Version: 5.0 / 2019-07-01

Introduction

VisionX actions allow the user to add more functionality/features to their applications. Actions are used to implement functions in workscreens, such as clicking a button named “Report” to open a report, or calculating the value `QUANTITY * PRICE` in the “SUM” column based on a value change in the “QUANTITY” column.

There are numerous examples for the use of actions. The following chapters will explain the functionality of actions using common examples.

We will begin with the basics.

Basics

Actions are created, edited and deleted via VisionX's GUI designer. They can be used for a number of events, such as, the click of a button. When the selected event occurs during the use of a workscreen, the respective action is performed.

Example: When the event “button click” occurs, the specified action is executed. For example, the action can show a a report.

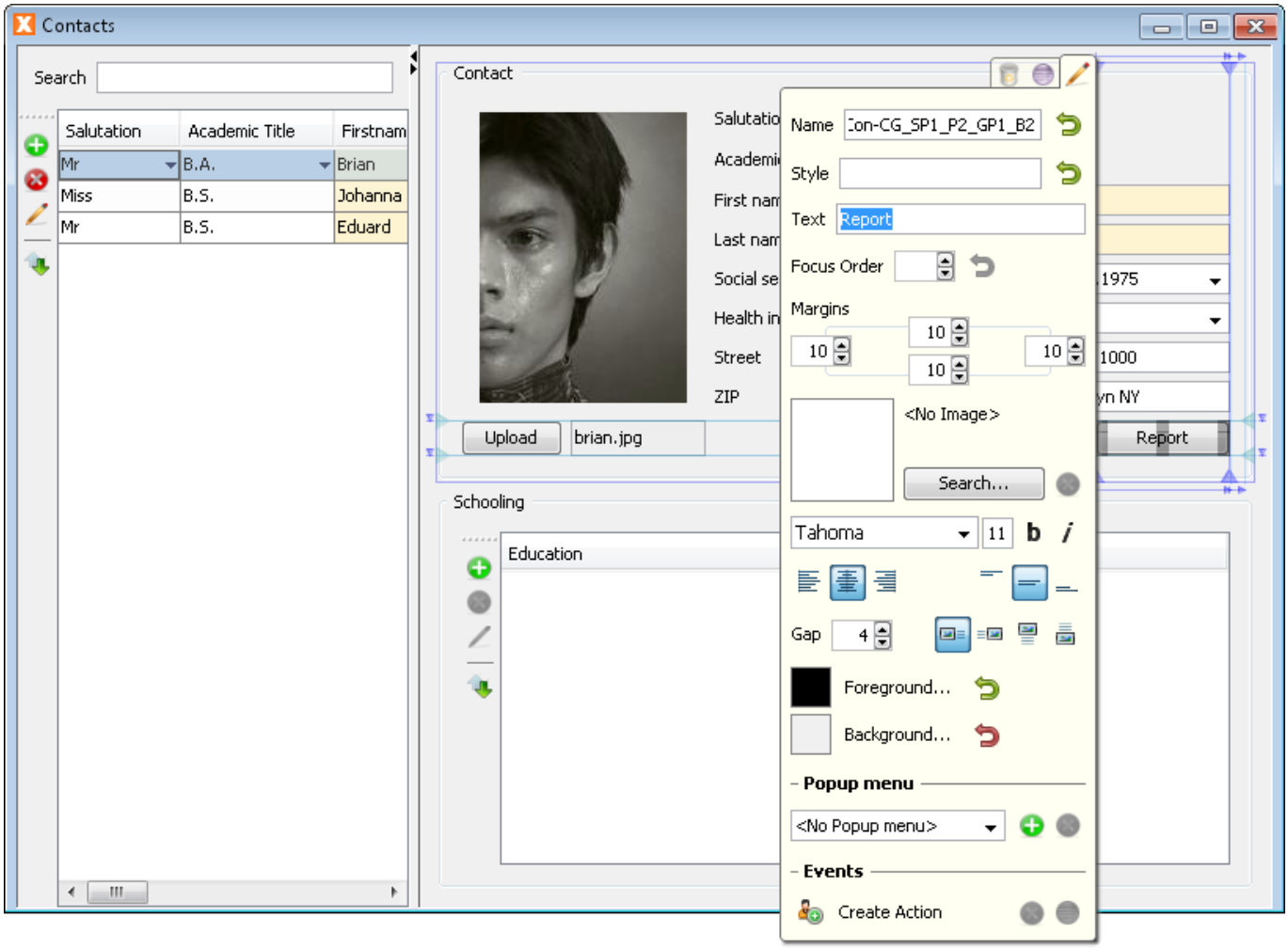
The following steps show how a report is displayed after the click of a button labeled “Report”.

Define Actions

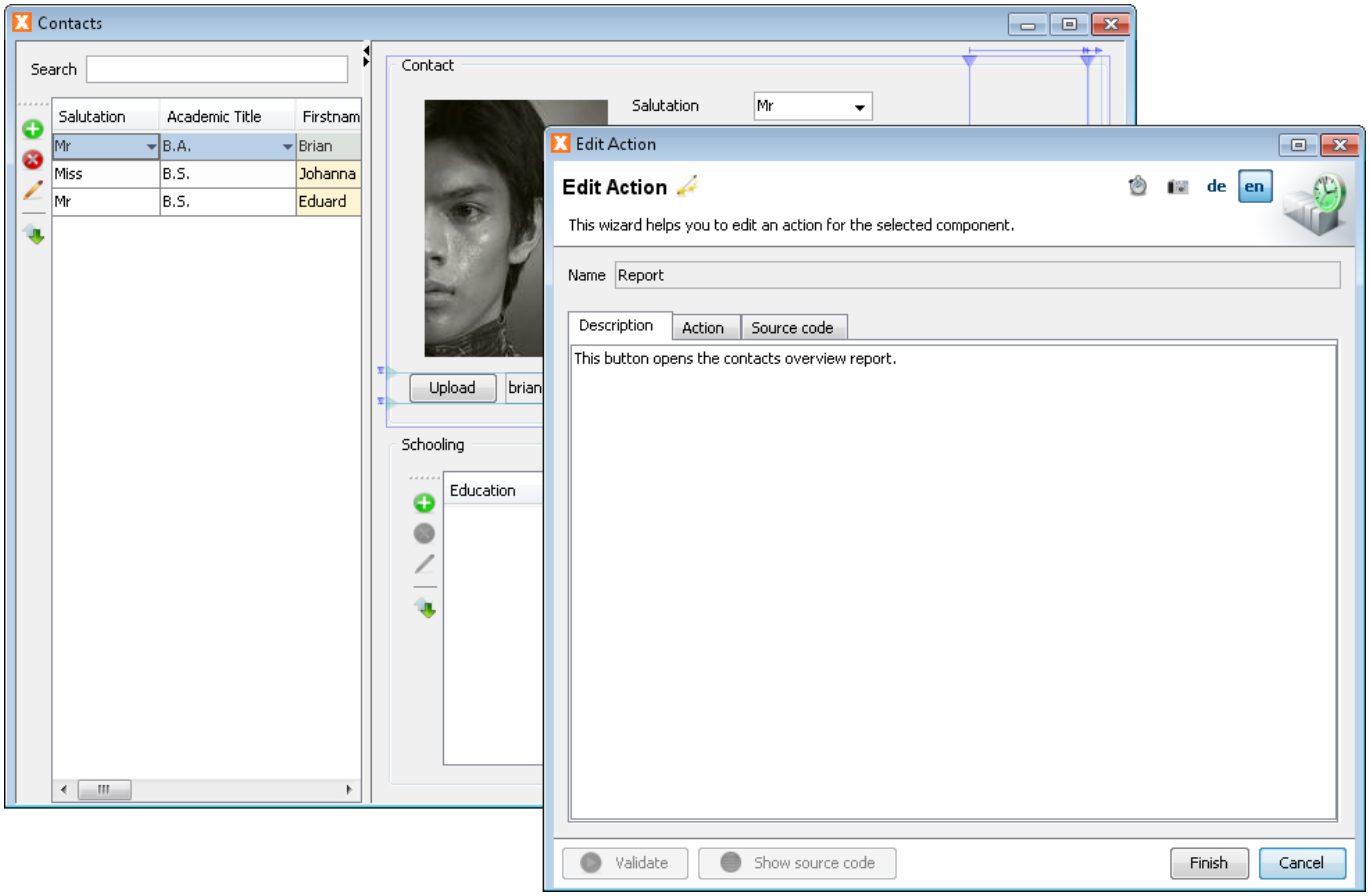
The following steps demonstrate the definition of a simple action.

Example: Displaying a report by clicking a button labeled “Report”.

1. Open a workscreen in the GUI designer - e.g, the “Contacts” workscreen.
2. Select the “Report” button in the GUI designer and click on the pencil icon.

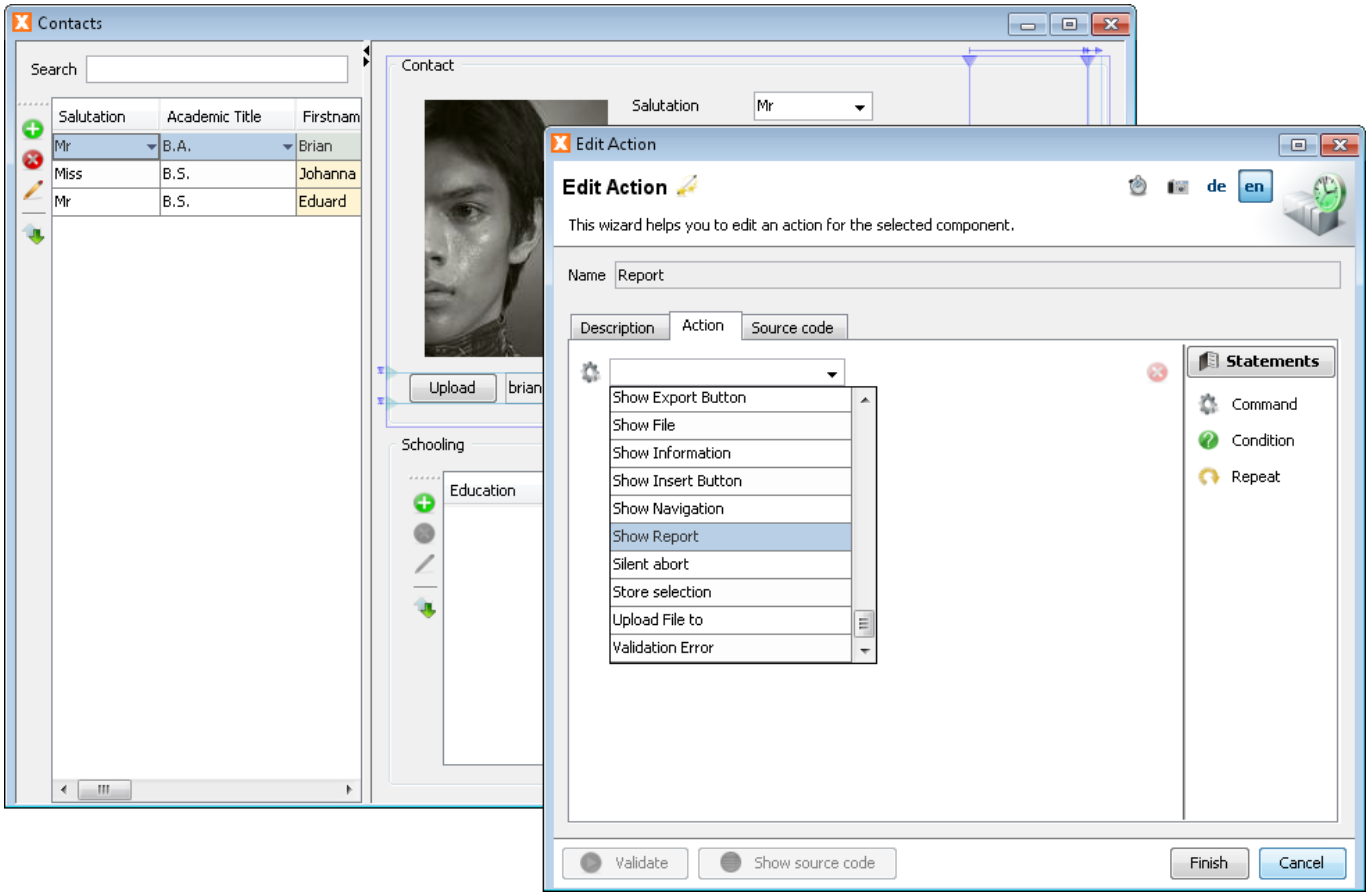


3. Click on the button “Create/Edit Action” to create/edit an action.

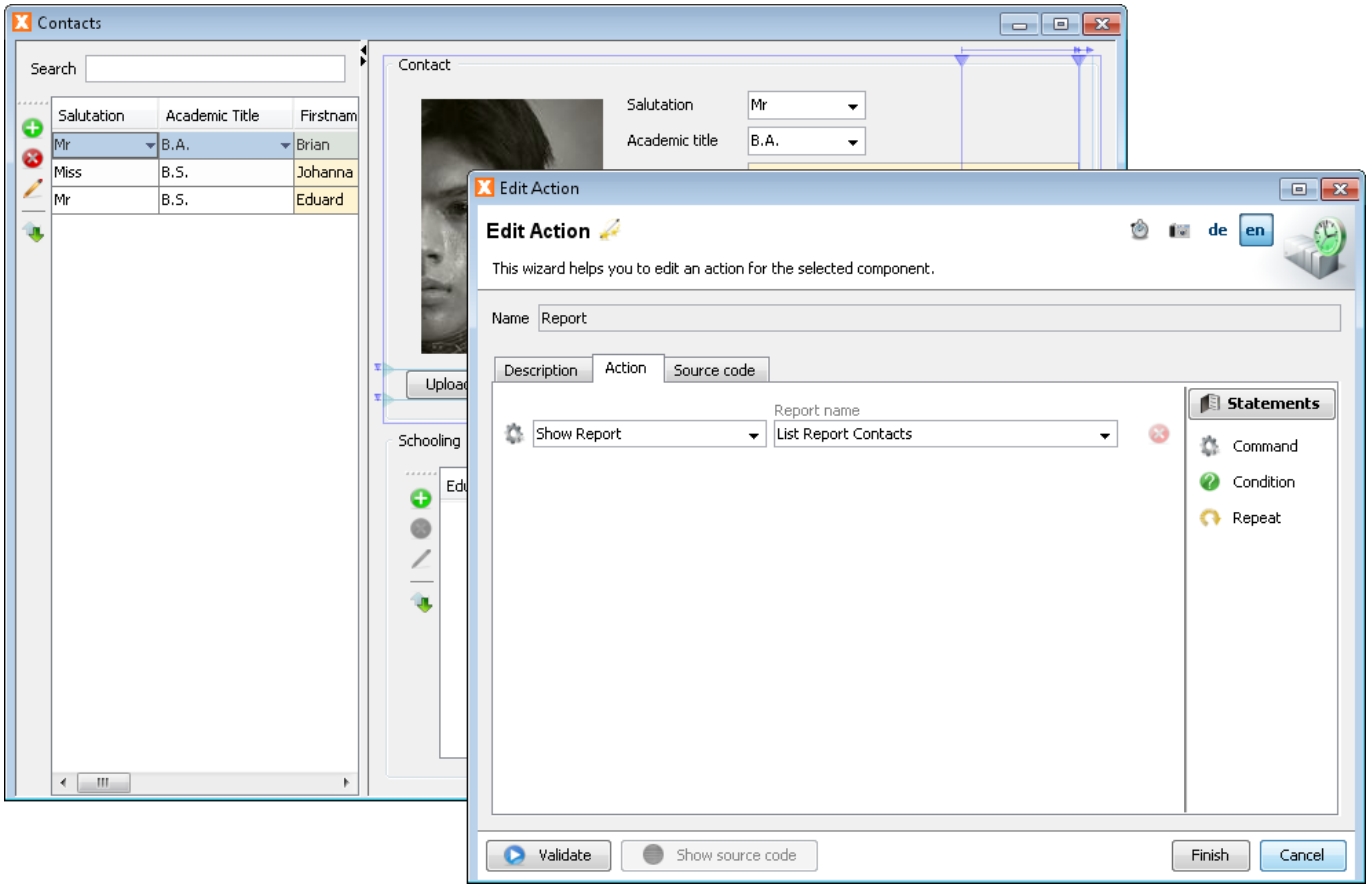


4. Enter a description of the action on the “Description” tab.

5. Click on the “Action” tab and select “Show Report”.



6. Select the desired report as parameter.



7. Click the “Verify” button and then “OK” to verify and save.

This completes the action. It can be tested by closing the GUI designer and changing to runtime mode. Clicking the “Report” button opens the “Contacts” report.

Also, all selection lists can be searched for values using *., e.g., “*show” to search for all commands containing “show”.

Documentation / Software Developer Specification

When an action is created, the first tab of the action wizard can be used to add a description. Our experience shows that, unfortunately, the documentation of applications is often neglected, or, when it is created, not kept up to date. VisionX offers a great advantage through the universal use of a centralized description throughout the entire application. Consistent updating of the documentation requires little effort and should not be neglected. It particularly simplifies and shortens communication efforts when parts of the development tasks are transferred to suppliers.

In VisionX, the description of the action on the first tab of the action wizard is used for the following:

- Description of the action in the specification document

  This text serves as specification of the action for the software developer!

  Specification documents make it easy to outsource certain functions to software providers / IT departments.

- Description of the action in the application's online help

- Description of the action in the function header (Javadoc) of the action-function in the Java source code

  Changes to the description text (Javadoc) in the source code are automatically transferred verbatim to the VisionX action editor.

  This makes it easy to keep the documentation up to date!

Command Categories

There are different types of commands in VisionX:

- Commands

  Commands execute predefined operations. They can include parameters that are considered when the action is performed.

- Conditions

  Conditions allow for the execution of commands depending on whether the condition is met or not.

  Example: If the currently registered user has the “Administrator” role, all fields on the workscreen are visible. If the user does not have this role, only selected fields are shown.

- Loops

  Loops are used to execute commands multiple times for the datasets of a table/view.

  Example: Sending an email to all contacts.

Commands

Commands execute predefined operations considering the specified parameters.

Formulas in Commands

A number of commands can include the use of formulas, e.g. “Calculate Value”.

Formulas allow for the calculation of values using basic arithmetic operations as well as a variety of group functions. For arithmetic operations +,-,*, / and brackets ( ) can be used, e.g. `12 *23 / 2 + (1 - [Items.Quantity] * [Items.Price]) - [Booking.Discount]`.

Group functions calculate results based on all values in a table column. A list is created using each row in the table column; the group function is then applied to this list. For example: `avg([Items.Price])` which returns the average of the prices in the table “Items”.

In addition, all functions are available in the variant to return `null` if all values in the column were `null`. These variants have the same name but are postfixed with “Null”, e.g. `avgNull([Items.Price])`. For additional information, please see the Wikipedia entry on SQL Null.

avg / avgNull

Returns the average of all values.

```
avg([Items.Price])
avgNull([Items.Price])
```

min / minNull

Returns the smallest value.

```
min([Items.Price])
minNull([Items.Price])
```

max / maxNull

Returns the largest value.

```
max([Items.Price])
maxNull([Items.Price])
```

count / countNull

Returns the number of entries.

```
count([Items.Price])
countNull([Items.Price])
```

sum / sumNull

Returns the sum of all values.

```
sum([Items.Price])
sumNull([Items.Price])
```

sumToCurrent / sumToCurrentNull

Returns the sum of values from the first row to the selected row.

```
sumToCurrent([Items.Price])
sumToCurrentNull([Items.Price])
```

first / firstNull

Returns the first value.

```
first([Items.Price])
firstNull([Items.Price])
```

last / lastNull

Returns the last value.

```
first([Items.Price])
firstNull([Items.Price])
```

List of All Commands

The following table shows all commands, descriptions, parameters, examples, and the resulting Java code.

Workscreen Commands

Open Screen

Opens the specified workscreen.

Parameters: `[List of all Workscreens]`

Example: Open Workscreen | User Management

```
getApplication().openWorkScreen("contacts.client.workscreens.ContactsWorkscr
een")
```

Close Screen

Closes the specified workscreen.

Parameters: `[List of all Workscreens]`

Example: Close Workscreen | User Management

```
((ProjX)getApplication()).close("contacts.client.workscreens.ContactsWorkscreen")
```

Center Screen

Centers the current workscreen (horizontally and vertically).

Example: Center Workscreen

```
center()
```

Close This Screen

Closes the current workscreen.

Example: Close Workscreen

```
close()
```

DataBook Commands

Fetch All

Fetches all rows of the specified databook.

Parameters: `[List of all databooks in a workscreen]`

Example: Fetch all rows | [Databook: Contacts]

Hint: This command is used to ensure that all rows of a databook/view are loaded and displayed (e.g., to search for specific entries).

```
rdbContacts.fetchAll()
```

Deselect

No row in the specified databook is selected.

Parameters: [List of all databooks in a workscreen]

Example: Deselect Row | [Databook: Contacts]

```
rdbContacts.setSelectedRow(-1)
```

Select First Row

The first row of the specified databook is selected.

Parameters: [List of all databooks in a workscreen]

Example: Select First Row | [Databook: Contacts]

```
rdbContacts.setSelectedRow(0)
```

Select Last Row

The last row of the specified databook is selected.

Parameters: [List of all databooks in a workscreen]

Example: Select last row | [Databook: Contacts]

```
rdbContacts.setSelectedRow(rdbContacts.getRowCount() - 1)
```

Select Last Inserted Row

Selects the last inserted row in the specified databook.

Parameters: [List of all databooks in a workscreen]

Example: Select last inserted row | [Databook: Contacts]

Hint: This command can be used to transfer data from the last inserted row in a databook to another databook or to use the data in a report. After the execution of this command, the values of the row's various columns can be accessed (e.g., using the “Set Value with Text” command).

```
DataBookUtil.selectLastInsertedRow(rdbContacts)
```

Select Row Having Value

Searches the specified databook for a row and selects that row. The selected table column is used to check if the value in the column is equal to the specified value. The search is applied to all rows of the table; the first row containing an exact match is selected.

Parameters: [List of all columns of all databooks in a workscreen], [Value for

which all rows are searched]

Example: Select row having value | [Contacts.Lastname] | Maria

```
DataBookUtil.selectRowHavingValue(rdbContacts,"Maria")
```

Store Selection

Stores the currently selected row in the specified databook to enable selection of that row at a later time.

Parameters: [List of all databooks in a workscreen]

Example: Store current selection | [Databook: Contacts]

Hint: This command can be used to restore the original selection after a number of operations.

```
DataBookUtil.storeSelection(rdbContacts)
```

Restore Selection

The previously saved selection in the specified databook is restored. If no selection is stored, no change is made.

Parameters: [List of all databooks in a workscreen]

Example: Restores selection | [Databook: Contacts]

Hint: This command can be used to restore the original selection after a number of operations.

```
DataBookUtil.restoreSelection(rdbContacts)
```

Insert Row Before

Inserts a new row in the specified databook before the currently selected row.

Parameters: [List of all databooks in a workscreen]

Example: Insert row before | [Databook: Contacts]

```
rdbContacts.insert(true)
```

Insert Row After

Inserts a new row in the specified databook after the currently selected row.

Parameters: [List of all databooks in a workscreen]

Example: Insert row after | [Databook: Contacts]

```
rdbContacts.insert(false)
```

Ask Before Delete

Asks the user to confirm before a row is deleted in the specified databook.

Parameters: `[List of all databooks in a workscreen]`

Example: Ask before delete | [Databook: Contacts]

```
ProjXUtil.addAskDeleteDialog(rdbContacts)
```

Calculate Value

Calculates the result of the specified formula (see Formulas in commands) and written in the specified table column.

Parameters: `[List of all columns of all databooks in a workscreen], [Formula including reference to columns in all tables of the workscreen]`

Example: Calculate value | [Contacts.Total] | [Contacts.Quantity] * [Contacts.Price]

Hint: This command can be used for various mathematical operations. The result is then written to a table column. For example, calculation of the total price based on quantity and unit price. See Formulas in commands for additional details regarding the use of formulas in commands.

```
Calc.set(new Var(rdbContacts,"TOTAL"),
    Calc.val(new Var(rdbContacts,"QUANTITY")) *
    Calc.val(new Var(rdbContacts,"PRICE")))
```

Set Value With Text

Inserts the specified text in the specified databook column.

Parameters: `[Target: List of all columns of all databooks in a workscreen], [Text including reference to columns in all tables of the workscreen]`

Example: Set value with text| [Contacts.Name] | [Contacts.Filename], [Contacts.Lastname]

Hint: This command can be used to populate text columns with fixed text or values from other columns. In the example below, the value from “Lastname”, the fixed text “, ” (comma + space) and the value from “Firstname” is written to the column “Name”: “Doe, John”.

```
Text.set(new Var(rdbContacts,"NAME"),
    Text.val(new Var(rdbContacts,"LASTNAME")) +
    ", " +
```

```
    Text.val(new Var(rdbContacts,"FIRSTNAME")))
```

Set Value

The value in a specified table column (source) is written to another table column (target).

Parameters: `[Target: List of all columns of all tables in the workscreen], [Source: List of all columns of all tables in the workscreen]`

Example: Set Value | [Contacts.Filename] | [Details.Filename]

Hint: This command can be used to copy values (text, date, numerical) from one table column to another. For example, [Contacts.Filename] = [Details.Filename]. As a result, the file name of the details table is copied to the file name of the contacts table.

```
DataBookUtil.set(new Var(rdbContacts,"FILENAME"),
    rdbDetails.getValue("FILENAME"))
```

Set Current Date

The current time and date are written to the specified table column.

Parameters: `[List of all columns of all tables in the workscreen]`

Example: Set Current Date | [Contacts.Birthday]

Hint: This command can be used to prefill all date columns with the current date. Another application is the creation of a log that shows when a record was added or changed. It is good practice to add changed_on, created_on table columns for all relevant tables!

```
Text.set(new Var(rdbContacts,"BIRTHDAY"),
    new Date())
```

Set Current User

The name of the current user is inserted in the specified databook column.

Parameters: `[List of all columns of all databooks in a workscreen]`

Example: Set current user | [Contacts.Username]

Hint: One application of this command is the creation of a log showing which user created or edited a dataset. It is good practice to add changed_by, created_by columns for all relevant databooks!

```
Text.set(new Var(rdbContacts,"LASTNAME"),
    getConnection().getUserName())
```

Set Filter Value

The specified filter is set with the specified value. All filter editors used in the workscreen can be used as filters. They can be placed on the workscreen via the GUI designer's data area. A full-text filter or a filter for a selected table column can then be defined. “Similar”, “equal”, “equal or smaller”, “equal or greater”, and “greater” can be used as filters. After the value is set, the filter is applied to all rows of the associated table, and only the resulting rows are visible.

Parameters: `[List of all filter editors of the workscreen], [Value for which all rows are searched]`

Example: Set filter value | [Search: Contacts.*] | Johanna

All datasets that contain“*Johanna*” are shown in the contacts table.

Hint: This command can be used to restrict access to information according to user roles. Role restriction at the data level!

- Administrator is able to see everything, no filter
- Staff member can only see their own datasets, filtered by username
- Department manager can only see own department's datasets, filtered by department

Another application of this command is the search for a specified dataset based on a master data list. For example, search for status “complete”, after which the “ID” column can be transferred from the master data table to the main table (e.g., to preset the status as default value).

```
edtSearch.setValue("Johanna")
```

Set Filter Current User

The current username is set as the filter value for the specified filter.

Parameters: `[List of all filter editors of the workscreen]`

Example: Set Filter Current User | [Search: Contacts.Username]

Hint: This command can be used to restrict access to information according to user roles. Role restriction at the data level!

- Administrator is able to see everything, no filter
- Staff member can only see their own datasets, filtered by username

```
edtSearch.setValue(getConnection().getUserName())
```

Set Column Readonly

The specified databook column is set to read-only.

Parameters: `[List of all columns of all databooks in a workscreen]`

Example: Set column readonly | [Contacts.Username]

Hint: This command sets both the column in the table as well as the associated editor on the form to read-only!

```
DataBookUtil.setColumnReadOnly(new Var(rdbContacts,"USERNAME"))
```

Set Column Editable

The specified databook column is set to read-write access.

Parameters: `[List of all columns of all databooks in a workscreen]`

Example: Set column editable | [Contacts.Username]

Hint: This command sets both the column in the table as well as the associated editor on the form to read-write!

```
DataBookUtil.setColumnEditable(new Var(rdbContacts,"USERNAME"))
```

Enable Insert in Table

Enables the insertion of rows in the specified databook.

Parameters: `[List of all databooks in a workscreen]`

Example: Enable insert in table | [Databook: Contacts]

```
rdbContacts.setInsertEnabled(true)
```

Disable Insert in Table

Disables the insertion of rows in the specified databook.

Parameters: `[List of all databooks in a workscreen]`

Example: Disable insert in table | [Databook: Contacts]

```
rdbContacts.setInsertEnabled(false)
```

Enable Edit in Table

Enables editing of data in the specified databook.

Parameters: `[List of all databooks in a workscreen]`

Example: Enable edit in table | [Databook: Contacts]

```
rdbContacts.setUpdateEnable(true)
```

Disable Edit in Table

Disables editing of data in the specified databook.

Parameters: [List of all databooks in a workscreen]

Example: Disable edit in table | [Databook: Contacts]

```
rdbContacts.setUpdateEnabled(false)
```

Enable Delete in Table

Enables deletion of rows in the specified table.

Parameters: [List of all databooks in a workscreen]

Example: Enable delete in table | [Databook: Contacts]

```
rdbContacts.setDeleteEnabled(true)
```

Disable Delete in Table

Disables deletion of rows in the specified table.

Parameters: [List of all databooks in a workscreen]

Example: Disable delete in table | [Databook: Contacts]

```
rdbContacts.setDeleteEnabled(false)
```

Enable Data Manipulation in Table

Enables all data manipulation (insert, edit, delete) in the specified databook.

Parameters: [List of all databooks in a workscreen]

Example: Enable data manipulation in table | [Databook: Contacts]

```
rdbContacts.setReadOnly(false)
```

Disable Data Manipulation in Table

Disables all data manipulation (insert, edit, delete) in the specified databook.

Parameters: [List of all databooks in a workscreen]

Example: Disable data manipulation in table | [Databook: Contacts]

```
rdbContacts.setReadOnly(true)
```

Delete Row

Deletes the currently selected row in the specified databook.

Parameters: [List of all databooks in a workscreen]

Example: Delete row | [DataBook: Contacts]

```
rdbContacts.delete();
```

Table View (Navigation Table) Commands

Show Insert Button

The “Insert” button on the toolbar of the specified table view is shown.

Parameters: [List of all table views in the workscreen]

Example: Show insert button| [Table view: Contacts]

```
navContacts.setInsertVisible(true)
```

Hide Insert Button

The “Insert” button on the toolbar of the specified table view is hidden.

Parameters: [List of all table views in the workscreen]

Example: Hide insert button | [Table view: Contacts]

```
navContacts.setInsertVisible(false)
```

Show Edit Button

The “Edit” button on the toolbar of the specified table view is shown.

Parameters: [List of all table views in the workscreen]

Example: Show edit button | [Table view: Contacts]

```
navContacts.setEditVisible(true)
```

Hide Edit Button

The “Edit” button on the toolbar of the specified table view is hidden.

Parameters: `[List of all table views in the workscreen]`

Example: Hide edit button | [Table view: Contacts]

```
navContacts.setEditVisible(false)
```

Show Delete Button

The “Delete” button on the toolbar of the specified table view is shown.

Parameters: `[List of all table views in the workscreen]`

Example: Show delete button | [Table view: Contacts]

```
navContacts.setDeleteVisible(true)
```

Hide Delete Button

The “Delete” button on the toolbar of the specified table view is hidden.

Parameters: `[List of all table views in the workscreen]`

Example: Hide delete button | [Table view: Contacts]

```
navContacts.setDeleteVisible(false)
```

Show Export Button

The “Export” button on the toolbar of the specified table view is shown.

Parameters: `[List of all table views in the workscreen]`

Example: Show export button | [Table view: Contacts]

```
navContacts.setExportVisible(true)

#### Hide Export Button
```

The “Export” button on the toolbar of the specified table view is hidden.

Parameters: [List of all table views in the workscreen]

Example: Hide export button | [Table view: Contacts]

```
navContacts.setExportVisible(false)
```

Show Navigation

The navigation toolbar is shown for the specified table view.

Parameters: [List of all table views in the workscreen]

Example: Show navigation | [Table view: Contacts]

```
navContacts.setToolBarVisible(true)
```

Hide Navigation

The navigation toolbar is hidden for the specified table view.

Parameters: [List of all table views in the workscreen]

Example: Hide navigation | [Table view: Contacts]

```
navContacts.setToolBarVisible(false)
```

Hide Column

The specified table column is hidden.

Parameters: [List of all columns of all tables in a workscreen]

Example: Hide column | [Contacts.Username]

Hint: This command is used to hide table columns irrespective of user roles.

```
DataBookUtil.hideColumn(new Var(rdbContacts,"USERNAME"));
```

Autoresize Columns

Automatically sets column widths for the selected table view. Widths are determined dynamically based on the database column width and the actual values in the columns.

Parameters: [List of all table views in the workscreen]

Example: Autoresize columns | [Table view: Contacts]

```
navContacts.setAutoResize(true)
```

Change Management Commands

Discard All Changes

All unsaved changes in all databooks of the current workscreen are discarded.

Example: Discard all changes

Hint: This command is used to discard charges on a form using a “Cancel” button.

```
getDataSource().restoreAllDataBooks()
```

Save

Saves all unsaved changes in all databooks of the current workscreen.

Example: Save

Hint: This command can be used before a report is shown or to save changes in a form using a “Save” button.

```
getDataSource().saveAllDataBooks()
```

Save Table

Saves all unsaved changes in the specified databook.

Example: Save table

Hint: This command can be used to save changes in a form using a “Save” button.

```
rdbContacts.saveAllRows()
```

Reload Table

Reloads all datasets in the specified databook from the database.

Example: Reload table

Hint: This command is used to keep changes made by other users or in other workscreens to the same databook current.

```
rdbContacts.reload()
```

Reload All

Reloads all databooks in the current workscreen from the database.

Example: Reload all

```
getDataSource().reloadAllDataBooks()
```

Reload All Screens

Reloads all databooks in all open workscreens from the database.

Example: Reload all screens

```
((ProjX)getApplication()).doReload()
```

Manual Save and Reload

The current workscreen is decoupled from the application toolbar. Clicking “Save” or “Reload” on the toolbar is ignored within the current workscreen. This is a useful feature for typical-form workscreens; it is a default setting for the generation of such workscreens in VisionX. It is, therefore, necessary to create separate “Save” and “Cancel” buttons for these workscreens with the respective functionality.

Example: Manual save and reload

```
setManualSaveAndReload(true)
```

Report Commands

Show Report

Shows the specified report. The report is either displayed in a browser or opened using the relevant program (e.g. Word, Excel), depending on how the PC is configured.

Parameters: `[List of all reports in the workscreen]`

Example: Show Report | Form Report Contacts

```
getApplication().getLauncher().showFileHandle(getFormReportContacts())
```

Download Report

Downloads the specified report. The report can be saved on a local hard drive and then opened.

Parameters: `[List of all reports in the workscreen]`

Example: Download report | Form Report Contacts

```
getApplication().getLauncher().saveFileHandle(getFormReportContacts())
```

Save Report

Saves the specified report in a table. The save location is defined via two parameters: The report's file name is saved in a text column of the table; the file itself is saved in the current row of a table column of the type Image or File.

Parameters: `[Filename: List of all columns of all databooks in a workscreen], [File: List of all columns of all databooks in a workscreen] | [List of all reports in the workscreen]`

Example: Save Report | [Contacts.Filename] | [Contacts.Image] | Form Report Contacts

Hint: This command can be used to save the reports generated by the application in a databook (e.g., lists, proposals and other business reports).

```
ProjXUtil.storeFileHandle(
    new Var(rdbContacts,"FILENAME"),
    new Var(rdbContacts,"IMAGE"),
    getFormReportContacts())
```

Import Report (Merge)

Imports the specified report. All of the report's data is imported to the relevant rows and fields of the workscreen. If “Merge” is selected, only changed, deleted, or new information is imported. Data that already exists is not added separately. If “Insert” is used, all data rows, including existing rows (tested for similarity) are transferred. This functionality is only available for XLSX (Excel) and XML reports that were created using VisionX. Of course, the report templates can be edited. This is command is used to import offline forms into the application.

Parameters: `[List of all reports in the workscreen]`

Example: Show report | Form Report2 Contacts (xls)

```
ProjXUtil.importFile(this,
"/reports/screens/ContactsWorkScreen$ContactsForm.xls", true)
```

Import Report (Insert)

See Import Report (merge) above

```
ProjXUtil.importFile(this,
"/reports/screens/ContactsWorkScreen$ContactsForm.xls", false)
```

File Upload/Download Commands

Upload File to

Uploads a file that is selected by the user and saves it in a databook. The save location is determined by two parameters: the name of the uploaded file is saved in a table's text column; the file itself is saved in the current row of a table column of the type Image or File. Parameters: `[Filename: List of all columns of all databooks in a workscreen]`, `[File: List of all columns of all databooks in a workscreen]`.

Example: Upload file | [Contacts.Filename] | [Contacts.Image]

Hint: This command can be used to saved files in a databook (e.g., emails, proposals and other business reports).

```
ProjXUtil.uploadFile(this,
    new Var(rdbContacts,"FILENAME"),
    new Var(rdbContacts,"IMAGE"))
```

Show File

Shows the specified file. The file is either displayed in a browser or opened using the relevant program (e.g. Word, Excel), depending on how the PC is configured. The file is selected via two parameters: the file name and a databook column. The file name can be selected via a table's text column or using fixed text. The data is taken from a column in the currently selected table row (type Image or File).

Parameters: `[Filename: List of all columns of all databooks in a workscreen]`, `[File: List of all columns of all databooks in a workscreen]`

Example: Show file | [Contacts.Filename] | [Contacts.Image]

Hint: This command is used to display documents that were saved in the application by the user.

```
ProjXUtil.showFile(this,
    (String)rdbContacts.getValue("FILENAME"),
    rdbContacts.getValue("IMAGE"))
```

Download File

Downloads the specified file. The file can be saved to a local hard drive and opened from there. The file is selected via two parameters: the file name and a databook column. The file name can be selected via a table's text column or using fixed text. The data is taken from a column in the currently selected table row (type Image or File).

Parameters: `[Filename: List of all columns of all databooks in a workscreen]`, `[File: List of all columns of all databooks in a workscreen]`

Example: Download file | [Contacts.Filename] | [Contacts.Image]

Hint: This command is used to download files that were saved in the application by the user.

```
ProjXUtil.downloadFile(this,
    (String)rdbContacts.getValue("FILENAME"),
    rdbContacts.getValue("IMAGE"))
```

Delete File

Deletes the specified file.

Parameters: `[Filename: List of all columns of all databooks in a workscreen], [File: List of all columns of all databooks in a workscreen]`

Example: Delete File | [Contacts.Filename] | [Contacts.Image]

```
ProjXUtil.deleteFile(*,*)
```

GUI Component Commands

Disable Component

Disables the specified GUI component. The element is grayed out and no changes can be made.

Parameters: `[List of all GUI components on the workscreen]`

Example: Disable component | [Input field: Contacts.Lastname]

```
edtLastName.setEnabled(false)
```

Enable Component

Enables the specified GUI component.

Parameters: `[List of all GUI components on the workscreen]`

Example: Enable component | [Input field: Contacts.Lastname]

```
edtLastName.setEnabled(true)
```

Hide Component

Hides the specified GUI component.

Parameters: `[List of all GUI components on the workscreen]`

Example: Hide component | [Input field: Contacts.Lastname]

```
edtLastName.setVisible(false)
```

Show Component

Shows the specified GUI component.

Parameters: [List of all GUI elements on the workscreen]

Example: Show component | [Input field: Contacts.Lastname]

```
edtLastName.setVisible(true)
```

Deselect Button

Deselects the specified button (checkbox, radio-, or toggle button).

Parameters: [List of all buttons (checkbox, radio- and toggle) on the workscreen]

Example: Deselect button | [Input field: Contacts.Lastname]

```
edtLastName.setSelected(false)
```

Select Button

Selects the specified button (checkbox, radio-, or toggle button).

Parameters: [List of all buttons (checkbox, radio- and toggle) on the workscreen]

Example: Select button | [Input field: Contacts.Lastname]

```
edtLastName.setSelected(true)
```

Enable Tab

Activates the specified tab so that it is not grayed out and can be selected.

Parameters: [Component: List of all of the workscreen's tab controls]

Parameters: [Tab: List of all tabs of the selected tab control]

Example: Enable tab | [TabsetPanel: tabsetPanelMain] | Tab 2

```
tabsetPanelMain.setEnabledAtIfExists("Tab 2",true)
```

Disable Tab

Deactivates the specified tab. It is grayed out and cannot be selected.

Parameters: [Component: List of all of the workscreen's tab controls]

Parameters: [Tab: List of all tabs of the selected tab control]

Example: Disable tab | [TabsetPanel: tabsetPanelMain] | Tab 2

```
tabsetPanelMain.setEnabledAtIfExists("Tab 2",false)
```

Select Tab

Selects the specified tab.

Parameters: [Component: List of all of the workscreen's tab controls]

Parameters: [Tab: List of all tabs of the selected tab control]

Example: Select tab | [TabsetPanel: tabsetPanelMain] | Tab 2

```
tabsetPanelMain.setSelectedIndexIfExists("Tab 2")
```

Set Tab Text

Sets the text of the specified tab.

Parameters: [Component: List of all of the workscreen's tab controls]

Parameters: [Tab: List of all tabs of the selected tab control]

Example: Set tab text | [TabsetPanel: tabsetPanelMain] | Tab 2 | Tab 2 New

```
tabsetPanelMain. setTextAtIfExists("Tab 2","Tab 2 New")
```

Set Label With Calculated Value

Calculates the result of the specified formula (see Formulas in commands) and writes it to the specified label (GUI element).

Parameters: [List of all labels (GUI elements) of the workscreen], [Formula including reference to columns in all tables of the workscreen]

Example: Set label with calculated value | [Contacts.Total] | [Contacts.Quantity] * [Contacts.Price]

Hint: This command can be used for various mathematical operations. The result is then written to a table column. For example, calculation of the total price based on quantity and unit price.

Additional details regarding the use of formulas in commands can be found in Formulas in commands.

```
Calc.setText(*,*,null) Calc.set(new Var(rdbContacts,"TOTAL"),
    Calc.val(new Var(rdbContacts,"QUANTITY")) *
    Calc.val(new Var(rdbContacts,"PRICE")), null)
```

Set Label Text

The specified text is written to the specified label (GUI element).

Parameters: `[Target: List of all labels (GUI-elements) of the workscreen], [Text including reference to columns in all tables of the workscreen]`

Example: Set Label Text| [Contacts.Name] | [Contacts.Filename], [Contacts.Lastname]

Hint: This command can be used to populate text columns with fixed text or values from other columns.

In the example on the right, the value from “Lastname”, the fixed text “, ” (comma + space) and the value from “Firstname” is written to the column “Name”: “Doe, John”.

```
lblLastName.setText()
Text.set(new Var(rdbContacts,"NAME"),
    Text.val(new Var(rdbContacts,"LASTNAME")) +
    ", " +
    Text.val(new Var(rdbContacts,"FIRSTNAME")))
```

Disable All Components

Disables all components in the given container.

Parameters: `[Target: List of all GUI components of the workscreen]`

Example: Disable all components | [Contacts.Panel]

```
ProjXUtil.setComponentsEnabled(pnlMain, false)
```

Enable All Components

Enables all components in the given container.

Parameters: `[Target: List of all GUI components of the workscreen]`

Example: Enable all components | [Contacts.Panel]

```
ProjXUtil.setComponentsEnabled(pnlMain, true)
```

Additional Commands

Logout

The user is logged out.

Example: Logout

```
((ProjX)getApplication()).doLogout(null)
```

Show Error

Shows a predefined message in an error dialogue.

Parameters: `[Error message with reference to all databook columns in the workscreen]`

Example: Show Error | [Input field: Contacts.Lastname] is missing!

```
showError(this,
    Text.val(new Var(rdbContacts,"LASTNAME")) +
    " is missing!")
```

Show Information

Shows a predefined message in an information dialogue.

Parameters: `[Information message with reference to all databook columns in the workscreen]`

Example: Show information | [Input field: Contacts.Lastname] is missing!

```
showInformation(this,
    Text.val(new Var(rdbContacts,"LASTNAME")) +
    " is missing!")
```

Validation Error

Shows a validation error. The specified text is shown on the workscreen in a validation error message.

Parameters: `[Validation error message with reference to all databook columns in the workscreen]`

Example: Validation error | [Input field: Contacts.Street] is a required field.

Hint: Validation errors are always used for the GUI element “field validation.” If email is empty, then validation error “...”!

```
throw new Exception(
    Text.val(new Var(rdbContacts,"STREET")) +
    " is a required field.")
```

Send Mail

Sends an email message. The parameters can reference databook columns or can be defined using fixed text.

Parameters: [STMP Server], [STMP Port], [SMTP User], [SMTP Password], [Sender], [Recipient], [Subject], [Mail Text], [Filename], [File]

Hint: It is good practice to create one or more tables containing the various parameters, to allow for easier configuration.

```
getConnection().callAction("sendMail",
    (String)rdbEmailserver.getValue("SMTPSERVER"),
    (String)rdbEmailserver.getValue("SMTPPORT"),
    (String)rdbEmailserver.getValue("USERNAME"),
    (String)rdbEmailserver.getValue("PASSWORD"),
    (String)rdbEmailserver.getValue("SENDER"),
    (String)rdbUsers.getValue("RECIPIENT"),
    "E-Mail subject",
    "Test\nhttp://sibvisions.com",
    "",
    "")
```

Open URL

Opens an URL. The parameter can reference databook columns or can be defined using fixed text.

Parameters: [URL]

```
getApplication().getLauncher().showDocument(
    (String)rdbUrls.getValue("URL"),
    null,
    "blank")
```

Call Server Action

Executes the given server action.

Parameters: [Action Name]

```
getConnection().callAction("action")
```

Silent Abort

Aborts the current operation silently and completely.

```
throw new SilentAbortException()
```

QR

QR Text

Converts the given text into a QR code and saves it in the given column.

Parameters: [List of all columns of all databooks in a workscreen], [List of all columns of all databooks in a workscreen]

Exmaple: QR Text | [Contacts.Qr] | [Contacts.Firstname]

```
QRUtil.setText(
    getConnection(),
    (String)rdbContacts.getValue("QR"),
    (String)rdbContacts.getValue("FIRSTNAME"))
```

QR Email

Converts the given email address into a QR code and saves it in the given column.

Parameters: [List of all columns of all databooks in a workscreen], [List of all columns of all databooks in a workscreen]

Exmaple: QR Text | [Contacts.Qr] | [Contacts.Email]

```
QRUtil.setEMail(
    getConnection(),
    (String)rdbContacts.getValue("QR"),
    (String)rdbContacts.getValue("EMAIL"))
```

QR Phone Number

Converts the given phone number into a QR code and saves it in the given column.

Parameters: [List of all columns of all databooks in a workscreen], [List of

all columns of all databooks in a workscreen]

Example: QR Text | [Contacts.Qr] | [Contacts.Phone]

```
QRUtil.setPhoneNumber(
    getConnection(),
    (String)rdbContacts.getValue("QR"),
    (String)rdbContacts.getValue("PHONE"))
```

QR Geo Location

Converts the given location into a QR code and saves it in the given column.

Parameters: [List of all columns of all databooks in a workscreen], [List of all columns of all databooks in a workscreen], [List of all columns of all databooks in a workscreen]

Exmaple: QR Text | [Contacts.Qr] | [Contacts.Latitude] | [Contacts.Longitude]

```
QRUtil.setText(
    getConnection(),
    (String)rdbContacts.getValue("QR"),
    (String)rdbContacts.getValue("LATITUDE"),
    (String)rdbContacts.getValue("LONGITUDE"),
    null)
```

QR Contact

Converts the given contact information into a QR code and saves it in the given column.

Parameters: [List of all columns of all databooks in a workscreen], [List of all columns of all databooks in a workscreen], [List of all columns of all databooks in a workscreen], [List of all columns of all databooks in a workscreen], [List of all columns of all databooks in a workscreen], [List of all columns of all databooks in a workscreen], [List of all columns of all databooks in a workscreen], [List of all columns of all databooks in a workscreen], [List of all columns of all databooks in a workscreen]

Exmaple: QR Text | [Contacts.Qr] | [Contacts.Name] | [Contacts.Company] | [Contacts.Title] | [Contacts.Phone] | [Contacts.Email] | [Contacts.Address] | [Contacts.Website] | [Contacts.Note]

```
QRUtil.setPhoneNumber(
    getConnection(),
    (String)rdbContacts.getValue("QR"),
    (String)rdbContacts.getValue("NAME"),
    (String)rdbContacts.getValue("COMPANY"),
    (String)rdbContacts.getValue("TITLE"),
    (String)rdbContacts.getValue("PHONE"),
    (String)rdbContacts.getValue("EMAIL"),
```

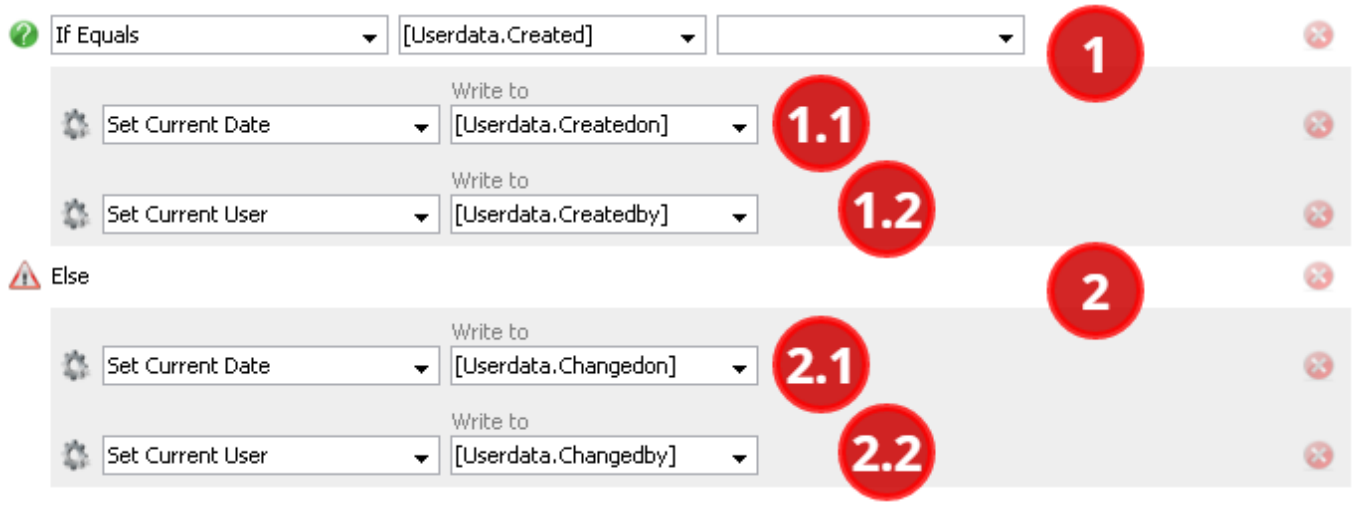
```
    (String)rdbContacts.getValue("ADDRESS"),
    (String)rdbContacts.getValue("WEBSITE"),
    (String)rdbContacts.getValue("NOTE"))
```

Conditions

Conditions allow for the execution of commands in dependence on whether the condition is met or not. One set of commands can be executed if the condition is met and another if the condition is not met.

This results in varying outcomes depending on the condition, e.g., if the user has the role “Administrator”, all databook columns are visible, otherwise only a selected number of columns is shown.

Example:



1. If the column “Userdata.Created” is empty, execute the following commands
    1. Write the current date to “Userdata.Createdon”
    2. Write the current user to “Userdata.Createdby”
2. Otherwise, execute the following commands
    1. Write the current user to “Userdata.Changedby”
    2. Write the current date to “Userdata.Changedon”

If Equals / If Not Equals / If Greater / If Greater or Equal / If Smaller / If Smaller or Equal

If the first parameter equals / does not equal / is greater than / is greater than or equals / is smaller than / is smaller than or equals the second parameter, then the commands in the “If” path are executed, otherwise the commands in the “Else” path are executed.

Parameters: `[List of all columns of all databooks in the workscreen, or a set value], [List of all columns of all databooks in the workscreen, or a set`

value]

```
if (Logical.equals(rdbUserdata.getValue("CREATEDON"),""))
{
    // "If" - Path
}
else
{
    // "Else" - Path
}
```

If Between / If Not Between

If the first parameter is greater than or equal to the second parameter and the first parameter is also smaller than or equal to the third parameter, then the commands in the “If” path are executed, otherwise the commands in the “Else” path are executed. “If not between” results in the opposite.

Parameters: `[List of all columns of all tables of the workscreen or a constant value], [List of all columns of all tables of the workscreen or a constant value]`

Hint: This command is often used to check a date field against a specified date range, or to check a number field against a specified range of values.

```
if (Logical.between(rdbUserdata.getValue("PRICE"),100, 1000))
{
    // "If" - Path
}
else
{
    // "Else" - Path
}
```

If a Row is Selected

If a row is selected in the specified table, then the commands in the “If” path are executed. Otherwise, the commands in the “Else” path are executed.

Parameters: `[List of all tables of the workscreen]`

```
if (Logical.equals(rdbUserdata.getValue("CREATEDON"),"")) {
    // "If" - Path
}
else
{
    // "Else" - Path
}
```

Has Role / Not Has Role

If the current user has the specified role, then the commands in the “If” path are executed. Otherwise, the commands in the “Else” path are executed.

Parameters: `[List of all roles in the application]`

Hint: This command can be used to restrict access to information according to user roles. Role restriction at the data level!

- Administrator is able to see everything, no filter
- Staff member can only see their own datasets, filtered by username

```
if (getApplication().hasRole("Administrator"))
{
    // "If" - Path
}
else
{
    // "Else" - Path
}
```

Is Enabled

If the specified GUI element is active (see command “Enable Component”), then the commands in the “If” path are executed, otherwise the commands in the “Else” path are executed.

Parameters: `[List of all GUI elements of the workscreen]`

```
if (edtLastName.isEnabled())
{
    // "If" - Path
}
else
{
    // "Else" - Path
}
```

Is Visible

If the specified GUI element is visible (see command “Show Component”), then the commands in the “If” path are executed. Otherwise, the commands in the “Else” path are executed.

Parameters: `[List of all GUI elements of the workscreen]`

```
if (edtLastName.isVisible())
{
```

```
    // "If" - Path
}
else
{
    // "Else" - Path
}
```

Is Button Selected

If the specified button is selected (checked) (see command “show element”), then the commands in the “If” path are executed. Otherwise, the commands in the “Else” path are executed.

Parameters: `[List of all GUI elements of the workscreen]`

```
if (edtLastName.isSelected())
{
    // "If" - Path
}
else
{
    // "Else" - Path
}
```

Is Tab Selected

If the specified tab is selected (see command “Select Tab ”), then the commands in the “If” path are executed. Otherwise, the commands in the “Else” path are executed.

Parameters: `[Element: List of all tab controls of the workscreen]`

Parameters: `[Tab: List of all tabs of the previously selected tab control]`

```
if (tabsetPanelMain.getSelectedIndex() == "Tab 2")
{
    // "If" - Path
}
else
{
    // "Else" - Path
}
```

Is Column Changed

If the columns is changed (value changed), then the commands in the “If” path are executed. Otherwise, the commands in the “Else” path are executed.

Parameters: `[List of all table columns in a workscreen]`

```
if (DataBookUtil.isChangedColumnName(pEvent,"LASTNAME"))
{
    // "If" - Path
}
else
{
    // "Else" - Path
}
```

Are Validations Ok

If all field validations on the same panel (GUI Element) of the validation result element are correct, then the commands in the “If” path are executed. Otherwise, the commands in the “Else” path are executed.

Parameters: `[List of all validation results]`

Hint: This condition is often used to perform and show all field validations. If validation errors (see command “Validation Error”) occur during the field validations, they are collectively shown in the specified validation result element.

What does “on the same panel” mean?

“Panel” means a GUI element that is merely a drawing area containing fields, labels, etc. VisionX performs all field validations that are on the same panel as the validation result.

```
if (validationResult.isValid())
{
    // "If" - Path
}
else
{
    // "Else" - Path
}
```

Is Invalid Email

If the first parameter is an invalid email address (W3C consortium definition), then the commands in the “If” path are executed. Otherwise, the commands in the “Else” path are executed.

Parameters: `[List of all columns of all tables in the workscreen or a fixed value]`

```
if (Logical.equals(rdbUserdata.getValue("CREATEDON"),""))
{
    // "If" - Path
```

```
}
else
{
    // "Else" - Path
}
```

Is Showing

This condition is true if the workscreen is showing and the operations “open” and “load” have been completed. The condition can be used for a events such as “After Row Selected” to determine if the event occurred during initialization or if it was triggered by the user.

```
if (isShowing())
{
    // "If" - Path
}
else
{
    // "Else" - Path
}
```

Is Importing

This condition is true if the “import” action is currently executed. The condition can be used for events such as “After Row Selected” to determine if the event occurred during initialization or if it was triggered by the user.

```
if (ProjXUtil.isImporting())
{
    // "If" - Path
}
else
{
    // "Else" - Path
}
```

Is Mobile Environment

This condition is true when the application is executed in a mobile environment.

```
if (getApplication().getLauncher().isMobileEnvironment())
{
    // "If" - Path
}
else
```

```
{
    // "Else" - Path
}
```

Is Desktop Environment

This condition is true when the application is executed in a desktop environment.

```
if (getApplication().getLauncher().isDesktopEnvironment())
{
    // "If" - Path
}
else
{
    // "Else" - Path
}
```

Is Html5 Environment

This condition is true when the application is executed in a Html5 environment.

```
if (getApplication().getLauncher().isWebEnvironment())
{
    // "If" - Path
}
else
{
    // "Else" - Path
}
```

Is Service Environment

This condition is true when the application is executed in a service environment.

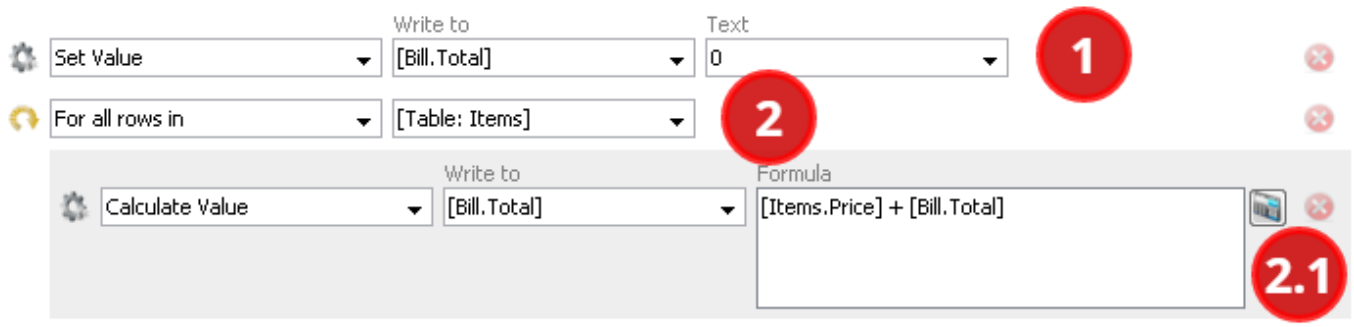
```
if
("APPSERVICES".equals(getApplication().getLauncher().getEnvironmentName()))
{
    // "If" - Path
}
else
{
    // "Else" - Path
}
```

Loop

Loops are used to execute commands multiple times for the datasets of a table/view.

Example: Create Sum

A typical application of a repetition is the generation of the sum from a number of individual values. The individual values (e.g., Table “Items”) are saved in a detail table, which is used as the parameter for the repetition.



1. The column “Bill.Total” is set to 0.
2. The following command is executed for all rows of the table “Items”:
    1. Calculate `[Bill.Total] + [Items.Price]` and write the resulting value to “[Bill.Total]”.

All prices are added and the total is saved in the “Bill.Total” column

Example: Keeping Subtotals Current

Another common example is the generation of subtotals after each individual entry. In the example below, the table “To-dos” contains a column “Value1” (e.g., for expenses). For each to-do row, we would like to know the amount of expenses that have accumulated up to and including the current to-do item.

The values in the “Value1” column are added from the first row to the currently selected row using the sumToCurrent([Todos.Value1]) group function. This function returns the subtotal of expenses in each To-do row.

If the expenses for an individual To-do item (“Value1” column) are changed, all sub-totals in all rows have to be recalculated. This calculation can be performed using a repetition involving all rows in the To-dos table. It is good practice to add this action as a “value change” event in the “To-dos” table. In addition, the currently selected row in the “To-dos” table should be saved and restored after the operation is completed.