You can also use ISO 8601 date format (for example: 2018-12-31T23:59:59.999-0700).</p> <p>If you want to define own date/time format just for this variable use additional parameter FE_name_FORMAT, where Example: &amp;FE_name_FORMAT=dd.MM.yyyy HH:mm (or using different format).</p> <p>Or you can use additional parameter dateFormat to define your own date/time format. Parameter dateFormat is global for all variables. Example: &amp;dateFormat=dd.MM.yyyy HH:mm:ss (or using different format).</p>	
Output type	<div> <div>Text</div> <div><b>Number</b></div> <div>Date / Time</div> </div>
Decimal places	<input type="text" value="2"/>
Show separate groups	<input type="checkbox"/>

OK

CANCEL

In output format can set the data type.

### Example of calling a Business Case with a URL:

[http://localhost/apparo/pages/userInterface.jsf?bc=BCNAME&FE\\_REPORT\\_VAR1=1234&backLink=%2Fcontent%2Ffolder%5B%40name%3D%27Apparo+Fast+Edit+Demonstration%27%5D](http://localhost/apparo/pages/userInterface.jsf?bc=BCNAME&FE_REPORT_VAR1=1234&backLink=%2Fcontent%2Ffolder%5B%40name%3D%27Apparo+Fast+Edit+Demonstration%27%5D)

In the URL has the report variable **REPORT\_VAR1** the value **1234**

This report variable can now be used in the Business Case or further processed.



#### 6.7.6.4 SQL Variables

There are 2 different types of SQL variables:

- **SQL variable (for all tables)**

SQL variable for executing commands in all tables. Each time you use the variable the associated SQL is executed. This variable contains the content of the first row, first column (depending on the SQL command)

- **SQL variable (for target table only)**

SQL variable for the Business Case target table. All filters of the Business Case are considered.

Example:

##### Variable for Business Case

Variable name

Variable description

Variable value

Data output format

SQL expression

select NVL(MAX(ID),0) + 1 from FESAMPLES.SAMPLE\_FORECAST

+

-

\*

/

&

|

^

||

=

>

<

>=

<=

(

)

'

OK

CANCEL

SYNTAX CHECK

☒ Calculate the variable value before each usage again

The main difference is that a **SQL-variable (for target table only)** automatic uses:

- The filter of the Business Case
- All security-dependent filters
- All Widget dependent filters

Therefore, the SQL of the variable must also use the target table so that the filter will also find the same column names.

**SQL variables (for target table only)** are very useful for calculations that relate to the target table - e.g. sum of all sales, as all the used filters are considered automatically.

Since the output changes when using filter widgets, usually this dynamic filter restriction must also be considered.

In a **SQL variable (for target table only)** this is done automatically, in opposite to a SQL variable (for all tables).

An SQL variable is always executed when it is used.  
As result, the first result value is used.

#### 6.7.6.5 Script variables

A script variable is a routine that returns a value. It is not connected to a database session.

##### Variable for Business Case

Variable name	<%city%>
Variable description	

Variable value	Data output format
<div> <div>Script body</div> <div> <p>Script language : javascript</p> <p>You can see a detailed JavaScript language description including examples by clicking the question mark icon placed next to the editor.</p> <p>Attention: If you want to use a Apparo variable in Javascript that contains text then you must use it in quotes, for example:</p> <pre>string.replace('&lt;%TEXT1%&gt;','&lt;%TEXT2%&gt;','text')</pre> <p>If a script variable must return value true or false then it must be a string like 'true'.</p> <pre> 1  var groups = afe.getGroupsByRegex('demo_office.*'); 2  var result = groups[0]; 3  if(result) { 4    result = result.substring(12); 5    result = result.toUpperCase(); 6  } 7  result; 8 9 10 11 12 13 14 15 16 17 18 19 20 21 </pre> <div>SYNTAX CHECK</div> <div> <input checked="" type="checkbox"/> Calculate the variable value before each usage again </div> </div> </div>	

The calculated value is returned by ,result'

You can use in the JavaScript routine SQL variables, reference variables and internal variables too. The Logic is defined by **JavaScript** and can be combined with SQL-Queries.

You can use scrip variables within database connection settings, but connection pooling will be disabled then.

##### Calculate the variable value before each usage again

If unchecked the variable will be calculated only once, when the BC starts

#### 6.7.6.6 Extended JavaScript functions

Fast Edit offers the possibility to use advanced features besides from the standard syntax. The corresponding examples can be found in the designer when you click on the question mark icon.

##### Custom script example returning a string value based on security group:

```
var groups = afe.getGroupsByRegex('.*');
var result = 'Security groups of the current user: ';
var i;
var group;
for(i = 0; i < groups.length; i++) {
    group = groups[i];
    result = result + group + ', ';
}
// returning the calculated result from script
result;
```

##### Custom function example returning a string value:

```
// declaring a function
function myCustomFunction() {
    var result = 'defaultStringValue';
    // complex algorithm to evaluate the result of the method
    return result;
}
// calling the declared function to return a value from script
myCustomFunction();
```

##### Example using custom functions declared in file

```
// If we have our custom functions declared in the text file we can use it in our script variable.
// In our example we have a file 'C:\scripts\myFunctions.js' with content: 'function myFunctionPlus(a, b)
{return a + b;}'
// We can 'include' this content into the script variable as follow:
<%FILE_CONTENT(C:\scripts\myFunctions.js)%>

// now we can use declared function
var x = myFunctionPlus(2, 1);

// variable 'x' now has value 3
x;
```

##### Example using Apparo variables:

```
// working with string variable, and adding a custom postfix
var result = '<%USER_NAME%>' + 'postfix';

// modifying a result of sql variable returning a numeric variable
var result = <%SQL_COUNT_VAR%> / 100;
```

#### Example using Apparo LANGUAGE variable in a column name:

```
// In this example Apparo must read the content of the widget reference variable <%PRODUCT_EN%> or
<%PRODUCT_DE%>.
// PRODUCT_EN for a user with english language and
// PRODUCT_DE for a user with german language:
```

```
var rc;
rc = '<%PRODUCT_<%LANGUAGE%>%>';
```

#### Example for calling a java class with return value:

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

#### Example for calling a java class with arguments and return value:

```
// In this example Apparo 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;
```

#### Example for dynamic variable resolving:

```
// In this example Apparo creates an instance of array and set current number of milliseconds (since
1.1.1970) for each element separately.
var args = []; // create new array
var i;
for(i = 0; i < 10; i++) {
    args[i] = afe.resolveVariable('TIME_MS');
}
```

#### 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);

var text;

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

text;
```

#### Example for getting the name and content of the current widget:

```
// In this example Apparo is reading the name and content of the current widget.
// This is helpful for defining the background colour of many similar widgets or defining default
value/constant value without creating many different variables.
var current_widget = '<%CURRENT_WIDGET_NAME%>';
var current_widget_content = afe.resolveVariable(current_widget);
var red_colour_background = 'false';

if (current_widget_content > 100) {
    red_colour_background = 'true';
}
else if (current_widget_content < 50 && current_widget == 'MEASURE1') {
    red_colour_background = 'true';
}

red_colour_background;
```

#### Example for storing content into a file:

```
// In this example Apparo store text content into 'myFile.txt' file.
var fileContent = 'This is file content.';
var success = afe.createFile('c:\\files\\myFile.txt', fileContent);
```

Example for executing a SQL query:

```
// In this example Apparo 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'");
```

#### Example for executing a SQL select:

```
// In this example Apparo 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];
}
```

#### Example for executing a SQL select and storing result into XML file:

```
// In this example Apparo executes SQL select to retrieve 'id', 'name' and 'price' values of all products in
table 'MyProduct' creates xml String and stores it into XML file.
var productsArray = afe.executeSqlSelect('select id, name, price from MySchema.MyProduct');
var i;
var rowData;
var xmlString = '<?xml version="1.0" encoding="UTF-8"?>\n<products>';
var xmlRow;

for(i=0; i < productsArray.length; i++) {
    rowData = productsArray[i];
    xmlRow = '\n\t<product id="' + rowData[0] + '" name="' + rowData[1] + '" price="' + rowData[2] + '" />';
    xmlString += xmlRow;
}

xmlString += '\n</products>';

var success = afe.createFile('c:\\myXmIs\\products.xml', xmlString);
success;
```

#### Example for executing a SQL query with parameters:

```
// In this example Apparo 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);
```

#### Example for executing a command:

```
// In this example Apparo executes command.
var returnValue = afe.executeCommand("c:\\scripts\\myfile.bat", "c:\\scripts");
```

#### Example for making a row read-only:

```
// In this example we want to make a data row read-only when the PROJECT_COMPLETION_DATE widget
// has value of a date in the past (the project is finished).
// First we must create a variable <%ROW_READONLY_VAR%> which will be used here.
// The goal is to return the same date as the widget if it is older then today, otherwise we will return
// dummy date.
// Returning date must be represented as string with proper format.

// dummy date in format of MM.dd.yyyy
var result='01.01.1990';

// today's date
var current_date = new Date();

// We need to read string value of PROJECT_COMPLETION_DATE widget.
// If the widget is storing timestamp then it's string value has format 'yyyy-MM-dd HH:mm:ss.S' e.g. '2015-
// 12-24 18:00:00.0'
// If the widget is storing number then it can be e.g. '42' or '42.1'
var end_date_string = '<%PROJECT_COMPLETION_DATE%>';

// If the PROJECT_COMPLETION_DATE is not specified then we don't want to make row read-only.
// If it has value then we must compare that date with today's date.
if (end_date_string.length > 0) {

// Here we are constructing the Date object from the string in order we can compare two dates.
var end_date = new Date();

// we must set correct Year, Month and Day from the end_date_string
end_date.setFullYear(end_date_string.substring(0,4));
// watch out here: months are calculated from 0 so we must decrease it's number
end_date.setMonth(end_date_string.substring(5,7)-1);
end_date.setDate(end_date_string.substring(8,10));

// now we can compare the dates
if (end_date < current_date) {

// again, we must use correct format: MM.dd.yyyy
var end_date_string_EN_format = "";
end_date_string_EN_format += end_date_string.substring(5,7);
end_date_string_EN_format += '.';
end_date_string_EN_format += end_date_string.substring(8,10);
end_date_string_EN_format += '.';
end_date_string_EN_format += end_date_string.substring(0,4);

result = end_date_string_EN_format;
}
}

// return the result
result;
```



#### Example for executing a SQL insert:

// In this example Apparo executes SQL insert and returns true if inserting is done successfully, otherwise returns false;

```
var x = afe.executeSqlInsert("insert into PRODUCTS(ID,DESCRIPTION) values (1, 'shirt')");
// variable 'x' is resolved as true if insert was done ok.
```

#### Example for reading Cognos session parameters

//In this example Apparo reads Cognos session parameter named 'userClassID'.

//Declaration of the result variable.  
var result = "";

//Call Apparo function that reads all the Cognos session parameters and return them as 2-dimensional String array.

```
var sessionParameters = afe.getSessionParameters();
```

```
var i = 0;
```

//Iterating over all returned parameters  
for(i; i < sessionParameters.length; i++) {

```
    //If parameter's name (second dimension with index 0) is 'userClassID' then we assign parameter's value
    (second dimension with index 1) into 'result' variable.
    if(sessionParameters[i][0] == 'userClassID') {
        result = sessionParameters[i][1];
    }
}
```

//Return the result  
result;

#### Example for exporting all Business Case data to file:

// In this example Apparo exports all Business Case data to file on filesystem and returns whether operation was successful.

// Note: Backslash symbols must be escaped, i.e. '\\' must be used.

```
var result = afe.exportAllRows('C:\\Users\\Administrator\\Documents\\allDataExport-
<%DATE_TIMESTAMP_SHORT%>.xlsx');
result;
```

#### Example for exporting selected Business Case data to file:

// In this example Apparo exports selected Business Case data to file on filesystem and returns whether operation was successful.

// Note: Backslash symbols must be escaped, i.e. '\\' must be used.

```
var result = afe.exportSelectedRows('C:\\Users\\Administrator\\Documents\\selectedDataExport-
<%DATE_TIMESTAMP_SHORT%>.xlsx');
result;
```

**Example for running Email Business Case:**

// In this example Apparo sends an e-mail for each modified row of the Table Business Case.

// First we must create an Email Business Case (e.g. 'NotificationEmailBc') that will be used for email sending.

// We can use variables of the Table Business Case in the Email Business Case definition.

// Next we must create a script file (e.g. 'sendingEmailNotification.js') containing single line:  
afe.runEmailBc('NotificationEmailBc');

// Then we must enable 'Enable Post row update execution' feature in the Table Business Case, set  
"Automatic execution of" to "Script on server"

// and choose the 'sendingEmailNotification.js' in the drop-down list 'Name'.

// With such Business Case setup an email will be send every time a row will be updated,  
// including Excel import (manually or using automatic server-side import or Business Case Email Import)

#### 6.7.6.7 Widget reference variables

It is possible to use the **current** content of a widget in within the filter of another widget. Other use cases are the usage within SQL and script variables or within labels.

##### Widget settings of database column **PRODUCT\_ID**

Widget type	Mapping & Other	Features	Lookup & Dropdown & Multiselect	Visual	Help texts	Data output format
Column name	<input type="text" value="PRODUCT_ID"/>					

The name of a widget reference variable is defined by the name of the column. Here: <%PRODUCT\_ID%>

##### Example:

It's possible to filter the values of a lookup widget depending on the value of another widget.

A Business Case has 2 widgets:

#1 Widget **PLANT** with the current plant value

#2 Lookup-Widget **DEPARTMENT** that shows all departments of the current selected plant.

Therefore the filter of the widget DEPARTMENT must be used as:

**PLANT\_DEP = <%PLANT%>**

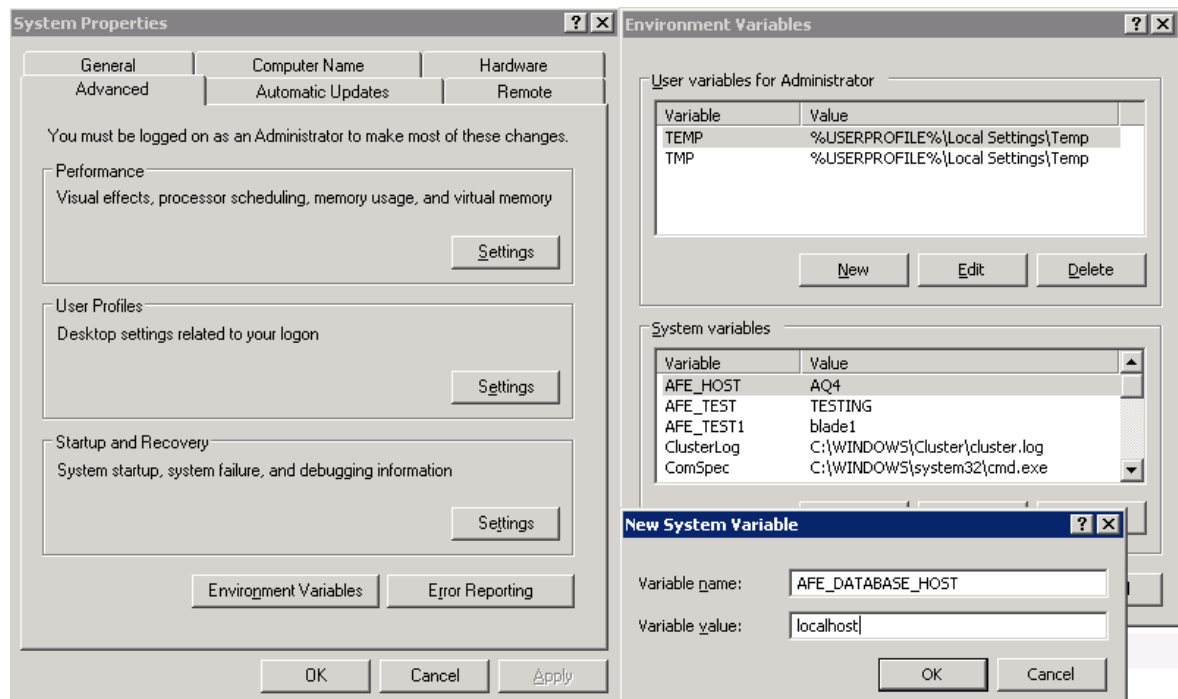
The column PLANT\_DEP must be part of the lookup database table of widget DEPARTMENT.

#### 6.7.6.8 Environment variables

All system variables starting with ,AFE\_‘ can be used within Business Cases and database connections.

You have to restart Apparo Fast Edit after defining the system variables.

Example for a Windows system variable:



In Windows are environment variables named ‘System variables’

Variables that are containing \_\_ (two underlines) in the name can be used too, but are not displayed in Apparo Designer in the variables list.

### 6.7.7 Inserting of new data rows

If enabled then insertion of new data rows is possible. In this case an empty data record will be shown below or above the existing data in the edit area.

It's possible to hide or place read-only widgets in the inserting area too. Optional it's possible to enable the insertion feature for certain user groups only.

**Inserting of new data rows** Features

If enabled then insertion of new data rows is possible. In this case an empty data record will be shown under the existing data in the inserting area. It's possible to hide or place read-only widgets in the inserting area too. Optional it's possible to enable the insertion feature for some user groups only.

Insertion of rows is enabled for: all users

Inserting area is placed: after editing area

☐ Inserting area is able to update an already existing data row  
Sometimes the user is inserting a new row in the inserting area but the row is already here.  
 If enabled then it is overwriting the existing data row without complaining.  
 If disabled then the user is seeing an error message.

#### Insertion of rows is enabled for

This provides three options, default is ,all users'

all users

- all users
- selected security groups
- if variable returns true

#### Inserting area is placed

Controls the placement of the insert area

after editing area

- before editing area
- after editing area

#### Inserting area is able to update an already existing data row

Sometimes the user is inserting a new row in the inserting area but there is already a row with the same primary key.

If enabled then it is overwriting the existing data row.  
 If disabled then the user is seeing an error message.

### 6.7.8 Editing of data rows

If editing is activated, it also activates this dialog here.

Editing of data rows

If editing is disabled then all editing widgets will be displayed as readonly.

Editing of data rows is enabled for

**Making whole data row in editing area readonly if ...**

If the selected widget value will match entered value, the entire record (row) will be read-only.

No widget selected

ADD

You can allow editing of data for all users, for certain user groups only or based on a variable check:

Editing of data rows is enabled for

all users  
selected security groups  
if variable returns true

#### Making whole data row in editing area readonly if ...

If the selected widget value will match entered value, the entire record (row) will be read-only. This feature is helpful if you work with different record states like 'open', 'closed' and just certain records must be updateable. Use English format to define numeric or date values.

The Excel import with copy & paste is recognizing the read-only rows too if the Excel import is using the edit area settings.

By clicking the ADD button you can select widgets that sets the row to be read-only based on a widget value

Editing widget

has value 
for

ADD

The value can also be a fix value or calculated by a variable.

#### Example for a calculated read-only column:

You create a hidden column named e.g. check\_column, with always contains the value 'true'. This can be done by using e.g. the string 'true' as constant value in insert & update case.

Now you can create a script variable to check different conditions and return the value 'true' if the conditions are met and the row shall be read-only. Otherwise, the variable returns 'false' and users can edit this row.

Making whole data row in editing area readonly if ...

If the selected widget value will match entered value, the entire record (row) will be read-only.

Editing widget  has value  for

ADD

## 6.7.9 Deleting of data rows

If deleting is activated, it is adding a delete button and selecting checkboxes. You can also activate the output of a security dialog here.

**Delete data rows manually** Features

Deleting of data rows is enabled for: all users

☒ Enable dialog window for deleting of selected data rows

**Delete dialog settings**

Language	Text
German	Alle markierten Zeilen entfernen?
English	Do you want to delete the selected <%SELECTED_ROWS_COUNT%> data rows?

Language	"Yes button" label	"No button" label
German	Ja	Nein
English	Yes	No

Delete Dialog Style

Font face: Arial    Size: 14    Style: Bold    Align: Center    Colour: #000000

Background colour: #FFFFFF

Dialog window size: Width: 350    Height: 150

You can also change the text of the delete query and the label of the button. Furthermore, it is possible to adapt the layout of the delete query.

Variables are allowed.

### Example

Do you really want to delete <%SELECTED\_ROWS\_COUNT%> rows?

If Delete all data rows on BC start is activated, the following dialog can be activated:

☒ Enable dialog window for deleting of all data rows

**Confirmation dialog settings for 'Delete All' button**

Language	Text
German	Alle Daten löschen?
English	Do you want to delete all data rows?

Language	"Yes button" label	"No button" label
German	Ja	Nein
English	Yes	No

Delete Dialog Style

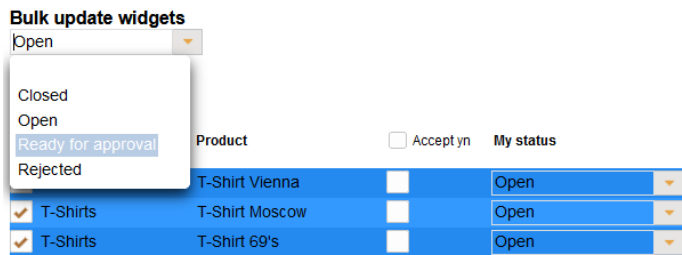
Font face: Arial    Size: 12    Style: Normal    Align: Left    Colour: #000000

Background colour: #FFFFFF

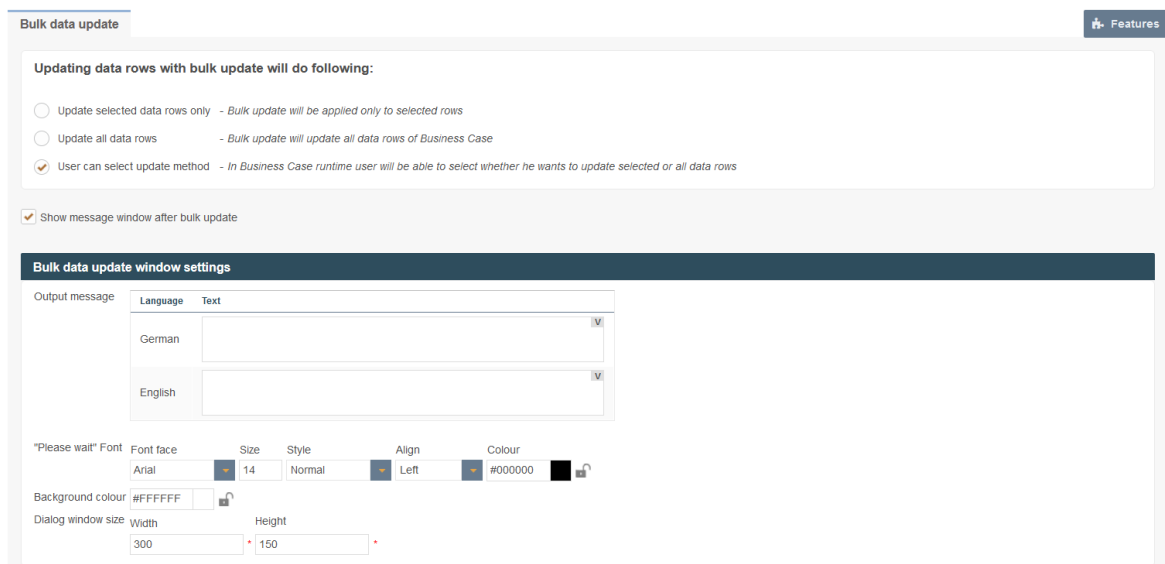
Dialog window size: Width: 350    Height: 150

### 6.7.10 Bulk data update

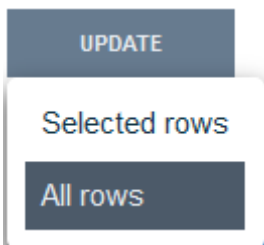
With the bulk update feature the user can update many rows with one mouse click. You can define bulk update widgets and with these widgets the user can set values for all selected rows. Hidden widgets can be updated/hidden bulk widgets with a constant value are allowed too.



In Designer, you can activate a message window after a successful bulk update.



You can choose the update option between Selected rows only or All rows, or leave it up to the user to decide:



The message window is displayed only if you define a text for the output message.

Variables are allowed.

#### Sample output message

<%BULK\_UPDATED\_ROWS%> records were updated.



#### 6.7.11 Excel Import

Excel is still one of the most powerful data processing programs:  
An ideal way to edit and present data in a simple way.

Unfortunately, Excel has disadvantages, the data is locked in a file.

Apparo Fast Edit offers several ways for the Excel import. Thus, the data can be tested auditable for errors and transferred in appropriate media (databases).

##### **Excel import options**

- With copy & paste directly from an open Excel file (Manual Import)
- By file import via the browser (File Import)
- Through automatic import from defined directories (Auto Import)
- By importing e-mail attachments (E-Mail Import)

The automatic import and import via e-mail attachment always requires a table Business Case, in which the settings for the (manual) import are defined.

### 6.7.11.1 General Excel Import

Run Close Save Cancel

General Excel import Import strategy Manual import using copy & paste Excel file import Automatic Excel file import Features

Main settings  
Standard buttons  
My own action buttons  
Filter data output  
Variables  
Inserting of new data rows  
Editing of data rows  
Delete data rows manually  
Bulk data update  
Excel import  
Excel export  
Data row validation  
Auditing of data changes

☐ User can use Excel import using copy & paste feature only  
☐ User can use Excel file import feature only  
☒ Enable dialog window with result message after Excel data row import

Dialog window size Width: 400 Height: 150

Language	Excel row import finish message
German	Anzahl Zeilen zum Importieren: <%IMPORTED_ROWS%>-  Zeilen wurden eingefügt: <%INSERTED_ROWS%>-  Zeilen wurden aktualisiert: <%UPDATED_ROWS%>- 
English	Number valid of rows to import: <%IMPORTED_ROWS%>-  Data rows inserted: <%INSERTED_ROWS%>-  Data rows updated: <%UPDATED_ROWS%>- 

Show description

## Options

### User can use Excel import using copy copy & paste feature only

If this option is activated, the business case only has the Excel import function with copy and paste. This means that immediately after starting the business case, the user only sees the Excel line import area, so this business case can only be used for Excel line import. If this business case uses the same primary keys as defined in the database table, it is helpful to disable the "Check primary key constraints before saving" feature to improve import performance.

### User can use Excel file import feature only

If this option is activated, the business case only has the import function for Excel files. The user can select an Excel file and import the data. This means that immediately after starting the business case, the user only sees the Excel file import page. This business case can therefore only be used for the Excel file import.

### Enable dialog window with result message after Excel data row import

After an Excel import the user can see a small finishing message. You can alter the text of this message here.

### Special Import variables

<b>IMPORTED_ROWS</b>	Count of processed rows
<b>INSERTED_ROWS</b>	Count of inserted rows
<b>UPDATED_ROWS</b>	Count of updated rows

### 6.7.11.2 Import strategy

General Excel import | **Import strategy** | Manual import using copy & paste | Excel file import | Automatic Excel file import | Features

☒ Insert new data row always

☒ Update existing data row always

Excel import strategy Import valid rows and ignore invalid rows

Autocommit after 1000 rows

☐ Write into a readonly widget too (for Copy & Paste)

☐ Write into a hidden widget too (for Copy & Paste)

☐ Check exact count of decimal places for numeric widgets

☒ For file import: User can edit the wrong data rows manually directly in web browser if the whole import file has < 100 wrong data rows

#### Insert new data row

If enabled then new data rows (the primary key values of this new data row are not found in the target table) are inserted

There are two options, either a new row is always inserted or only with the prior examination (via JavaScript variable)

always

always

only if script variable returns true

#### Update existing data row

If this is enabled, existing rows will be overwritten (if the primary key combination is used twice), either always or by variable checking.

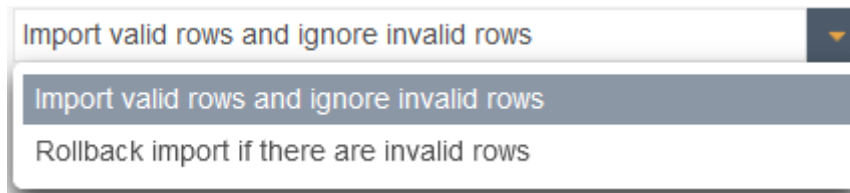
always

always

only if script variable returns true

### Excel import strategy

With this feature you can configure the behavior of an Excel import.



You can select between a complete rollback if there is invalid content (no data will be imported) or whether only valid content will be imported.

### Autocommit after 1000 rows

Apparo Fast Edit will commit the database transaction after defined number of rows has been processed in Excel import.

If value is 0 or no value is defined then this feature is disabled.

If the above setting is set to "rollback import if there are invalid rows" than this feature is disabled.

### Write into a readonly widget too

If enabled then Excel import is overwriting the value of an read-only widget too

### Write into a hidden widget too

If enabled then Excel import is expecting a value for a hidden widget too

### Check exact count of decimal places for numeric widgets

When enabled then numeric values to be imported must exactly match the specified count of decimal places defined in the widget's data output format setting (must be set to "number").

### 6.7.11.3 Manual Import using copy & paste

This function allows direct import of data rows from Excel by means of copy and paste into this business case. The user can select many Excel rows (even more than 100,000 rows are possible), press the Excel import button and paste them into the text area. Of course, the order of the Excel columns must be the same as in the Business Case. Read-only and hidden widgets are not used for mapping, but only if they have a constant value. It is not allowed to import Excel cell values that span multiple rows, use file import in this case.

#### Column list description

You can define a description text that is helpful for the user to know all expected Excel columns. All variables are useable and HTML tags are possible too. You can use the internal variable `<%EXPECTED_COLUMNS%>` that has a list of all expected columns using the widget labels. All hidden or read-only widgets are expecting no Excel column value but the default/constant values of the hidden/read-only widgets are used automatically.

#### Mapping from Excel

If you want to import into different widgets than of the editing/inserting area then you can define an individual mapping for Excel file import too.

#### 6.7.11.4 Excel file import

This feature enables a manual import of an Excel data file into Apparo Fast Edit. The user can select an Excel file and the Business Case is importing the complete file.

Important: You must define a mapping of Excel columns like A, B, C and the associated widget. Just define the Excel column name in the right widget. You can define the count of header rows that must be ignored in tab "Excel file import".

The screenshot shows the 'Excel file import' tab in the Apparo Fast Edit interface. The interface is divided into several sections:

- General Excel import:** Includes a 'Header row count' field set to 2 and an 'Error file format' dropdown set to 'Same as input file'.
- Excel file name:** A section with a text area for the file name and a 'Language' dropdown set to 'German'. Below it, there are two input fields for 'Excel file name' and 'Excel sheet name', both with 'German' selected in the language dropdown.
- Excel sheet name:** A section with a 'Language' dropdown set to 'German' and two input fields for 'Excel sheet name', both with 'German' selected in the language dropdown.
- Data file import description:** A section with a 'Language' dropdown set to 'German' and two input fields for 'Data file import description', both with 'German' selected in the language dropdown.
- Mapping strategy:** A dropdown menu set to 'Using Excel column names like A,B,C'.
- Mapping Excel to widgets:** A section with a 'Mapping strategy' dropdown set to 'Using Excel column names like A,B,C' and a 'Mapping Excel to widgets' dropdown set to 'Using Excel column labels of first row for mapping'.

At the bottom, there is a table with columns: Widget type, Database column, Excel column name (like B or BA), Default value, and Constant value.

#### Options

##### Header row count

This number of rows are ignored during the import

##### Error file format

Format of a file containing errors and their descriptions, a user can download after an import that finished with errors.

The screenshot shows the 'Error file format' dropdown menu. The options are:

- Same as input file
- Excel 2007 or newer (XLSX)
- Excel 2003 or older (XLS)
- Comma-separated values (CSV)

### Excel file name

You can specify that the name of the Excel file must comply with a naming convention, e.g. it must begin with "Controlling", etc.

You can define a regular expression for this.

If the setting is empty, the file names are not checked.

### Excel sheet name

An Excel document can contain many Excel worksheets.

If this setting is empty, the first (from left to right) worksheet is imported.

You can enter a worksheet name for each language and you can use \* and ? Example: Turnover\*

If there are many worksheets that match this name, the user must select the correct worksheet.

You can also use item numbers, e.g. #2 for the 2nd sheet or #1 for the first. You can also use variables.

### Mapping Excel to widgets

For the import a mapping necessary. This mapping is defining all Excel columns that must imported using this Business Case .

If you input for example Excel column F for the 1. widget then all values of Excel column F will be imported into 1. widget.

If the Excel document has no value in a cell and the mapped widget has a default value defined then Apparo Fast Edit is using automatically the default value.

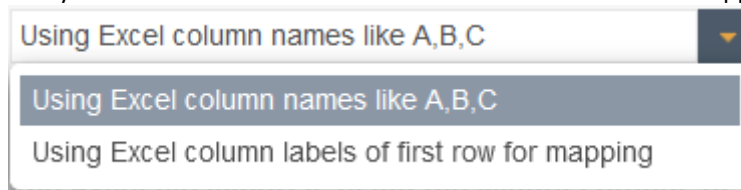
If a widget has a constant value then this value will be used for import depending on the setting (use constant value in insert case only or in insert/update case).

### Data file import description

Contains the text of the select file dialog

### Mapping strategy

Lets you choose between Excel columns or header title for the mapping.



## Mapping Excel to widgets

Widget type	Database column	Excel column name (like B or BA)	Default value	Constant value
Input field	OFFICE_ID	<input type="text"/>	1	
Lookup dropdown (for all tables)	PRODUCT_LINE_ID	<input type="text" value="a"/>		
Lookup dropdown (for all tables)	PRODUCT_ID	<input type="text" value="b"/>		

To import data from an Excel file, it is necessary to assign all the required Excel columns to the widgets. If a widget has no Excel assignment, it does not use the default value and the constant value depending on the widget settings.

The Excel file import only uses the widget settings of the insertion area.

The import requires a mapping between the Excel column and the widget. The mapping defines the Excel columns to be imported.

For example, if you enter an 'F' in the first widget, the Excel column F will be imported into the first widget. Default values are only used if they are assigned to an Excel column and only when inserting a new data row.

Constant values are always used, regardless of the mapping to the Excel column. The setting for use, either only when inserting a new row or when inserting and updating, can be found in the widget settings.

## Settings for a CSV file for importing

When importing CSV files additional settings are required.

Settings for a CSV file for importing

Character set of import file

Field separator

Use language defined separator
☐

Quote mark

### Character set of import file

Contains a list of the available character sets.

### Field separator

Definition of the field separator. Using the next setting it is possible to use an own separator for each used language.

Hint: If you need tab character as separator use "\t"

### Use language defined separator

When checked then Apparo Fast Edit detects language type from file (for example from filename\_en.xlsx as "en") and uses separator defined for detected language from language messages.

### Quote mark

Definition of the quote mark - character used to enclose fields containing a separator, usually "



#### 6.7.11.5 Automatic Excel file import

It is possible to automatically import files that the server can access (e.g. Excel files).

In this case, Apparo Fast Edit checks whether files according to a certain file mask exist in a file directory of your choice.

If so, these files are imported into the working directory. After the import, these files are stored in the history files directory.

The settings "Field separator" and "Header number" are also used for manual file import.

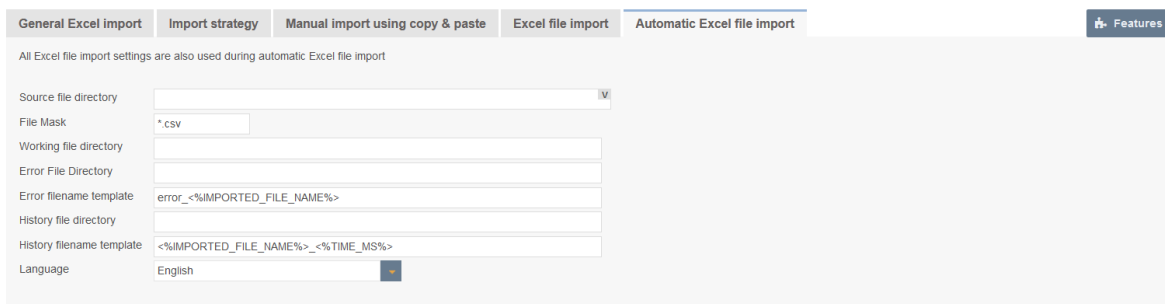
The time interval for looking into the feed directory can be defined in the settings of the current client.

Automatic file import means that the business case does not have to be started manually. After activating the function, the import takes place within the entered time interval.

If the feed directory is located on a second server, the Windows user who starts Apparo Fast Edit must have the rights to access the directory on this second server.

If there are many files ready for import, the file with the oldest change timestamp will be imported first.

**All Excel file import settings are also used during automatic Excel file import.**



The screenshot shows the 'Automatic Excel file import' tab in the Apparo Fast Edit settings. The window has a title bar with 'Features' on the right. Below the title bar, there are five tabs: 'General Excel import', 'Import strategy', 'Manual import using copy & paste', 'Excel file import', and 'Automatic Excel file import'. The 'Automatic Excel file import' tab is selected. Below the tabs, there is a text label: 'All Excel file import settings are also used during automatic Excel file import'. The main area contains several input fields: 'Source file directory' (with a browse button), 'File Mask' (with a dropdown menu showing '\*.csv'), 'Working file directory', 'Error File Directory', 'Error filename template' (with a text field containing 'error\_<%IMPORTED\_FILE\_NAME%>'), 'History file directory', 'History filename template' (with a text field containing '<%IMPORTED\_FILE\_NAME%>\_<%TIME\_MS%>'), and 'Language' (with a dropdown menu showing 'English').

### Options

#### Source file directory

Defines the file directory in which Apparo Fast Edit looks for files to import. It is looking after each 'n' seconds into this directory.

The path can be: \\servername\folder1\folder2 or x:\folder1\folder2 or <%VARIABLENAME%>\folder or <%VARIABLENAME%>. The variable must deliver a correct path.

If empty then no automatic import will occur on this Business Case .

#### File Mask

The importable files must have the defined file mask.

File mask can contain wildcards ? and \*.

Example: \*.xls

Caution: If the filename matches the file mask of multiple Business Cases, the used Business Case will be random. If empty, no automatic import will occur on this Business Case .

### **Working file directory**

Optionally Apparo Fast Edit is moving the files to the 'Working file directory' first and then the import process is starting.

If empty, working file and directory will be the same as the feed file.

### **Error File Directory**

File directory for error files with the error messages. If empty, no error results will be stored.

Error filename template

### **Template file name for error files.**

The usage of variables is possible, for example:

<%IMPORTED\_FILE\_NAME%>

name of the imported file (without path)

<%TIME\_MS%>

numeric (UNIX) timestamp

If empty, no error results will be stored.

### **History file directory**

After import the files are moved into this file directory. If empty, no history will be stored.

### **History filename template**

Template file name for history files.

Mask can contain placeholders <%PlaceholderName%> where PlaceholderName is one of:

IMPORTED\_FILE\_NAME    name of the imported file (without path)

TIME\_MS                numeric (UNIX) timestamp

If empty, no history will be stored.

### **Language**

Language definition (important for formatting such as formatting of date).

#### 6.7.12 Verify that all the files were imported

In a multistage import file, where e.g. the 2nd import step depends on the full completion of the 1st file import, you can ensure by a script that the first import step is complete and that No more files need to be imported.

Strategy:

1. All files for step 1 are copied to the respective source directories
2. Via the script "autoImportChecker" you are waiting until all files have been imported from Step 1, i.e. the import of Step 1 is complete
3. All files for step 2 will be copied to the relevant source file directories

[APPARO\_HOME]\FastEdit\import\autoImportChecker.bat or.  
[APPARO\_HOME]/FastEdit/import/autoImportChecker.sh

Possible parameter:

-clientId <clientId> Example: -clientId QA  
When using this optional parameter, only the client "QA" is checked

-- afeURL <URL> Example: -afeURL http://localhost:18000/apparo  
When using this optional parameter, not the local installation is checked, but the one from the URL

Examples:

autoImportChecker.bat -clientId QA

This call checks all Business Cases from client QA whether it will import files at the moment or in the future. Here, the local Apparo application server is queried.  
The script is terminated only when no file imports are expected.

In the log file " autoImportcheckerResult.log" you will find the relevant log entries.

### 6.7.13 Excel export

With Excel row export the user can export data into an Excel file output and/or into the client clipboard. You can define an optional header, widget labels output, separation character and so on.

#### 6.7.13.1 General

General Excel export using CSV format Features

Exporting into Excel is enabled for: all users

Export to Excel: all selected rows

Mapping to Excel: 1. widget = Excel column A, 2. widget = Excel column B, ...

Exported file name: ApparoExport\_<%TIME\_MS%>

Output the widget labels into first row: ☒

Optional export header title:

Language	Optional header title of Excel export
German	Datenexport aus Apparo Fast Edit
English	Data export from Apparo Fast Edit

### Options

#### Exporting into Excel is enabled for

The export can be disabled for all users or for selected security groups

#### Export to Excel

Defines which rows shall be exported

all selected rows

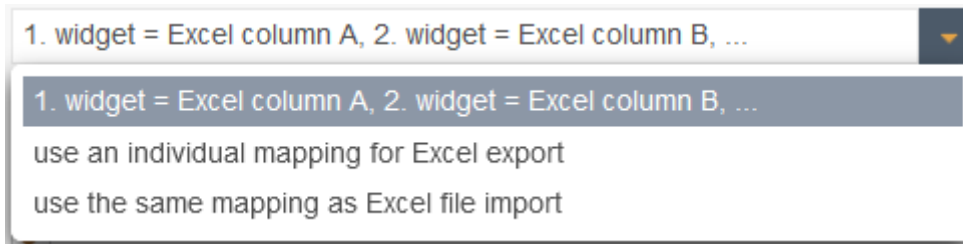
all selected rows

all visible rows of all pages

all visible rows of current page

## Mapping to Excel

The mapping links Excel columns with the corresponding Database columns.



**There are two different mapping strategies:**

The first visible widget is mapped to Excel column A, the second visible widget to Excel column B, and so on

Individual mapping - you can define for each widget the target Excel column in the widget settings. In this way not all widget values must be exported and the ordering is free definable.

### Exported file name

Template for the name of the exported file.

Variables can be used.

The file extension (XLS, XLSX, CSV) is added automatically.

### Output into an Excel file (client side)

Exports the data in an Excel file

### Output the widget labels into own row

Shows an additional Excel row for the labels of the widgets

### Optional Export header title

Is a headline defined, this will be displayed in the first Excel row. All variables can be used here.

### 6.7.13.2 Excel export using CSV format

For the Excel export to a CSV file, further options are available.

General

Excel export using CSV format

Character set of export file
ISO-8859-1

Overwrite default column separator
☒

Language	Field separator
German	.
English	,

#### Character set of export file

Contains a list of settable character sets.

#### Overwrite default column separator

Each installed language in Apparo Fast Edit has an own Excel column separator because Excel is using for some different languages different separators.

If the expected Excel column separator of your Excel version is not equal to the default separator of a language then you can here define the right Excel column separator, use \t for tabulator.

## 6.7.14 Copying of data rows

This can be used to copy rows within the target table.

Copying of data rows

Features

This can be used to copy a data row within a database target table.  
Rules regarding the copying of data:

- In the same window. Data within the same window can be copied
- In the new window. Marked rows will be used in a new window including all primary keys. With this, the data and the primary keys can be altered before copying takes place. The primary keys will always be shown.

Following variables are helpful to define a special widget behavior:

<%ACTIVE\_COPY\_WINDOW%> - Variable has value "true" if the current window is the window for copying rows  
<%COPY\_ROW\_MODE%> - Variable has value "true" if the copy feature is applied.

Select copy rows method:

☒ In the same window - Data within the same window can be copied.
   
☐ In the new window - Copy selected rows into a new window

☒ The user can copy the data row if following variable returns true <%check\_true%>

### In the same window:

Data within the same window can be copied

### In the new window:

Marked rows will be used in a new window including all primary keys.

With this, the data and the primary keys can be altered before copying takes place.

The primary keys are always shown.

### Copy depending on a variable

Permission to copy data lines can be made dependent on a variable that returns true if the conditions are met.

### Users view

Header area with the headline and logos

APPARO<sup>®</sup> Group

Filter area

SEARCH RESET FILTERS

Bulk update widgets

Edit selected rows before copying

Product line	Product	My status	Amount/Year	Quarter I	Quarter II	Quarter III	Quarter IV	Last changed by	Comment	Last change from
T-Shirts	T-Shirt Vienna	Open	5100	1000	600	2000	1200	administrator		09.07.2021
T-Shirts	T-Shirt October	Open	5100	1000	600	2000	1200	Anonymous		13.07.2021
T-Shirts	T-Shirt blue	Open	600	111	111	111	111	Anonymous		13.07.2021
T-Shirts	T-Shirt Vienna3	Open	1100	100	800	100	100	Anonymous		13.07.2021
T-Shirts	T-Shirt Moscow	Open	1100	100	800	100	100	administrator		09.07.2021
T-Shirts	T-Shirt 69's	Open	600	111	111	111	111	administrator		09.07.2021

OK

CANCEL

Calculation area

10000

OK

CANCEL

CLOSE & SAVE

DELETE

IMPORT

EXPORT

EXPORT TO EXCEL

EXPORT ROW REPORT

FILE REPORT

PRINT

Footer

Before copying the records can be edited. It is recommended to change the primary key.

#### 6.7.15 Checking primary key

##### ▼ Checking primary key



#### Checking primary key

Enable checking of PK before changing data in database.

YES

DB-columns that are used as primary keys must NOT be of data type CHAR.

#### Enable checking of PK before changing data in database

If enabled then the Business Case is checking the primary key for being unique. That is helpful if the primary key has no own unique index/primary key constraint in the database table.

It is possible to use database tables that have no primary key defined in the table.  
In this case this feature must be used to simulate a primary key.

#### Not activated:

A primary key can supply more than one hit, helpful, e.g. in denormalized tables.

Warning: The primary key is used to uniquely identify the data row to be stored. If the key exists multiple times, the value of more than one line can be changed or deleted.



### 6.7.16 Data row validator

Enables validation of input when inserting or updating data

**You can:**

- Access all widget content via widget reference variables
- Use SQL variables
- Define own error texts, which are output automatically

Data row validation

Data row validator

```

1  var a = afe.resolveVariable('FORECAST');
2  var b = <%FORECAST2%>;
3  var c = <%FORECAST3%>;
4  var d = <%FORECAST4%>;
5  var e = <%FORECAST5%>;
6  // prepare empty result, what means that row data is valid
7  var result = '';
8  if (a < (b+c+d+e)) {
9      if ('<%LANGUAGE%>' == 'en') {result = 'Sum of quarters is greater than the amount per year';
10     } else {
11         result = 'Summe der Quartale ist größer als das Jahr';
12     }
13 }
14 if (a == null || a==0) {if ('<%LANGUAGE%>' == 'en') {
15     result = 'Please enter an amount per year';
16 } else {
17     result = 'Bitte geben Sie die Anzahl je Jahr an';
18 }
19 // return the result
20 result;
21 ~

```

SYNTAX CHECK

**Technical:**

You define a JavaScript routine that can access widget reference variable or SQL variable. An example can be obtained by clicking the question mark icon.

If the data row contains an error, an appropriate error message will be displayed automatically.

**Hint:** There is tutorial video 0410 Data Quality which handles this feature in the Training menu of the Apparo Designer

#### 6.7.17 Data transaction handling

##### ▼ Data transaction handling



#### Data transaction handling

Auto-commit: Store all data changes immediately into the database

YES

#### **Auto-commit: Store all data changes immediately into the database**

All data changes are committed immediately. The user can't roll back data changes, if the user leaves the Business Case and closing the window with right upper corner x-icon, then all data changes are stored too.

If this feature is enabled, then every change is stored ASAP (committed).

The Excel data import is committed only once at the end in success case, because of performance reasons.

### 6.7.18 Automatic scripts and database procedures

It is possible to start a shell-script, database function/procedure or SQL-script before Business Case /server side file import starts, and/or after it is finished before forced Excel row import starts, and/or after it is finished after the user has inserted or updated data row

With Pre/Post-Execution it is possible to run automatically a script or a database procedure/function at certain moments.

**Other options are:**

#### JavaScript commands

Allows you to enter JavaScript commands that will be executed when the event is triggered

#### SQL commands

Allows you to enter SQL commands that will be executed when the event is triggered

#### Database procedure

Allows you to run stored procedures or functions with parameters that will be executed when the event is triggered. Variables are allowed.

Return values can be stored within a variable and used within the Business Case, except for events after closing the Business Case.

#### Anonymous database block

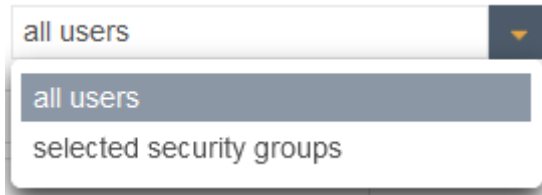
Is similar to a stored procedure, but can be entered directly within the Business Case. You don't need database access.

Allows you to run an anonymous database block with parameters that will be executed when the event is triggered. Variables are allowed.

#### Script on server

Allows you to run a script (batch-, SQL- or JavaScript file) with parameters that will be executed when the event is triggered. Variables are allowed.

This behavior can be defined for all or for users that are members of a specified group.  
If the current user is member of a specified group then just the shell-script, database function/procedure, SQL-script or JavaScript file of this group is executed only.



In all other cases the default script/function/procedure is called only.

The commands are executed using the same database session like the Business Case and are separated by a semicolon.

Currently, Apparo Fast Edit is supporting Oracle, Microsoft SQL Server, IBM DB2, Sybase ASE/IQ (chained mode only) and Teradata databases.

#### 6.7.18.1 Pre Business Case execution

Allows to run automatically a script or database function/procedure if the user is starting the Business Case

#### 6.7.18.2 Post Business Case execution in success case

Allows to run automatically a script or database function/procedure after the user has closed the Business Case with 'Ok' or 'Close' button

#### 6.7.18.3 Post Business Case execution in failure case

Allows to run automatically a script or database function/procedure after the user has closed the Business Case with 'CANCEL' or 'X' button)

Fast Edit is checking the browser state by default every 180 seconds, it may take up to 3 minutes after closing the BC with 'X' before the script/procedure is executed.

#### 6.7.18.4 Post insert execution

Allows to run automatically a script or database function/procedure after a new row was inserted

*This insert can be done:*

- From inserting area (Table Business Case)
- From insert mode (Single Business Case)
- From Excel file import
- From Excel row import using copy and paste
- From automatic server import
- From automatic import of email data-file attachments
- By copying row/s in the same window

The procedure or script will **NOT be executed after modifying a row in edit area.**

#### 6.7.18.5 Post update execution

Allows to run automatically a script or database function/procedure after a row was updated

*This update can be done:*

- From inserting area (Table Business Case)
- From insert mode (Single Business Case)
- From Excel file import
- From Excel row import using copy and paste
- From automatic server import
- From automatic import of email data-file attachments
- By copying row/s in the same window
- After modifying a row in edit area

Optionally, a query window to activate that appears when the user updates a row of data from the input area.

#### 6.7.18.6 Post Excel import execution

Allows to run automatically a script or database function/procedure after any kind of Excel import has finished

*All Apparo Fast Edit variables can be used here, including:*

- <%IMPORTED\_ROWS%> count of imported rows
- <%INSERTED\_ROWS%> count of inserted rows
- <%UPDATED\_ROWS%> count of updated rows
- <%IMPORTED\_FILE\_NAME%> file name of the imported file (if applicable)
- <%EXCEL\_IMPORT\_ID%> An unique ID of type string for each Excel import

#### 6.7.18.7 Post Widget cell value update

Allows to run automatically a script or database function/procedure if the user is changing a widget cell value (and leaving the widget)

**Post widget cell value update execution (allows to run automatically a script or database function/procedure after a widget value was updated)**

Maximum number of parallel executing threads:

☒ Enable dialog window that will be shown after user has updated a widget value from editing area.

**Dialog window settings**

'Please wait' message text: All variables can be used in this input field

Language	'Please wait' message text
German	Bitte warten...
English	Please wait...

'Please wait' message font: Font face:  Size:  Style:  Align:  Colour:

After the user changes a widget value, a script / database procedure / anonymous database block is called for this change.

The call is made per widget value, i.e. if the user changes 10 values, the action is called 10 times.

The calls can be made in parallel if multiple threads are allowed.

The maximum number of parallel threads can be set per Business Case.

Optional a message window can be activated.

**The action will be defined on widget level:**

#### Widget settings of database column OFFICE\_ID

**Widget type** **Mapping & Other** **Features** **Actions** **Visual** **Help texts** **Data output format**

☒ Call a script/procedure if the value of this widget was changed by user in a web browser

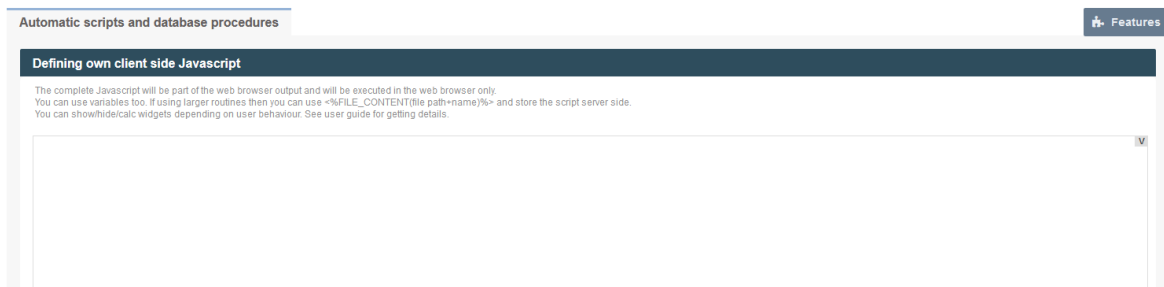
Note: If the user is changing many values of this widget then it is possible to call the script/procedure in a parallel way for reducing waiting time.

Automatic execution of:

Name:

#### 6.7.18.8 Defining own client side Javascript

Allows the use of client side custom JavaScript functions and the use of special Apparo JavaScript methods like `afe.callClassMethod` as mentioned earlier in this chapter.



Especially for this area you can use special methods to read and change widgets values

A video guide using these methods to implement a planning feature is available in the Apparo Designer in the training menu or online on [movies.apparo.solutions](http://movies.apparo.solutions) and also described later in this user guide in the chapter "Planning application using a Table Business Case".

#### Special Apparo Methods

##### *getAfeTableWidgetValue*

Supported are

- `getAfeTableWidgetNumValue` – reads and handles the values as numeric values
- `getAfeTableWidgetStringValue` - reads and handles the values as string values

##### *setAfeTableWidgetValue*

Supported are

- `setAfeTableWidgetNumValue` – writes the values as numeric values
- `setAfeTableWidgetStringValue` - writes the values as string values

### 6.7.19 Auditing of data changes

The audit function you can use to document all data changes.

There are 2 different types of audit:

#### 6.7.19.1 Simple Auditing

To save the audit information into the target table.

The screenshot shows the 'Simple auditing settings' window. It has a tab 'Auditing of data changes' and a 'Features' button. The main area is a table with the following structure:

Simple auditing settings	
	Inserting a new row case
User name column	USER_ID
Date column	STAMP
State (U,I,D) column	
Row edit type column	
In delete case delete data row physically	<input type="checkbox"/>

There is also a section for 'Updating or deleting row case' with similar fields for 'USER\_ID' and 'STAMP'.

It is possible to save the user name, date and time and the type of change for each row in the target table.

There are 2 different types of changes possible:

- The user adds a new row
- The user deletes or modifies a row.

The following states are possible: U = Update, I = Insert, D = Delete

#### Options

##### User name column

Stores the name of the user

##### Date column

Date column for storing update or insert date and time

##### State (U, I, D) column

The database column in that the state (U=Update, I=Insert, D=Delete) will be stored.

##### Row edit type column

In this auditing column the row edit type can be stored. The row edit type (type of string) is describing the way of editing.

##### In delete case delete data row physically

Physically delete row(s) with 'D' flag from table. If disabled then all deleted rows get the state 'D' and are not physically deleted.



### 6.7.19.2 Detailed Auditing

Storing detailed audit information in a separate audit database table helpful if every small change (eg a column) with name, timestamp, old&new values etc. should be documented.

Detailed auditing settings			
Auditing database table	SAMPLES_ADV_AUDITING		
Auditing column for user name	USERNAME		
Date column	CHANGE_DATE		
State (U,I,D) column	STATE_TYPE		
Row edit type column	ROW_EDIT_TYPE		
Custom value column	CUSTOM_VALUE		
	<%AFE_BC_FOLDER%>		
Target table name column	TARGET_TABLE		
Business Case identifier column	BCID		
SQL statement column	SQL_COMMAND		
Summary change column			

Widget mapping			
Editing widget	Auditing column for the old value		Auditing column for the new value
OFFICE_ID (NUMBER)	OLD_NUM_1		NEW_NUM_1
PRODUCT_LINE_ID (NUMBER)	OLD_NUM_2		NEW_NUM_2
PRODUCT_ID (NUMBER)	OLD_NUM_3		NEW_NUM_3

### Options

#### Database schema

The database schema in that the auditing table is already stored.

#### Auditing database table

The database table for the auditing data.

#### Auditing column for user name

The database column of the auditing table in that the user name who has changed data will be stored.

#### Date column

The auditing column for storing the date/time of the data change.

#### State (U, I, D) column

The auditing table column in that the state must be stored (U=update, I=insert, D=delete).

#### Row edit type column

In this auditing column the row edit type can be stored.

#### Custom value column

In this auditing column a custom value with variables can be stored that is stored in the auditing table only.

#### Target table name column

In this auditing column the name of the target table of this Business Case can be stored.

#### Business Case identifier column

In this auditing column the Business Case ID (short name) can be stored.

#### SQL statement column

In this auditing column the SQL statement can be stored. Be sure that this column can store a long text.

#### Summary change column

This text contains all data changes in one string like oldValue=1, newValue=2,..

## 6.7.20 Data History

Apparo Fast Edit can historicize a record (slowly changing dimension type 1 and 2).  
Information about "**Slowly changing dimension**", see:

[http://en.wikipedia.org/wiki/Slowly\\_Changing\\_Dimensions](http://en.wikipedia.org/wiki/Slowly_Changing_Dimensions)

**Note: For a historicizing the database must be able to perform "save points".**

**Since the Sybase / Teradata JDBC driver does not support this feature, the historicization of records within a Sybase or Teradata database is not possible.**

This function automatically copies data rows when they are modified.  
It automatically manages the current record and makes it possible to either overwrite or historicize records within time frame definitions.

The user usually sees only the current line and not the data changes or deletions (if only virtually), the new rows are simply copies of the original lines.

Hint: There is detailed description document with use case on our website under Docs or download it from [<<here>>](#)

Main

Advanced

Features

Using history functionality means that Apparo is automatically copying a data row if the user is changing a value into it and it is managing the 'date from'/'date to' columns automatically. Depending on the decision strategy (time resolution) it is updating the row or copying including time frame update. The row group normally contains all used primary key widgets. It must not contain date\_from & date\_to columns. For auto-handling the date from widget, it requires e.g. the '<CURRENT\_DATE>' variable set as constant value in insert case or as default value. Please set the output type of this widget to at least date with time (medium). For milliseconds use the custom output type e.g. 'dd.MM.yyyy:hh:mm:ss.SSS'. Please read this [document](#) for more details.

Use columns as row group

OFFICE\_ID

PRODUCT\_LINE\_ID

PRODUCT\_ID

SALES

STATUS\_ID

STATE\_REVISION\_ID

FORECAST

FORECAST2

FORECAST3

FORECAST4

FORECAST5

FORECAST6

The history functionality is helpful if the user is updating data values or creating new data rows and the Business Case must create automatically a copy of this complete data row to be able to see the history of the data changes.

The Business Case is managing automatically 'date from', 'date to' and 'current' columns of the target table. With these date columns it is possible to see the time dependencies of the changes. Apparo is combining data rows together to a 'row group'. A 'row group' are data rows that are storing detail information about an entity, for example a 'product entity' has many different prices over the time.

Depending on the decision strategy (time resolution) it is definable when the Business Case is updating the current row or making a copy of this data row. If for example the decision strategy is 'same hour' and there are 2 changes in the same hour then the second change is just updating the data row. If the second change is for example 5 hours later then the Business Case is creating a new data row including update of 'date from', 'date to' and 'current' column. Please don't use widgets(columns) for the history feature that contain read/write expressions.

Example:  
The Business Case is managing the price of a product. The 'row group' is the column 'product\_id'. The primary key is 'product\_id' and 'date\_from'.  
'product\_id' 'date\_from' 'date\_to' 'price'  
5 20.12.2010 31.12.2999 100

Now the user is changing the price to '105', editing date: 05.01.2011  
5 20.12.2010 04.01.2010 100  
5 05.01.2010 31.12.2999 105

Time resolution

Same second

If there are 2 or more data changes into a row in the same time frame then Apparo will update the row only. If the next change is outside of the time frame then Apparo is copying automatically this row and changing the 'date from' and 'date to' columns automatically too. Special case: If simple auditing is used with specified 'state (U,I,D)' column then deleting a data row will insert a new data row with flag 'D' instead of setting the flag to the same data row. (= 'Slowly changing dimension' strategy)

'Valid from' date column

VALID\_FROM\_TS

☐ Validate that the new 'valid from' value precede the latest existing 'valid from' value of this row group.

The 'valid from' database column of the target table is used for storing the begin of the time frame for a row. Hints: This column must be a part of the primary key.

The database column type must be able to store the complete datetime. If you want to use milli second time resolution then these types are possible: MS SQL Server: datetime2, Oracle: timestamp, IBM DB2: timestamp

'Valid to' date column

VALID\_TO\_TS

The 'last active' database column of the target table is used to mark the last record of a row group. In most cases, this is the current entry, but it can also be an entry that lies in the future. This setting is optional. It is automatically managed by the Business Case.

'Last active' flag column

CURRENT\_FLAG

The 'last active' database column of the target table is used for marking the last active row of a group. This setting is optional. It is automatically managed by the Business Case.

The Business Case is managing automatically 'date from', 'date to' and "current" columns of the target table. With these date columns it is possible to see the time dependencies of the changes.

## Background

Apparo Fast Edit is combining data rows together to a 'row group'. A 'row group' are data rows that are storing detail information about an entity, for example a "product entity" has many different prices over the time.

Please don't use widgets (columns) for the history feature that contain read/write expressions.

## Time resolution

If there are 2 or more data changes into a row in the same time frame then Apparo Fast Edit will update the row only. If the next change is outside of the time frame then Apparo Fast Edit is copying automatically this row and changing the 'date from' and 'date to' columns automatically too.

### 'Valid from' date column

The 'valid from' database column of the target table is used for storing the begin of the time frame for a row.

**Validate that the new 'valid from' value precede the latest existing 'valid from' value of this row group.**

Useful for manually entered valid from values

HINT: this column must be a part of the primary key

### 'Valid to' date column

The 'valid to' database column of the target table is used for storing the end of the time frame for a row. This setting is optional. It is automatically managed by the Business Case .

### Use for current flag

The 'current' database column of the target table is used for marking the current row of a group. This setting is optional. It is automatically managed by the Business Case .

## Advanced settings

It contains the settings for the infinity date, the current flag and other settings.

Main	Advanced
Date for infinity	<input type="text" value="12.31.2999 0:0:0"/> The infinity date is used in the 'date to' column for the current data row. The date format is MM.dd.yyyy H:m:s where <ul style="list-style-type: none"> <li>d - day of month (1-31)</li> <li>M - month of year (1-12)</li> <li>yyyy - year</li> <li>H - hour</li> <li>m - minute</li> <li>s - second</li> </ul> e.g. 1.1.2999 1:45:45 or 12.31.2999 11:11:11
History flag for current row	<input type="text" value="1"/> * Here you can define the value for the current column for the data row with the current values Specify number in format "###.###.###" or for text you can use any character
History flag for not current row	<input type="text" value="0"/> * Here you can define the value for the current column for all non-current data rows Specify number in format "###.###.###" or for text you can use any character
Set historical entries to read-only	<input type="checkbox"/> If enabled then data records with a valid-to date that lie in the past cannot be deleted or changed.
Don't delete entries physically	<input type="checkbox"/> If enabled and the user deletes a record with valid-to date in the future, the entry is not physically deleted, but the valid-to date is set to the current date and will be historicized.
Users can enter custom valid-to dates	<input type="checkbox"/> If enabled, the user can set custom valid-to dates for the current row. By default, the valid-to date is taken from the feature settings (infinity date) and used for the current row.
Use advanced constant value settings for 'valid-from' widget	<input checked="" type="checkbox"/> If enabled, the use of constant values of the widget 'valid-from' can be defined in a more detailed way.

**Advanced constant value settings for 'valid-from' widget**

The constant value for inserting a new history group entry	<input type="text" value="&lt;%TIMESTAMP%&gt;"/> V Use format 'yyyy-MM-dd HH:mm:ss.SSS' e.g. 2019-09-25 14:59:59.000
The constant value for editing an existing group entry	<input type="text" value="&lt;%TIMESTAMP%&gt;"/> V Use format 'yyyy-MM-dd HH:mm:ss.SSS' e.g. 2019-09-25 14:59:59.000
If activated, the user can manually overwrite the set constant values	<input type="checkbox"/>

If left empty, no constant values are used.  
Set constant values are valid for all security groups if using security group-dependent constant values.

## Date for infinity

The infinity date is used in the 'date to' column for the current data row.

The date format is: MM.dd.yyyy H:m:s

Where

- d - day of month (1-31)
- M - month of year (1-12)
- yyyy - year
- H - hour
- m - minute
- s - second

e.g. 1.1.2999 1:45:45 or 12.31.2999 11:11:11

### **History flag for current row**

Here you can define the value for the current column for the data row with the current values  
Specify number in format "###,###.###" or for text you can use any character.

In most cases the value '1' is used here.

### **History flag for not current row**

Here you can define the value for the current column for all non-current data rows  
Specify number in format "###,###.###" or for text you can use any character

In most cases the value '0' is used here.

### **Set historical entries to read-only**

If enabled then data records with a valid-to date that lie in the past turns to read-only and cannot be deleted or changed.

### **Don't delete entries physically**

If enabled and the user deletes a record with valid-to date in the future, the entry is not physically deleted. In this case the valid-to date is set to the current date instead and will be historicized.

### **Users can enter custom valid-to dates**

If enabled, the user can set manually custom valid-to dates for the current row.  
By default is the valid-to date taken from the feature settings (infinity date) and used for the current row automatically.

### **Use advanced constant value settings for 'valid-from' widget**

If enabled, the use of constant values of the widget 'valid-from' can be defined in a more detailed way.  
If this feature is activated, but the constant values for inserting and/or editing were left empty, then no constant values are used.  
Set constant values are valid for all security groups if using security group-dependent constant values.

### **The advanced constant value settings for 'valid-from' widget offers three options:**

1. *The constant value for inserting a new history group entry*

You can use custom or predefined variables here (e.g. <%TIMESTAMP%> or <%CURRENT\_DATE%>) or a fix date value with the format 'yyyy-MM-dd HH:mm:ss.SSS'  
e.g. 2019-09-25 14:59:59.000

2. *The constant value for editing an existing group entry*

You can use custom or predefined variables here (e.g. <%TIMESTAMP%> or <%CURRENT\_DATE%>) or a fix date value with the format 'yyyy-MM-dd HH:mm:ss.SSS'  
e.g. 2019-09-25 14:59:59.000

3. *If activated, the user can manually overwrite the set constant values*

Users can enter manually custom values instead of using the given constant values.

### 6.7.21 Security

This limits the general access to the Business Case (whitelist).

Only users who are member of at least one of the entered groups can use this Business Case.

Security groups are to be entered separated by commas.

Security

Comma separated list of security group names. Only users, who are members of at least one of these groups, will be able to open this Business Case.  
Wildcards \* and % can be used in group names both representing any number of characters. Example: controlling\* will accept groups like controllingAfrica, controllingUSA, controllingEurope etc.

Security\_Group\_A,Security\_Group\_B,  
Security\_Group\_C,Security\_Group\_D

Security group

Wildcards \* and % can be used in group names both representing any number of characters.

Example: controlling\* will accept groups like controllingAfrica, controllingUSA, controllingEurope etc.

## 6.7.22 Limited Access

The limited access limits the possibilities of a Business Case to output data only.

In the limited access mode can:

- No data be entered or changed (neither manually, yet over the Excel Import)
- No scripts or database procedures (functions) get started
- Buttons only limited be used

Limited access (readonly mode)

Features

If this Business Case is limited then all widgets in edit area are read-only and:

- "Save" button is hidden
- "Insert" button is hidden
- Inserting is disabled
- Excel row/column import button is disabled
- Manual data file import is disabled
- Locking is disabled
- Pre/post execution is disabled

Select limited access type:

☒ Limited for all

- The complete Business Case is read-only

☐ Limited for security groups

- Only members of the defined roles or groups can not change data

☐ Limited if variable returns 'true' value

- Limited if the result of a variable is 'true'

The limited access may be restricted

- for specific security groups
- when a variable returns true (for example, to avoid entering of data during maintenance periods)

### 6.7.23 My own database error messages

When the database an error returned is the original message shown by default. With this feature you can define own more understandable messages for your users.

The field SQL status is optional, but may help to group error messages.

If you want to define your own messages, you must first import the DB type template file, stored in the 'dbmessages' path.

#### Example:

To define a custom message for Oracle Code ORA-02291 enter '02291' in the field "SQL Error Code".

My own database error messages

Features

Delete

Custom SQL messages

<input type="checkbox"/>	Database type	Language	SQL code	SQL state	Message text
<input type="checkbox"/>	Oracle	English	0001		Duplicate key! Please check the ID you entered.

Add new SQL message


Database type	Language	SQL code	SQL state	Message text	Action
Oracle	German				Add



## 7 Single Business Cases (SBC)

A single Business Case (SBC) is used to represent a single data set (database row). A typical application is a data entry screen or a detailed view.

The functions and settings of the SBC are substantially identical to those of Table Business Cases. This chapter focuses on the features and the settings that apply only to the single Business Case .


**Apparo Fast Edit**

**administrator**
**Demo**

**Dynamic Forms**

Based on the selection of values in certain widgets, depending widgets will be shown up.  
**If the checkbox 'Special optical features' is checked, a description field is shown**  
**If the kind of transmission is 'Manual', another widget is shown, which requires to enter the amount of gears**

Car name	Honda CRV
Color	red
Special optical features?	<input type="checkbox"/>
Transmission	Manual
Number of gears	11
Drive	all wheel drive 4x4
Driver	Fero

Page: 1 / 5 **ENTER NEW DATA RECORD**

OK
CANCEL
CLOSE
DELETE

User view of a SBC, the widgets are visually divided into 2 columns.

### 7.1 Structure of the SBC

- **Header area** - with the title and description
- **Data area** - where the widgets are arranged in columns
- **Navigation pane** - used to navigate between records and the switch button for the data input mode
- **Button area** - contains the default and user-defined buttons
- **Optional footer area** - for info and graphics

## 7.2 Arrangement of the widgets in the SBC

The widgets in the SBC can be output with multiple columns, the order is determined by the line.

The screenshot shows the 'Widgets' tab in the SBC configuration tool. The interface includes a top navigation bar with tabs: Target table, Header, Footer, Visual, Colours, Widgets (selected), Row ordering, and Link into Portal. Below the navigation bar are buttons for '+ Add' and 'X Delete'. The main area is titled 'Editing widgets' and displays a single column of widget configurations. Each widget is represented by a header bar with a 'Drag here' icon and a list of fields with their respective data types (PK, RO, H, NN) and checkboxes.

Widget	Fields	PK	RO	H	NN
Widget 1	ID, Input field, Id	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Widget 2	NAME, Lookup dropdown (for all tables), Car name	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Widget 3	COLOR, Lookup dropdown (for all tables), Color	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Widget 4	GEAR, Input field, Number of gears	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Widget 5	OPTICAL_FEATURES, Checkbox, Special optical features?	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Widget 6	DESCRIBE_OPTICS, Text area, Describe optics	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Widget 7	TRANSMISSION, Lookup dropdown (for all tables), Transmission	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

**Designer view:** The arrangement of the widget by columns and rows

### Move widget(s) to another column

Use the mouse to drag the widget to the desired position

The screenshot shows the 'Widgets' tab in the SBC configuration tool, displaying a multi-column layout. The interface is similar to the previous one, but the widgets are arranged in four columns. The first three columns contain the same widget configurations as in the previous screenshot. The fourth column is empty, with a 'Drop here to place the widget' label. The 'TRANSMISSION' widget is highlighted in green, indicating it is being moved or is the active widget.

Widget	Fields	PK	RO	H	NN
Widget 1	ID, Input field, Id	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Widget 2	NAME, Lookup dropdown (for all tables), Car name	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Widget 3	COLOR, Lookup dropdown (for all tables), Color	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Widget 4	GEAR, Input field, Number of gears	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Widget 5	OPTICAL_FEATURES, Checkbox, Special optical features?	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Widget 6	DESCRIBE_OPTICS, Text area, Describe optics	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Widget 7	TRANSMISSION, Lookup dropdown (for all tables), Transmission	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

### Dynamic Forms

Based on the selection of values in certain widgets, depending widgets will be shown up.  
If the checkbox 'Special optical features' is checked, a description field is shown  
If the kind of transmission is 'Manual', another widget is shown, which requires to enter the amount of gears

Row 1  
Row 2  
Row 3  
Row 4

	Column	Column 2	Column 3
Car name	Honda CRV	Special optical features? <input checked="" type="checkbox"/>	Transmission Manual
Color	red	Describe optics	Drive all wheel drive 4x4
Number of gears	11		
Driver	Fero		

Page: 1 / 5 ENTER NEW DATA RECORD

OK CANCEL CLOSE DELETE

User view

## 7.3 Visual

Here you define the general optical settings for the single Business Case .  
These settings are different from those in Table Business Case

Target table	Header	Footer	Visual	Colours	Widgets	Row ordering	Link into Portal	Features						
Label width (px)		<input type="text" value="100"/> *												
Widget width (px)		<input type="text" value="200"/> *												
Visual column label widths		<input type="text" value="100,105,100"/>												
Visual column widget widths		<input type="text" value="200,205,200"/>												
Gap width between data rows(px)		<input type="text" value="2"/> *												
Hide application header		<input type="checkbox"/>												
Enable a general button bar for rich text widgets		<input type="checkbox"/>												
Enable dialog window with error message in case of error		<input checked="" type="checkbox"/>												
Window background image URL		<input type="text" value=""/> V												
Show just the first data row only		<input type="checkbox"/>												
No data to display message		<table border="1"> <thead> <tr> <th>Language</th> <th>No data to display message</th> </tr> </thead> <tbody> <tr> <td>German</td> <td><input type="text" value="Keine Daten zum Anzeigen"/> V</td> </tr> <tr> <td>English</td> <td><input type="text" value="No data to display"/> V</td> </tr> </tbody> </table>							Language	No data to display message	German	<input type="text" value="Keine Daten zum Anzeigen"/> V	English	<input type="text" value="No data to display"/> V
Language	No data to display message													
German	<input type="text" value="Keine Daten zum Anzeigen"/> V													
English	<input type="text" value="No data to display"/> V													

### Options

#### Label width (px)

Width of the label in pixels

#### Widget width (px) \*

Width of the input area of the widget

#### Visual column label widths

Defines the visual column label widths. If no value is defined for visual column label width then 'Label Width' property is used as default value; e.g.

100,150,200	3 visual columns with label widths 100 (px), 150 (px) and 200 (px)
100,,200	3 visual columns with label widths 100 (px), 'Label Width' (px) and 200 (px)
,,200	3 visual columns with label widths 'Label Width' (px), 'Label Width' (px) and 200 (px)

IMPORTANT: Negative numbers are not valid.

### Visual column widget widths

Defines the visual column widget widths. If no value is defined for visual column widget width then 'Widget Width' property is used as default value; e.g.

100,150,200	3 visual columns with widget widths 100 (px), 150 (px) and 200 (px)
100,,200	3 visual columns with widget widths 100 (px), 'Widget Width' (px) and 200 (px)
,,200	3 visual columns with widget widths 'Widget Width' (px), 'Widget Width' (px) and 200 (px)

IMPORTANT: Negative numbers are not valid

### Gap width between data rows(px)

The optical gap between data rows in pixels, default is 2

### Hide application header

Hides the blue application header

### Enable a general button bar for rich text widgets

When this feature is enabled, the user sees only a button bar with buttons for bold, italic, underline, etc. for changing the text style in rich text widgets.

This button bar is necessary when using text area type widgets with rich text functionality (bold, italic, underline, use different colors). The general button bar is visible as in Microsoft Word. If it is disabled, each text area with rich text functionality has its own button bar.

### Enable dialog window with error message in case of error

If this setting is enabled than a pop-up dialog window will be displayed after each error.

### Window background image URL

It is possible to use own background picture using an URL.

### Show just the first data row only

When enabled only first data row will be displayed; otherwise additional buttons "<<" and ">>" will be displayed to show previous respectively next data row.

### No data to display message

Message presented to user when there are no data to display

## 7.4 Starting the Single Business Case in Insert mode using URL parameter

The URL parameter insertMode=true allows to start the SBC in insert mode.

Usage example (Cognos):

```
http://SERVER_NAME/ibmcognos/cgi-bin/cognos.cgi?b_action=xts.run&m=portal/bridge.xts&c_env=/portal/FastEditBusinessCase.xml&c_mode=post&bc=BC_NAME&clientid=CLIENT_NAME&backLink=%252Fcontent%252Ffolder%255B%2540name%253D%2527Apparo%2BFast%2BEEdit%2527%255D&insertMode=true
```

## 8 Business Case Sets (Set)

Sets group multiple Business Cases in a tab view. The Business Cases can be accessed with tabs and edited comfortable.

### 8.1 Selection and positioning of Business Cases in the set (Set)

**Main settings**

Identifier / Short name: Set

Business Case name: Set

Security group:

Select Business Cases

Available	Selected
\Demo page #11 small applications\SAMPL MASTER PROD - SMALL APP	\Demo page #11 small applications\SAMPL MASTER PROD LINES - ENTRY
\Demo page #11 small applications\SAMPL MASTER PROD PRICE	\Demo page #11 small applications\SAMPL MASTER PROD DETAILS - App
	\Demo page #11 small applications\SAMPL MASTER PROD LIST - APP

☒ Show just Business Cases of current folder only

☒ Show folder path

Notes

In 'Available' you find all existing Business Cases.

By double-clicking or using the arrow keys, these are assigned to the set.

The positioning within the set is also done via arrow keys or the mouse.

By holding down the Ctrl key you select multiple Business Cases and move it to the desired position.

## 8.2 Colors

In colors you can set the color of the tabs (tab):

Colours

Inactive tab background colour

#E7E7E7

\*

Inactive tab text colour

#336699

\*

Active tab background colour

#336699

\*

Active tab text colour

#FFFFFF

\*

## 8.3 Tab Widths

In Table width you define the width of the tabs

Tab widths

Business Case name	Tab width
SAMPLES - product lines	200
SAMPLES - product details	200
SAMPLES - product list	200

## 8.4 Global Set filters

A global filter is a connection between different filter widgets of different Business Cases of a Business Case Set. That is helpful if some Business Cases of this Set must be filtered in the same way when if the user is jumping to another Business Case.

Example: All Business Cases must filter for the same product and the user is selecting the product just once. It is possible to use many different global filters parallel, e.g. for product and for product-line.

All existing filter widgets of the Business Cases in the Set are listed here.

To create a global Set filter, move all related filters to 'Selected filter widgets' and hit OK.

### Global filter



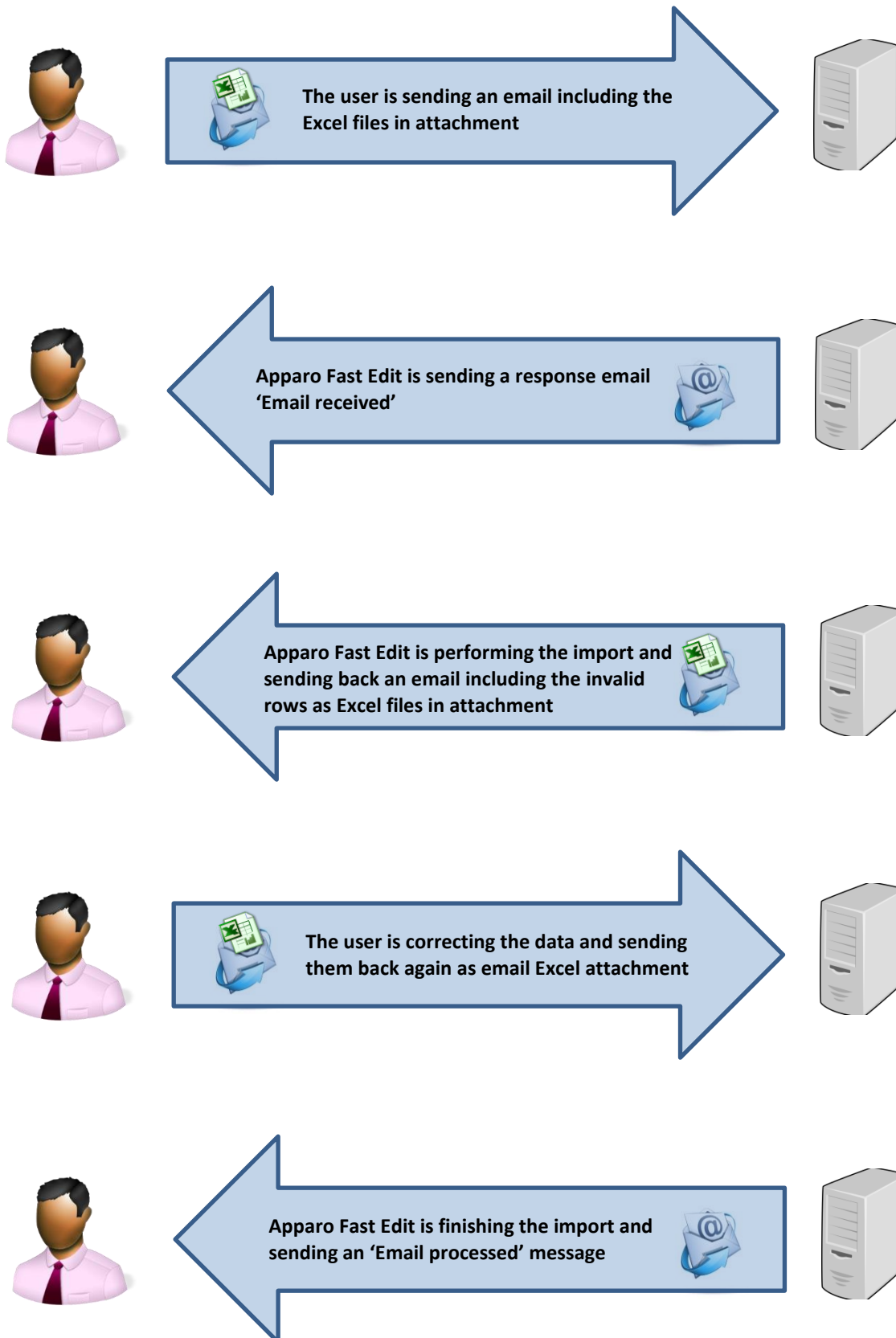
## 9 E-Mail Import Business Case (EIBC)

The Excel Email Import feature enables you to import data that is stored in Excel files (CSV, XLS, XLSX) from email attachments.

That means the user can send an email with **Excel files in attachment and the data of these Excel files will be imported automatically in your relational databases.**

The user is getting automatically answer-emails that are informing the user about the progress or data quality issues.







All activities can be logged in database table, the emails and attachments can be stored physically on the server.



## 9.1 Creating a new Business Case of type 'Email import'

When clicking on New Business Case in the Business Case list the following selection list will appear:

Please select type of Business Case you want to create now

	<b>Table</b>	A table Business Case is showing many data rows on the same page. The user can filter the data, edit, import from Excel, export to Excel and so on.
	<b>Single</b>	A single Business Case is showing just one data row only.
	<b>Set</b>	A grouping of multiple Business Cases (table/single) for more comfortable usage. You can define global filters that are filtering all Business Cases automatically too.
	<b>Email import</b>	Importing Excel data directly by email - send Excel sheets using email attachments and Apparo will import the Excel data directly into the database including file uploads. No web browser is necessary, just an email.
	<b>Email</b>	An eMail Business Case is a definition of an email text including usage behavior and can be used in another Business Cases of type 'table' or 'single' only. In these Business Cases it is possible to define buttons that can use this eMail Business Case.
	<b>Action</b>	Purpose of Action Business Case is to execute scripts or database procedures that can be called from a report/HTML page. Usage of AJAX and Javascript for automatically executing in the background is possible too.

CANCEL

Click on Email Import to create a new 'Email Import Business Case'

For the setup we will need a pre-defined email connection and at least one database connection, used for the Business Case that will perform the import.

These Business Cases are also containing all definitions for securing the data quality.

The Business Cases must have activated the Excel file import feature.

### 9.1.1 New Business Case - Main Settings

The main settings require the following settings:

- **Identifier:** The short name of the Business Case (must be unique)
- **Business Case Name:** This name will appear as name when we link the Business Case to the portal
- **Email connection:** The email connection for sending and receiving emails
- **Internal description:** Optional. For documentation purposes only.

## Email Import Business Case (EIBC) - Main settings

Identifier / Short name	<input type="text" value="Sales_figures"/>	*
Business Case name	<input type="text" value="Email import sales"/>	*
Email connection	<input type="text" value="Email2"/>	▼ *
Enabled	<input checked="" type="checkbox"/>	
Notes	<input type="text" value="For importing the monthly sales via email"/>	

Fill all necessary fields and click 'Next' to create the Business Case

## 9.2 Overview of all possible settings

Once the Business Case is created we will see the following overview.

Here you can save and close the Business Case and click through the tabs of the settings:

- **Introduction:** Contains usage examples and explanations
- **Main Settings:** Contains the main settings and the server path for storing emails and attachments
- **Importing Groups:** Contains the import groups, the detailed settings how shall be imported
- **eMails:** Here you can define the text of a failure email, for the case that no import group is matching
- **Logging:** Contains the logging settings, details like user name can be mapped here to a database column
- **Variables:** Contains predefined variables and you can create own JavaScript variables

The screenshot shows the 'Main Settings' tab selected in the sidebar. The main content area displays the 'Email file system storage' settings. The fields are as follows:

Field	Value
Identifier / Short name	Sales_figures
Business Case name	Email import sales
Email connection	Email2
Enabled	<input checked="" type="checkbox"/>
Notes	For importing the monthly sales via email

### 9.3 Main Settings

This tab contains, beside from the main settings, the path for storing emails and attachments physically on the server.

Main

Email file system storage

Path for storing the email content

<%AFE\_HOME\_DIR%>\emails

V

Email file system storage

### 9.4 Importing Groups

An import group contains the definitions of which attachments are expected and which business case should perform the import. It also contains the security settings, log settings and the reply email texts.

Import Groups

+ Add new import group

List of defined import groups for this email import Business Case

Order number	Change order	Import group name	Description	Enabled	Actions
1	⬇ ⬆ ⬇	> Group1		<input checked="" type="checkbox"/>	✕

Adding a new import group

## 9.5 Importing group settings

### 9.5.1 Main group settings

It contains the import group name (unique) and an optional description text. You can enable or disable the import group here.

#### Email import group settings

The screenshot shows a dialog box titled "Email import group settings". It has four tabs: "Main group settings", "Business Cases", "Email texts", and "Security". The "Main group settings" tab is active. It contains the following fields:

- Import group name:** A text input field containing "Group1". There is a red asterisk to its right.
- Description:** A larger text input field, currently empty.
- Enabled:** A checkbox that is checked.
- Error handling strategy:** A dropdown menu showing "Rollback import if there are invalid rows".
- Error list file format:** A dropdown menu showing "XLSX".
- Language:** A dropdown menu showing "English". There is a red asterisk to its right.

At the bottom of the dialog are two buttons: "OK" (green) and "CANCEL" (orange).

#### Import group name

The unique name for this import group

#### Description

Optional: Short description of a use case. For example, "Import profile for collecting monthly sales data".

#### Enabled

The import group can be disabled here.

#### Error handling strategy

When there is a data error in an email attachment, there are two possible strategies to deal with it. Either the entire import process is aborted and no data is written, or the process skips rows with errors and saves the good ones. This value overrides the 'Excel import strategy' of the Table Business Cases.

#### Error list file format

If an import was completed successfully and some errors occurred, the user will receive the errors as an attachment file in the specified format.

For CSV format, you must specify a separator. If you want to use a tab, please enter '\t'.

#### Language

For validation errors in email attachments, users receive validation messages in the selected language.

## 9.5.2 Business Cases

Contains the mapping to all Business Cases that are defined to perform the import. When an email from a valid sender arrives, Apparo Fast Edit automatically analyses the structure of the attachments and comparing the structure with the defined import groups.

### Email import group settings

Main group settings
Business Cases
Email texts
Security

ADD NEW BUSINESS CASE

Please notice, that an email, in order to be accepted by this import group, all its attachments (excluding images) must be mapped to the defined table Business Case files. For example if the email contains a text file attachment that stays unused, the email will not be accepted by the import group.

Table Business Cases to be executed in following order					
Order number	Change order	Table Business Case identifier	Notes	Excel file attachments	Actions
1	⬅ ➡	> SAMPL WF WORKFLOW2		> *.excel	✕

OK
CANCEL



### 9.5.2.1 Add new attachment

You need to define at least one email attachment for every defined importing Business Case.

**Email file attachment settings - Excel file**

**Basic settings**   **Advanced Excel document data constraints**

*Please note, that only the first sheet of an excel file will be imported.*

File name pattern:  \*

Description:

Min. and max. occurrences:  –  \*

Language:  ▼

Creating a new file attachment

The following properties are expected:

- **File name pattern:** Defines the allowed file extension (\*.excel allows all Excel files: .xls, .xlsx, .csv)
- **Description:** For the internal documentation
- **Starting row:** For the case it contains a header in row 1, we start the import in row 2
- **Min and Max occurrences:** The minimum should be at least 1 – the user gets an error email if the attachment contains less attachments than expected
- **Language:** Important for language sensitive data types e.g. dates and numbers

### 9.5.2.2 Advanced Excel document data constraints

This feature is optional:

Here you can define the expected data column types, this feature allows Apparo Fast Edit to better distinguish similar Excel file attachments.

Email file attachment settings - Excel file

Basic settings

Advanced Excel document data constraints

Starting row

1

\*

Expected row and column counts

Number of data columns

Minimum row count

Maximum row count

Expected data column types

This functionality is not supported for CSV files.

Column name (eg. A, B or AA)

A

Column type

Text

Short description

ADD COLUMN TYPE

Column name (eg. A, B or AA)	Column type	Short description	Actions
No column definitions found			

#### Expected row and column counts

Here you can define the number of columns and the minimum/maximum number of rows for valid Excel imports.

#### Expected data column types

Here you can define all expected columns in detail

### 9.5.3 Email texts

Contain the bodies of different auto response emails.  
Optional. When empty, no email will be sent.

There different kinds of response emails:

- **'Matching email import group found'**: Sent when email received
- **'Confirmation email'**: Sent when confirmation by user is necessary
- **'Security constraints not met'**: Sender does have the required rights for the import
- **'Email processing cancelled because of error'**: Sent in case of data errors and the import is set to 'Cancel the import in case of errors'
- **'Errors occurred, but import was performed'**: Sent when the import is finished with errors
- **'Email successfully imported'**
- **'Limited access prevented email processing'**: The feature 'limited access' is activated and prevents the import
- **'Error list'**: Email with file attachment containing all erroneous rows

#### Email import group settings

Main group settings	Business Cases	Email texts	Security
<b>Matching email import group found</b>			
Email subject	[Data Import] <%ORIG_EMAIL_SUBJECT%> / Ticket <%IMPORT_TICKET_ID%> / Info - email received <span>V</span>		
Email body	<p>This is an automatically generated email by Apparo. Ticket number: &lt;%IMPORT_TICKET_ID%&gt;</p> <p>Data structure of your email attachments is correct. They are going to be imported now.</p> <p>You will receive additional emails informing you about the import progress.</p>		
<b>Email successfully imported</b>			
Email subject	[Data Import] <%ORIG_EMAIL_SUBJECT%> / Ticket <%IMPORT_TICKET_ID%> / Success - all files from your email have been imp <span>V</span>		
Email body	<p>This is an automatically generated email by Apparo. Ticket number: &lt;%IMPORT_TICKET_ID%&gt;</p> <p>Your email attachments were imported completely successfully.</p> <p>The import is finished now.</p>		
<b>Security constraints not met</b>			
Email subject	[Data Import] <%ORIG_EMAIL_SUBJECT%> / Ticket <%IMPORT_TICKET_ID%> / Error - access denied <span>V</span>		
Email body	<p>This is an automatically generated email by Apparo. Ticket number: &lt;%IMPORT_TICKET_ID%&gt;</p> <p>Your email didn't meet security criteria. Maybe it didn't contain a required keyword or it was not sent from an expected email address.</p>		

Auto response email texts

#### 9.5.4 Security

The email import can be secured:

- by limiting the allowed email senders (comma separated list of email addresses)
- by limiting the email senders based on a security group:  
the user account including email address must be stored in an MS Active Directory system
- by using a text keyword that must be delivered in the subject or body of the email
- by enabling a confirmation email (an automated email is returned to the sender, which has to be confirmed within a defined timeframe)
- by a list of trusted email servers (only emails of listed servers are accepted)

All emails can be encrypted using SSL

The general access can be restricted by using the limited access feature in the tab 'Security':

- **No limitations:** Default value, no restrictions
- **Limited for all:** Nobody can use this import group
- **Limited for variable value:** Not useable if a variable return 'true' – e.g. a variable returns true during the time period when the database is performing maintenance tasks

#### Email import group settings

Main group settings	Business Cases	Email texts	Security
Allowed email sender addresses		<input type="text"/>	
Security keywords		<input type="text"/>	
Email confirmation required		<input type="checkbox"/>	
Confirmation reply must come within		15 minutes	
Check if the email address of the sender is defined in the local security system		<input checked="" type="checkbox"/>	
Authorized security groups		<input type="text"/>	
Business Case limited access		<input checked="" type="radio"/> No limitation (default) <input type="radio"/> Limited for all <input type="radio"/> Limited for variable value	

OK CANCEL

## 9.6 eMails

It contains the general error message for the case that no matching import group could be found to perform the import.

This can have different causes:

- Erroneous setup of import groups
- Erroneous attachments (e.g. file does not match the file import template)
- The import group can be temporary disabled by the administrator
- Disabled by a variable (e.g. a time controlled variable to avoid issues during a maintenance period)

The screenshot shows the 'eMails' configuration window in the Apparo software. On the left is a sidebar with navigation links: Introduction, Main Settings, Import Groups, eMails (selected), Logging, and Variables. At the top of the main area are buttons for Close, Save, and Cancel. The main content area is titled 'eMails' and displays a configuration for an email template. The 'Email subject' field contains the text: '[Data Import] <%ORIG\_EMAIL\_SUBJECT%> / Ticket <%IMPORT\_TICKET\_ID%> / Error - no matching profile found'. The 'Email body' field contains a multi-line text template: 'This is an automatically generated email by Apparo. Ticket number: <%IMPORT\_TICKET\_ID%> You have sent an email to the Apparo system. The system analyzed your email attachments but the structure of your Excel sheets or the number of attachments are not assignable to the defined import possibilities. Therefore the system imported nothing and the import stopped. Please inform your Apparo designer who is responsible for the email Excel import functionality.' Below this, there is a section for 'Internal error occurred and processing stopped' with an 'Email subject' field containing: '[Data Import] <%ORIG\_EMAIL\_SUBJECT%> / Ticket <%IMPORT\_TICKET\_ID%> / Error - Internal error occurred'.

General error message

## 9.7 Logging

All events can be logged into an own database table.

In order to log all possible values the table will need the following columns:

- **Column for client name:** What client was used for the import
- **Column for sender address:** What sender address tried to import
- **Column for event timestamp:** Timestamp
- **Column for ticket ID:** Ticket ID, unique ID for the import event
- **Column for storage path:** Where is the email and attachment stored
- **Column for Business Case ID:** What Business Case performed the import
- **Column for importing group name:** What import group performed the import
- **Column for the import message:** Plain text with error message
- **Column for the log severity:** Can be warning, error, info or debug
- **Column for the message code:** A number representing the message

You can log all email import related events into a custom database table.  
The following definition can be used to create such a table:

### IBM DB2:

```
CREATE TABLE EMAIL_IMPORT_LOG (
    LOG_SEQUENCE_NUMBER
INTEGER,
    CLIENT_NAME VARCHAR(255),
    SENDER_ADDRESS
VARCHAR(255),
    EVENT_TIMESTAMP
TIMESTAMP,
    TICKET_ID INTEGER,
    STORAGE_PATH
VARCHAR(2000),
    BUSINESS_CASE_ID
VARCHAR(500),
    IMPORT_GROUP_NAME
VARCHAR(500),
    IMPORT_MESSAGE
VARCHAR(4000),
    LOG_SEVERITY VARCHAR(255),
    MESSAGE_CODE INTEGER);
```

### Oracle:

```
CREATE TABLE EMAIL_IMPORT_LOG (
    LOG_SEQUENCE_NUMBER
INTEGER,
    CLIENT_NAME VARCHAR2(255),
    SENDER_ADDRESS
VARCHAR2(255),
    EVENT_TIMESTAMP
TIMESTAMP,
    TICKET_ID INTEGER,
    STORAGE_PATH
VARCHAR2(2000),
    BUSINESS_CASE_ID
VARCHAR2(500),
    IMPORT_GROUP_NAME
VARCHAR2(500),
    IMPORT_MESSAGE
VARCHAR2(4000),
    LOG_SEVERITY VARCHAR2(255),
    MESSAGE_CODE INTEGER);
```

### MS SQL Server:

```
CREATE TABLE EMAIL_IMPORT_LOG (
    LOG_SEQUENCE_NUMBER
INTEGER,
    CLIENT_NAME VARCHAR(255),
    SENDER_ADDRESS
VARCHAR(255),
    EVENT_TIMESTAMP DATETIME,
    TICKET_ID INTEGER,
    STORAGE_PATH
VARCHAR(2000),
    BUSINESS_CASE_ID
VARCHAR(500),
    IMPORT_GROUP_NAME
VARCHAR(500),
    IMPORT_MESSAGE
VARCHAR(4000),
    LOG_SEVERITY VARCHAR(255),
    MESSAGE_CODE INTEGER);
```

## 9.8 Variables

You can use JavaScript to compute advanced calculations, and the result can be used in importing Business Cases as any other variable.

There is a list with pre-defined variables, ready to use.

Close Save Cancel

Introduction
Main Settings
Import Groups
eMails
Logging
Variables

Variables

+ Add
X Delete

User defined variables

Variable name	Variable type
<%VAR%>	Script variable

Internal variables

Internal variables ready for use

Variable name	Variable description
<%SAFE_HOME_DIR%>	Folder on the server which contains Apparo settings
<%SAFE_BG_NAME%>	Name of currently opened Business Case
<%SERVER_NAME%>	Name of application server where Apparo is running
<%NEW_UNIQUE_VALUE%>	Unique value (everytime variable is resolved, its value will be unique)
<%CURRENT_DATE%>	Current date and time
<%DATE%>	Current date
<%TIMESTAMP%>	Current date and time

[151]

## 10 E-mail Business Cases (EBC)

An e-mail Business Case is used to send e-mails.  
It contains the definitions, such as subject and body.  
Content, recipient, etc. can be made dynamic with variables.

An e-mail Business Case is called usually by button froms Single or Table Business Cases.

**An e-mail Business Case can access all the widget reference variables of the current line.**  
All other variables can also be used.

### Comment via eMail and button-click

Sender	demo@apparo.solutions
Recipient(s)	@apparo.solutions
Subject	Apparo Fast Edit Demonstration
	<div>ADD ATTACHMENT</div>
Text	<div> <div> <i>¶</i> <b>A</b> <b>B</b> <i>/</i> <u>U</u> <math>\frac{1}{3}</math> <math>\frac{1}{4}</math> <math>\frac{1}{5}</math> <math>x_2</math> <math>x^2</math> <math>\frac{1}{2}</math> <math>\frac{1}{3}</math> <math>\frac{1}{4}</math> <math>\frac{1}{5}</math> <math>\frac{1}{6}</math> <math>\frac{1}{7}</math> <math>\frac{1}{8}</math> <math>\frac{1}{9}</math> <math>\frac{1}{10}</math> <math>\frac{1}{11}</math> <math>\frac{1}{12}</math> <math>\frac{1}{13}</math> <math>\frac{1}{14}</math> <math>\frac{1}{15}</math> <math>\frac{1}{16}</math> <math>\frac{1}{17}</math> <math>\frac{1}{18}</math> <math>\frac{1}{19}</math> <math>\frac{1}{20}</math> <math>\frac{1}{21}</math> <math>\frac{1}{22}</math> <math>\frac{1}{23}</math> <math>\frac{1}{24}</math> <math>\frac{1}{25}</math> <math>\frac{1}{26}</math> <math>\frac{1}{27}</math> <math>\frac{1}{28}</math> <math>\frac{1}{29}</math> <math>\frac{1}{30}</math> <math>\frac{1}{31}</math> <math>\frac{1}{32}</math> <math>\frac{1}{33}</math> <math>\frac{1}{34}</math> <math>\frac{1}{35}</math> <math>\frac{1}{36}</math> <math>\frac{1}{37}</math> <math>\frac{1}{38}</math> <math>\frac{1}{39}</math> <math>\frac{1}{40}</math> <math>\frac{1}{41}</math> <math>\frac{1}{42}</math> <math>\frac{1}{43}</math> <math>\frac{1}{44}</math> <math>\frac{1}{45}</math> <math>\frac{1}{46}</math> <math>\frac{1}{47}</math> <math>\frac{1}{48}</math> <math>\frac{1}{49}</math> <math>\frac{1}{50}</math> <math>\frac{1}{51}</math> <math>\frac{1}{52}</math> <math>\frac{1}{53}</math> <math>\frac{1}{54}</math> <math>\frac{1}{55}</math> <math>\frac{1}{56}</math> <math>\frac{1}{57}</math> <math>\frac{1}{58}</math> <math>\frac{1}{59}</math> <math>\frac{1}{60}</math> <math>\frac{1}{61}</math> <math>\frac{1}{62}</math> <math>\frac{1}{63}</math> <math>\frac{1}{64}</math> <math>\frac{1}{65}</math> <math>\frac{1}{66}</math> <math>\frac{1}{67}</math> <math>\frac{1}{68}</math> <math>\frac{1}{69}</math> <math>\frac{1}{70}</math> <math>\frac{1}{71}</math> <math>\frac{1}{72}</math> <math>\frac{1}{73}</math> <math>\frac{1}{74}</math> <math>\frac{1}{75}</math> <math>\frac{1}{76}</math> <math>\frac{1}{77}</math> <math>\frac{1}{78}</math> <math>\frac{1}{79}</math> <math>\frac{1}{80}</math> <math>\frac{1}{81}</math> <math>\frac{1}{82}</math> <math>\frac{1}{83}</math> <math>\frac{1}{84}</math> <math>\frac{1}{85}</math> <math>\frac{1}{86}</math> <math>\frac{1}{87}</math> <math>\frac{1}{88}</math> <math>\frac{1}{89}</math> <math>\frac{1}{90}</math> <math>\frac{1}{91}</math> <math>\frac{1}{92}</math> <math>\frac{1}{93}</math> <math>\frac{1}{94}</math> <math>\frac{1}{95}</math> <math>\frac{1}{96}</math> <math>\frac{1}{97}</math> <math>\frac{1}{98}</math> <math>\frac{1}{99}</math> <math>\frac{1}{100}</math> </div> <div> <p>Dear ,</p> <p>The order needs your attention.</p> <p><a href="#">Click here to call up this case in a browser window.</a></p> <p>Reason:</p> <p>Regards</p> <p>administrator</p> </div> </div> <div> <div>SEND EMAIL</div> <div>CANCEL</div> </div>



## 10.1 Creating an EBC

When you create an e-mail Business Cases you have to fill first, like all other types of Business Case , the general settings.

The email connection is used only to send and can also be used in other e-mail Business Cases.

The optional security group ensures that only authorized users can send e-mails.

A number of security groups are to be entered separated by a comma.

### Main email Business Case settings

Identifier / Short name	<input type="text" value="Email_BC"/>	*
Business Case name	<input type="text" value="Email_BC"/>	*
Email connection	<input type="text" value="Email2"/>	▼ *
Business Case security group	<input type="text" value="Administrators, GroupOne"/>	
Notes	<div></div>	

## 10.2 Header and Footer

In the header or footer, you can define captions and descriptions, specify fonts and styles and insert logos. In the title, the description and the logo URL variables can be used.

Run

Close

Save

Cancel

Main settings

Header

Footer

Email properties

Email body

Email attachments

Button titles

Header

Title & Description

Language	Title	Description
German	Kommentar via eMail und Button-Klick	
English	Comment via eMail and button-click	

Title style

Font face	Size	Style	Align	Colour
Arial	14	Bold	Left	#000000

Description style

Font face	Size	Style	Align	Colour
Arial	12	Normal	Left	#000000

Background colour

#FFFFFF

Left logo URL

Right logo URL

### 10.3 E-mail properties

Here you can define the sender e-mail, the recipient list and their settings.

**Email properties**

**Sender & recipients**

Sender email address: demo@apparo.solutions

☐ Try to use automatically the email address of the user if the email is stored in the security system

☐ User can change the sender email address

Recipient(s): Please input into this field all email recipients separated by comma. Optional the user can edit this list too.

<%mail%>@apparo.solutions

☐ User can change the recipient list for this email

**Subject**

Subject: Apparo Fast Edit Demonstration

☐ User can change the email subject

**Settings**

Definition of size of the textarea for the email body text. This textarea is visible if the user can edit the email body text.

Widget width(px): 600

Label width(px): 80

☐ User can decide if for the following email(s) settings can be changed again or not

#### Sender & recipients

##### Sender email address

Does the indicated in the e-mail sender address this need not match the e-mail sender from the e-mail link. Variables can be used.

Options:

- Try to use automatically the email address of the user if the email is stored in the security system
- Users can change the sender address

##### Recipient(s)

Contains all recipients, separated by commas. Variables can be used.

Optionally, the user can modify the list.

## Subject

Contains the subject of the e-mail, variables can be used.  
Optional user may change the subject.

## Settings

Defines the size of the text area for the e-mail text (visible if the user is allowed to change the e-mail text).

- Widget Width (px)
- Label Width (px)

## 10.4 E-Mail body

Contains the 'E-mail Body', also known as e-mail text.  
If you use formatted text, the email HTML format is used.  
You can use all the variables of the calling Business Cases.

## 10.5 Email attachments

Send e-mail attachments with this e-mail, either user-selected or specified files from the server or from a selected file widget.

## 10.6 Button titles

Contains the label of the buttons in all installed languages

Button titles			
Language	Add email attachment button	Send email button	Don't send email button
German	Anhänge dazufügen	E-mail senden <input type="text" value="V"/>	Abbrechen <input type="text" value="V"/>
English	Add attachment	Send eMail <input type="text" value="V"/>	Cancel <input type="text" value="V"/>

## 11 Action Business Cases (ABC)

Action BC is helpful for adding database actions like changing data, calling scripts server side without user interactivity directly into a Cognos report using HTML.

An Action BC can contain own web output too, even buttons like yes/no are possible.  
The window of an Action BC can close automatically.

Additional it is possible to run an Action BC **without user interface**.  
In this case the communication to the Apparo server will be done using AJAX.

### 11.1 Possible Actions

An overview of the 4 options and their meaning, use case are following in Use Cases

- ☐ Enable Pre Business Case action execution
- ☐ Enable Post Business Case action execution in success case
- ☐ Enable Post Business Case action execution in failure case
- ☐ Enable Exit Business Case action execution

#### Enable Pre Business Case action execution

The default case, an action is started when the Action BC starts

#### Enable Post Business Case action execution in success case

Another action can be started when the user clicks on the green OK button in the closing dialog

#### Enable Post Business Case action execution in failure case

Another action can be started when the user clicks on the red Cancel button in the closing dialog

#### Enable Exit Business Case action execution

Another action can be started when the user clicks on the green OK or red Cancel button in the closing dialog

*The first action usually starts a script or calculation or writes something to the database, while the second action usually triggers e.g. an email notification.*

## 11.2 Use Cases

Action BCs can be run without user interaction, started when a JavaScript event triggers by calling the Action BC URL

Or


Action BCs can be run with user interaction, started by a JavaScript event when the user e.g. clicks a checkbox, or simply when clicking an URL button or a hyperlink.

*Basically, there are two operation modes:*







- Without output window, the silent mode, the user does not know that an Action BC has been started.
- With output window, issues a notification showing the process output, e.g. a calculation

### 11.3 Creating a new Action Business Case

Select Action to start a new Action BC

 Apparo

Please select type of Business Case you want to create now

	<b>Table</b>	A table Business Case is showing many data rows on the same page. The user can filter the data, edit, import from Excel, export to Excel and so on.
	<b>Single</b>	A single Business Case is showing just one data row only.
	<b>Set</b>	A grouping of multiple Business Cases (table/single) for more comfortable usage. You can define global filters that are filtering all Business Cases automatically too.
	<b>Email import</b>	Importing Excel data directly by email - send Excel sheets using email attachments and Apparo will import the Excel data directly into the database including file uploads. No web browser is necessary, just an email.
	<b>Email</b>	An eMail Business Case is a definition of an email text including usage behavior and can be used in another Business Cases of type 'table' or 'single' only. In these Business Cases it is possible to define buttons that can use this eMail Business Case.
	<b>Action</b>	Purpose of Action Business Case is to execute scripts or database procedures that can be called from a report/HTML page. Usage of AJAX and Javascript for automatically executing in the background is possible too.

CANCEL

### 11.4 Defining the main settings

If the used action contains SQL commands, you need to select a database connection.

Run Close Save Cancel

Main settings  
 Header  
 Footer  
 Visual settings  
 Actions  
 Buttons  
 Variables  
 Link into Portal

Main settings

Identifier / Short name  
 Business Case name  
 Database connection  
 Show output  
 Close automatically  
 Business Case security group  
 Notes

Action BC with output  
 Action BC with output  
 SAMPLES  
 This Action Business Case will display output  
☐  
  
 calling an Action BC that is calling a script and showing Output.



## 11.5 Header & Footer

## 11.6 Visual settings

Contains the texts and visual settings of the message windows.

The description message is displayed when the BC action is started.

The "Please wait" message is displayed while the action is being executed.

Action output text contains the text of the Action has finished message and may contain return values.

*Other settings:*

### Hide application header

Hides the blue application header

### Window background

Defines the color of the window background

### Window background image URL

Shows a background image instead of the color.

## 11.7 Actions

Contains the actions and gives the selection **Automatic execution of**

### 11.7.1 Javascript commands

Allows server sided JavaScript and the use of variables and all custom Javascript methods - [see here](#)

☒ Enable Pre Business Case action execution

**Pre Business Case execution (allows to run automatically a script or database function/procedure if the user is starting the Business Case)**

Automatic execution of: Javascript commands for all users

JavaScript definition

Return value variable: <%RETURN\_VALUE%>

### 11.7.2 SQL commands

Allows the direct use of all SQL commands

☒ Enable Pre Business Case action execution

**Pre Business Case execution (allows to run automatically a script or database function/procedure if the user is starting the Business Case)**

Automatic execution of: SQL commands for all users

SQL definition

Return value variable: <%RETURN\_VALUE%>

### 11.7.3 Database stored procedures and functions

☒ Enable Pre Business Case action execution

**Pre Business Case execution (allows to run automatically a script or database function/procedure if the user is starting the Business Case)**

Automatic execution of: Database procedure for all users

Name: return COMMENT\_FUNC('Action BC calling', <%REPORT\_VAR1%>)

Return value variable: <%RETURN\_VALUE%>

To call a database function or procedure:

[calling convention] procedure/function\_name ( argument1, argument2, ..., argumentN ).

ATTENTION: Please use the same upper and lower case for schema and procedure/function as defined in your database.

If the database connection of this procedure/function is the same as the one for the business case, then the procedure/function will be executed within the same database transaction.

The procedure may not commit or rollback the existing transaction, but it may start its own inner (named) transaction (if supported by the database) or use savepoints.

For character or string arguments, use the ' character to enclose the argument. Use at least one space between [calling convention] and procedure name.

Your parameters can include Apparo variables, for example: <%USERNAME%>, <%CURRENT\_DATE%>, <%BC\_NAME%>, <%PRIMARY\_KEY%>.

Do not enclose Apparo variables with apostrophes or quotes.

For the complete list of variables, see the Variables chapter.

[Calling convention] is one of these:

**If you are using an Oracle or IBM DB2 database:**

- return - For calling a stored function that returns a value.

**If you are using an MS SQL Server database:**

- Calling functions on SQL Server is not supported. It is possible to have a return value from a procedure, but [Calling convention] must be empty in this case.
- Please use "SET NOCOUNT ON;" in the beginning of your SQL Server procedure. Then it is possible to use SQL commands in your procedure without affecting the return value.

**If you use a Sybase database:**

- select - For calling a stored function that returns a value.

**If you are using a Teradata database:**

- return macro - For calling a Teradata macro that returns a value.
- macro - for calling a Teradata macro that does not return a value
- return - For calling a stored function that returns a value.

**If you are using an SAP HANA database:**

- select - For calling a stored function that returns a value.

**Hint:** [Calling convention] must be empty when you call a stored procedure.

#### 11.7.4 Executing anonymous block

The anonymous block serves as database procedure, which can be executed without having direct database access.

☒ Enable Pre Business Case action execution

**Pre Business Case execution (allows to run automatically a script or database function/procedure if the user is starting the Business Case)**

Automatic execution of: Anonymous database block for all users

Anonymous database block definition

Return value variable: <%RETURN\_VALUE%>

#### 11.7.5 Calling a script or batch file on the server

You can call all script files located in the scripts folder as set in the Apparo Configuration Manager.

**Supported are:**

- Batch and executable files (.bat, .sh, .exe)
- SQL files (.sql)
- JavaScript files (.js)

☒ Enable Pre Business Case action execution

**Pre Business Case execution (allows to run automatically a script or database function/procedure if the user is starting the Business Case)**

Automatic execution of: Script on server (batch file, SQL file or Javascript file) for all users

Name: sendOrderToSupplier.bat

Parameters for batch file only (separated with space): <%VARIABLE1%> <%VARIABLE2%>

Return value variable: <%RETURN\_VALUE%>

## 11.8 Buttons

Here you can enable/disable buttons.

Button type	Button label	Enabled	Order
> OK	OK	<input checked="" type="checkbox"/>	↓ ↑
> Cancel	Abbrechen	<input type="checkbox"/>	↓ ↑

By clicking the linked button type, you can change the button title for each installed language:

Language	Label
German	OK
English	OK

Redirect URL:

Gap (right):

OK CANCEL

Also possible: Defining a redirect URL, which redirects the user after clicking

## 11.9 Variables

When calling the Action BC via URL, all parameters passed to the Action BC require their own report variable

### Example:

We want to pass one parameter and created one report variable:

### Syntax:

*&Report\_Var\_1=VALUE*

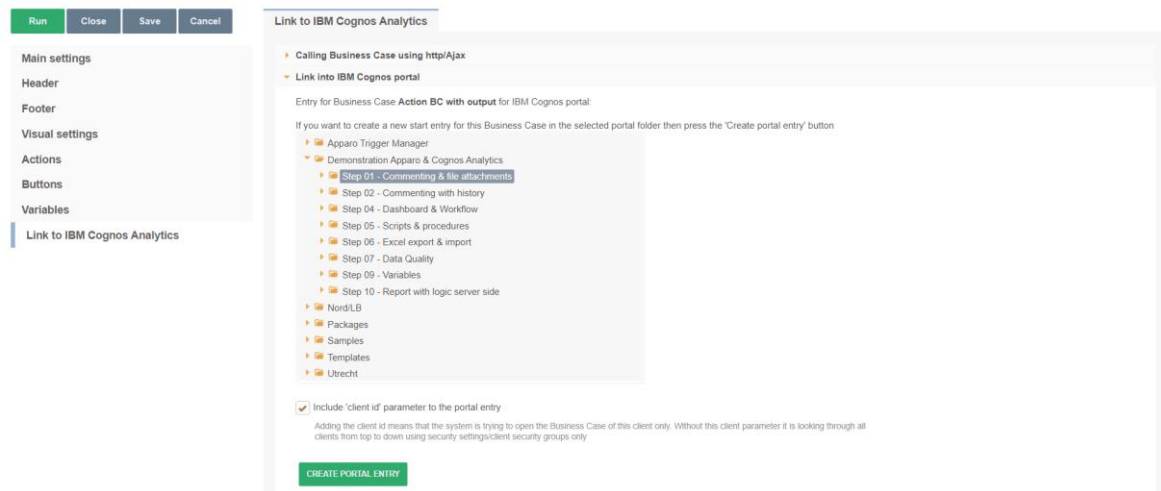
**The URL would look like this:**

*/ibmcognos/bi/v1/apparoBusinessCase.html?bc=Action+BC+with+output&clientid=Demo+g2&Report\_Var\_1=VALUE*

The variable contains the value VALUE and can be used as parameter when calling the action.

## 11.10 Link into IBM Cognos Analytics

Gives 2 options, the 1<sup>st</sup> option will be explained in detail later in this chapter.  
The 2<sup>nd</sup> option let you directly link the action BC to the Cognos portal.  
Select the folder and press the button Create portal entry.

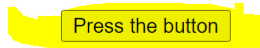


## 11.11 Action Business Case with user interface output

If screen output has been activated (see above), then buttons and texts can be defined.  
The output is in a window.  
Optionally, the window can also be closed automatically when the script / procedure has finished.

### 11.12 Action Business Case without user interface output

It is possible to place buttons and checkboxes directly in a Cognos report.  
These widgets can use report calculations and are calling an **Action Business Case** without additional output.



A Business Case is called automatically if the user is pressing a button or setting a checkbox in a list.

ID	custom ID	HTML item	yes/no	Year	Jan	Feb	March	April
10	40	<input type="checkbox"/>	0	2019	0	0	0	0
10	40	<input type="checkbox"/>	0	2020	2	0	0	0
10	40	<input type="checkbox"/>	0	2021	0	0	5551	0
15	20	<input type="checkbox"/>	0	2019	15000	12000	580	79600
15	40	<input type="checkbox"/>	0	2020	0	0	0	0
15	40	<input type="checkbox"/>	0	2020	0	0	0	0
20	30	<input type="checkbox"/>	0	2020	8000	9000	10000	18000
Overall - Summary		Overall - Summary	0	14139	23002	21000	16131	97600

#### 11.12.1 Placing a button into a Cognos report

In IBM CA, Javascript programs must be stored in separate files on the server side.  
These files are then linked to a report **using custom control widgets**.

1. Create a new blank report or open an existing report
2. Create a HTML item with the following content:

```
<button type="button" name="apparoAjaxButton"
value="FE_VAR1=123&FE_VAR2=456">Press the button</button>
```

You can use instead of a text content a report calculation too. This is helpful if you want to calculate the delivered parameter values.

3. Add a new **custom control widget** with name **apparo ajax custom control** into the report.  
**Hint:** It must be placed "above" the HTML item widget.





Settings for „**apparo ajax custom control**“:

☐ Custom control

GENERAL	
Description	apparo ajax custom control
UI type	UI with event propagation
Module path	/ibmcognos/bi/v1/apparoTableModule.html
Configuration	{ "bcid": "Action+BC+no+output", "clientid": "Demo+g2", "elementName": "apparoAjaxButton" }
Height	50px
Width	
MISCELLANEOUS	
Name	apparo ajax custom control

Please use this configuration, just copy and paste it:

```
{
  "bcid": "Action1",
  "clientid": "Demo",
  "elementName": "apparoAjaxButton"
}
```

Hint: Parameter values must be in **uuencoded** format.

Use this URL for calling the page

**/ibmcognos/bi/v1/apparoTableModule.html**

#### Meaning:

bcid	Business Case ID
clientid	Client ID
elementName	Identifier, this will be used in the HTML controls too

The File apparoTableModule.html is stored in the [IBM\_GATEWAY]\webcontent file directory. You can use another file (just copy this file) but then you must add this new file name in the web server configuration too. See directory “bi”, ARR-settings.

If the user is clicking the button then the Action BC of custom control will be started because attribute name...

```
<button type="button" name="apparoAjaxButton"
value="FE_VAR1=123&FE_VAR2=456">Press the button</button>
```

...and attribut **elementName** in custom control action1...

```
{
  "bcid": "Action1",
  "clientid": "Demo",
  "elementName": "apparoAjaxButton"
}
```

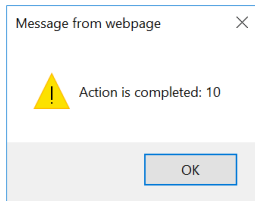
...having the same value.

Important: The report must run with full interactivity:

Report	⌵
RUNNING & VALIDATING	
Burst options	
Run with full interactivity	Yes
STYLE	
Report styles reference	

**Result:**

If the user is pressing button1 then the user should see a window like here with the return value of the Business Case:



That means, when the user presses the button, the Action Business Case "Action1" of the client "Demo" is called with the parameters VAR1 and VAR2.

This also means, this Action Business Case needs the report variables VAR1 and VAR2.

### 11.12.2 List with checkbox column

1. Create a new empty report or open existing report, define a list widget with data.
2. Add a HTML item to the list as a new column, change it to "**Expression**" instead of "**Text**" with the following example content (parameters must be adapted):

```
'<input name="apparoAjaxCheckbox" type="checkbox"
value="par1=1&FE_customer=' +number2string( [Query1].[CUSTOMER_ID1])
+'&bc=Action+BC+for+checkbox+in+a+list&clientid=Demo+g2"/>'
```

#### Description:

clientid client ID

bc Business Case ID

name Connection name that is used in the Javascript file

Hint: Parameter values must be in **uuencoded** format.

In the example the parameters **FE\_par1** and **FE\_customer** are passed. Therefore in the Action BC a report variable "**par1**" (without FE\_) and "**customer**" must be defined.

Additionally the parameter **FE\_checkbox\_state** that contains the current status of the checkbox will be passed **automatically**.

In the next step, this report variables **<%par1%>**, **<%customer%>** and **<%checkbox\_state%>** can be used as a call parameter for a database procedure or for a script.

Important: The report must run with full interactivity:

Report	⌵
<b>RUNNING &amp; VALIDATING</b>	
Burst options	
Run with full interactivity	Yes
<b>STYLE</b>	
Report styles reference	

Example:

ID	custom ID	HTML item	yes/no
10	40	< HTML item >	0
10	40	< HTML item >	0
10	40	< HTML item >	0
15	20	< HTML item >	0
15	40	< HTML item >	0
15	40	< HTML item >	0
20	30	< HTML item >	0
Overall - Summary		Overall - Summary	0

Output:

ID	custom ID	HTML item	yes/no
10	40	<input type="checkbox"/>	0
10	40	<input type="checkbox"/>	0
10	40	<input type="checkbox"/>	0
15	20	<input type="checkbox"/>	0
15	40	<input type="checkbox"/>	0
15	40	<input type="checkbox"/>	0
20	30	<input type="checkbox"/>	0
Overall - Summary		Overall - Summary	0

Now you see the new checkbox column.

1. Add a new **custom control** outside of the List widget:

Settings:

☐ Custom control

GENERAL	
Description	Control for checkboxes
UI type	UI with event propagation
Module path	/ibmcognos/bi/v1/apparoTableModule.html
Configuration	{ "bcid": "Action+BC+for+checkbox+in+a+list", "clientid": "Demo+g2", "elementName": "apparoAjaxCheckbox" }
Height	
Width	
MISCELLANEOUS	
Name	

Configuration

×

bcid	Action+BC+for+checkbox+in+a+list
clientid	Demo+g2
elementName	apparoAjaxCheckbox



OK

Apply

Cancel

Hint: Use + instead of blank, use uuencoded characters only.

You can take this configuration using copy & paste:

```
{
  "bcid": "Action+BC+for+checkbox+in+a+list",
  "clientid": "Demo+g2",
  "elementName": "apparoAjaxCheckbox"
}
```

The URL for the module:

**/ibmcognos/bi/v1/apparoTableModule.html**

If the user now selects / deselects a checkbox within the list, the Action Business Case defined in the URL is called.

### 11.12.3 Adapting the message window

The output of the message window is defined in the demonstration file apparoTableModule.html and can easily be adapted. The file is stored in the Cognos Gateway installation, file directory “webcontent”. You will find the settings here:

```

apparoTableModule - Notepad
File Edit Format View Help
define( ['jquery'], function($) {

    //to disable showing prompt message after the AJAX execution set variable showPromptAfterAjax to false
    var showPromptAfterAjax = true;

    //to enable notification that is automatically closed after the AJAX execution set variable showNotificationAfterAjax to true;
    var showNotificationAfterAjax = false;

    //Module constructor
    function ApparoPageModule() {
    };

    //Module mandatory function
    ApparoPageModule.prototype.draw = function(oPage) {

        //html element name, BusinessCaseId and ClientId is set in configuration
        var conf = oPage.configuration;
        if (!conf){
            alert('Missing Apparo configuration.');
```

**var showPromptAfterAjax = true;**

Shows a message window with ok button, once the called action is finished

**var showNotificationAfterAjax = true;**

Shows a “Please wait” notification while the action is being executed and then autohiding

If both are true then two message windows will be displayed, we recommend to set only one option to true.

If both are set to false, the action is started but there will be no notification.

The default settings are:

```

var showPromptAfterAjax = true;
var showNotificationAfterAjax = false;
```

**Hint:** For testing it can be helpful to delete the browser cache.

Contact to the Apparo support: [support@apparo.solutions](mailto:support@apparo.solutions)

## 12 Business logic serverside

With Apparo Fast Edit it is possible to define your own business logic or smaller programs:

- Logic within a script variable
- Logic within a Javascript script
- Logic within the Row validator

The programming language used is JavaScript, which is executed on **the server side**.

### 12.1 Example of a script variable

#### Variable for Business Case

Variable name	<%GROUPS%> *
Variable description	

Variable value	Data output format
----------------	--------------------

**Script body**

Script language : javascript

You can see a detailed JavaScript language description including examples by clicking the question mark icon placed next to the editor.

Attention: If you want to use a Apparo variable in Javascript that contains text then you must use it in quotes, for example:

```
string.replace('<%TEXT1%>', '<%TEXT2%>', 'text')
```

If a script variable must return value true or false then it must be a string like 'true'.

```

1 var groups = afe.getGroupsByRegex('.*');
2 var result = 'Security groups of the current user: ';
3 var i;
4 var group;
5 for(i = 0; i < groups.length; i++) {
6   group = groups[i];
7   result = result + group + ', ';
8 }
9 // returning the calculated result from script
10 result;
11
12
13
14
15
16
17
18
19
20
~~

```

SYNTAX CHECK

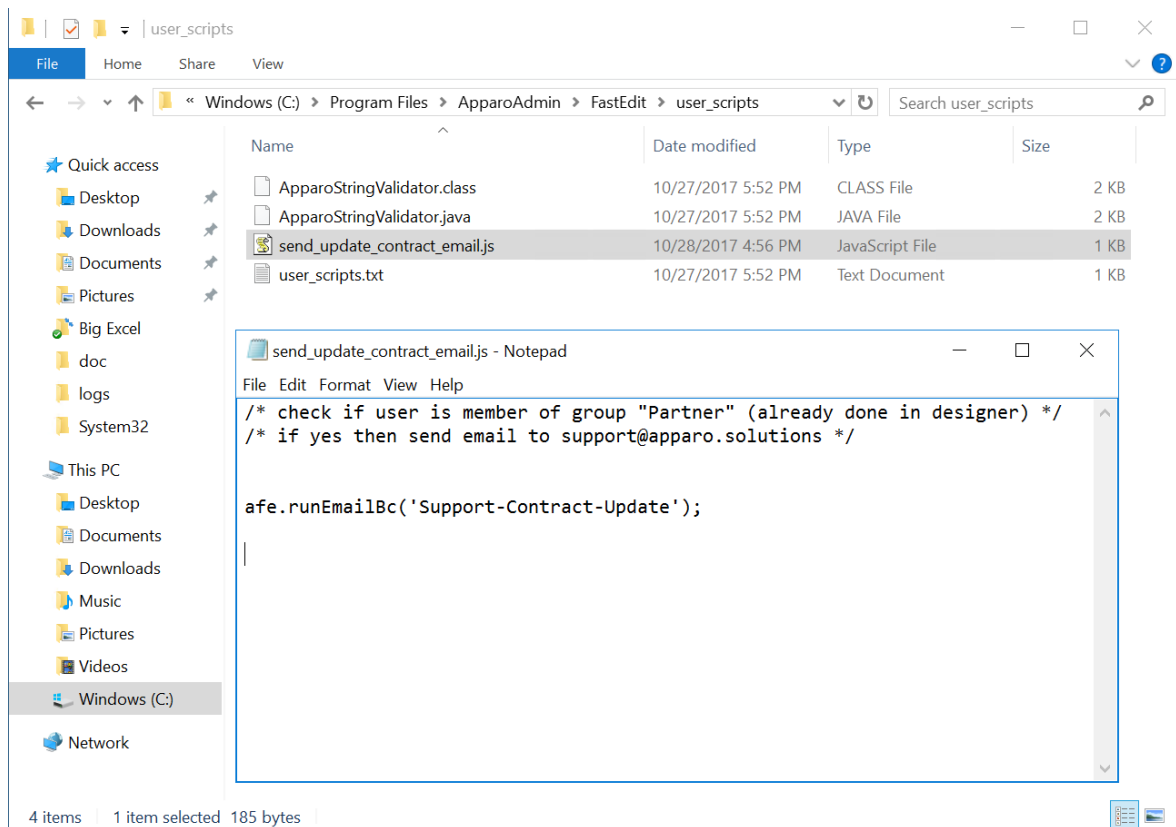
☒ Calculate the variable value before each usage again

OK	CANCEL
----	--------

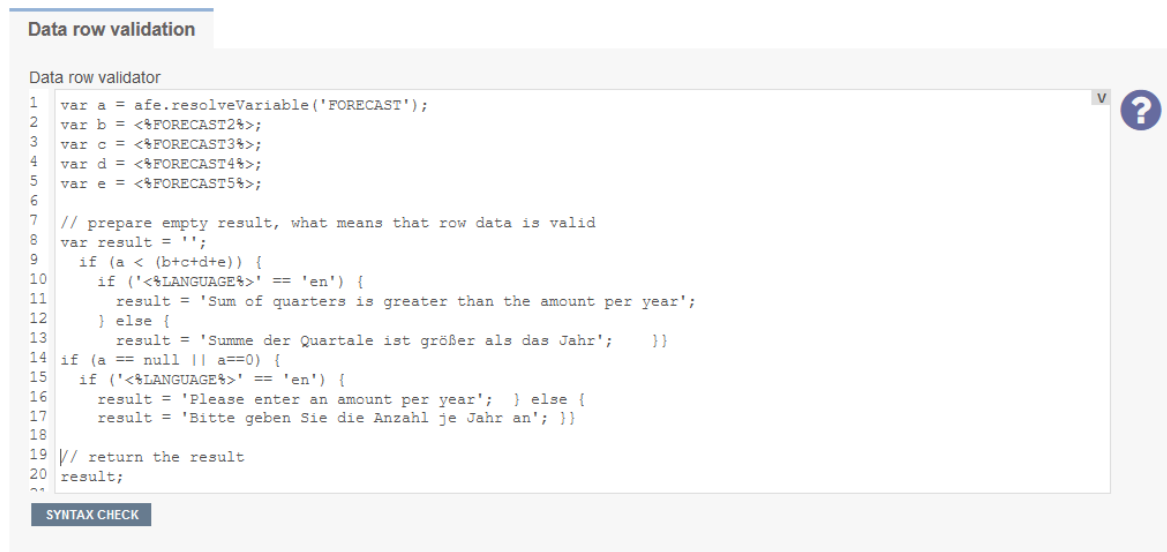


## 12.2 Example of a JavaScript script file

This file can be called up by Apparo Fast Edit and is saved as a file on the server:



## 12.3 Example of the Row-Validator



## 12.4 Additional Apparo Fast Edit Methods

- **afe.callClassMethod**(className, methodName, args) - calls a compiled Java class. This must be stored on the server side.
- **afe.createFile**(pathName, content) - Creates a file with the provided content.
- **afe.executeCommand**(command, homeDir) - Executes the specified cmd command.

Example:

```
afe.executeCommand('C:\\Program Files\\Apparo\\FastEdit\\user_scripts\\batchfile.bat "20";C:\\Program Files\\Apparo\\FastEdit\\user_scripts\\');
```

- **afe.executeSql**(sqlQuery) - Executes an SQL command and returns the first column of the first row.
- **afe.executeSql**(sqlQuery, parametersArray) - Executes an SQL command with a parameter list and returns the first column of the first row.
- **afe.executeSqlInsert**(sqlInsert) – executing a SQL insert and returning true = success, false=failure
- **afe.executeSqlSelect**(sqlSelect) - Executes the specified SQL query and returns the result as a two-dimensional object array of rows and columns.
- **afe.exportAllRows**(filename) - Exports all visible (i. e. all filters are considered) data rows into an Excel file on the server side. Supported output formats: xls, xlsx and csv. The use of variables is possible.  
*Example:* afe.exportAllRows (' <%AFE\_HOME\_HOME\_DIR%>/data-export/<%AFE\_CLIENT\_ID%>/<%AFE\_BC\_ID%>\_<%DATE\_TIMESTAMP\_SHORT%>.xlsx');  
Attention: Please use \\ as path separator, e. g. afe. exportAllRows (' c: \\filename. csv');
- **afe.exportSelectedRows**(filename) - Exports all selected data rows into an Excel file on the server side. Supported output formats: xls, xlsx and csv. The use of variables is possible.  
*Example:* afe.exportSelectedRows (' <%AFE\_HOME\_HOME\_DIR%>/data-export/<%AFE\_CLIENT\_ID%>/<%AFE\_BC\_ID%>\_<%DATE\_TIMESTAMP\_SHORT%>.xlsx');  
Attention: Please use \\ as path separator, e. g. afe. exportAllRows (' c: \\filename. csv');
- **afe.getGroupsByRegex**(regex) - Returns an array of security group names corresponding to the ' regular expressions' argument. Example: afe. getGroupsByRegex (' CLIENT. \*')
- **afe.getSessionParameters**() - Reads the Cognos session parameters and returns them as a 2-dimensional string array
- **afe.resolveVariable**(variableName) - Returns the value of a variable
- **afe.runEmailBc**(emailBusinessCaselId) - Defines the e-mail business case to be executed, i. e. an e-mail is sent..
- **afe.refreshEmbeddedBC**(iframeName, BusinessCaselId, clientId, additionalUrlParameters) – The current Business Case can refresh another Business Case that is embedded in a report too. Parameters are possible

You will find examples of use when you call the ?-icon in the script window.:



## 12.5 Example for the usage of Apparo Fast Edit Methods in Javascript

### 12.5.1 Custom script example returning a string value based on security group

```
var groups = afe.getGroupsByRegex('.*');
var result = 'Security groups of the current user: ';
var i;
var group;
for(i = 0; i < groups.length; i++) {
    group = groups[i];
    result = result + group + ', ';
}
// returning the calculated result from script
result;
```

### 12.5.2 Refresh another embedded Business Case in an IBM Cognos Analytic report

Usage possible in a script file (e.g. refresh.js) or in a script variable:

```
afe.refreshEmbeddedBC('iframe-Name', 'BusinessCase-Id', 'p1=4711&FE_reportvar=test');
```

You can find the iframe-Name in the iframe HTML of the embedded Business Case.

The embedded Business Case will be refreshed as soon as possible.

That means if the first Business Case that contains this code will refresh its output then the 2. Business Case will be refreshed too.

### 12.5.3 Custom function example returning a string value

```
// declaring a function
function myCustomFunction() {
    var result = 'defaultStringValue';
    // complex algorithm to evaluate the result of the method
    return result;
}
// calling the declared function to return a value from script
myCustomFunction();
```

#### 12.5.4 Example using custom functions declared in file

```
// If we have our custom functions declared in the text file we can use it in our script variable.
// In our example we have a file 'C:\scripts\myFunctions.js' with content: 'function myFunctionPlus(a, b)
{return a + b;}'
// We can 'include' this content into the script variable as follow:
<%FILE_CONTENT(C:\scripts\myFunctions.js)%>

// now we can use declared function
var x = myFunctionPlus(2, 1);

// variable 'x' now has value 3
x;
```

#### 12.5.5 Example using Fast Edit variables

```
// working with string variable, and adding a custom postfix
var result = '<%USER_NAME%>' + 'postfix';

// modifying a result of sql variable returning a numeric variable
var result = <%SQL_COUNT_VAR%> / 100;
```

Example using Fast Edit LANGUAGE variable in a column name:

```
// In this example Fast Edit must read the content of the widget reference variable <%PRODUCT_EN%>
or <%PRODUCT_DE%>.
// PRODUCT_EN for a user with english language and
// PRODUCT_DE for a user with german language:

var rc;
rc = '<%PRODUCT_<%LANGUAGE%>>';
```

#### 12.5.6 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;
```

#### 12.5.7 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;
```

#### 12.5.8 Example for dynamic variable resolving

```
// In this example Fast Edit creates an instance of array and set current number of milliseconds (since
1.1.1970) for each element separately.
var args = [];    // create new array
var i;
for(i = 0; i < 10; i++) {
    args[i] = afe.resolveVariable('TIME_MS');
}
```

#### 12.5.9 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);

var text;

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

text;
```

#### 12.5.10 Example for getting the name and content of the current widget

```
// In this example Fast Edit is reading the name and content of the current widget.
// This is helpful for defining the background colour of many similar widgets or defining default
value/constant value without creating many different variables.
var current_widget= '<%CURRENT_WIDGET_NAME%>';
var current_widget_content = afe.resolveVariable(current_widget);
var red_colour_background = 'false';

if (current_widget_content > 100) {
    red_colour_background = 'true';
}
else if (current_widget_content < 50 && current_widget == 'MEASURE1') {
    red_colour_background = 'true';
}

red_colour_background;
```

#### 12.5.11 Example for storing content into a file

```
// In this example Fast Edit store text content into 'myFile.txt' file.
var fileContent = 'This is file content.';
var success = afe.createFile('c:\\files\\myFile.txt', fileContent);
```

#### 12.5.12 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'");
```

#### 12.5.13 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];
}
```

#### 12.5.14 Example for executing a SQL select and storing result into XML file

```
// In this example Fast Edit executes SQL select to retrieve 'id', 'name' and 'price' values of all products in
table 'MyProduct' creates xml String and stores it into XML file.
var productsArray = afe.executeSqlSelect('select id, name, price from MySchema.MyProduct');
var i;
var rowData;
var xmlString = '<?xml version="1.0" encoding="UTF-8"?>\n<products>';
var xmlRow;

for(i=0; i < productsArray.length; i++) {
    rowData = productsArray[i];
    xmlRow = '\n\t<product id="' + rowData[0] + '" name="' + rowData[1] + '" price="' + rowData[2] + '" />';
    xmlString += xmlRow;
}

xmlString += '\n</products>';

var success = afe.createFile('c:\\myXmIs\\products.xml', xmlString);
success;
```

#### 12.5.15 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);
```

#### 12.5.16 Example for executing a command

```
// In this example Fast Edit executes command.
var returnValue = afe.executeCommand("c:\\scripts\\myfile.bat", "c:\\scripts");
```

#### 12.5.17 Example for making a row read-only

```
// In this example we want to make a data row read-only when the PROJECT_COMPLETION_DATE widget
has value of a date in the past (the project is finished).
// First we must create a variable <%ROW_READONLY_VAR%> which will be used here.
// The goal is to return the same date as the widget if it is older then today, otherwise we will return dummy
date.
// Returning date must be represented as string with proper format.

// dummy date in format of MM.dd.yyyy
var result='01.01.1990';

// today's date
var current_date = new Date();

// We need to read string value of PROJECT_COMPLETION_DATE widget.
// If the widget is storing timestamp then it's string value has format 'yyyy-MM-dd HH:mm:ss.S' e.g. '2015-
12-24 18:00:00.0'
// If the widget is storing number then it can be e.g. '42' or '42.1'
var end_date_string = '<%PROJECT_COMPLETION_DATE%>';

// If the PROJECT_COMPLETION_DATE is not specified then we don't want to make row read-only.
// If it has value then we must compare that date with today's date.
if (end_date_string.length > 0) {

// Here we are constructing the Date object from the string in order we can compare two dates.
var end_date = new Date();

// we must set correct Year, Month and Day from the end_date_string
end_date.setFullYear(end_date_string.substring(0,4));
// watch out here: months are calculated from 0 so we must decrease it's number
end_date.setMonth(end_date_string.substring(5,7)-1);
end_date.setDate(end_date_string.substring(8,10));

// now we can compare the dates
if (end_date < current_date) {

// again, we must use correct format: MM.dd.yyyy
var end_date_string_EN_format = "";
end_date_string_EN_format += end_date_string.substring(5,7);
end_date_string_EN_format += '.';
end_date_string_EN_format += end_date_string.substring(8,10);
```

```

    end_date_string_EN_format += '.';
    end_date_string_EN_format += end_date_string.substring(0,4);

    result = end_date_string_EN_format;
}
}

// return the result
result;

```

#### 12.5.18 Example for reading Cognos session parameters

```

//In this example Fast Edit reads Cognos session parameter named 'userClassID'.

//Declaration of the result variable.
var result = "";

//Call Fast Edit function that reads all the Cognos session parameters and return them as 2-dimensional
String array.
var sessionParameters = afe.getSessionParameters();

var i = 0;

//Iterating over all returned parameters
for(i; i < sessionParameters.length; i++) {

    //If parameter's name (second dimension with index 0) is 'userClassID' then we assign parameter's value
    (second dimension with index 1) into 'result' variable.
    if(sessionParameters[i][0] == 'userClassID') {
        result = sessionParameters[i][1];
    }
}

//Return the result
result;

```

#### 12.5.19 Example for exporting all Business Case data to file

```

// In this example Fast Edit exports all Business Case data to file on filesystem and returns whether
operation was successful.
// Note: Backslash symbols must be escaped, i.e. '\\' must be used.

var result = afe.exportAllRows('C:\\Users\\Administrator\\Documents\\allDataExport-
<%DATE_TIMESTAMP_SHORT%>.xlsx');
result;

```

#### 12.5.20 Example for exporting selected Business Case data to file

```

// In this example Fast Edit exports selected Business Case data to file on filesystem and returns whether
operation was successful.
// Note: Backslash symbols must be escaped, i.e. '\\' must be used.

var result = afe.exportSelectedRows('C:\\Users\\Administrator\\Documents\\selectedDataExport-
<%DATE_TIMESTAMP_SHORT%>.xlsx');
result;

```



#### 12.5.21 Example for running Email Business Case

*// In this example Fast Edit sends an e-mail for each modified row of the Table Business Case.*

*// First we must create an Email Business Case (e.g. 'NotificationEmailBc') that will be used for email sending.*

*// We can use variables of the Table Business Case in the Email Business Case definition.*

*// Next we must create a script file (e.g. 'sendingEmailNotification.js') containing single line:  
afe.runEmailBc('NotificationEmailBc');*

*// Then we must enable 'Enable Post row update execution' feature in the Table Business Case, set  
"Automatic execution of" to "Script on server"*

*// and choose the 'sendingEmailNotification.js' in the drop-down list 'Name'.*

*// With such Business Case setup an email will be send every time a row will be updated,*

*// including Excel import (manually or using automatic server-side import or Business Case Email Import)*

## 13 Business logic & widget control in the web browser

Own JavaScript business logic can be executed **automatically** when the user is in the

- **Single Business Case or**
- **Table Business Case**

in insert or editing mode:

- check and uncheck a **checkbox**
- exits an **input field** (or pressing the Enter key)
- selects a value in a **lookup widget (for all tables)**

After that, a JavaScript routine can be automatically executed in the browser to change other widget values:

- Widget Label
- Widget Label with variables
- Input field widget
- Checkbox

Attention: Only the widgets of the **current** data row can be changed, as well as all calculation widgets.

**Example:**

1. The user sets a checkbox or changes a number in an input field and exits this input field.
2. The self-defined JavaScript routine is started

The routine can now read values from other widgets and change the widget value of the type label, label with variables or input field without a submit.

In a table business case, the current values of a column can also be summed up and output in a calculation widget, for example.

### Limitations

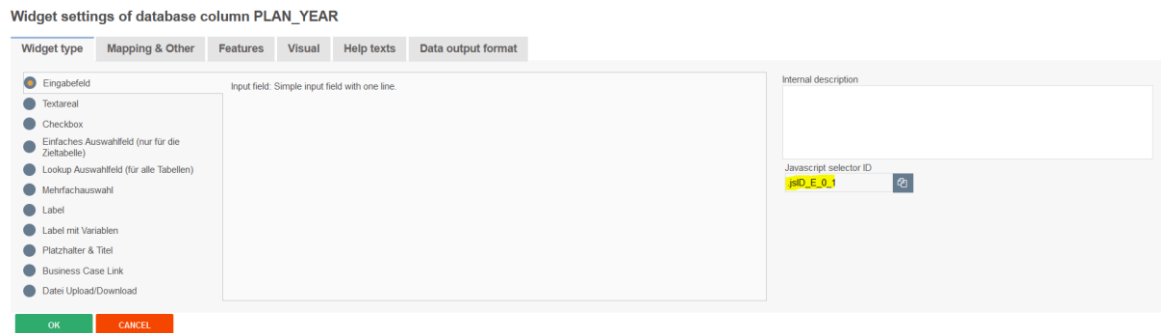
The execution in the web browser naturally results in some restrictions:

- Variables can only be used to a limited extent, they are calculated on the server side.
- You could start an Action Business Case with `window.open`, but the (intermediate) results of the calculations cannot be saved in this way, as access to the widget references of the calculations is missing.

### 13.1 JavaScript Selektor ID

For unique assignment, each widget has a JavaScript selector ID.

The JavaScript selector ID can be found in the widget settings under Widget Type:



To copy the ID, you can use the button to the right of the ID.

#### 13.1.1 Structure of the ID

**The selector ID looks like this:**

.jsID\_E\_0\_0

**The first part of the ID is the abbreviation for JavaScript ID**

.jsID\_E\_0\_0

**The second part of the ID describes the area in which the widget is used**

.jsID\_E\_0\_0

E stands for edit area and C stands for calculation area.

**The two digits stand for the respective column and number in which the widget is used**

.jsID\_E\_0\_0

**The counter** starts at zero and is always incremented by 1.

**The first digit** identifies the column in which the widget is located and is used in Single Business Cases. In

Table Business Cases, the counter is always 0, as no columns are used.

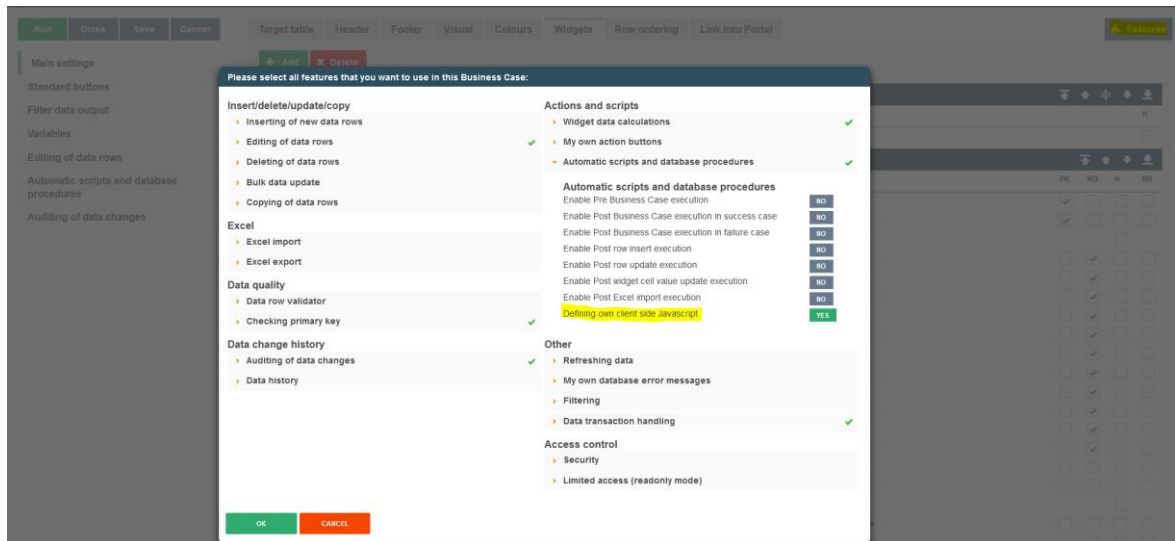
**The second digit** represents the consecutive numbering of the widgets.

If you change the order of the widgets, the ID also changes.

## 13.2 Use in a Table Business Case

### 13.2.1 Activating the feature

Activate the feature "Defining own client-side JavaScript" under Features in the widget settings:



### 13.2.2 Available JavaScript methods

The general format for **get** and **set** methods:

**getAfeWidgetValue(targetElementSelector)**

must be extended in the table business case in the edit area as follows:

getAfeTableWidgetValue(sourceElement, targetElementSelector).

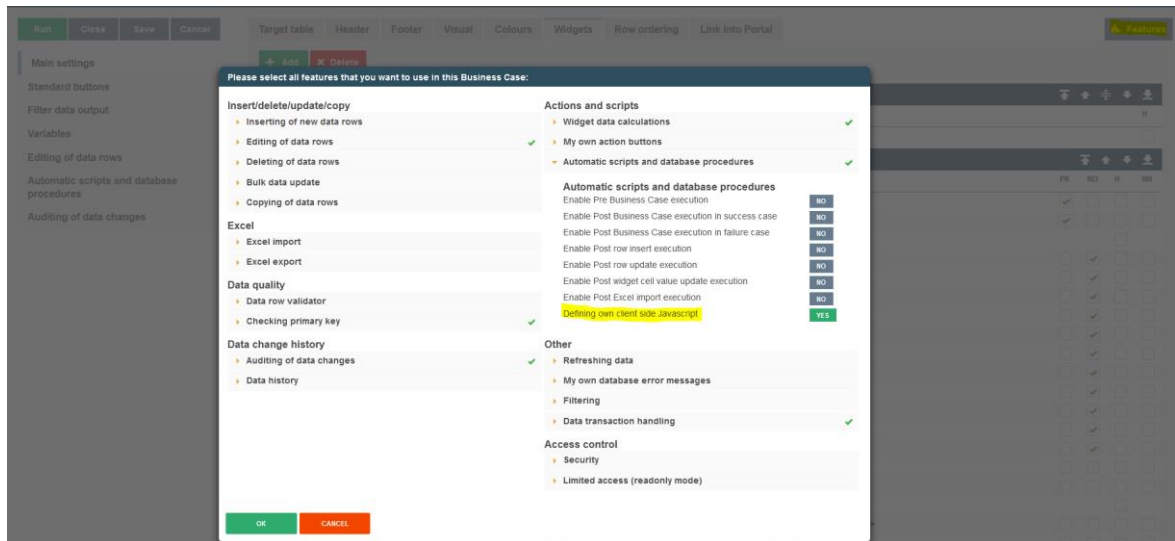
**Table** is a hint for the routine and sourceElement refers to the current row and **is always** 'this'.

Goal	Command
<b>Read</b> the value of a widget of the current row, Value is a <b>number</b>	getAfeTableWidgetNumValue(this, '.jsID_E_0_1');
<b>Read</b> the value of a widget of the current row, Value is a <b>string</b>	getAfeTableWidgetStringValue(this, '.jsID_E_0_1');
<b>Write</b> to a widget of the current row, Value is a <b>number</b>	setAfeTableWidgetNumValue(this, '.jsID_E_0_2', calcValueNum);
<b>Write</b> to a widget of the current row, Value is a <b>string</b>	setAfeTableWidgetStringValue(this, '.jsID_E_0_2', calcValueNum);
<b>Read</b> the value of a <b>calculation widget</b> , Value is a <b>number</b>	getAfeWidgetNumValue('.jsID_E_0_1');
<b>Write</b> a value to a calculation widget, Value is a <b>string</b>	setAfeWidgetStringValue('.jsID_C_0_1', 'CHANGED');
<b>Reading a Lookup Widget Label</b>	var myLabelValue = getAfeTableWidgetLookupLabel(this, '.jsID_E_0_3');
<b>Read/aggregate all numeric values of a column of the current page</b>	getAfeTableColumnFunction('.jsID_E_0_9', 'sum') ;  getAfeTableColumnFunction('.jsID_E_0_9', 'min') ;  getAfeTableColumnFunction('.jsID_E_0_9', 'max') ;  getAfeTableColumnFunction('.jsID_E_0_9', 'avg') ;  Null values are calculated with 0-values

### 13.3 Use in a Single Business Case

#### 13.3.1 Activating the Feature

Activate the feature "Defining own client side JavaScript" under Features in the widget settings:



#### 13.3.2 Available JavaScript methods

Goal	Command
<b>Read</b> the value of a widget, Value is a <b>number</b>	<code>getAfeWidgetNumValue('.jsID_E_0_1');</code>
<b>Read</b> the value of a widget, Value is a <b>string</b>	<code>getAfeWidgetStringValue('.jsID_E_0_1');</code>
<b>Write</b> to a widget, Value is a <b>number</b>	<code>setAfeWidgetNumValue('.jsID_E_0_2', calcValueNum);</code>
<b>Write</b> to a widget, Value is a <b>string</b>	<code>setAfeWidgetStringValue('.jsID_E_0_2', calcValueNum);</code>
<b>Read</b> the value of a <b>calculation widget</b> , Value is a <b>number</b>	<code>getAfeWidgetNumValue('.jsID_E_0_1');</code>
<b>Write</b> a value to a calculation widget, Value is a <b>string</b>	<code>setAfeWidgetStringValue('.jsID_C_0_1', 'CHANGED');</code>
<b>Reading a Lookup Widget Label</b>	<code>var myLabelValue = getAfeWidgetLookupLabel('.jsID_E_0_3');</code>

## 13.4 Read/Write Widget Values

### 13.4.1 Reading widget values

With the method ***getAfeWidgetValue(JavaScriptSelektorID)*** you can read any widget value.  
The *JavaScriptSelektorID* identifies the widget whose value we want to read.

For numeric values (numbers) use *getAfeWidgetNumValue* and  
For string values use *getAfeWidgetStringValue*.

### 13.4.2 Writing widget values

With the method ***setAfeWidgetValue(JavaScriptSelektorID, value)*** you can write values into widgets.  
The *JavaScriptSelektorID* identifies the widget we want to write to and **value** identifies the value (e.g. a number) to write.

For numeric values (numbers) use *setAfeWidgetNumValue* and  
For string values (strings) use *setAfeWidgetStringValue*.

### 13.4.3 Example function

In this example, the value of the widget with reference ID **.jsID\_E\_0\_0** is read and the value of the widget \* 2 is stored back into the widget **.jsID\_E\_1\_2** when the user exits the widget **.jsID\_E\_0\_0**.

```
$(document).on('change', '.jsID_E_0_0', function(){  
  var myValue = getAfeWidgetNumValue('.jsID_E_0_0');  
  setAfeWidgetNumValue('.jsID_E_1_2', myValue * 2);  
})
```

### 13.4.4 In detail

```
$(document).on('change', '.jsID_E_0_0', function()
```

Starts a JavaScript ***function()*** when a value in the widget ***'jsID\_E\_0\_0'*** in the web browser ***\$(document)*** is changed ***on('change')***.

**Attention:** If the widget is a **lookup (for all tables)**, then the setting " User can enter into widget for selecting a value" must be **switched off**.

The content of the JavaScript function is enclosed in the curly brackets.

```
{ var myValue = getAfeWidgetNumValue('.jsID_E_0_0');
```

Defines **var** and fills the JavaScript variable ***myValue*** with the value of the widget ***'jsID\_E\_0\_0'***.

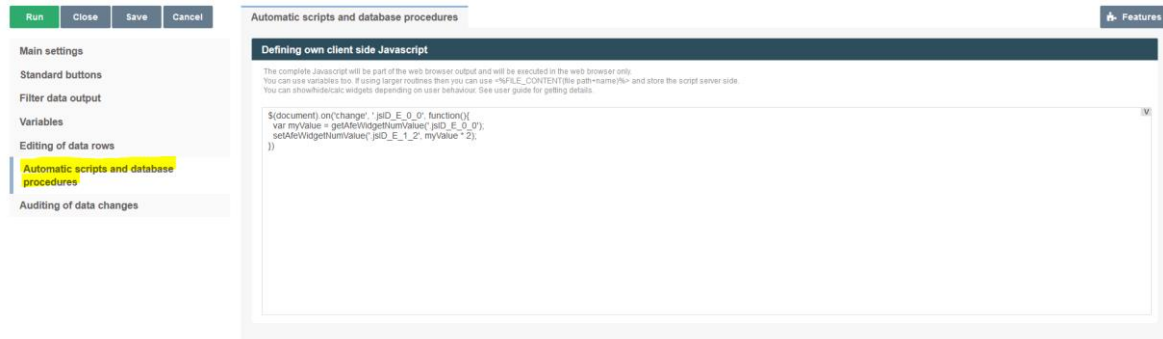
```
setAfeWidgetNumValue('.jsID_E_1_2', myValue * 2); }
```

Writes the content of the JavaScript variable ***myValue*** multiplied by ***2 \* 2*** into the widget ***'jsID\_E\_1\_2'*** as numeric value ***setAfeWidgetNumValue***

## Note

Numeric value is important here, the system can thus automatically use language-specific number formats, otherwise there would be problems e.g. when using different decimal separators (123.45 and 123,45)

### 13.4.5 Use in Apparo





### 13.5 Possibilities of a Checkbox

Checkboxes can be set independently of the used value (usually 0 and 1 or Y and N).  
With true the checkbox is checked, with false the checkbox is unchecked.

```
setAfeWidgetStringValue('jsID_E_0_3', true);
```

This call sets the checkmark (=true) in the checkbox widget with the reference '**jsID\_E\_0\_3**'.  
Since true is a string we use the method **setAfeWidgetStringValue**.

**Checkboxes can also be easily hidden.**

In the following example, the checkbox is hidden depending on its value:

**Example:**

```
$(document).on('change', 'jsID_E_0_3', function(){  
    var myValue = getAfeWidgetNumValue('jsID_E_0_3');  
    if(true == myValue) {  
  
        document.querySelector('jsID_E_0_4').parentElement.parentElement.parentElement.style.display = "none";  
  
    }  
    else {  
  
        document.querySelector('jsID_E_0_4').parentElement.parentElement.parentElement.style.display = "table-row";  
    }  
});
```

If the checked checkbox is set, the checkbox widget **jsID\_E\_0\_4** is **hidden** or otherwise (re)displayed.

### 13.6 Disabling Row Selection Checkbox

The row selection checkboxes that are displayed when, for example, the "Delete rows" feature is active or when using custom buttons that work for all selected rows, can be disabled row by row depending on widget values (or calculations).

These rows can then not be deleted or the action on the button cannot be triggered for this row.

#### Method:

```
hideTableRowSelection(JavascriptSelectorId, 'widget value');
```

```
hideTableRowSelection('.jsID_E_0_5', 'test');
```

In this example, all rows whose value of the widget with ID .jsID\_E\_0\_5 contains the value 'test' become unselectable.

#### 13.6.1 Application example

In a business case, rows should be non-deletable depending on the workflow status "Ready for approval".

<input type="checkbox"/>	Year	Month	Office	Product line	Product	My status	Revision status	Plan data	Plan2	Pla
<input type="checkbox"/>	2017	1	London office	Trousers	Talli	Ready for approval	Open	4007	500	
<input type="checkbox"/>	2017	1	London office	Trousers	Talli	open	Open	5555	0	

Lookup widgets can be used directly, but assuming the selection checkbox is to be hidden based on a calculation, we need to output the result of the calculation in a 'label with variables' type widget beforehand. Therefore, we output the value of the lookup widget here as a substitute in a label with variables, which can optionally also be hidden:

#### Widget settings

Widget type

Mapping & Other

Features

Visual

Help texts

Data output format

Label value

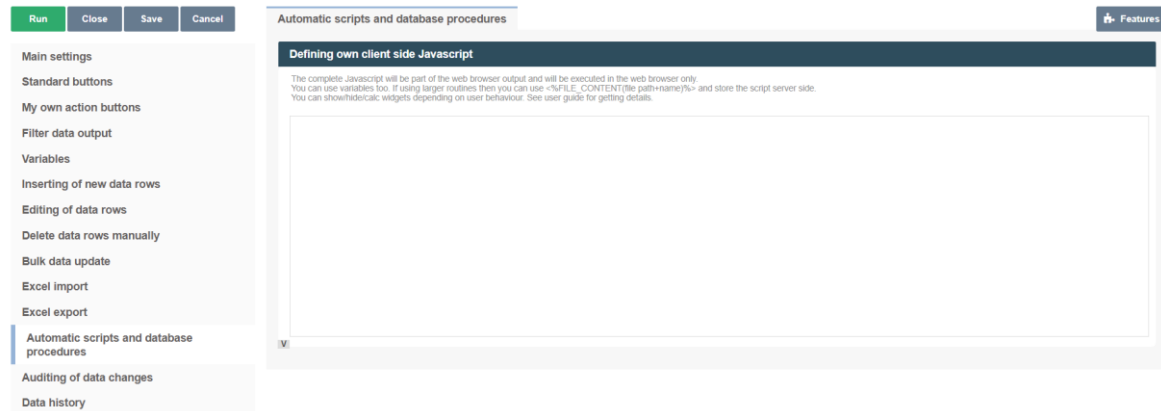
<%STATUS\_ID%>

☐ Hide value of this widget if used variable is empty

OK

CANCEL

As label value we simply use the widget reference variable of the status widget. In a calculation, 'true' or similar would usually be output.



In the window "Define own client side Javascript" we enter the following function:

```
function onAfeFormReload() {
    $(document).ready(function(){
        //for multiple widgets or values the method can be called more then once.
        //Note: function onAfeFormReload() can be used only once.
        hideTableRowSelection('.jsID_E_0_27', '2');
    })
}
```

In detail:

**function onAfeFormReload() {**

Calls a special Apparo function that, as soon as the business case starts or is reloaded (e.g. when OK is clicked), starts the following JavaScript:

**\$(document).ready(function(){**

Once the web page has finished loading, start a function with the following command(s):

**hideTableRowSelection('.jsID\_E\_0\_27', '2');**

Hide the selection checkbox if the value of the widget of type 'Label with variables' '**.jsID\_E\_0\_27**' is equal to two (lookup ID value for "Ready for approval").

**Result:**

<input type="checkbox"/>	Year	Month	Office	Product line	Product	My status	Revision status	Plan data	Plan2	Pl:
<input checked="" type="checkbox"/>	2017	1	London office	Trousers	Talli	Ready for approval	Open	4007	500	
<input type="checkbox"/>	2017	1	London office	Trousers	Talli	open	Open	5555	0	

The user can now no longer delete this row.

### 13.7 Show and hide a widget

For a Table Business Case:

- **hideAfeTableWidget(this,<javascript selector ID>);**
- **unhideAfeTableWidget(this, <javascript selector ID>');**

For a Single Business Case:

- **hideAfeWidget<javascript selector ID>;**
- **unhideAfeWidget(<javascript selector ID>;**

Note: Even if the widget is hidden, the value is stored in the database destination table.

Example:

```
$(document).on('change', '.jsID_E_0_2', function() {

    let ID = getAfeTableWidgetNumValue(this, '.jsID_E_0_2');
    if(ID == 2022) {
        hideAfeTableWidget(this, '.jsID_E_0_12' ); // hide widget of current row if year is 2022
    }
    else {
        unhideAfeTableWidget(this, '.jsID_E_0_12' ); // show widget of current row
    }
});
```

## 13.8 Possibilities of Lookup Widgets

Lookup widgets can only be read, but not set.

### 13.8.1 Lookup key values

Lookup key values can be read like this:

**In the table business case**

```
getAfeTableWidgetNumValue(sourceElement, targetElementSelector)
```

```
var myLabelValue = getAfeTableWidgetNumValue(this, '.jsID_E_0_4');
```

**In the single business case**

```
getAfeWidgetNumValue(targetElement)
```

```
var myLabelValue = getAfeWidgetNumValue('.jsID_E_0_4');
```

### 13.8.2 Lookup output values (Label)

The lookup output values can be read like this:

**In the table business case**

```
getAfeTableWidgetLookupLabel(sourceElement, targetElementSelector)
```

```
var myLabelValue = getAfeTableWidgetLookupLabel(this, '.jsID_E_0_4');
```

**In the single business case**

```
getAfeWidgetLookupLabel(targetElement)
```

```
var myLabelValue = getAfeWidgetLookupLabel('.jsID_E_0_4');
```

### 13.9 Aggregate all values of a column in a table Business Case

It is possible to perform calculations over all used rows of a widget (=column).

The column must be numeric.

All values of only the current page (=all visible data rows) are taken into account.

The sum can be output e.g. in a calculation widget.

Product	Sum year	January	February	March	April	May	June	July	August	September	October	November	December	Workflow
T-Shirt Vienna	3,000.00	0.00	0.00	200.00	300.00	2,000.00	300.00	120.00	200.00	-30.00	-30.00	-30.00	-30.00	open
Lueneburg	50,000.00	0.00	0.00	200.00	300.00	3,450.00	300.00	120.00	200.00	11,357.50	11,357.50	11,357.50	11,357.50	Ready for approval
Bags New York	-39,857.50	0.00	0.00	200.00	300.00	3,450.00	300.00	120.00	200.00	1,857.50	1,857.50	-50,000.00	1,857.50	open
New Yorker	20,000.00	0.00	0.00	400.00	500.00	500.00	5,005.00	500.00	500.00	3,148.75	3,148.75	3,148.75	3,148.75	Ready for approval
Nightblue	300.00	0.00	0.00	800.00	500.00	500.00	5,005.00	500.00	500.00	-1,876.25	-1,876.25	-1,876.25	-1,876.25	open
Gilbert	5,000.00	0.00	0.00	1,200.00	500.00	500.00	5,005.00	500.00	500.00	-801.25	-801.25	-801.25	-801.25	open
Luxor	8,000.00	0.00	0.00	800.00	500.00	500.00	5,005.00	500.00	500.00	48.75	48.75	48.75	48.75	Ready for approval
Madox	10,000.00	0.00	0.00	800.00	500.00	500.00	5,005.00	500.00	500.00	548.75	548.75	548.75	548.75	open
	56,442.50	0.00	0.00	4,600.00	3,400.00	11,400.00	25,925.00	2,860.00	3,100.00	14,253.75	14,253.75	-37,603.75	14,253.75	

#### getAfeTableColumnFunction(targetColumnSelector, functionName)

**targetColumnSelector** refers to the widget for which the function is to be calculated over all rows.

**functionName** refers to the function and can be one of the following:

- **Sum:** getAfeTableColumnFunction('.jsID\_E\_0\_3', 'sum')
- **Minimum:** getAfeTableColumnFunction('.jsID\_E\_0\_3', 'min')
- **Maximum:** getAfeTableColumnFunction('.jsID\_E\_0\_3', 'max')
- **Average:** getAfeTableColumnFunction('.jsID\_E\_0\_3', 'avg')

#### 13.9.1 Example for sum over one column

For this we need a calculation widget (without content) of type Label with variables for the output.

The output value is calculated and entered by the script.

```
var Write_Sum;
```

With this we define the JavaScript variable **Write\_Sum**

```
Write_Sum = getAfeTableColumnFunction('.jsID_E_0_3', 'sum');
```

Herewith we fill the variable **Write\_Sum** with the calculation of the sum

**getAfeTableColumnFunction('.jsID\_E\_0\_3', 'sum')** over all visible rows of the widget **'.jsID\_E\_0\_3'**.

```
setAfeWidgetNumValue('.jsID_C_0_0', Write_Sum);
```

Here we write **setAfeWidgetNumValue('.jsID\_C\_0\_0', Write\_Sum)**; the calculation stored in the variable **Write\_Sum** to the calculation widget **'.jsID\_C\_0\_0'**.

### 13.10 Use of variables

As already mentioned the use of variables is only possible to a limited extent. If these are to be included in calculations, they must first be output in a widget of the type label with variables and then read from there with the get method.

Widget settings

Widget type Mapping & Other Features Visual Help texts Data output format

Input field  
Text area  
Checkbox  
Simple dropdown (target table only)  
Lookup dropdown (for all tables)  
Multi select lookup  
Label  
Label with variables  
Spacer & Title  
Business Case link  
File upload/download

Label with variables: Displaying static text and content of variables.

Internal description:

JavaScript selector ID: jsID\_E\_0\_20

OK CANCEL

The JavaScript selector ID of the widget is required.

Widget settings

Widget type Mapping & Other Features Visual Help texts Data output format

Label value: <%offsetMay%>

Hide value of this widget if used variable is empty: ☐

OK CANCEL

The variable is output as a label value.

Widget settings

Widget type Mapping & Other Features Visual Help texts Data output format

Hiding

☐ Hide this widget in the editing area

☐ Hide this widget in the inserting area

☒ Hide this widget in edit and inserting area for: all users

OK CANCEL

Optionally, the widget can also be hidden as 'hidden'.

**Example:**

```
setAfeTableWidgetNumValue(this, 'jsID_E_0_7', calcValueNum+
getAfeTableWidgetNumValue(this, 'jsID_E_0_20')
```

With **setAfe** the value of the JavaScript variable **calcValueNum** is written into the widget '**jsID\_E\_0\_7**', added with the value of the Apparo variable **<%offsetMay%>**, output in widget '**jsID\_E\_0\_20**'.

### 13.11 Using server side variables

SQL variables and script variables can also be recalculated at runtime via Javascript and the result can be reused in the browser.

Example:

```
var srcElement;

$(document).on('change', '.jsID_E_0_3', function(event) {
    srcElement = this;
    let dropdownKey = getAfeTableWidgetValue(srcElement, '.jsID_E_0_3');
    let dropdownValue = getAfeTableWidgetLookupLabel(srcElement, '.jsID_E_0_3');

    afeReadVariableJs([
        {name:'variableName', value:'myFeVariable'},
        {name:'callbackMethodName', value:'myJsMethod'},
        {name:'FE_ID', value: dropdownKey},
        {name:'FE_COLOUR', value: dropdownValue} ]);
})

$(document).on('change', '.jsID_E_0_4', function(event) {
    srcElement = this;
    let dropdownKey = getAfeTableWidgetValue(srcElement, '.jsID_E_0_4');
    let dropdownValue = getAfeTableWidgetLookupLabel(srcElement, '.jsID_E_0_4');

    afeReadVariableJs([
        {name:'variableName', value:'myFeVariable'},
        {name:'callbackMethodName', value:'mySecondJsMethod'},
        {name:'FE_ID', value: dropdownKey},
        {name:'FE_COLOUR', value: dropdownValue} ]);
})

function myJsMethod(result) {
    setAfeTableWidgetValue(srcElement, '.jsID_E_0_1', 'color code is: '+result[0]);
}

function mySecondJsMethod(result) {
    setAfeTableWidgetValue(srcElement, '.jsID_E_0_1', 'second color code is: '+result[0]);
}
```

If the value of the drop-down widget is changed ('.jsID\_E\_0\_3' or '.jsID\_E\_0\_4'), then the request is sent to the server using the Javascript method **afeReadVariableJs** to resolve the variable **<%myFeVariable%>** and update the text widget '.jsID\_E\_0\_1' according to the result.



The `afeReadVariableJs` method has these parameters:

- **variableName:** Name of the variable to be recalculated
- **callbackMethodName:** Name of the Javascript method in the browser that is automatically called after the calculation. This also supplies the calculation result.
- **FE\_x:** Parameters are sent to the server as `<%FE_x%>` variables. This means that the variable must begin with `FE_`, the rest of the name is freely definable.

In the variable itself, this parameter can then also be used with `<%FE_x%>`. For example, `<%FE_PRODUCT_ID%>`

All parameters must be of type String or Number.

The used script variable:

```
//declare the result variable
var resultArray;

var id = '<%FE_ID%>';
var colour = '<%FE_COLOUR%>';

//check if the dropdown values are filled
if(id && colour) {
    //select values from the database
    resultArray = afe.executeSqlSelectOneRow("select CODE, ID, COLOUR from BASE_LOOKUP where ID =
<%FE_ID%> and COLOUR = '<%FE_COLOUR%>' ");
}
else {
    //otherwise return empty values
    resultArray = ["", ""];
}

resultArray;
```

### 13.12 Use of larger JavaScript programmes

With increasing complexity, it is advisable to outsource the JavaScript to an external file.

With the variable:

```
<%FILE_CONTENT(path+file)%>
```

you can import the contents of the file again.

Example call of the file myJsFunctions.txt:

```
<%FILE_CONTENT(D:\My Data\script\myJsFunctions.txt)%>.
```

The following replaces the variable with the contents of the file:

```
myAlert(); // <- Example content of the file.
```

### 13.13 Enter key for calling the JavaScript routine

The Enter key is normally used to simulate a click on the OK key.

However, if the user is to be able to start the calculations with the enter key, the function "ready" must be extended:

```
$(document).ready(function(){  
// The business case has been started, the Calc widget is set to 0  
setAfeWidgetNumValue('.jsID_C_0_0', 0);  
  
    disableFormSubmitOnEnter();  
})
```

With **disableFormSubmitOnEnter();** the enter key is converted and only calls the JavaScript routine.

### 13.14 Updating the entered numbers to match the existing number format

When entering numbers, e.g. 1000, after clicking the OK button it will be adjusted to the set widget number format, e.g. 1,000.00.

When using the script method, we instead have to call a method to achieve the formatting without having to click the OK button.

Instead of just calling the `calculateYearSum` function:

```
$(document).on('change', '.jsID_E_O_20', function() {
  calculateYearSum(this);
})
```

We call the method that formats the input values:

```
$(document).on('change', '.jsID_E_O_20', function() {
  setInputWidgetNumFormat(this); // format the numeric and string input values according to the widget settings
  calculateYearSum(this);
})
```

### 13.15 Example of a Table Business Case for planning

In this example, a small planning application is developed.

1. The entered annual total is distributed over the annual months, ignoring the previous months (these are "frozen").
2. If the user has entered a monthly plan value, the annual total is automatically updated.
3. For May, an additional value is automatically added from a hidden "Label with variables" widget.
4. The monthly totals for all selected products and the yearly total are automatically calculated.

**Demo of distributing yearly budget to 12 month**

☐ Product id  
☐ Bags New York  
☐ Gilbert  
☐ Lueneburg  
☐ Luxor  
☐ Madox

SEARCH RESET FILTERS

Product	Sum year	January	February	March	April	May	June	July	August	September	October	November	December	Workflow	Comment for	Last user
T-Shirt Vienna	3,000.00	0.00	0.00	200.00	300.00	2,000.00	300.00	120.00	200.00	-30.00	-30.00	-30.00	-30.00	open		sales
Lueneburg	50,000.00	0.00	0.00	200.00	300.00	3,450.00	300.00	120.00	200.00	11,357.50	11,357.50	11,357.50	11,357.50	Ready for approval		administ
Bags New York	-39,857.50	0.00	0.00	200.00	300.00	3,450.00	300.00	120.00	200.00	1,857.50	1,857.50	-50,000.00	1,857.50	open		administ
New Yorker	20,000.00	0.00	0.00	400.00	500.00	500.00	5,005.00	500.00	500.00	3,148.75	3,148.75	3,148.75	3,148.75	Ready for approval		administ
Nightblue	300.00	0.00	0.00	800.00	500.00	500.00	5,005.00	500.00	500.00	-1,876.25	-1,876.25	-1,876.25	-1,876.25	open		administ
Gilbert	5,000.00	0.00	0.00	1,200.00	500.00	500.00	5,005.00	500.00	500.00	-801.25	-801.25	-801.25	-801.25	open		administ
Luxor	8,000.00	0.00	0.00	800.00	500.00	500.00	5,005.00	500.00	500.00	48.75	48.75	48.75	48.75	Ready for approval		administ
Madox	10,000.00	0.00	0.00	800.00	500.00	500.00	5,005.00	500.00	500.00	548.75	548.75	548.75	548.75	open		administ
	56,442.50	0.00	0.00	4,600.00	3,400.00	11,400.00	25,925.00	2,860.00	3,100.00	14,253.75	14,253.75	-37,603.75	14,253.75			

OK CLOSE

You can find this example business case in the public demo

<https://demo.apparo.services>

Demonstration Apparo Fast Edit

Business Case List

administrator

Demonstration

Go to Portal

+ New

Edit

+ New

X Delete

Copy/Move

Import

Export

Filter

Business Case Folders

Demonstration

Master Data (MDM)

Planning

Standalone Demo

Start

Business Case ID

Name

Type

Connection name

Target table/view

Last change user

Last change date

Planning year

> Planning year with auto-distributing month values + sums

Table

SAMPLES

SAMPLE\_FORECAST5

administrator

8/23/21 11:29 AM

> SAMPL APP SALES

> SAMPLES - Sales

Set

Administrator

11/28/17 2:48 PM

> SAMPL APP SALES MAN

> SAMPLES - Sales manager

Table

SAMPLES

SAMPLE\_SALES

Administrator

11/28/17 2:48 PM

> SAMPL MASTER PLAN DETAILS

> SAMPLES - product details

Single

SAMPLES

SAMPLE\_PRODUCT

Administrator

11/28/17 2:48 PM

> SAMPL PLAN SALES PLAN

> SAMPLES - sales planning

Table

SAMPLES

SAMPLE\_SALES\_PLANNING

Administrator

11/28/17 2:48 PM

Show description

Connections

Business Cases

Settings

Clients

Training

About Anaplan

#### Hint

In the training menu of the Apparo Designer, you can find the making of videos for this Business Case.

## The Script:

```
// sum of a product was changed:
$(document).on('change', '.jsID_E_0_1', function(){

// get current value of the sum widget:
var myValue = getAfeTableWidgetNumValue(this, '.jsID_E_0_1');

// Date calculations...
var currentDate = new Date();
var currentMonth = currentDate.getMonth() + 2;
var months = 13 - currentMonth;

//calculate sum of values in the past
var sumOfPast = 0;
if (currentMonth >= 2) { sumOfPast = getAfeTableWidgetNumValue(this, '.jsID_E_0_3'); };
if (currentMonth >= 3) { sumOfPast = sumOfPast + getAfeTableWidgetNumValue(this, '.jsID_E_0_4'); };
if (currentMonth >= 4) { sumOfPast = sumOfPast + getAfeTableWidgetNumValue(this, '.jsID_E_0_5'); };
if (currentMonth >= 5) { sumOfPast = sumOfPast + getAfeTableWidgetNumValue(this, '.jsID_E_0_6'); };
if (currentMonth >= 6) { sumOfPast = sumOfPast + getAfeTableWidgetNumValue(this, '.jsID_E_0_7'); };
if (currentMonth >= 7) { sumOfPast = sumOfPast + getAfeTableWidgetNumValue(this, '.jsID_E_0_8'); };
if (currentMonth >= 8) { sumOfPast = sumOfPast + getAfeTableWidgetNumValue(this, '.jsID_E_0_9'); };
if (currentMonth >= 9) { sumOfPast = sumOfPast + getAfeTableWidgetNumValue(this, '.jsID_E_0_10'); };
if (currentMonth >= 10) { sumOfPast = sumOfPast + getAfeTableWidgetNumValue(this, '.jsID_E_0_11'); };
if (currentMonth >= 11) { sumOfPast = sumOfPast + getAfeTableWidgetNumValue(this, '.jsID_E_0_12'); };
if (currentMonth == 12) { sumOfPast = sumOfPast + getAfeTableWidgetNumValue(this, '.jsID_E_0_13'); };

// change values of future months and current only
// toFixed(2) rounds the value to 2 decimal places, but the result is string. The '+' in the begin convert this string into English number
var calcValueNum = + (((myValue-sumOfPast) / months).toFixed(2));

if (currentMonth == 1) { setAfeTableWidgetNumValue(this, '.jsID_E_0_3', calcValueNum); }
if (currentMonth <= 2) { setAfeTableWidgetNumValue(this, '.jsID_E_0_4', calcValueNum); }
if (currentMonth <= 3) { setAfeTableWidgetNumValue(this, '.jsID_E_0_5', calcValueNum); }
if (currentMonth <= 4) { setAfeTableWidgetNumValue(this, '.jsID_E_0_6', calcValueNum); }

// for May additional the calculated value of E_0_17 must be added.
// this is an example how to use calculated values from the database, e.g. SQL. The value is stored in a hidden widget of type "Label with variables"
if (currentMonth <= 5) { setAfeTableWidgetNumValue(this, '.jsID_E_0_7', calcValueNum+ getAfeTableWidgetNumValue(this, '.jsID_E_0_17') ); }

if (currentMonth <= 6) { setAfeTableWidgetNumValue(this, '.jsID_E_0_8', calcValueNum); }
if (currentMonth <= 7) { setAfeTableWidgetNumValue(this, '.jsID_E_0_9', calcValueNum); }
if (currentMonth <= 8) { setAfeTableWidgetNumValue(this, '.jsID_E_0_10', calcValueNum); }
if (currentMonth <= 9) { setAfeTableWidgetNumValue(this, '.jsID_E_0_11', calcValueNum); }
if (currentMonth <= 10) { setAfeTableWidgetNumValue(this, '.jsID_E_0_12', calcValueNum); }
if (currentMonth <= 11) { setAfeTableWidgetNumValue(this, '.jsID_E_0_13', calcValueNum); }
if (currentMonth <= 12) { setAfeTableWidgetNumValue(this, '.jsID_E_0_14', calcValueNum); }

// because the particular values may be rounded, recalculate the SUM in order to reflect the sum of rounded values
calculateYearSum(this);
})

$(document).on('change', '.jsID_E_0_3', function(){
// january value was changed
calculateYearSum(this);
})

$(document).on('change', '.jsID_E_0_4', function(){
// February value was changed
calculateYearSum(this);
})

$(document).on('change', '.jsID_E_0_5', function(){
calculateYearSum(this);
})

$(document).on('change', '.jsID_E_0_6', function(){
calculateYearSum(this);
})

$(document).on('change', '.jsID_E_0_7', function(){
calculateYearSum(this);
})

$(document).on('change', '.jsID_E_0_8', function(){
calculateYearSum(this);
})

$(document).on('change', '.jsID_E_0_9', function(){
```

```

    calculateYearSum(this);
})

$(document).on('change', '.jsID_E_0_10', function(){
    calculateYearSum(this);
})

$(document).on('change', '.jsID_E_0_11', function(){
    calculateYearSum(this);
})

$(document).on('change', '.jsID_E_0_12', function(){
    calculateYearSum(this);
})

$(document).on('change', '.jsID_E_0_13', function(){
    calculateYearSum(this);
})

$(document).on('change', '.jsID_E_0_14', function(){
    calculateYearSum(this);
})

function calculateYearSum(elem) {
// make a sum of all months of the current product
var yearSum = getAfeTableWidgetNumValue(elem, '.jsID_E_0_3');
yearSum = yearSum + getAfeTableWidgetNumValue(elem, '.jsID_E_0_4');
yearSum = yearSum + getAfeTableWidgetNumValue(elem, '.jsID_E_0_5');
yearSum = yearSum + getAfeTableWidgetNumValue(elem, '.jsID_E_0_6');
yearSum = yearSum + getAfeTableWidgetNumValue(elem, '.jsID_E_0_7');
yearSum = yearSum + getAfeTableWidgetNumValue(elem, '.jsID_E_0_8');
yearSum = yearSum + getAfeTableWidgetNumValue(elem, '.jsID_E_0_9');
yearSum = yearSum + getAfeTableWidgetNumValue(elem, '.jsID_E_0_10');
yearSum = yearSum + getAfeTableWidgetNumValue(elem, '.jsID_E_0_11');
yearSum = yearSum + getAfeTableWidgetNumValue(elem, '.jsID_E_0_12');
yearSum = yearSum + getAfeTableWidgetNumValue(elem, '.jsID_E_0_13');
yearSum = yearSum + getAfeTableWidgetNumValue(elem, '.jsID_E_0_14');

// recalculate all month sums of all products
calculateColumnSums();

// set sum of year of current product
setAfeTableWidgetNumValue(elem, '.jsID_E_0_1', yearSum);
}

function calculateColumnSums() {

// recalculates all month sums

var m;
m = getAfeTableColumnFunction('.jsID_E_0_3', 'sum');
setAfeWidgetNumValue('.jsID_C_0_3', m);

m = getAfeTableColumnFunction('.jsID_E_0_4', 'sum');
setAfeWidgetNumValue('.jsID_C_0_4', m);

m = getAfeTableColumnFunction('.jsID_E_0_5', 'sum');
setAfeWidgetNumValue('.jsID_C_0_5', m);

m = getAfeTableColumnFunction('.jsID_E_0_6', 'sum');
setAfeWidgetNumValue('.jsID_C_0_6', m);

m = getAfeTableColumnFunction('.jsID_E_0_7', 'sum');
setAfeWidgetNumValue('.jsID_C_0_7', m);

m = getAfeTableColumnFunction('.jsID_E_0_8', 'sum');
setAfeWidgetNumValue('.jsID_C_0_8', m);

m = getAfeTableColumnFunction('.jsID_E_0_9', 'sum');
setAfeWidgetNumValue('.jsID_C_0_9', m);

m = getAfeTableColumnFunction('.jsID_E_0_10', 'sum');
setAfeWidgetNumValue('.jsID_C_0_10', m);

m = getAfeTableColumnFunction('.jsID_E_0_11', 'sum');
setAfeWidgetNumValue('.jsID_C_0_11', m);

m = getAfeTableColumnFunction('.jsID_E_0_12', 'sum');
setAfeWidgetNumValue('.jsID_C_0_12', m);

```

```

m = getAfeTableColumnFunction('.jsID_E_0_13', 'sum');
setAfeWidgetNumValue( '.jsID_C_0_13', m);

m = getAfeTableColumnFunction('.jsID_E_0_14', 'sum');
setAfeWidgetNumValue( '.jsID_C_0_14', m);

// calc total sum and display it, value is the sum of all products
setAfeWidgetNumValue( '.jsID_C_0_1',
  getAfeTableColumnFunction('.jsID_E_0_3', 'sum') +
  getAfeTableColumnFunction('.jsID_E_0_4', 'sum') +
  getAfeTableColumnFunction('.jsID_E_0_5', 'sum') +
  getAfeTableColumnFunction('.jsID_E_0_6', 'sum') +
  getAfeTableColumnFunction('.jsID_E_0_7', 'sum') +
  getAfeTableColumnFunction('.jsID_E_0_8', 'sum') +
  getAfeTableColumnFunction('.jsID_E_0_9', 'sum') +
  getAfeTableColumnFunction('.jsID_E_0_10', 'sum') +
  getAfeTableColumnFunction('.jsID_E_0_11', 'sum') +
  getAfeTableColumnFunction('.jsID_E_0_12', 'sum') +
  getAfeTableColumnFunction('.jsID_E_0_13', 'sum') +
  getAfeTableColumnFunction('.jsID_E_0_14', 'sum') );
}

$(document).ready(function(){
// Business Case was started, this function will be called automatically, the calc widget are updated

// pressing enter key means new event and not making submit
disableFormSubmitOnEnter();

// calc month sums:
calculateColumnSums();

})

function onAfeFormReload() {
  $(document).ready(function(){
// Business Case after submit (e.g. pressing OK button) is calling this function automatically

// enter key means new event
disableFormSubmitOnEnter();

// calc month sums:
calculateColumnSums();
  })
}

$(document).on('focus', '.jsID_E_0_1', function(){

// the user has clicked into the sum widget. Now this function is called automatically.
// This is helpful if you want to make calculations directly after user clicked into a widget

// ... place for activities

})

```

## 14 Business logic in a Cognos report

With an Action Business Case, you can incorporate actions directly into existing Cognos reports.

The user can set / delete a checkbox or press a button.

An Action Business Case is then called automatically, which can, for example, execute SQL.

Press the button

A Business Case is called automatically if the user is pressing a button or setting a checkbox in a list.

ID	custom ID	HTML item	yes/no	Year	Jan	Feb	March	April
10	40	<input type="checkbox"/>	0	2019	0	0	0	0
10	40	<input type="checkbox"/>	0	2020	2	0	0	0
10	40	<input type="checkbox"/>	0	2021	0	0	5551	0
15	20	<input type="checkbox"/>	0	2019	15000	12000	580	79600
15	40	<input type="checkbox"/>	0	2020	0	0	0	0
15	40	<input type="checkbox"/>	0	2020	0	0	0	0
20	30	<input type="checkbox"/>	0	2020	8000	9000	10000	18000
Overall - Summary		Overall - Summary	0	14139	23002	21000	16131	97600

[You will find a detailed description in the chapter "Action Business Case \(ABC\)"](#)



## 15 Primary Keys & Not Null columns

Each column (also many columns at the same time) can be used as the primary key.

In this case it is not important how the primary key is defined within the database table.  
In Apparo Fast Edit you can define a completely different primary key.

Apparo Fast Edit is using the primary key in insert and update case only.

The definition in the database will be not used.

Likewise only the null/not null-definitions from Fast Edit will be used.  
The definition in the database will be not be used.

## 16 Adding Business Cases into Portal, Cognos reports or Dashboards

You have following possibilities – just open the right entry.

Target table	Header	Footer	Visual	Colours	Widgets	Row ordering	Link to IBM Cognos Analytics
Here you can define a new Cognos portal entry for this Business Case/Set or add it into an existing Report Studio report.							
<ul style="list-style-type: none"> <li>▶ Link into IBM Cognos portal</li> <li>▶ Link into IBM Cognos Report Studio report</li> <li>▶ Link into Dashboard</li> </ul>							

### 16.1 New entry in Cognos portal

Target table	Header	Footer	Visual	Colours	Widgets	Row ordering	Link to IBM Cognos Analytics
Here you can define a new Cognos portal entry for this Business Case/Set or add it into an existing Report Studio report.							
<div> <div>▼ Link into IBM Cognos portal</div> <div> <p>Please select the Cognos portal folder in that the new entry shall be stored.</p> <p>Entry for Business Case <b>SAMPLES - product lines</b> for IBM Cognos portal:</p> <p>If you want to create a new start entry for this Business Case in the selected portal folder then press the 'Create portal entry' button</p> <div> <div>▼</div> <div>Apparo</div> </div> <div> <input checked="" type="checkbox"/> Include 'client id' parameter to the portal entry           <p>Adding the client id means that the system is trying to open the Business Case of this client only. Without this client parameter it is looking through all clients from top to down using security settings/client security groups only</p> <div>CREATE PORTAL ENTRY</div> </div> </div> </div>							
<ul style="list-style-type: none"> <li>▶ Link into IBM Cognos Report Studio report</li> <li>▶ Link into Dashboard</li> </ul>							

Select the target folder and press “CREATE PORTAL ENTRY”. After that you will see a new entry in the portal for starting this Business Case.

## 16.2 Linking Business Case into a report

### Possibility #1: Link to call the Business Case in own window

You can add this HTML into a list.

```
'<a href="javascript:void(0)" onclick="window.open('/ibmcognos/bi/v1/apparoBusinessCase.html?bc=SAMPL+MASTER+PROD+LINES&clientid=Demo'+
'&p1=' + [DEFINE HERE PRIMARY KEY VALUE 1 AND USE number2string() IF IT IS A NUMBER]'+
"" _blank","directories=0, location=0, menubar=0, scrollbars=0, status=1, titlebar=0, toolbar=0, top=20, left=20, width=900, height=400");">'+
'Label of Hyperlink'
+'</a>'
```

### Possibility #2: Embed this Business Case into a Report Studio report (the same window like report):

Do not use it as an element of list.

```
'<iframe src="/ibmcognos/bi/v1/apparoBusinessCase.html?bc=SAMPL+MASTER+PROD+LINES&clientid=Demo&embedded=true'+
'&p1=' + [DEFINE HERE PRIMARY KEY VALUE 1 AND USE number2string() IF IT IS A NUMBER]'+
"" name="apparo" height="600" width="600"></iframe>'
```

### Possibility #3: Embed this Business Case into a Report Studio report in a list (the same window like report):

Please add the first HTML part inside the list to define links and second HTML part outside the list to define output area of this Business Case.

```
'<a href="/ibmcognos/bi/v1/apparoBusinessCase.html?bc=SAMPL+MASTER+PROD+LINES&clientid=Demo&embedded=true'+
'&p1=' + [DEFINE HERE PRIMARY KEY VALUE 1 AND USE number2string() IF IT IS A NUMBER]'+
"" target="apparo">edit...</a>'
```

Put this HTML in a new HTML item. It is defining the output position and size of the Business Case.

```
<iframe name="apparo" height="600" width="600" frameborder="0" />
```

## Copy the URL

The URL always contains placeholders for all set primary keys and for all containing report variables.

### 16.3 Adding a Business Case link

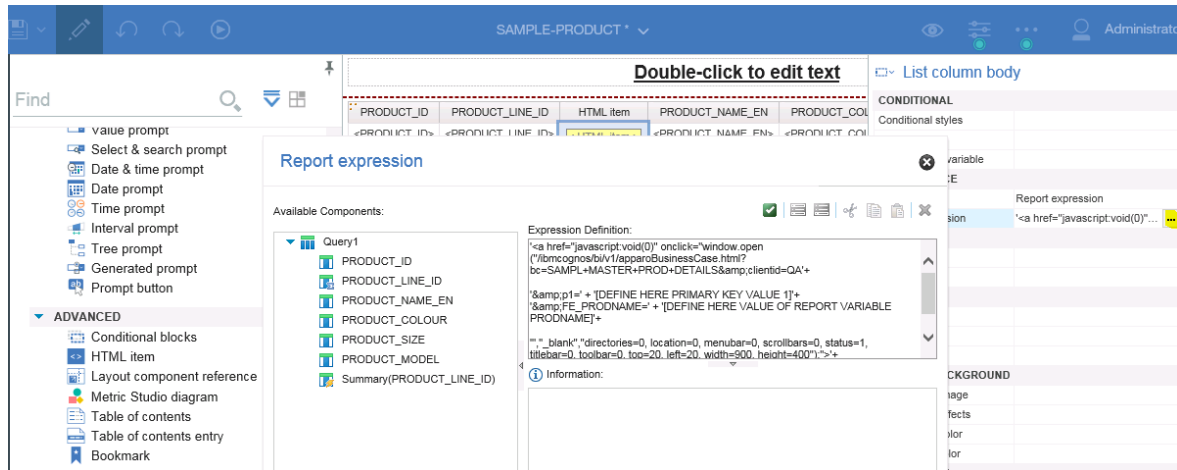
Pick an HTML item from the toolbox and add it to your list.

The screenshot shows the APPARO interface with a table titled 'Double-click to edit text'. The table has columns: PRODUCT\_ID, PRODUCT\_LINE\_ID, HTML item, PRODUCT\_NAME\_EN, PRODUCT\_COLOUR, PRODUCT\_SIZE, and PRODUCT\_MC. The 'HTML item' column contains three rows with the value '< HTML item >'. The 'Overall - Summary' row has the value '< Summary(PRODUCT\_LINE\_ID) >'. On the left, the 'Find' panel shows a list of components, with 'HTML item' highlighted under the 'ADVANCED' section.

Now open the properties and change the source type to 'Report expression'

The screenshot shows the APPARO interface with the 'HTML item' selected. The 'Properties' panel on the right is open, showing the 'HTML SOURCE' section. The 'Source type' is set to 'Report expression'. The 'Report expression' field contains the code: '<a href="javascript:void(0)" on'. The 'Find' panel on the left shows the 'HTML item' highlighted under the 'ADVANCED' section.

Then open the expression editor and paste the copied URL there (yellow marked)



### Example URL:

```
'<a href="javascript:void(0)"
onclick="window.open("/ibmcognos/bi/v1/apparoBusinessCase.html?bc=SAMPL+MASTER+PROD+DETAILS
&amp;clientid=QA'+

'&amp;p1=' + '[DEFINE HERE PRIMARY KEY VALUE 1]'+
'&amp;FE_PRODNAME=' + '[DEFINE HERE VALUE OF REPORT VARIABLE PRODNAME]'+

'", "_blank", "directories=0, location=0, menubar=0, scrollbars=0, status=1, titlebar=0, toolbar=0, top=20,
left=20, width=900, height=400");">'+
'Label of Hyperlink'
+'</a>'
```

### Now edit the URL and replace the string in the brackets

```
'<a href="javascript:void(0)"
onclick="window.open("/ibmcognos/bi/v1/apparoBusinessCase.html?bc=SAMPL+MASTER+PROD+DETAILS
&amp;clientid=QA'+

'&amp;p1=' + number2string([Query1].[PRODUCT_ID])+
'&amp;FE_PRODNAME=' + [Query1].[PRODUCT_NAME_EN]+

'", "_blank", "directories=0, location=0, menubar=0, scrollbars=0, status=1, titlebar=0, toolbar=0, top=20,
left=20, width=900, height=400");">'+
[Query1].[PRODUCT_NAME_EN]
+'</a>'
```

### Hints:

For the product ID we must use the function **number2string**.

*Report variables* are transporting any value you need from the report to the same named report variable in the Business Case. They always have the syntax in the URL: **FE\_VarName**

Instead of using Hyperlink as link name, you can use values as here the product name for the link

Running the report shows the result:

PRODUCT_ID	PRODUCT_LINE_ID	HTML item	PRODUCT_NAME_EN	PRODUCT_COLOUR	PRODUCT_SIZE	PRODUCT_MODEL	PRODUCT_MANUF
350	6	<a href="#">Bino</a>	Bino	blue	M		Bino
440	2	<a href="#">T-Shirt Moscow</a>	T-Shirt Moscow	green	M		Adidas
480	6	<a href="#">Bino</a>	Bino	blue	M		Bino
490	2	<a href="#">T-Shirt Moscow</a>	T-Shirt Moscow	green	M		Adidas
500	1	<a href="#">test</a>	test	green	M		test abc
510	4	<a href="#">Alon</a>	Alon	green	XL		uuuu
520	4	<a href="#">Gilbert</a>	Gilbert	yellow	XL		Hugo Boss
390	1	<a href="#">test</a>	test	green	M		test abc
400	4	<a href="#">Alon</a>	Alon	green	XL		uuuu
410	4	<a href="#">Gilbert</a>	Gilbert	yellow	XL		Hugo Boss
420	4	<a href="#">Luxor</a>	Luxor	green	XL		Hugo Boss
430	4	<a href="#">Madox</a>	Madox	white	M		Escada

When clicking the hyperlink the mapped Business Case opens

The screenshot shows the APPARO interface with a table of products on the left and a detailed view of the 'Bino' product on the right. The table lists various products with their IDs, line IDs, and names. The detailed view for 'Bino' (Product ID 350) shows its attributes: Product line (Bikinis), Product name (Bino), Colour (blue), Size (M), Model, and Manufacturer (Bino). It also displays the start date (28.02.2015) and a price history section. The interface includes a search bar, a sidebar with navigation icons, and a top bar with a 'New' button.

#### Hints:

When using the p1 parameter (=primary key value), the opened Business Case shows only results for this value.

When using a Table Business Case, the p1 parameter can be deleted to show all data unfiltered.

It is also possible to use other values, like the product line for filtering, to show all results of the clicked product line.

## 17 Usage of Cognos Prompts for filtering in Business Cases

You can use prompt values in a Business Case for filtering too.

The screenshot shows the APPARO Group Business Case Designer interface. On the left, a multi-select prompt is open, listing items: Alon, Architect, Bags New York, Bino Man, Blue Cap, and Dark Cap. Below the list are buttons for 'Select all', 'Deselect all', and 'Finish'. The main area displays a table with columns: PRODUCT\_ID, PRODUCT\_LINE\_ID, OFFICE\_ID, and SALES. The table contains data for product 14,400 and its summary. Below the table, a 'Conditional formatting demo' is shown with row and column conditions. At the bottom, a table displays data for 2017 T-Shirts, including product line, product, amount by quarter, and last change information.

PRODUCT_ID	PRODUCT_LINE_ID	OFFICE_ID	SALES
14,400	90	46	
14,400 - Summary	90	46	
14,400	Overall - Summary		
Overall - Summary	90	46	

Year	Product line	Product	Amount/Year	Quarter I	Quarter II	Quarter III	Quarter IV	Last changed by	Last change from
2017	T-Shirts	T-Shirt October	99	400	101	10	10	Administrator	04.21.2018
2017	T-Shirts	T-Shirt Vienna	10	100	100	100	100	administrator	03.29.2017

In this example, the selected prompt values are passed to the integrated Business Cases for filtering.

Method:

In the Business Case, **define a new report variable with the output type Text.**

Then use this report variable as the default value in a multi-select filter:

The screenshot shows the 'Widget settings of database column PRODUCT\_ID' dialog box. The 'Mapping & Other' tab is selected. The 'Column name' is set to 'PRODUCT\_ID'. The 'Default value' is set to 'USE LABELS:<%products%>' for all users. There are 'OK' and 'CANCEL' buttons at the bottom.

"USE LABELS" means that no IDs are expected, but a list of the output texts - e.g. MARCO POLO, TELIAND but not 20, 50

The integrated HTML for calling the Business Case must be extended:

#### Report expression

Available Components:

- Query1
  - PRODUCT\_ID
  - SALES
  - PRODUCT\_LINE\_ID
  - OFFICE\_ID
  - Summary(PRODUCT\_ID)
  - Summary(PRODUCT\_LINE\_ID)
  - Summary(OFFICE\_ID)
  - Summary(SALES)
- Query2
  - PRODUCT\_ID
  - PRODUCT\_NAME\_EN

Expression Definition:

```
'<iframe src="/ibmcognos/bi/v1/apparoBusinessCase.html?
bc=SAMPL+WF+Conditional&clientid=Demo&embedded=true' +
&FE_products=' +ParamDisplayValue('Parameter_product')+
'" name="apparo" height="600" width="1000" frameborder="0" ></iframe>'
```

Information:

Tips Errors

The 3rd line with FE\_products provides a list of the selected products to the Business Case.



## 18 Output of multiple Business Cases in one report including synchronizing

You can place multiple Business Cases in a Report Studio report:

The screenshot displays the Apparo Excel Application Replacement interface. The main window shows a list of business cases with columns for Order ID, Details, Comment, Order date, Ship date, Ship mode, Customer ID, Customer name, Product ID, Sales, Quantity, and PROFIT. A sidebar on the left allows filtering by location (Aberdeen, Abilene, Akron, Albuquerque, Alexandria, Allen) and includes a 'Finish' button. Below the sidebar, a 'Summary of the selected orders' section shows a calculated target of 19. At the bottom, a detailed view of a selected business case (CA-2014-102988) is shown, including fields for Order ID, Order date, Ship date, Ship mode, Discount, Customer ID, Customer name, Segment, Country, City, State, Postal code, Region, Product ID, Category, Sub-category, Product name, Sales, Quantity, and Profit. A 'Forward this case for further processing' button is visible at the bottom right.

The individual Business Case output can be refreshed if necessary. Individual parameters such as primary key or a report variable can also be transferred.

Thus, e.g. In the upper screen, the right business case can be refreshed including values depending on user behavior in the left Business Case.

**This removes even more complex Excel applications with IBM Cognos Analytics and Apparo.**

Advantages:

- **Always up-to-date data**, no import to Excel necessary
- There is **always only one current version of the program**, which can be accessed by all users at any time via browser
- **Data import from Excel** is simple, fast and quality assured
- **Auditing, Logging:** Any data change can be logged
- **Security** - the security system of IBM Cognos Analytics is used

The update request is controlled by JavaScript method:

**afe.refreshEmbeddedBC('iframeName', 'Business Case ID', 'parameters like p1 or report variables')**

**iframeName:**

This is the name of the HTML iframe that is containing the Business Case output.

**Business Case ID:**

The Business Case ID of the business case that is to be reissued. That You can also issue different business cases depending on your business logic

**Additional parameters:**

Here you can pass parameters such as primary key or report variables

Example: p1 = 100 & FE\_Variable =<%WIDGET\_VALUE%>

## 18.1 Usage of the JavaScript-Method

afe.refreshEmbeddedBC can be used in a **Script-Variable** or in a **JavaScript-file**.

### 18.1.1 Usage in a Script-Variable

```
var rc="";
afe.refreshEmbeddedBC('Details','Product details','p1=<%SEARCH_KEY_PRODUCT_LINE_ID%>');
rc;
```

The refresh is requested as soon as this variable is executed and the Business Case output is updated. Therefore this variable could be placed in the Business Case header or footer.

### 18.1.2 Usage in a Javascript-file

Content of file refresh.js (this file must be stored serverside in the default file directory C:\Program Files\Apparo\FastEdit\user\_scripts )

```
afe.refreshEmbeddedBC('Details','Product details','p1=<%SEARCH_KEY_PRODUCT_LINE_ID%>');
```

This Javascript-file must be called ...

- Using Pre/Post-Scripts, for example post-insert
- Execution using an own button

## 18.2 Usage of HTML hyperlink

You can also define a hyperlink with HTML in a "label with variable" widget that the user can use to refresh immediately:

```
<a
href="/ibmcognos/bi/v1/apparoBusinessCase.html?bc=PROD+LIST&clientid=Demo&embedded=true&p1
=<%PRODUCT_LINE_ID%>" target="PRODLIST">Show product details</a>
```

This example calls the Business Case with the ID "PROD LIST" and the client "Demo".  
The first primary key should have the current row value from the PRODUCT\_LINE\_ID widget (<% PRODUCT\_LINE\_ID%>).  
The iframe name is "PRODLIST" and the column displays "Show product details".

The refresh is executed as soon as the user clicks on the hyperlink.

## 19 Apparo database repository

All the settings and definitions, beside from the logos and scripts, are stored in the **Apparo database repository**.

For data storage purposes, it is to be recommended that at regular intervals, the repository be saved in the form of a database backup.

**The repository is server-independent which means that it can be moved across to another server without changes being necessary (i.e. from the development server to the productive server).**

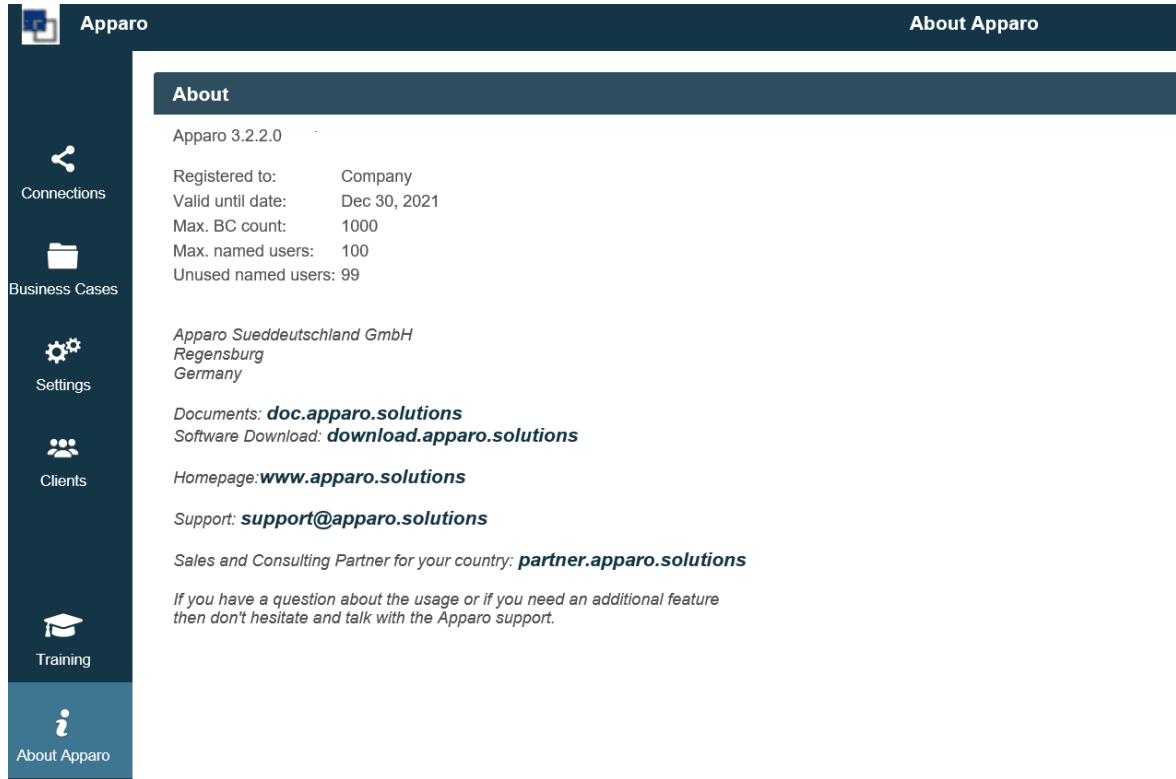
**Several Apparo Fast Edit instances can use the same repository at the same time.**

An automatic repository-update will ensue with the installation of a new Apparo Fast Edit version. After this, the **older** Apparo Fast Edit versions will be **unable to use the same updated repository**.

## 20 About Apparo

In 'About Apparo' you get in the first line information about the program version and the build.

The next block contains information about the global license key, including the registrar, the expiry date and the maximum number of business cases and users.



**About**

Apparo 3.2.2.0

Registered to:	Company
Valid until date:	Dec 30, 2021
Max. BC count:	1000
Max. named users:	100
Unused named users:	99

Apparo Sueddeutschland GmbH  
Regensburg  
Germany

Documents: **doc.apparo.solutions**  
Software Download: **download.apparo.solutions**  
Homepage: **www.apparo.solutions**  
Support: **support@apparo.solutions**  
Sales and Consulting Partner for your country: **partner.apparo.solutions**

*If you have a question about the usage or if you need an additional feature then don't hesitate and talk with the Apparo support.*

## 21 Addendum

### 21.1 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

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*

Now (without restarting Apparo) you can use this file in the Apparo Designer:

#### Widget settings of database column **PRODUCT\_LINE\_ID**

Widget type	Mapping & Other	Features	Lookup & Dropdown & Multiselect	Visual	Help texts	Data output format
Output type		<div> Number  Currency  Percentage  Date / Time  Text  Use type of output column </div>				
Decimal places		2				
Show separate groups		<input type="checkbox"/>				
How to show negative number		<div> with minus sign  with minus sign and in red colour </div>				
<b>Data quality check</b>						
Custom validator Java 8 class		<div> TesterPK  ApparoStringValidator  <b>TesterPK</b> </div>				
Interval of old value (%)						
Interval						
Sample format text font						

## 21.2 Creating an encrypted password

Passwords in repository and XML files are encrypted using AES256.  
It is possible to use encrypted pass strings instead of plain passwords

- Within the Apparo Configuration Manager
- In Apparo Designer in database and email connections

### Syntax

The file is located in [Apparo-Home]/FastEdit/etc and can be called via script or command line:

CreatePassword.bat/sh PIN PASSWORD

The PIN is always "T9puG" and is used to protect the master password, just like the pin of a credit card.

### Output

The standard output is the encrypted password. To use it later, the prefix 'CRYPTED:' must be included.

Example output:

"CRYPTED:usUa6Jilr6PGOjta+QFEeCUacDtj,BBDydcIfIDC73p+e2O+P8Mau"

### Use Case

In a company the Designer users must not know the database passwords, therefore the database administrator calls the encryptor with the parameter 'T9puG' and the password 'secret\_password123'.  
The output is like "CRYPTED:xxxxx".

He copies the whole string including the prefix and sends it to the Apparo Designer user who inserts it into the Designer password field (database connection password or eMail account password).

### 21.3 DB-Session Handling

If a user is starting a Business Case then automatically the Business Case is taking an own database session. A database session is used by one Business Case at the same time only.

If connection pooling is enabled then it is taking the session from the connection pool.

If connection pooling is disabled then Apparo Fast Edit is opening a new database session.

Apparo Fast Edit can manage database transactions.

This feature is helpful if the user want to cancel changes and rollback all changes.

If the user is pressing OK, CLOSE or CANCEL button then it has impact to the database transaction too.

That means “commit” or “rollback” is used.

If the Business Case is using the “Auto-Commit” feature then after every update/insert/delete command an additional “commit” is used and the transaction is closed automatically.

If the user is closing the Business Case in a correct way (pressing OK or CLOSE button) then the database transaction is closed with a “commit” command too. That means there are no locks because of the usage of this Business Case after the Business Case is closed by the user.

If the connection pooling is enabled then the database connection will be moved back to the pool.

If the connection pooling is disabled then the database connection will be closed.

If the user is closing a Business Case with **just closing the complete window** without pressing OK, CLOSE or CANCEL button then the database session/transaction management is different:

Apparo Fast Edit is testing automatically every minute if a Browser window that is used for running a Business Case is still open. That means if the user closed the Business Case in a non-official way then the database session is closed **automatically** 5-6 minutes later using rollback.

#### Calling database procedures and functions:

Using Oracle or IBM DB/2 then it is possible to use the same database transaction like Apparo Fast Edit is using for this Business Case.

If using MS SQL Server then using the same transaction is not possible. Therefore using commit or rollback is not allowed. Solution: Define an own transaction.

## 21.4 Usage of external web javascript frameworks like jQuery

It is possible to include javascript frameworks like jQuery for improving the output of a Business Case. jQuery is already included in Apparo Fast Edit.

How to use it:

Following jQuery example should be displayed in a single/table Business Case:

```
<button type="button" id="myButton">Start Animation</button>
<div id="myDiv" style="background:green; height:50px; width:50px;" />
<script>
$(document).ready(function(){
    $("#myButton").click(function(){
        $("#myDiv").animate({ width: '300px' });
    });
});
</script>
```

Open the Business Case and choose tab “Header” and paste it into the header description:

Target table	Header	Footer	Visual	Colours	Widgets	Row ordering	Link into Portal
Title & Description							
	Language	Title	Description				
	German	Produktliste	<pre>\$(document).ready(function(){     \$("#myDiv").animate({ width: '300px' }); }); &lt;/script&gt;</pre>				
	English	Product list	<pre>&lt;button type="button" id="myButton"&gt;Start Animation&lt;/button&gt; &lt;div id="myDiv" style="background:green; height:50px; width:50px;" /&gt; &lt;script&gt;</pre>				
Title style							
	Font face	Size	Style	Align	Colour		
	Arial	16	Bold	Left	#54B4EB		
Description style							
	Font face	Size	Style	Align	Colour		
	Arial	12	Normal	Left	#000000		
Background colour							
	#FFFFFF						
Left logo URL							
Right logo URL							



Run the Business Case:

The button and the green area are the result of the jQuery definition.

### Product list

Product list with links to product details and price history.--- Content of the Qlik variable: default

Start Animation

Search product line

Product name

SEARCH

RESET FILTERS

<input type="checkbox"/>	Details	Product line	Product name english *	Product ID	Product colour	Product size	Product manufacturer	Price valid from	Current price
<input type="checkbox"/>	> Details	Jackets	Madox	170	white	M	Escada	01.15.2009	> US\$ 50
<input type="checkbox"/>	> Details	Underwear	Lino outdoor	340	green	L	Lino	02.22.2015	> US\$ 55
<input type="checkbox"/>	> Details	T-Shirts	Architect	470	black	L	Woodstock	01.26.2017	> US\$
<input type="checkbox"/>	> Details	T-Shirts	T-Shirt 69's	480	white	M	Woodstock	02.21.2017	> US\$ 29
<input type="checkbox"/>	> Details	Bags	Softbag	460	yellow	XL	Bags United	01.13.2017	> US\$
<input type="checkbox"/>	> Details	Jackets	Wrangler	450	white	M	Holister	08.10.2016	> US\$ 55
<input type="checkbox"/>	> Details	Bikinis	Bino Man	350	blue	M	Bino	02.28.2015	> US\$ 4
<input type="checkbox"/>	> Details	T-Shirts	T-Shirt Madox	440	black	M	Adidas	01.15.2009	> US\$ 40

## 21.5 Hidding of the top header of Business Case output

Please put

```
<script type="text/javascript">
document.getElementById('headerPanel').style.display = 'none'; </script>
```

into the header setting of the Business Case. Now the top header is hidden..