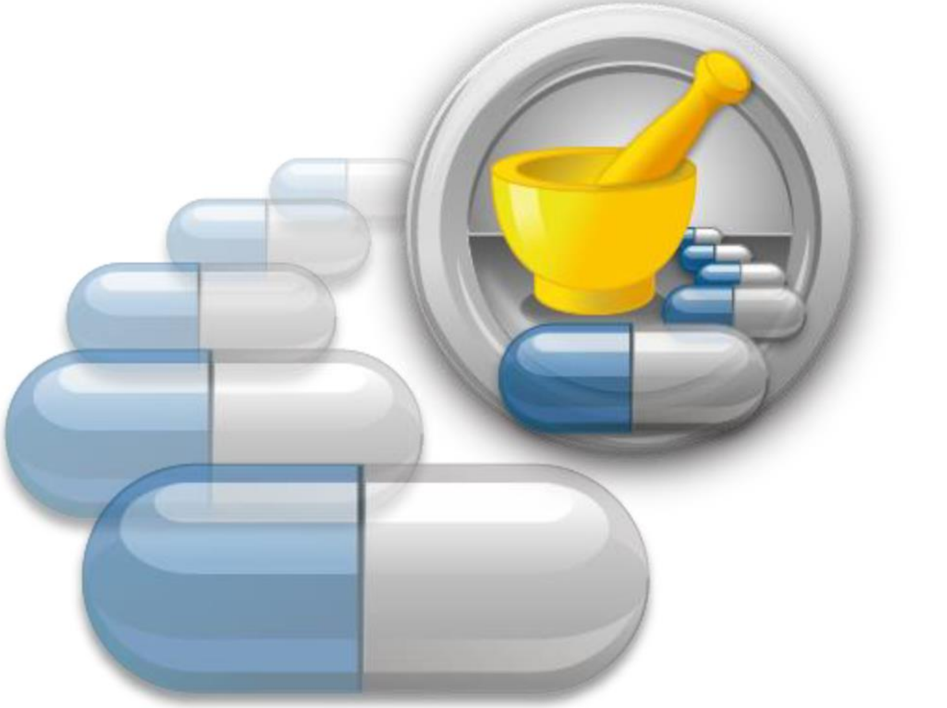
LISTEN.
THINK.
SOLVE.®

PharmaSuite®



RECIPE AND WORKFLOW DESIGNER
VOLUME 1: INTRODUCTION AND BASICS

RELEASE 8.4
USER MANUAL

PUBLICATION PSRDV1-UM005E-EN-E-DECEMBER-2017
Supersedes publication PSRD-UM005D-EN-E

Allen-Bradley · Rockwell Software

Rockwell Automation

Contents

Figures

Recipe and Workflow Designer

Recipe and Workflow Designer of PharmaSuite is a graphical workbench for building and maintaining master recipes, master workflows, and their component building blocks. With its functions for status and version control, it covers the entire life cycle of a master recipe, master workflow, or building block. The workbench provides material flow control on the basis of material parameters, information flows for values from process parameters, privilege parameters for access rights, capability parameters, and equipment parameters.
For each structure level of a master recipe, master workflow, or building block, the system presents an SFC graph (sequential function chart).
PharmaSuite Recipe and Workflow Designer is available in several modes: as Recipe Designer - Batch, Recipe Designer - Device, and Workflow Designer. The permission to switch between the modes depends on a user's access rights and is available from the **Window** menu of Recipe Designer - Batch (see "Main Menu Bar", "Window" section in Vol. 2), Recipe Designer - Device (see "Main Menu Bar", "Window" section in Vol. 3), or Workflow Designer (see "Main Menu Bar", "Window" section in Vol. 4), respectively.

This section contains important information about the basic principles of working with Recipe and Workflow Designer. Please read this section carefully, because it provides a solid background for all operations you may wish to perform with your system.

Later sections will explain how to perform the specific tasks in the system. We assume you are familiar with the conventions described in the following sections and the fundamentals of working with a personal computer.

Typographical Conventions

This documentation uses typographical conventions to enhance the readability of the information it presents. The following kinds of formatting indicate specific information:

Bold typeface	Designates user interface texts, such as ■ window and dialog titles ■ menu functions ■ panel, tab, and button names ■ box labels ■ object properties and their values (e.g. status).
Italic typeface	Designates technical background information, such as ■ path, folder, and file names ■ methods ■ classes.
CAPITALS	Designate keyboard-related information, such as ■ key names ■ keyboard shortcuts.
`Monospaced typeface`	Designates code examples.

TIP

Instructions in this manual are based on Windows 7. Select the appropriate commands if you are using a different operating system.

Screen Layout

The basic screen layout of Recipe and Workflow Designer holds the following user interface components:

- menus and toolbars for accessing the functions of Recipe and Workflow Designer
- Graph Window with upper and lower tab bar navigation
- two navigation support windows, Map and Explorer
- Setlist window for graph-building support
- Property window for displaying the properties of recipe, workflow, and building block elements, their headers, and source building blocks
- Phase Preview window for displaying how a phase will be rendered during execution
- Messages window for displaying all error, warning, or information messages pertaining to the active master recipe, master workflow, or building block
- status bar with general information.

Since you can run Recipe and Workflow Designer in different modes, Recipe Designer - Batch, Recipe Designer - Device, and Workflow Designer, which are streamlined for their specific areas of application, some of the basic components look slightly different between the modes and provide specialized functions.

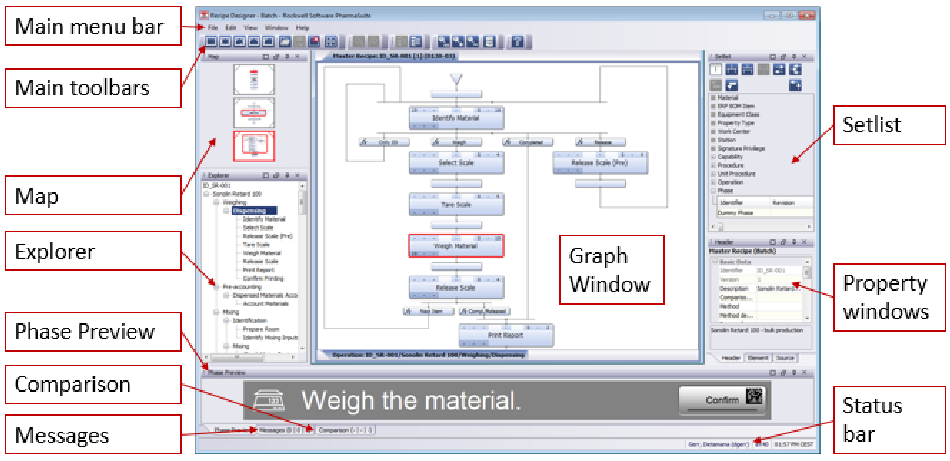


Figure 1: Screen layout of Recipe Designer - Batch

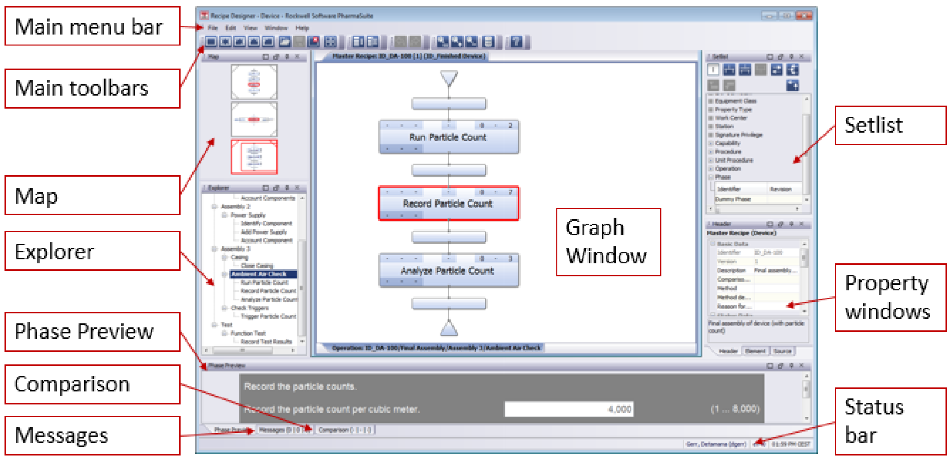


Figure 2: Screen layout of Recipe Designer - Device

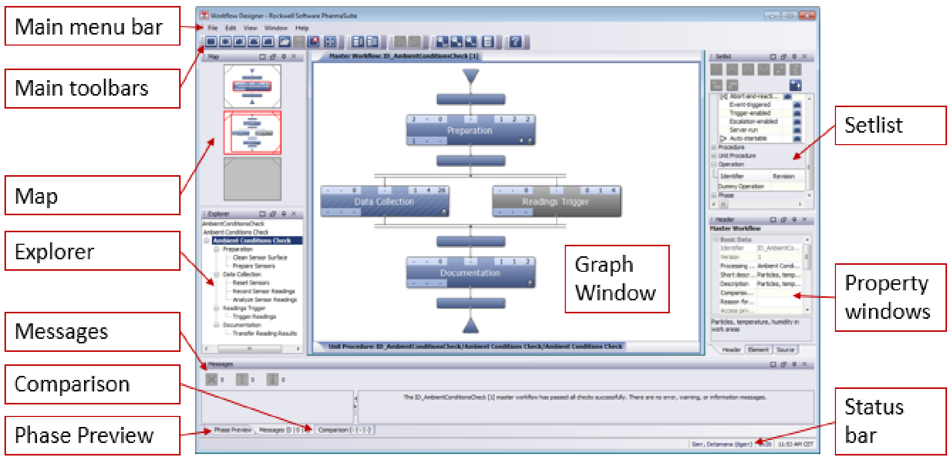


Figure 3: Screen layout of Workflow Designer

Panel Management

When you start Recipe and Workflow Designer for the first time, it opens with the default screen layout, which holds the dockable main menu bar and main toolbars at the top, the Graph Window in the center of the screen, two dockable panels for Map and Explorer to the left of the Graph Window, two dockable panels for the Setlist and the Property windows to its right, and two dockable panels for the Phase Preview and the Messages window at the bottom. You can, however, rearrange the panels if you prefer a different layout.

When you exit the system, Recipe and Workflow Designer saves the current panel layout and will start your next session with this layout. You can, however, always revert back to the default layout (page 6).

To change the position and size of the panels, click a panel header bar and drag the panel to the desired position on the screen. The system displays the new panel size and position with a gray preview frame. Panels can be

- free floating
- docked horizontally or vertically
- nested into another panel, accessible for switching by tab.

TIP

To move a menu bar or a toolbar you have to use their drag handles, which are located at the left margins of horizontal bars and at the top margins of vertical bars. If you pull a toolbar out as floating panel, you can close the panel.
To reopen it, proceed as follows:

- Right-click anywhere on the menu bar to open a shortcut menu.
- Select the toolbar. The system displays it again at the location where you closed it before.

With the other functions of the shortcut menu you can configure if the bars are **Rearrangable**, **Hidable**, or **Floatable**.

Each panel header provides a toolbar for further panel resizing and movements:

☐ 🗗 Maximize/restore
Maximizes a panel at its current position or restores it to its original size.

🗗 ☐ Toggle floating
Floats or unfloats a panel. An unfloated panel reverts to its original position and size.

⊥ ⊣ Toggle auto-hide
Auto-hides a docked panel. It can be accessed by tab. If you revoke the auto-hide, the panel reverts to its original position and size.

Close
Closes a panel. To reopen it, open the **View** menu and select the panel for display.

When the number of open tabs on a tab bar exceeds the available width of the tab bar, the system displays the following buttons to support your navigation:

Scroll backward
Scrolls the tab bar to the left.

Scroll forward
Scrolls the tab bar to the right.

Close
Displays a list of all open tabs from which you can select a tab you wish to close. The currently active tab is indicated by bold font.

To manage your screen layout changes, the system provides the **Window** menu with the following functions:

- Undo layout change
  Revokes the last layout change you have performed. You can undo up to 100 actions, thus you can step by step revoke the last 100 layout changes you have performed. Once you have performed a layout change, the menu function changes to a more precise description of the undo action, such as **Undo resizing** or **Undo dragging**.

- Redo layout change
  Redoes the last layout change you have revoked with the **Undo layout change** function. You can redo up to 100 actions, thus you can step by step redo the last 100 layout changes you have revoked. Once you have performed a layout change, the menu function changes to a more precise description of the undo action, such as **Redo resizing** or **Redo dragging**.

- Save user layout
  Saves the current window layout and overwrites the layout that was last saved by you on this computer.

- Load user layout
  Loads the last layout you have saved on this computer with the **Save user layout** function.

- Reset layout
  Resets the window layout to the system-defined default layout. This function does not affect the saved user layout, which can be restored with the **Load user layout** function.

Chapter 2

Basic Concepts of Recipe and Workflow Designer

Based on the concepts of the S88 and S95 standards, Recipe and Workflow Designer provides a comprehensive framework for creating, configuring, and managing master recipes and master workflows, along with their building blocks, components, and structures:

- Building block (page 7)
- Procedure (page 8)
- Unit procedure (page 8)
- Operation (page 8)
- Phase (page 8)
- SFC graph (page 8)
- MFC data (page 9)
- Change requests for mass changes (page 9)

TIP

Please note that PharmaSuite provides a tool to export master recipes, master workflows and building blocks from one database and import them into another database of the same release. With this tool you can develop, test, and verify recipes, workflows, and building blocks in a test environment and only move them to your productive environment when they are ready for use.
For further information, please refer to "Exporting and Importing Master Recipes, Master Workflows, and Building Blocks" in "PharmaSuite Technical Manual Administration".

What Is a Building Block?

Building blocks are the individual structural components on any level of a master recipe or master workflow that can be used in building a recipe or workflow structure: procedure, unit procedure, operation, or phase.

What Is a Procedure?

The procedure is the highest level in the (recipe) hierarchy and defines the strategy for carrying out a major processing action such as making a batch. It is defined in terms of an ordered set of unit procedures.

What Is a Unit Procedure?

According to S88, a unit procedure consists of an ordered set of operations that causes a contiguous production sequence to take place within a unit. Only one operation is presumed to be active in a unit at any time. An operation is carried to completion in a single unit. However, multiple unit procedures of one procedure may run concurrently, each in different units.
However, PharmaSuite allows modeling and execution of parallel operations in compliance with general SFC rules.

What Is an Operation?

An operation is an ordered set of phases that defines a major processing sequence that takes the material being processed from one state to another, usually involving a chemical or physical change. It is often desirable to locate operation boundaries at points in the procedure where normal processing can safely be suspended.

What Is a Phase?

The smallest element of procedural control that can accomplish a process-oriented task is a phase.

What Is an SFC Graph?

PharmaSuite uses SFC graphs to model recipes and their structures.

Sequential function chart (SFC) programming offers a graphical method of organizing the program. The three main components of an SFC are steps, actions, and transitions. Steps are merely chunks of logic, i.e., a unit of programming logic that accomplishes a particular control task. Actions are the individual aspects of that task. Transitions are the mechanisms used to move from one task to another.

In PharmaSuite, steps are represented as building blocks.

What Is Material Flow Control (MFC)?

Material flow control (MFC) defines the material flow between the unit procedures of a master recipe, master workflow, or procedure building block. MFC uses the material input and output parameters defined with the phases to specify during which unit procedures the materials enter the production process and how they are moved and transformed during processing until their flow ends with the produced material.

The flow of materials is represented by three types of MFC items:

- An input item represents a material that enters into a process step, such as a raw material.
- A transfer item represents a material that moves from one process step to the next, such as an intra material.
- An output item represents the final product that leaves the last process step. There can only be one output per master recipe.
  Only Dispense stand-alone master recipes can have more than one output item.

What Is a Change Request?

The **Change request** function can replace a specific approved building block with an updated approved version of the building block in any valid or scheduled master recipe, master workflow, and approved or archived building block. It automatically creates editable copies of all affected components, replaces the building block, and moves the new versions of the components to the status the initial version had. Performing this type of change manually on a large number of components would be both time-consuming and error-prone.

What Is a Confidential Object?

The concept of confidential objects protects the intellectual property of recipes, workflows, building blocks, orders, ERP BOMs, and related data from unauthorized access. The system provides a specific access rights type to define access privileges that allow users to maintain protected recipes, workflows, and orders.

A master recipe or master workflow can be protected by assigning an explicit access privilege or inheriting the access privilege from the used ERP BOM or the selected material (product). Orders and workflows based on a protected master recipe or master workflow inherit the access privilege automatically. A user without the suitable access privilege is not able to

- view a protected master recipe or master workflow,
- start a protected order or workflow for execution,

- log in at a station on which a protected order or workflow runs,
- see a protected order or workflow in PharmaSuite for Production Responses,
- print the batch report, weighing report, or workflow report of a protected order or workflow,
- open a change request that contains a protected object,
- see any transaction history data created by a protected order or workflow.

If several confidential objects are linked with each other (for example, when a confidential building block is used in a recipe or a confidential workflow is assigned to an order), only the access privilege defined for the master object is required to access the linked objects with the master object as starting point. In order to link two confidential objects, they do not need to be protected by the same access privilege. A user, however, needs to have both access privileges to create the link.

Example scenario:
A workflow and an order require different access privileges. A user with access to both can assign the workflow to the order. Afterwards, another user who only has the access privilege of the order can print the batch report and review exceptions of the workflow as long as he starts the review from the order.

Managing Building Blocks

The functions for managing building blocks allow you to open existing custom building blocks for editing, create new custom building blocks from scratch, save elements of existing recipes, workflows, or building blocks as new custom building blocks, and locate all occurrences of an existing custom building block.

The meta data of a building block, such as its description, status (page 29), and revision information is maintained as part of its header data.

Header data is not only accessible through the **Header** property window from Recipe Designer - Batch (see "Header Property Window" in Vol. 2), Recipe Designer - Device (see "Header Property Window" in Vol. 3) or Workflow Designer (see "Header Property Window" in Vol. 4), but also when building blocks are created from scratch (page 15) or from existing elements (page 16), copied (page 17), or renamed (page 18).

A building block is uniquely defined by two mandatory properties, **Identifier** and **Revision**. This allows you to maintain several variants of a building block at the same time. Revisions have to be assigned manually; there is no automatic generation mechanism implemented so that you have full flexibility with respect to defining a naming or numbering scheme for your revisions.

TIP

Please note that revisioning is not linked to the status graph of building blocks. Thus you can have several **Approved** revisions of one building block at the same time.

To open an existing building block, proceed as follows:

1. In the main menu bar, open the **File** menu and select **Open** or in the File toolbar, click the **Open** button.
   The system displays the **Open** dialog.
2. In the **Filter** tool of the **Open** dialog, select the object type (procedure, unit procedure, operation, phase) of the building block you wish to open.
3. Double-click the building block you wish to work on.
   The system opens a new tab in the upper tab bar of the Graph Window. It displays the header of the building block.

To create a new building block from scratch, proceed as follows:

1. In the main menu bar, open the **File** menu and select the **New <object type>** function for the building block you wish to create (procedure, unit procedure, operation, phase) or in the File toolbar, click the respective **New <object type>** button.
   The system displays the **New <Object Type>** dialog (page 15).
2. Type the **Identifier** and the **Revision** of the new building block in the respective boxes.
3. Click the **OK** button.
   The system opens a new tab in the upper tab bar of the Graph Window. It displays the header component of the new building block.
4. Double-click the header to create the next lower level in the building block structure. The Graph Window updates to display the new element enclosed by a start and end step.
5. Continue to build the structure of the new building block until you have reached the phase graph on the operation level.
   In the Map and the Explorer, you can watch the building block structure grow.

TIP

Phase building blocks represent the lowest level of the building block hierarchy. Thus they only have a header component without an underlying graph structure.

To create a new building block by copying an existing master recipe, master workflow, or building block element, proceed as follows:

1. Open the master recipe, master workflow, or building block that contains the element you wish to save as new building block.
2. Navigate to the structure level whose graph contains the element and select it.
3. Right-click anywhere on the background in the Graph Window to open the shortcut menu. Select the **Create building block from selected** function.
   The system displays the **Create <Object Type>** dialog (page 16).
4. The text boxes are pre-filled with the data of the element you are about to copy. Adjust the data or type a new data. The **Identifier** and **Revision** boxes are mandatory.
   If you wish to add the new building block to the Setlist immediately, make sure the **Add to Setlist** option is selected.
   If you wish to open the new building block in the Graph Window immediately, make sure the **Open in new tab** option is selected.
5. Click the **OK** button.
   The new building block is now available for selection from the **Open** dialog and the Universe.

To copy an existing building block, proceed as follows:

1. Open the building block you wish to copy.
2. In the main menu bar, open the **File** menu and select **Save <building block identifier> as**.
   The system displays the **Save <Object Type> as** dialog (page 17).
3. The text boxes are pre-filled with the data of the element you are about to copy. Adjust the data or type a new data. The **Identifier** and **Revision** boxes are mandatory.
4. Click the **OK** button.
   The system opens a new tab in the upper tab bar of the Graph Window with the new building block.

Once you have created a new building block structure from scratch, your Graph Window holds the basic graph structure with one dummy building block on each level of the hierarchy. To complete your building block, proceed as follows:

1. In the View toolbar, click the **Open Universe** button (page 99).
   The system opens the Universe where you can select all the components, inputs, and outputs you wish to use in your building block and feed them into the Setlist.
2. Navigate to the structure level on which you wish to build the graph.
3. Use the toggle buttons of the Setlist toolbar to define how you will add a Setlist item to the graph (sequence, selection branch, simultaneous branch).

   TIP
   Please note that to replace the first, system-generated element of a structure level with a building block from the Setlist you can use the **Replace** button (page 99).

4. Double-click a building block in the Setlist to add it in the defined manner to your graph.

   TIPS
   Please note that only in the Graph Window can you insert loops.
   Right-click the building block where you wish the loop to start, to open the process workflow toolbar. Click the **Loop** button (page 98) and select a suitable loop endpoint to close the loop.

   Specific restrictions apply to Dispense unit procedure graphs and operations:
   There can only be one operation that is marked as Dispense-specific within the graph.
   A Dispense operation cannot be combined with other non-Dispense operations.
   This means that a Dispense unit procedure must contain exactly one Dispense operation.

5. If your graph contains branches or loops, configure the transition conditions that are relevant to the decision points.
6. Use the toggle buttons of the Setlist toolbar and double-click the respective parameter to add local input and output materials or process parameters to your elements.
7. To configure the parameters of the elements, click one of their parameter buttons located in the corners of the building block.
   The system opens the Parameter Panel where you can define all parameter properties.

   TIP
   If you work on a procedure building block for a master recipe that holds material parameters that need to be tracked by material flow control, be sure to mark them as **MFC-relevant** by selecting this option in the Parameter Panel.

8. For procedure building blocks for master recipes or master workflows with MFC-relevant material parameters you can now define the building block's internal material flows. In the main menu bar of Recipe Designer - Batch (see "Main Menu Bar", "View" section in Vol. 2), Recipe Designer - Device (see "Main Menu Bar", "View" section in Vol. 3), or Workflow Designer (see "Main Menu Bar", "View" section in Vol. 4), open the **View** menu and select the **Material Flow Control** function or use the keyboard shortcut ALT+C to open the **Material Flow Control** tab of Recipe Designer - Batch (see "MFC Data" in Vol. 2), of Recipe Designer - Device (see "MFC Data" in Vol. 3), or of Workflow Designer (see "MFC Data" in Vol. 4) in the lower tab bar. Now define how the materials you have marked as MFC-relevant flow through the unit procedures of your procedure graph.
9. When you have completed all graphs of your building block and have configured all parameters, check if its structure and parameters are valid.
   Open the Messages window to view all messages the system has returned for your building block. Make all necessary changes.

   TIP
   Only if there are no errors listed for the building block in the Messages window can you move it to the **Approved** status (page 29).

Dialogs of Building Block Management

To manage building blocks in both Recipe Designer and Workflow Designer, the system provides specific dialogs for the following functions:

- Creating a building block (page 16) from an existing graph component,
- Creating a new building block (page 15) from scratch,
- Renaming (page 18) an existing building block,
- Saving a building block under a new name to create a copy (page 17),
- Locating the occurrence of a building block throughout all recipes, workflows, and building blocks by compiling a usage list (page 19).

New Building Block

The **New <Data Type>** dialog supports you in selecting a suitable identifier and revision for your new building block.

TIP

If your building blocks will be subject to automatic creation of new revisions through change requests, it is recommended that revision numbers contain at least one numeric integer value. When the system creates a new revision of a building block while executing a change request, it increases the last integer value of a revision number to the next available integer.
Examples:
[1] -> [2]
[A] -> [3] if revisions [1] and [2] exist already.
[1.0] -> [1.1]
[2B] -> [3B]
[2B1] -> [2B2]
[1B] -> [4B] if revisions [2B] and [3B] exist already.

The content of the **Identifier** box functions as search criterion that controls which data objects are displayed in the list of **Available Revisions**. The search is restricted to the **Identifier** column and will disregard all objects that contain the given string of characters in any other column.

Once you have typed the **Identifier**, the system lists all objects of this object type that contain this string of characters in their identifiers.
By default, the list is sorted alphabetically with respect to the **Identifier** column of the object, but you can adjust both the sort and the column order (page 131).
To fill the **Identifier** and **Revision** boxes with the data of one of the listed objects, double-click the respective list row.

TIP

Please note that Recipe and Workflow Designer takes a snapshot of the database when you access an object type. This means that the list of data objects remains unaware of changes made to database objects or new objects added by other users while you browse through it. To synchronize the snapshot with the central database, click the **Refresh from database** button.

Now you can easily define a suitable revision for your new building block and type data in any of the other text boxes.
If the building block is confidential and needs to have restricted access permissions, you can select a suitable access privilege. The list of available access privileges only contains those privileges that are also assigned to your current login.

Only when you have filled in both mandatory boxes, is the **OK** button enabled and you can click it to save and close the form.

CREATE BUILDING BLOCK

The **Create <Data Type>** dialog supports you in selecting a suitable identifier and revision for your new building block.

TIP

If your building blocks will be subject to automatic creation of new revisions through change requests, it is recommended that revision numbers contain at least one numeric integer value. When the system creates a new revision of a building block while executing a change request, it increases the last integer value of a revision number to the next available integer.
Examples:
[1] -> [2]
[A] -> [3] if revisions [1] and [2] exist already.
[1.0] -> [1.1]
[2B] -> [3B]
[2B1] -> [2B2]
[1B] -> [4B] if revisions [2B] and [3B] exist already.

All text boxes are preset with the data of your original building block.
The content of the **Identifier** box functions as search criterion that controls which data objects are displayed in the list of **Available Revisions**. The search is restricted to the **Identifier** column and will disregard all objects that contain the given string of characters in any other column.

Once you have typed the **Identifier**, the system lists all objects of this object type that contain this string of characters in their identifiers.
By default, the list is sorted alphabetically with respect to the **Identifier** column of the object, but you can adjust both the sort and the column order (page 131).
To fill the **Identifier** and **Revision** boxes with the data of one of the listed objects, double-click the respective list row.

TIP

Please note that Recipe and Workflow Designer takes a snapshot of the database when you access an object type. This means that the list of data objects remains unaware of changes made to database objects or new objects added by other users while you browse through it. To synchronize the snapshot with the central database, click the **Refresh from database** button.

Now you can easily define a suitable identifier and revision for your new building block and type data in any of the other text boxes.
If the building block is confidential and needs to have restricted access permissions, you can select a suitable access privilege. The list of available access privileges only contains those privileges that are also assigned to your current login.
To make your new building block immediately available for use from the Setlist, make sure the **Add to Setlist** option is selected.

To open your new building block immediately in the Graph Window for further changes, check the **Open in new tab** option.
Only when you have filled in both mandatory boxes, is the **OK** button enabled and you can click it to save and close the form.

Save Building Block as

The **Save <Data Type> as** dialog supports you in selecting a suitable identifier and revision for your new building block.

TIP

If your building blocks will be subject to automatic creation of new revisions through change requests, it is recommended that revision numbers contain at least one numeric integer value. When the system creates a new revision of a building block while executing a change request, it increases the last integer value of a revision number to the next available integer.
Examples:
[1] -> [2]
[A] -> [3] if revisions [1] and [2] exist already.
[1.0] -> [1.1]
[2B] -> [3B]
[2B1] -> [2B2]
[1B] -> [4B] if revisions [2B] and [3B] exist already.

All text boxes are preset with the data of your original building block.
The content of the **Identifier** box functions as search criterion that controls which data objects are displayed in the list of **Available Revisions**. The search is restricted to the **Identifier** column and will disregard all objects that contain the given string of characters in any other column.

Once you have typed the **Identifier**, the system lists all objects of this object type that contain this string of characters in their identifiers.
By default, the list is sorted alphabetically with respect to the **Identifier** column of the object, but you can adjust both the sort and the column order (page 131).
To fill the **Identifier** and **Revision** boxes with the data of one of the listed objects, double-click the respective list row.

TIP

Please note that Recipe and Workflow Designer takes a snapshot of the database when you access an object type. This means that the list of data objects remains unaware of changes made to database objects or new objects added by other users while you browse through it. To synchronize the snapshot with the central database, click the **Refresh from database** button.

Now you can easily define a suitable identifier and revision for your new building block and type data in any of the other text boxes.
If the building block is confidential and needs to have restricted access permissions, it inherits its access privilege from the original building block. The inherited access privilege cannot be changed. If the original building block is not confidential, you can select a suitable access privilege. The list of available access privileges only contains those privileges that are also assigned to your current login.
To make your new building block immediately available for use from the Setlist, make sure the **Add to Setlist** option is selected.
Only when you have filled in both mandatory boxes, is the **OK** button enabled and you can click it to save and close the form.

RENAME BUILDING BLOCK

The **Rename <Data Type>** dialog supports you in selecting a suitable new identifier for your building block.

All text boxes are preset with the data of your original building block.
The content of the **Identifier** box functions as search criterion that controls which data objects are displayed in the list of **Available Revisions**. The search is restricted to the **Identifier** column and will disregard all objects that contain the given string of characters in any other column.

Once you have typed the **Identifier**, the system lists all objects of this object type that contain this string of characters in their identifiers.
By default, the list is sorted alphabetically with respect to the **Identifier** column of the object, but you can adjust both the sort and the column order (page 131).
To fill the **Identifier** and **Revision** boxes with the data of one of the listed objects, double-click the respective list row.

TIP

Please note that Recipe and Workflow Designer takes a snapshot of the database when you access an object type. This means that the list of data objects remains unaware of changes made to database objects or new objects added by other users while you browse through it. To synchronize the snapshot with the central database, click the **Refresh from database** button.

Now you can easily define a suitable identifier for your building block. You can also type or change data in any of the other text boxes.
If the building block is confidential, it has an access privilege assigned. The access privilege cannot be changed.
Only when you have filled in both mandatory boxes, is the **OK** button enabled and you can click it to save and close the form.

Usage List

The **Usage List** supports you in locating the occurrences of a building block in recipes, workflows, and building blocks. It differentiates between direct and indirect usages of a building block:

- **Direct usage** of a building block refers to occurrences in which the building block in question is directly accessible for editing or replacement within the analyzed structure (master recipe, master workflow, or building block). This means that neither its direct parent object nor a higher-level parent object may be **Approved**, unless the higher-level object is the root object of the analyzed structure.

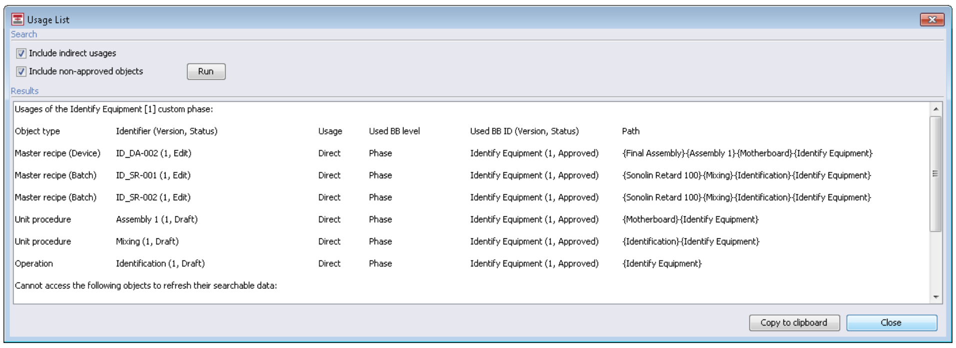
  TIP
  If the root object itself is **Valid** or **Approved**, you need to perform a **Save as** action to be able to access the building block.

- **Indirect usage** of a building block refers to occurrences in which the building block in question is not directly accessible for editing or replacement within the analyzed structure (master recipe, master workflow, or building block) because its direct parent object or a higher-level parent object are **Approved**, unless the higher-level object is the root object of the analyzed structure.

TIPS

Please note that each time you compile a usage list for a building block, the system refreshes the data on which it performs the search. For this purpose it has to access all relevant objects (master recipes, master workflows, and building blocks) on the database to determine if they have been modified since the last time a usage list was compiled.
If there are other users who work with the system on the same database, some objects may be locked as they are currently being edited and are thus not accessible for refreshing. In this case, the usage list compiles with the potentially obsolete data it has retrieved before its previous run, but lists all objects it could not refresh in a specific section.

Please note that your access rights determine which components are considered when the system compiles the usage list. Confidential recipes, workflows, or building blocks, which you are not allowed to access, are not shown in the list.



Usages of the Identify Equipment [1] custom phase:

Object type	Identifier (Version, Status)	Usage	Used BB level	Used BB ID (Version, Status)	Path
Master recipe (Device)	ID_DA-002 (1, Edit)	Direct	Phase	Identify Equipment (1, Approved)	{Final Assembly}-{Assembly 1}-{Motherboard}-{Identify Equipment}
Master recipe (Batch)	ID_SR-001 (1, Edit)	Direct	Phase	Identify Equipment (1, Approved)	{Sonolin Retard 100}-{Mixing}-{Identification}-{Identify Equipment}
Master recipe (Batch)	ID_SR-002 (1, Edit)	Direct	Phase	Identify Equipment (1, Approved)	{Sonolin Retard 100}-{Mixing}-{Identification}-{Identify Equipment}
Unit procedure	Assembly 1 (1, Draft)	Direct	Phase	Identify Equipment (1, Approved)	{Motherboard}-{Identify Equipment}
Unit procedure	Mixing (1, Draft)	Direct	Phase	Identify Equipment (1, Approved)	{Identification}-{Identify Equipment}
Operation	Identification (1, Draft)	Direct	Phase	Identify Equipment (1, Approved)	{Identify Equipment}

Cannot access the following objects to refresh their searchable data:

Figure 4: Usage list

The **Usage List** dialog provides the following functions:

- Search options to refine your search, which are both selected by default:
  - **Include indirect usages**
    Determines if the usage list also shows all occurrences of the building block in question, no matter if its usage is direct or indirect.
  - **Include non-approved objects**
    Determines if the usage list also shows occurrences of the building block in question where it is included in master recipes or master workflows that have a status other than **Valid** or in building blocks that are not **Approved**.
- To compile the usage list with the selected options, click the **Run** button.
- To copy the complete list to the clipboard for subsequent pasting into a spreadsheet application or a text editor, click the **Copy to clipboard** button. To retain its basic formatting in the application into which it is pasted, the list holds blanks and tabs as separator characters.
- To close the dialog, click the **Close** button.

SFC Graph Components of Building Blocks

Building blocks can exist on any level of the SFC graph hierarchy below master recipes or master workflows. Thus there are building block header components for procedures (page 20), unit procedures (page 22), operations (page 24), and phases (page 27). Generally, the system indicates building block graphs with a yellow background color in the Graph Window.

Procedure Header

A **Procedure Header** tab contains exactly one procedure step.

The building block image of a procedure shows the identifier of the procedure in the center.
If the procedure is based on an **Approved** source building block, the image displays a marker icon.
To rename the procedure, open the **File** menu and select the **Rename <procedure identifier>** function.

The upper and lower left corners are reserved for input parameters and output parameters, such as material inputs and outputs. The upper right corner provides space for parameters that are not part of an input/output flow, such as privilege and process parameters.

- When there are parameters defined for a building block, the system displays their numbers on the respective parameter buttons in the corners of the building block.
- When there are no parameters defined, but the building block expects parameters, the system indicates this by displaying a red zero (0) on the respective buttons.

- When there are parameters defined, but the building block expects a higher number of parameters, the system indicates this by displaying the number of defined parameters in red on the respective buttons.
- When no parameters can be defined, the system indicates this by dashes (-) on the respective buttons.

The numbers displayed for procedures are the sum of all local parameters of the procedure plus all the parameters of its child elements (unit procedures, operations, and phases).
The count of equipment parameters sums up all equipment classes, equipment entities, and individual property types assigned to the phases of the procedure as well as all work centers assigned to its unit procedures.

When you hover over a parameter button, the system displays its type in a tooltip, regardless of the parameter's availability (**Material**, **Container**, **Equipment**, **Transition**, **Privilege**, **Capability**, **Process**).

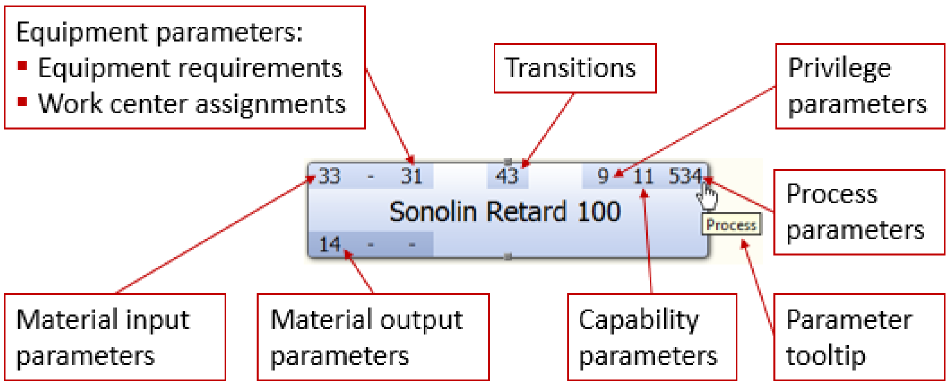


Figure 5: Recipe Designer - Batch: procedure building block - header

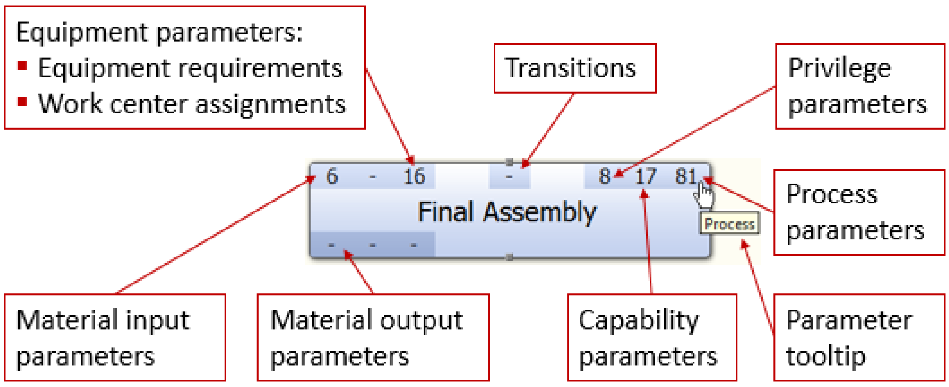


Figure 6: Recipe Designer - Device: procedure building block - header

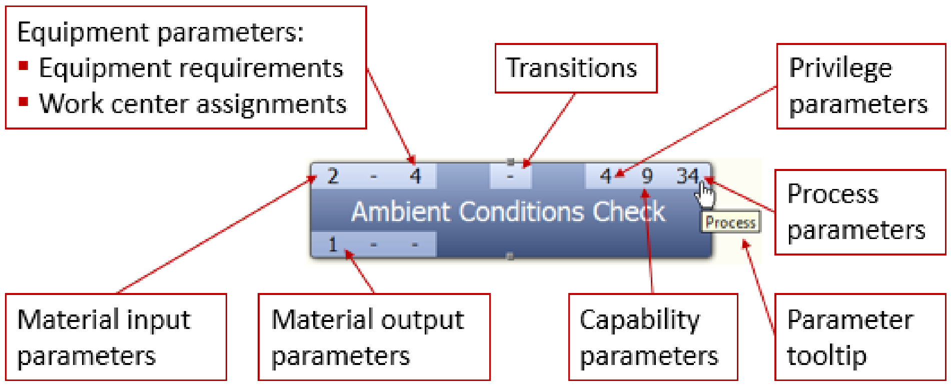


Figure 7: Workflow Designer: procedure building block - header

Right-click anywhere on the background to open a shortcut menu with the following functions:

TIP

Which of the functions are enabled also depends on the graph components you have selected.

- Go back (ALT+LEFT)
  Returns to the previously active tab in the lower tab bar within the currently active main component, if you navigated to your current tab from another one.
- Close sub-tab (CTRL+W)
  Closes the currently active tab of the lower tab bar.

UNIT PROCEDURE HEADER

A **Unit Procedure Header** tab contains exactly one unit procedure step.

The building block image of a unit procedure shows the identifier of the unit procedure in the center.
If the unit procedure is based on an **Approved** source building block, the image displays a marker icon.
To rename the unit procedure, open the **File** menu and select the **Rename <unit procedure identifier>** function.

The upper and lower left corners are reserved for input parameters and output parameters, such as material inputs and outputs. The upper right corner provides space for parameters that are not part of an input/output flow, such as privilege and process parameters.

- When there are parameters defined for a building block, the system displays their numbers on the respective parameter buttons in the corners of the building block.
- When there are no parameters defined, but the building block expects parameters, the system indicates this by displaying a red zero (0) on the respective buttons.

- When there are parameters defined, but the building block expects a higher number of parameters, the system indicates this by displaying the number of defined parameters in red on the respective buttons.
- When no parameters can be defined, the system indicates this by dashes (-) on the respective buttons.

The numbers displayed for unit procedures are the sum of all local parameters of the unit procedure plus all the parameters of its child elements (operations and phases).
The count of equipment parameters sums up

- the work centers assigned to the unit procedure,
- the stations assigned to the operations of the unit procedure, and
- the equipment classes assigned as requirements to the phases of the unit procedure.

When you hover over a parameter button, the system displays its type in a tooltip, regardless of the parameter's availability (**Material**, **Container**, **Equipment**, **Transition**, **Privilege**, **Capability**, **Process**).

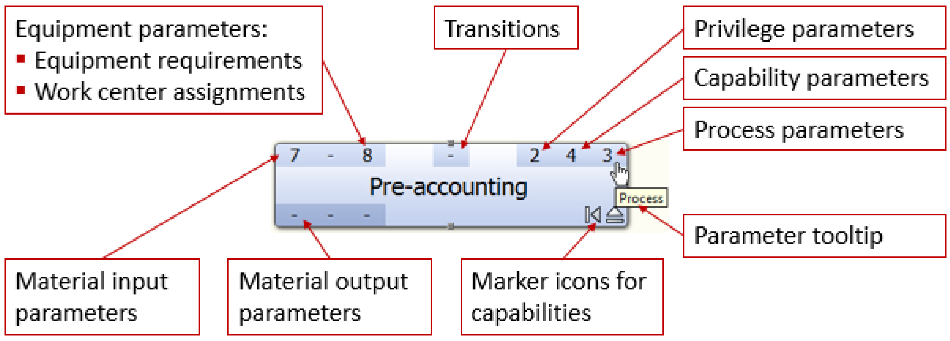


Figure 8: Recipe Designer - Batch: unit procedure building block - header

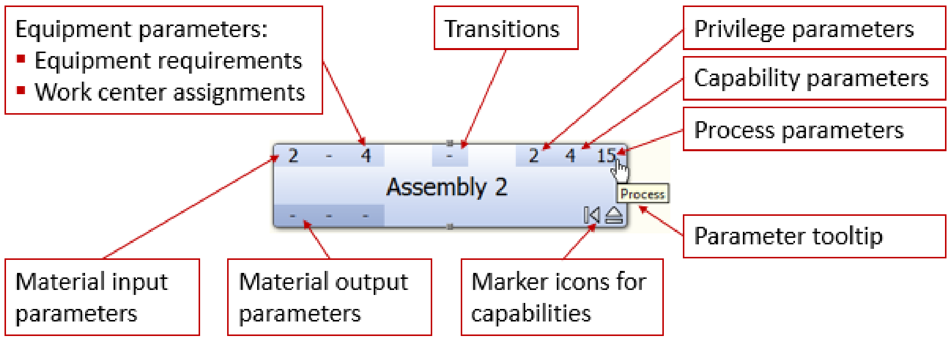


Figure 9: Recipe Designer - Device: unit procedure building block - header

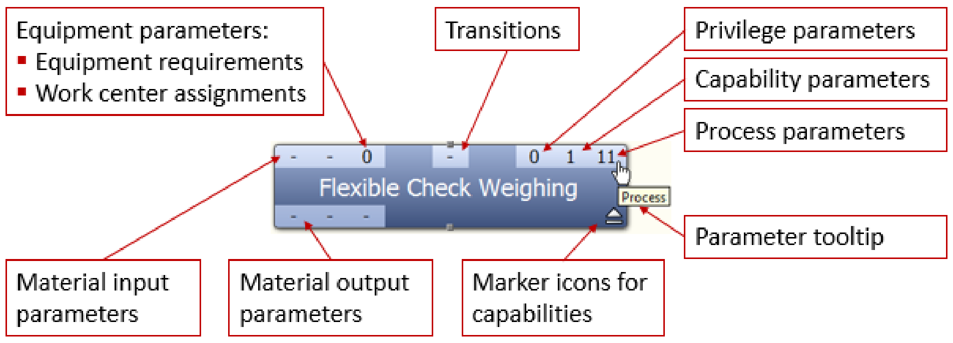


Figure 10: Workflow Designer: unit procedure building block - header

A unit procedure that contains a Dispense operation is automatically marked as Dispense-specific and has the following restrictions apply to its graph:

- There can only be one operation that is marked as Dispense-specific within the graph.
- A Dispense operation cannot be combined with other non-Dispense operations.

This means that a Dispense unit procedure must contain exactly one Dispense operation.

Right-click anywhere on the background to open a shortcut menu with the following functions:

TIP

Which of the functions are enabled also depends on the graph components you have selected.

- Go back (ALT+LEFT)
  Returns to the previously active tab in the lower tab bar within the currently active main component, if you navigated to your current tab from another one.
- Close sub-tab (CTRL+W)
  Closes the currently active tab of the lower tab bar.

Operation Header

An **Operation Header** tab contains exactly one operation step.

The building block image of an operation shows the identifier of the operation in the center.
If the operation is based on an **Approved** source building block, the image displays a marker icon.
To rename the operation, open the **File** menu and select the **Rename <operation identifier>** function.

The upper and lower left corners are reserved for input parameters and output parameters, such as material inputs and outputs. The upper right corner provides space for parameters that are not part of an input/output flow, such as privilege and process parameters.

- When there are parameters defined for a building block, the system displays their numbers on the respective parameter buttons in the corners of the building block.
- When there are no parameters defined, but the building block expects parameters, the system indicates this by displaying a red zero (0) on the respective buttons.
- When there are parameters defined, but the building block expects a higher number of parameters, the system indicates this by displaying the number of defined parameters in red on the respective buttons.
- When no parameters can be defined, the system indicates this by dashes (-) on the respective buttons.

The numbers displayed for operations are the sum of all local parameters of the operation plus all the parameters of its child phases.
The count of equipment parameters sums up

- the stations assigned to the operations of the unit procedure and
- the equipment classes assigned as requirements to the phases of the unit procedure.

When you hover over a parameter button, the system displays its type in a tooltip, regardless of the parameter's availability (**Material**, **Container**, **Equipment**, **Transition**, **Privilege**, **Capability**, **Process**).

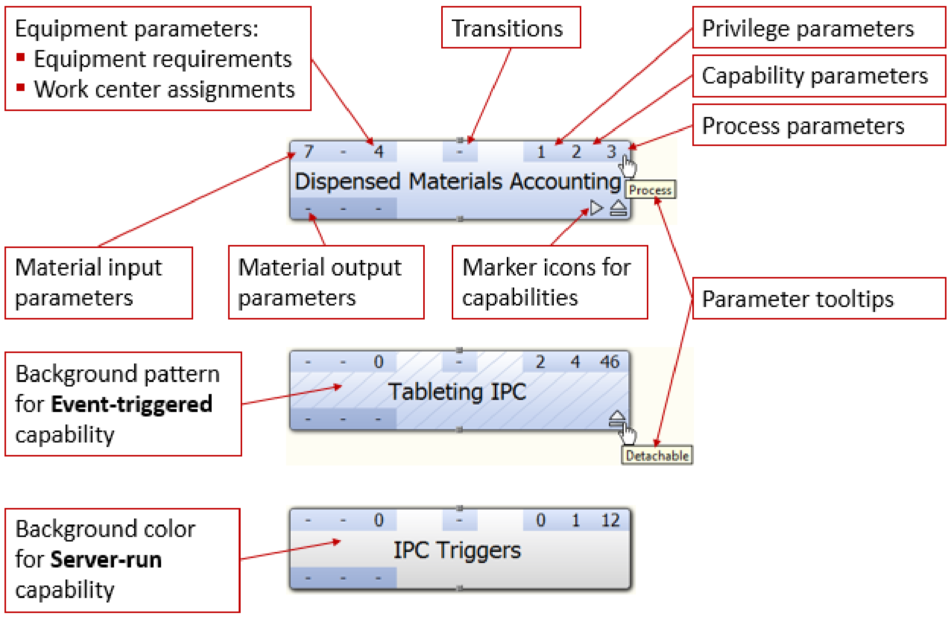


Figure 11: Recipe Designer - Batch: operation building block - header

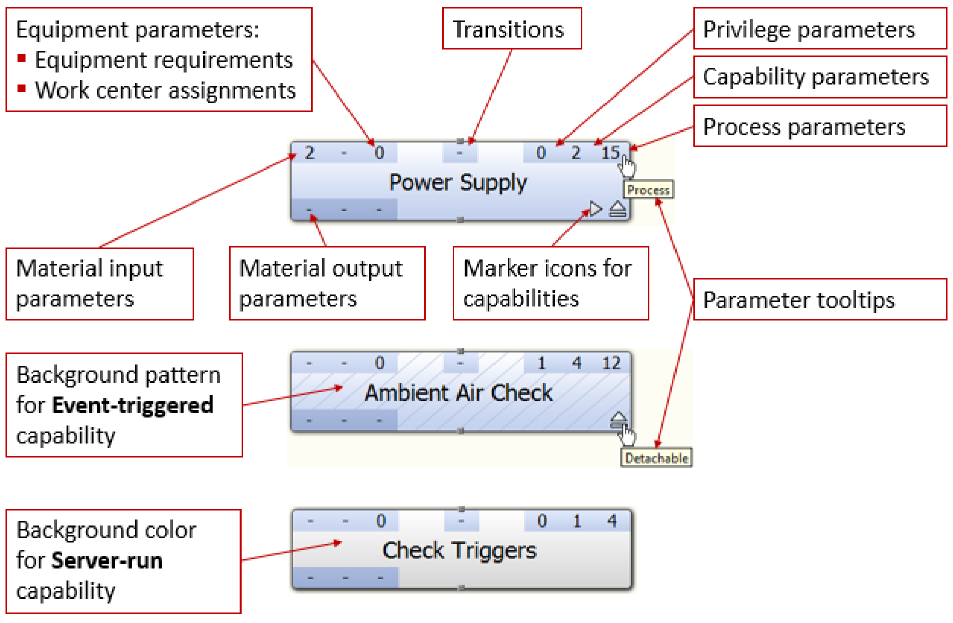


Figure 12: Recipe Designer - Device: operation building block - header

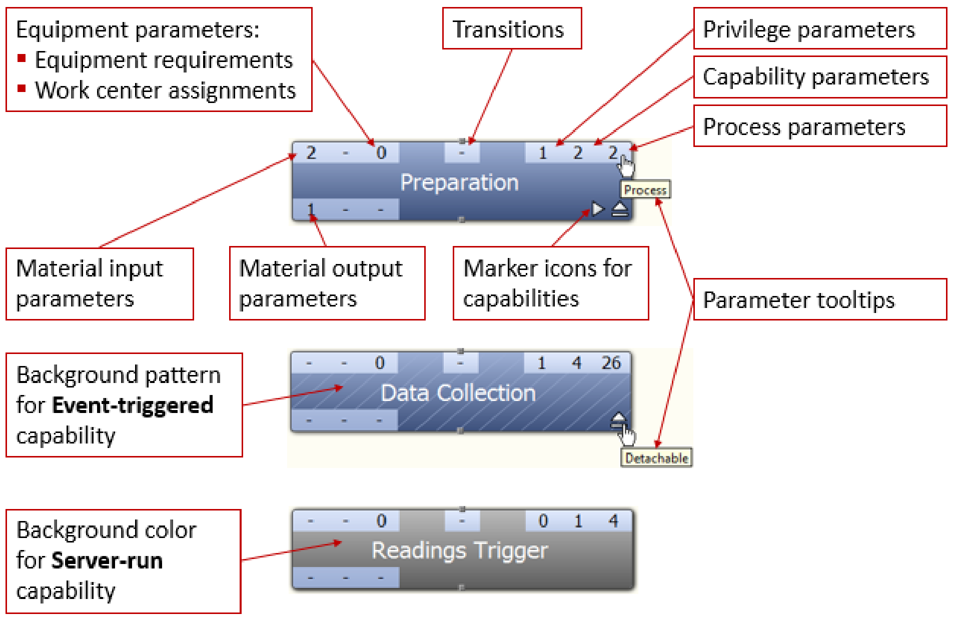


Figure 13: Workflow Designer: operation building block - header

When you draw Dispense phases into your graph to form a default Dispense operation, you additionally have to mark the operation itself as Dispense-specific. To do so, select the **Dispense** option in its **Header** property window (see "Header Property Window" in Vol. 2).

By marking an operation as Dispense-specific, you place the following restrictions on its usage in a unit procedure graph:

- There can only be one operation that is marked as Dispense-specific within the graph.
- A Dispense operation cannot be combined with other non-Dispense operations.

This means that a Dispense unit procedure must contain exactly one Dispense operation.

Right-click anywhere on the background to open a shortcut menu with the following functions:

TIP

Which of the functions are enabled also depends on the graph components you have selected.

- Go back (ALT+LEFT)
  Returns to the previously active tab in the lower tab bar within the currently active main component, if you navigated to your current tab from another one.
- Close sub-tab (CTRL+W)
  Closes the currently active tab of the lower tab bar.

PHASE HEADER

A **Phase Header** tab contains exactly one phase step.

The building block image of a phase shows the identifier of the phase in the center.
If the phase is based on an **Approved** source building block, the image displays a marker icon.
To rename the phase, open the **File** menu and select the **Rename <phase identifier>** function.

The upper and lower left corners are reserved for input parameters and output parameters, such as material inputs and outputs. The upper right corner provides space for parameters that are not part of an input/output flow, such as privilege and process parameters.

- When there are parameters defined for a building block, the system displays their numbers on the respective parameter buttons in the corners of the building block.
- When there are no parameters defined, but the building block expects parameters, the system indicates this by displaying a red zero (0) on the respective buttons.
- When there are parameters defined, but the building block expects a higher number of parameters, the system indicates this by displaying the number of defined parameters in red on the respective buttons.
- When no parameters can be defined, the system indicates this by dashes (-) on the respective buttons.

The numbers displayed for phases are the sum of all local parameters of the phase.

When you hover over a parameter button, the system displays its type in a tooltip, regardless of the parameter's availability (**Material**, **Container**, **Equipment**, **Transition**, **Privilege**, **Capability**, **Process**).

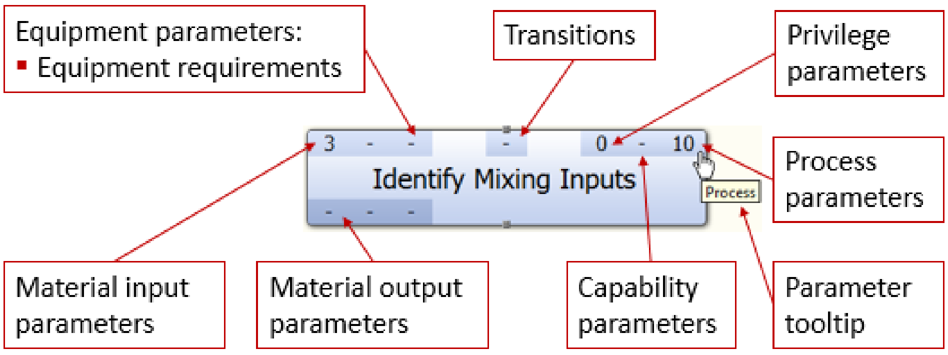


Figure 14: Recipe Designer - Batch: phase building block - header

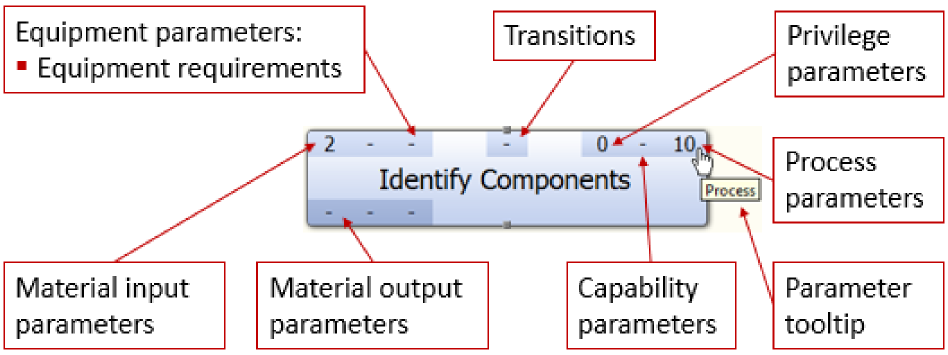


Figure 15: Recipe Designer - Device: phase building block - header

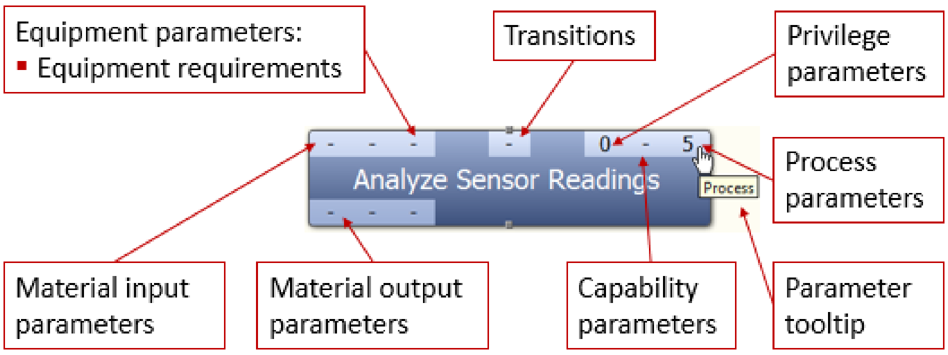


Figure 16: Workflow Designer: phase building block - header

Status Handling of Building Blocks

Building blocks only exist within the applications of Recipe and Workflow Designer so that their statuses are not affected by actions that take place in other applications of PharmaSuite. Thus all status changes have to be triggered explicitly from within either Recipe Designer or Workflow Designer.

Building Block Statuses

During their life-cycle, building blocks can assume the following statuses:

- **Draft**: The building block has been created and saved. It can be edited.
- **Approved**: The building block has been reviewed successfully. When an **Approved** building block is pulled as element into a recipe or workflow, its locked parameters and transitions are frozen and cannot be unlocked or edited. Parameters and transitions that are not locked can be modified. The SFC graphs on the structure levels below its header are read-only.
- **Archived**: The building block cannot be drawn as element into a recipe or workflow anymore. It is thus no longer listed in the Universe and is removed from the Setlist. However, it can still be opened.

Depending on the system's configuration, a status change can require a user to enter a single or double electronic signature as proof of authorization.

Changing the Status of Building Blocks

Building block status changes do not have a specific dialog but are integrated with an electronic signature.

To change the status of a building block, proceed as follows:

1. From the **File** menu, select **Change status of <building block identifier>** or click the **Change status** button (page 97) in the Status toolbar.

   TIP
   Make sure to have saved your building block, otherwise the **Change status of <object identifier>** function is disabled.

   The system displays the **Electronic Signature** dialog. The **Description** box displays the status change you are about to perform.

2. Execute the electronic signature (page 132).
   Setting a building block from **Draft** to **Approved**, causes the following behavior:
   - The entire building block becomes read-only, indicated by the (R) marker on its tab in the upper tab bar.
   - The background color of the Graph Window changes from yellow to gray.

- Transitions that are not specified at least by an identifier are locked automatically.
- When the approved building block is pulled as element into another building block, a recipe, or a workflow, its header displays an **Approved** marker (page 103) in the Explorer and on its component image in the Graph Window.
  Parameters and transitions that were locked on the **Approved** building block are frozen and cannot be unlocked or edited, while parameters and transitions that were not locked on the **Approved** building block are now again open for editing.

TIP
Please note that the behavior of the MFC graph of an **Approved** procedure building block corresponds to that of non-locked parameters. Once you pull the procedure building block into a master recipe its MFC graph is no longer read-only and can be modified.

Setting a building block from **Approved** to **Archived** causes the following behavior:

- Since an **Archived** building block is no longer available for use, it is removed from the Universe and from the Setlist.
- You can, however, still open an **Archived** building block, if you wish to copy or view it.

TIP

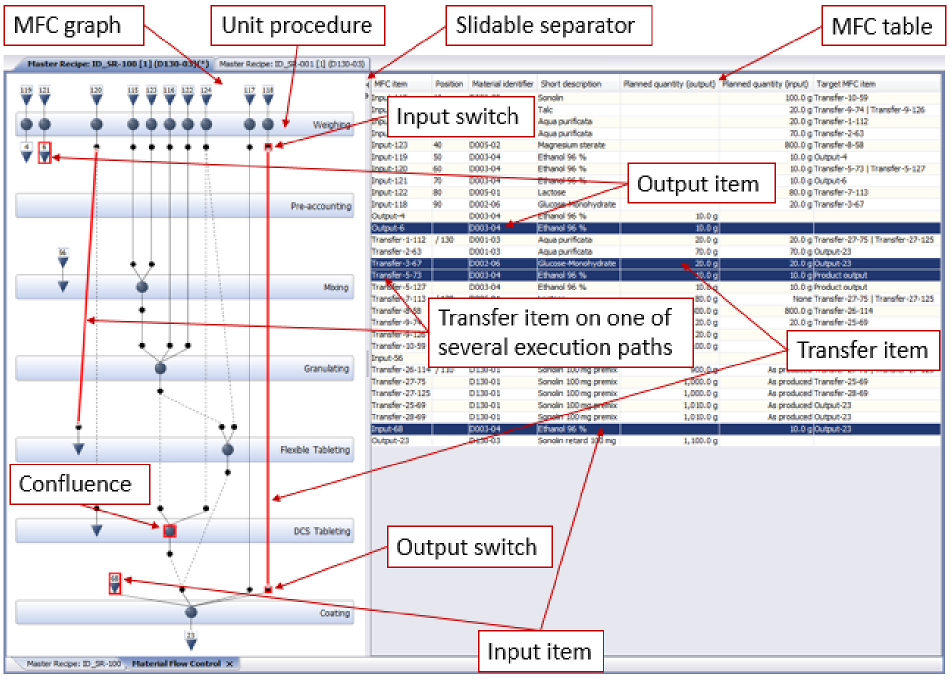
The status transitions performed on a building block and its approver or archiver, if applicable, are recorded in Recipe and Workflow Designer with the **Header** properties of the building block.

MFC Data of Building Blocks

Recipe and Workflow Designer determines the MFC-relevant data of a procedure building block from the attributes defined for its material parameters. The **Material Flow Control** tab consists of two panels:

- The graph panel (page 31) on the left shows the MFC graph that results from the material parameters defined on phase level. For building blocks to be used in batch recipes, you can edit the MFC graph and define the flow of materials by merging or splitting the MFC graph components.
- The table panel (page 35) on the right holds the non-editable table of MFC items. It lists all input, output, and transfer items and updates along with all changes made to the MFC graph.

The panels are divided by a slidable separator and can thus be freely resized, if necessary.



MFC item	Position	Material identifier	Short description	Planned quantity (output)	Planned quantity (input)	Target MFC item
[illegible]	[illegible]	[illegible]	Sonolin		100.0 g	Transfer-10-59
[illegible]	[illegible]	[illegible]	Talc		20.0 g	Transfer-9-74 | Transfer-9-126
[illegible]	[illegible]	[illegible]	Aqua purificata		20.0 g	Transfer-1-112
[illegible]	[illegible]	[illegible]	Aqua purificata		70.0 g	Transfer-2-63
Input-123	40	D005-02	Magnesium stearate		800.0 g	Transfer-8-58
Input-119	50	D003-04	Ethanol 96 %		10.0 g	Output-4
Input-120	60	D003-04	Ethanol 96 %		10.0 g	Transfer-5-73 | Transfer-5-127
Input-121	70	D003-04	Ethanol 96 %		10.0 g	Output-6
Input-122	80	D005-01	Lactose		80.0 g	Transfer-7-113
Input-118	90	D002-06	Glucose-Monohydrate		20.0 g	Transfer-3-67
Output-4		D003-04	Ethanol 96 %	10.0 g		
Output-6		D003-04	Ethanol 96 %	10.0 g		
Transfer-1-112	/ 130	D001-03	Aqua purificata	20.0 g	20.0 g	Transfer-27-75 | Transfer-27-125
Transfer-2-63		D001-03	Aqua purificata	70.0 g	70.0 g	Output-23
Transfer-3-67		D002-06	Glucose-Monohydrate	20.0 g	20.0 g	Output-23
Transfer-5-73		D003-04	Ethanol 96 %	10.0 g	10.0 g	Product output
Transfer-5-127		D003-04	Ethanol 96 %	10.0 g	10.0 g	Product output
Transfer-7-113	[illegible]	[illegible]	[illegible]	80.0 g	None	Transfer-27-75 | Transfer-27-125
Transfer-8-58	[illegible]	[illegible]	[illegible]	800.0 g	800.0 g	Transfer-26-114
Transfer-9-74	[illegible]	[illegible]	[illegible]	20.0 g	20.0 g	Transfer-25-69
Transfer-9-126	[illegible]	[illegible]	[illegible]	20.0 g	[illegible]	[illegible]
Transfer-10-59	[illegible]	[illegible]	[illegible]	100.0 g	[illegible]	[illegible]
Input-56	[illegible]	[illegible]	[illegible]		[illegible]	[illegible]
Transfer-26-114	/ 110	D130-01	Sonolin 100 mg premix	900.0 g	As produced	[illegible]
Transfer-27-75		D130-01	Sonolin 100 mg premix	1,000.0 g	As produced	Transfer-25-69
Transfer-27-125		D130-01	Sonolin 100 mg premix	1,000.0 g	As produced	Transfer-28-69
Transfer-25-69		D130-01	Sonolin 100 mg premix	1,010.0 g	As produced	Output-23
Transfer-28-69		D130-01	Sonolin 100 mg premix	1,010.0 g	As produced	Output-23
Input-68		D003-04	Ethanol 96 %		10.0 g	Output-23
Output-23		D130-03	Sonolin retard 100 mg	1,100.0 g		

Figure 17: Material Flow Control tab in the Graph Window

MFC Graph Components

The MFC graph displays how input materials enter and flow through the unit procedures, where they are transformed during the process until they become the final output. Apart from the changes you can perform actively to model the material flow, there are some external events that also update the MFC graph:

- changes made in the Parameter Panel, to the **MFC-relevant** or the **Position** attributes of a material parameter,
- structural changes to the SFC graph, such as the deletion of a phase, operation, or unit procedure with MFC-relevant material parameters.

The MFC graph consists of the following components:

Unit procedure

The individual unit procedures of the procedure are displayed as horizontal bars and hold all material nodes defined for their respective phases.

MFC input

An MFC input is displayed as triangle with an outgoing connector and represents a material input parameter configured with one of the procedure's phases. The number it shows corresponds to the numeric suffix of the MFC item identifier, which the system generates and assigns automatically.

When you hover over an MFC input, the system displays the following attributes as tooltip:

Material identifier, **Short description**, **Position** (if available), **Planned quantity** (if available) or **Planned quantity mode** (depending on its mode), and its path within the unit procedure (operation/phase).

Double-click an MFC input to open the Parameter Panel on the procedure level. The system shows the affected material input parameter as selected.

MFC output

An MFC output is displayed as triangle with an incoming connector and represents a material output parameter configured with one of the procedure's phases. The number it shows corresponds to the numeric suffix of the MFC item identifier, which the system generates and assigns automatically.

When you hover over an MFC output, the system displays the following attributes as tooltip:

Material identifier, **Short description**, **Position** (if available), **Planned quantity** (if available), and its path within the unit procedure (operation/phase).

Double-click an MFC output to open the Parameter Panel on the procedure level. The system shows the affected material output parameter as selected.

Confluence

A confluence is displayed as circle and represents the point in the material flow when an input is processed and thus converted into an output or merged with other inputs to form an output. A confluence can receive one or more inputs, but can only issue one output.

When you hover over a confluence that is connected to one or more inputs, its tooltip lists the attributes of all of its inputs.

When you hover over a confluence that is not connected to an input, its tooltip lists the attributes of its output.

Merged with final output
A confluence node is replaced by an output triangle when its input is merged directly with the final output instead of flowing into the unit procedure's confluence.
When you hover over a merged with final output node, its tooltip lists the attributes of its input.

Connector
Connector lines represent the material flow between inputs, outputs, their switches, and confluences. A connector between switches is an **MFC transfer** item. Connectors drawn as dotted lines represent optional flows, which result from selection branches between unit procedures.
Double-click an MFC transfer to open the Parameter Panel on the procedure level. The system shows the affected material input and output parameters as selected.
When you hover over a connector, the system displays a tooltip with the **Material identifier**, and **Short description**, and (if available) the **Position** of the material it represents.

Right-click anywhere on the background to open a shortcut menu with the following functions:

TIP

Which of the functions are enabled also depends on the graph components you have selected.

- Merge (CTRL+M)
  Available if you select two or more mergeable nodes. Merges the selected nodes to form a transfer or a confluence. After the merge operation, the resulting new component shows selected.
- Merge with ...
  Available if you select one node, which has one or more potential merge targets in the graph. The system lists the potential targets in a cascading sub-menu and indicates them in the graph by changing their node color to orange. When you hover over an item in the list of potential merge targets, the system changes its node color to red. Select one of the potential targets from the sub-menu to merge the nodes. After the merge operation, the resulting new component shows selected.

- Merge with final output (CTRL+O)
  Available if you select one or more confluence nodes of positions that are relevant to the quality of final output, but do not enter directly into the process. The confluence transforms into an output triangle to indicate that it is merged directly into the final output instead of the unit procedure's confluence. After the merge operation, the resulting new component shows selected.

- Merge automatically (CTRL+E)
  Merges all nodes whose merge targets can be determined unambiguously by the system. Nodes that have more than one potential merge target remain unmerged and need to be merged manually.

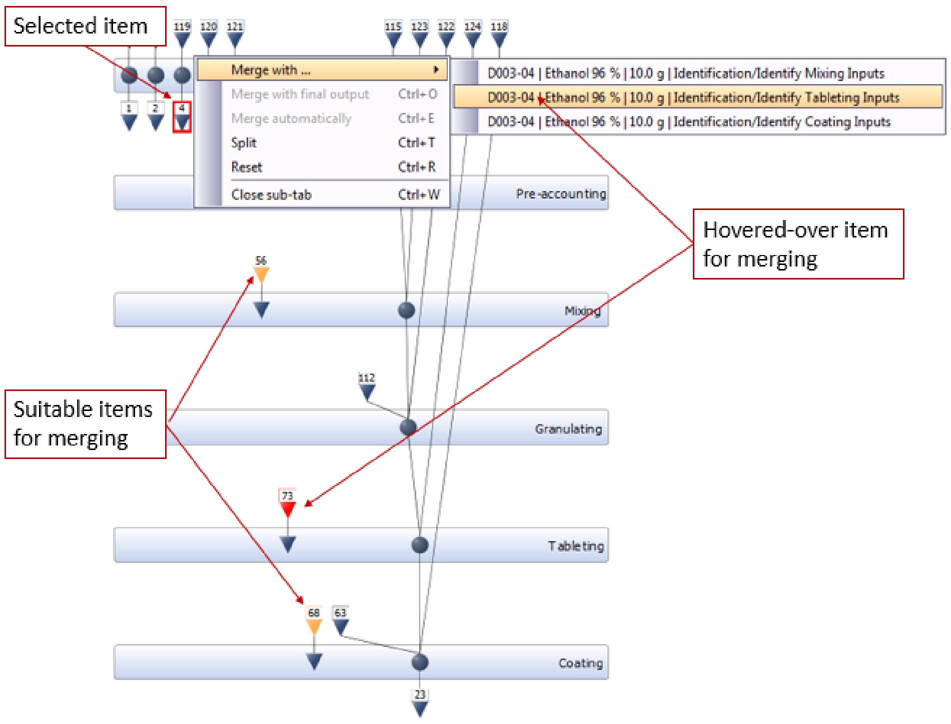

Selected item
Merge with ...
Merge with final output Ctrl+O
Merge automatically Ctrl+E
Split Ctrl+T
Reset Ctrl+R
Close sub-tab Ctrl+W
D003-04 | Ethanol 96 % | 10.0 g | Identification/Identify Mixing Inputs
D003-04 | Ethanol 96 % | 10.0 g | Identification/Identify Tableting Inputs
D003-04 | Ethanol 96 % | 10.0 g | Identification/Identify Coating Inputs
Hovered-over item for merging
Pre-accounting
Mixing
Granulating
Tableting
Coating
Suitable items for merging

Figure 18: Color support for merge operations

- Split (CTRL+T)
  Splits the selected component. After the split operation, the affected components show selected and thus can easily be remerged. The type of the component you select determines the behavior of the split function:
  - Splitting a connector dissolves the connection between the two nodes it connects. This can lead to the recreation of initial nodes that existed before any of the nodes were merged.
  - Splitting an input or an output node removes the input or output from its current confluence. This recreates its initial confluence node that has only one connection to the input or output node from which you triggered the split.

  - Splitting a confluence dissolves the entire confluence and recreates all initial confluence nodes of its inputs and its output.
- Reset (CTRL+R)
  Resets the MFC graph to its initial, unmerged state.
- Close sub-tab (CTRL+W)
  Closes the currently active tab of the lower tab bar.

MFC Table

The MFC table lists all MFC input, output, and transfer items of the procedure building block.
You cannot edit the data shown in the table, but selecting or multi-selecting listed items highlights the corresponding components in the MFC graph. Hovering over a table cell displays its content as tooltip. The table updates as a result of the following types of events:

- merge or split operations performed on the MFC graph,
- changes made in the Parameter Panel, to the **MFC-relevant** or the **Position** attributes of a material parameter,
- structural changes to the SFC graph, such as the deletion of a phase, operation, or unit procedure with MFC-relevant material parameters.

Double-click a list row in the MFC table to open the Parameter Panel on the procedure level. The system shows the affected material parameters as selected, which means that for MFC inputs and outputs, the system highlights just one material parameter, whereas for MFC transfers, two material parameters are highlighted.

The MFC table provides the following data for each of the listed items:

- **MFC item**
  Unique identifier of the item. It consists of the MFC type (input, output, transfer) and, for inputs and outputs, a count number. An MFC transfer item shows the numbers of the output and the input items from which it was merged.
  The identifiers are persistent within the MFC context, which means that when you unselect the **MFC-relevant** option of a material parameter in the Parameter Panel and then re-select it again, it receives a new identifier. The same applies when you delete an MFC-relevant material parameter and re-insert it.

> **TIP**
> Please note that the MFC item identifiers of building blocks are renumbered when you draw the building block into a master recipe or master workflow.

- **Position**
  You have to define a position number for each MFC-relevant input parameter that is also a direct MFC input item. Material input parameters that are merged into an MFC transfer item during the definition of MFC data do not need to have a position, unless they belong to an Inline Weighing operation. For material output parameters, positions are optional. If position numbers are defined, however, they must be unique throughout each execution path of a master recipe, master workflow, or building block. If you do not define the positions, the system indicates in the Messages window of Recipe Designer - Batch (see "Messages" in Vol. 2), Recipe Designer - Device (see "Messages" in Vol. 3), or Workflow Designer (see "Messages" in Vol. 4) which input parameters lack their position numbers.
  The positions are presented in any list of material inputs or outputs, for example during material identification on the shop floor, and are also used as default sorting criterion.

  TIP
  The processing sequence of materials for Dispense orders is controlled by their weighing material types as the system does not allow a compensator item to be processed unless its active substance material has been completed. Similarly, the processing sequence in Inline Weighing can be controlled by a corresponding check. For this reason the positions of the materials should reflect their processing sequence, so that an active material has a lower position number and thus be listed before its compensator, which again should precede its filler.

  Defined with the material parameter.

- **Material identifier**
  Drawn from the material parameter.

- **Short description**
  Drawn from the material parameter.

- **Planned quantity (output)**
  For output or transfer items of material parameters with **As defined** as planned quantity mode, it shows the planned quantity as defined with the material parameter.
  For output or transfer items of material parameters with other planned quantity modes it shows the respective mode, either **None** or **As produced**.

- **Planned quantity (input)**
  For input or transfer items of material parameters with **As defined** as planned quantity mode, it shows the planned quantity as defined with the material parameter.
  For input or transfer items of material parameters with other planned quantity modes it shows the respective mode, either **None** or **As produced**.

- **Target MFC item**
  Indicates the transfer or direct output item into which the selected item flows. In a valid MFC graph, all items flow either into a transfer or into the final direct outputs. If there are several potential targets, since the recipe contains selection branches between unit procedures that cause several executions paths, all potential targets are listed.

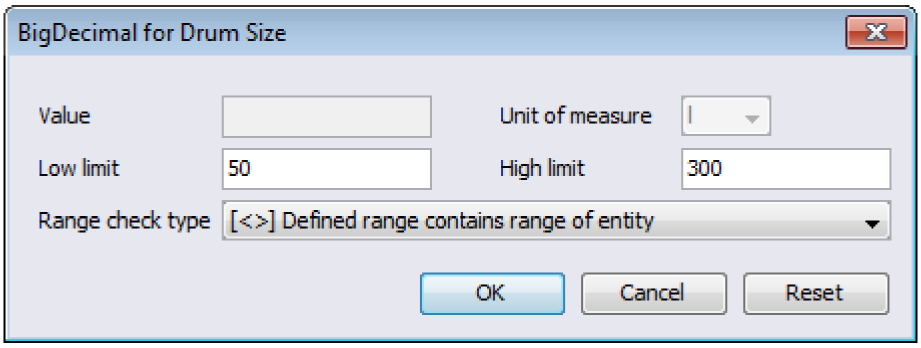
The list of MFC items is sorted first by type, then by position, and finally by MFC item identifier.

Editors

Recipe and Workflow Designer provides specific editors to facilitate easy editing of the values in the Parameter Panel's **Results** list, the expressions of transition conditions, the properties listed in a property window, or date input boxes on **Change status** dialogs.

BigDecimal Editor

The BigDecimal editor supports you with entering a number or a range between two numbers. It allows to define a unit of measure for the value or range.


BigDecimal for Drum Size

Value:
Unit of measure: l
Low limit: 50
High limit: 300
Range check type: [<>] Defined range contains range of entity

Figure 19: BigDecimal editor for property requirement

- When you define a property requirement, you can either give an exact value or a range including its range check type. The unit of measure is specified with the property type and cannot be changed.
  You can leave non-required boxes blank.

TIP

When defining a range you can set the range's low and high limits and specify how the defined range is matched against the actual value or range provided by an equipment entity identified during execution:

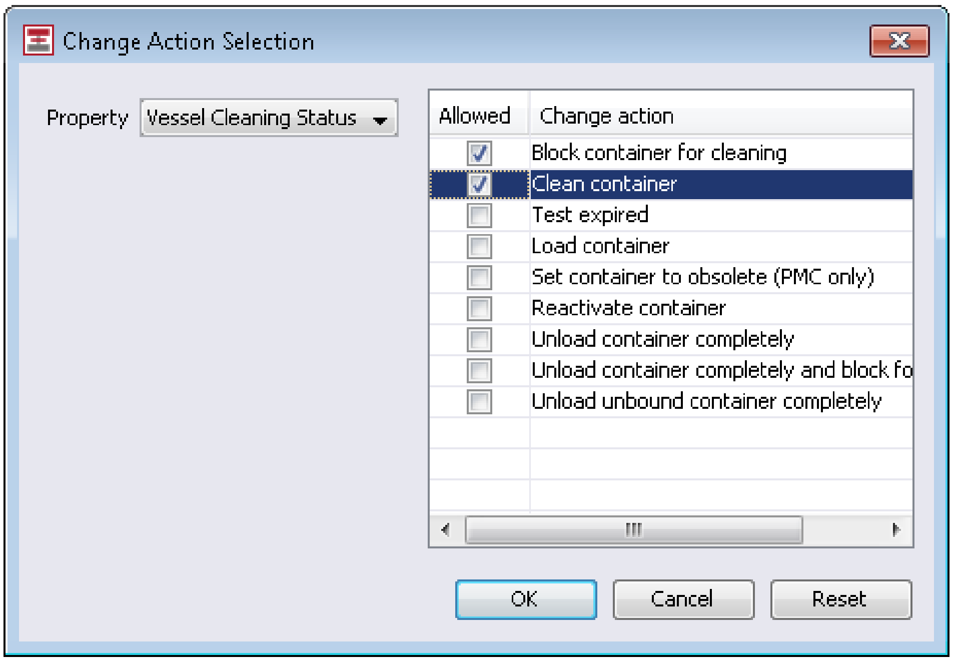
- [<>] Defined range contains range of entity
  The option provides for equipment requirements where the actual value or range of an equipment entity must lie within the defined limits of minimum and maximum values.
  This check type would be suitable to cover a situation when you have an equipment unit that can be run with vessels of various sizes. Your requirement would, for example, specify a range between 300 l and 500 l and the volume of the identified vessel would have to be within this range. So vessels with a volume of 350 l or 500 l could be identified successfully, whereas a vessel with a volume of 550 l would cause an exception.
- <[]> Defined range is contained in range of entity
  The option provides for equipment requirements where the actual range of an equipment entity must at least be able to cover the defined range.
  This check type would be suitable to cover a situation when you have a mixer that can run with a wide range of speeds. Your requirement would, for example, specify a range between 300 rpm and 600 rpm since the volatile material to be processed must be treated with mixing cycles at 350, 450, and 550 rpm but not at a lower speed to prevent lumping and not at a higher speed to prevent overheating. So a mixer capable of running between 50 rpm and 2000 rpm would be identified successfully, whereas a mixer that runs at ranges between 200 rpm and 550 rpm would cause an exception.

- To save your data and close the BigDecimal editor, click the **OK** button.
- To close the BigDecimal editor without saving your changes, click the **Cancel** button.
- To reset your value or range to their last saved form, click the **Reset** button.

Change Action Selection Editor

The Change Action Selection editor supports you with

- selecting a FlexibleStateModel property with its bundle of possible change actions an operator can perform during execution,
- defining which of the possible change actions of the selected property are available to an operator during execution.



Change Action Selection

Property: Vessel Cleaning Status

Allowed	Change action
✓	Block container for cleaning
✓	Clean container
☐	Test expired
☐	Load container
☐	Set container to obsolete (PMC only)
☐	Reactivate container
☐	Unload container completely
☐	Unload container completely and block fo
☐	Unload unbound container completely

Figure 20: Change Action Selection editor

- To save your options and close the Change Action Selection editor, click the **OK** button.
- To close the Change Action Selection editor without saving your changes, click the **Cancel** button.
- To reset your selected change actions to their last saved form, click the **Reset** button.

Date/Time Picker Editor

The Date/Time Picker editor supports you with entering a date with or without a time. You can either type the date or time directly in the input box or open the calendar control to pick it. Depending on the data expected by the input box you are about to fill, the calendar either only provides date picking or it holds an additional spin control to pick a time as well. The calendar opens with the date (and time) that is currently displayed in the input box. It highlights the selected date in orange and displays the current date with a red border. When you have selected a date (and time), the calendar control closes automatically.

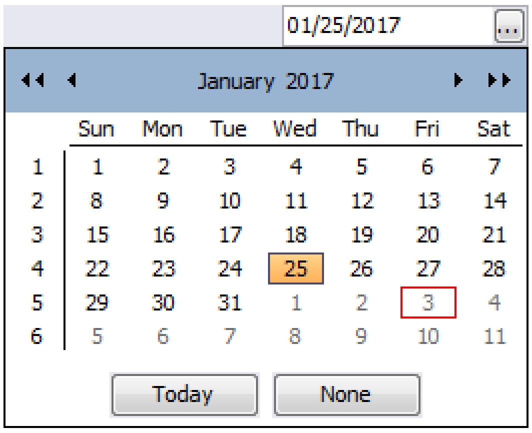
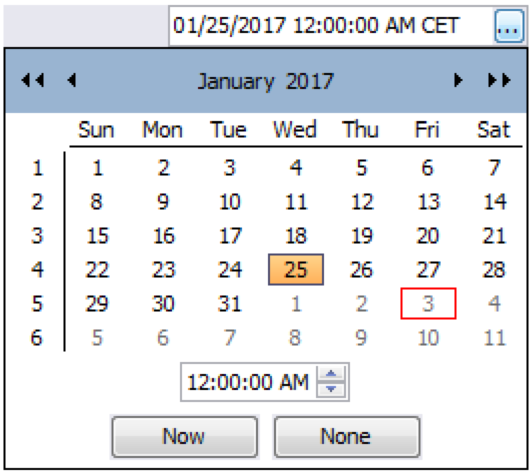

01/25/2017

Figure 21: Date/Time Picker editor


01/25/2017 12:00:00 AM CET

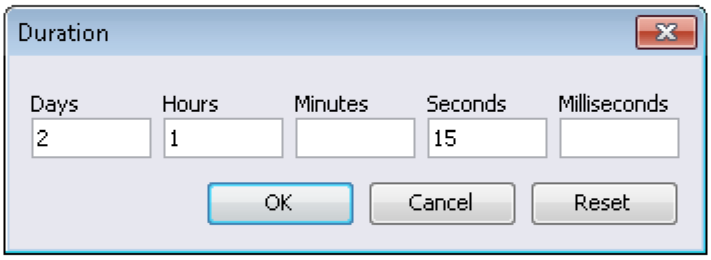
12:00:00 AM

Figure 22: Date/Time Picker editor

- To navigate the calendar to other months or years, use the controls in the calendar header.
  - Months: SINGLE ARROW keys or option list that appears when you click the current month name.
  - Years: DOUBLE ARROW keys or spin control that appears when you click the current year.
- To edit a time, click the position (hours, minutes, seconds) in the time text box you wish to change. Then use the spin control or the UP and DOWN ARROW keys to increase or decrease the number.
- In the Date Picker, to select the current date, click the **Today** button.
- In the Date and Time Picker, to select the current date and time, click the **Now** button.
- To clear the date and or time from the input box, click the **None** button.
- To close the calendar control without changing the date or time, click anywhere in the dialog outside the calendar control or press the ESC key.

Duration Editor

The Duration editor supports you with entering a duration with days as largest and milliseconds as smallest unit.



Days	Hours	Minutes	Seconds	Milliseconds
2	1		15	

Figure 23: Duration editor

- Fill the text boxes as required. You can leave non-required boxes blank. To save your value and close the Duration editor, click the **OK** button or press the ENTER key.

TIP
Please note that the system does not allow you to type values for the various time units that exceed the maximum number of each time unit and would move it to the next higher unit. Thus, for hours the maximum is 23, for minutes and seconds it is 59, and for milliseconds it is 999.

- To close the Duration editor without saving your changes, click the **Cancel** button or press the ESC key.
- To reset your duration values to their last saved form, click the **Reset** button.

Expression Editor

The Expression editor supports you with defining

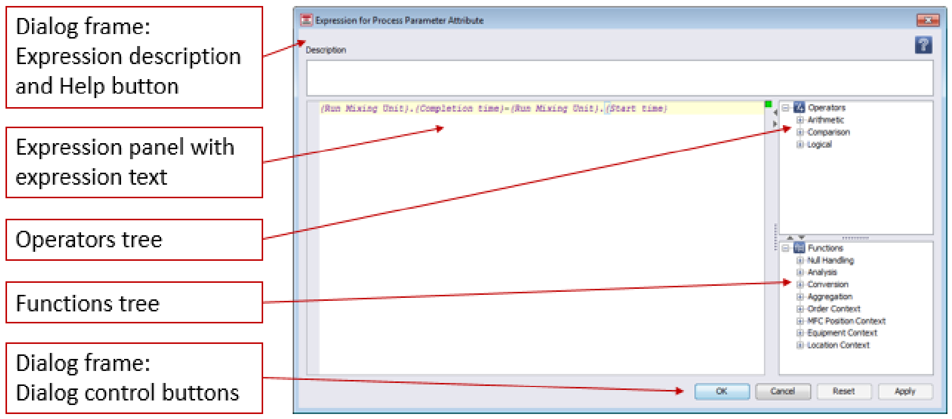
- expressions that provide inputs to process parameter attributes in Recipe Designer - Batch (see "Process Parameters" in Vol. 2), Recipe Designer - Device (see "Process Parameters" in Vol. 3), or Workflow Designer (see "Process Parameters" in Vol. 4), in order to model the information flow throughout your master recipe, master workflow, or building block
- specific transition conditions, in Recipe Designer - Batch (see "Transition" in Vol. 2), Recipe Designer - Device (see "Transition" in Vol. 3), or Workflow Designer (see "Transition" in Vol. 4), which are required if your graph contains decision points such as selection branches or loops
- expressions that define complex rules for equipment requirements in Recipe Designer - Batch (see "Equipment Requirement Parameters" in Vol. 2), Recipe Designer - Device (see "Equipment Requirement Parameters" in Vol. 3), or Workflow Designer (see "Equipment Requirement Parameters" in Vol. 4).

Expressions can calculate values, reference outputs from previously processed phases, or perform a combination thereof.

Expression Editor for Process Parameter Attributes

The Expression editor for process parameter attributes consists of

- the dialog frame for expression meta data
- the expression panel (page 47) that holds the actual text of an expression
- two resizable tree panels that list available operators (page 60) and functions (page 67).



```
(Run Mixing Unit).(Completion time)-(Run Mixing Unit).(Start time)
```

Figure 24: Expression editor for process parameter attributes

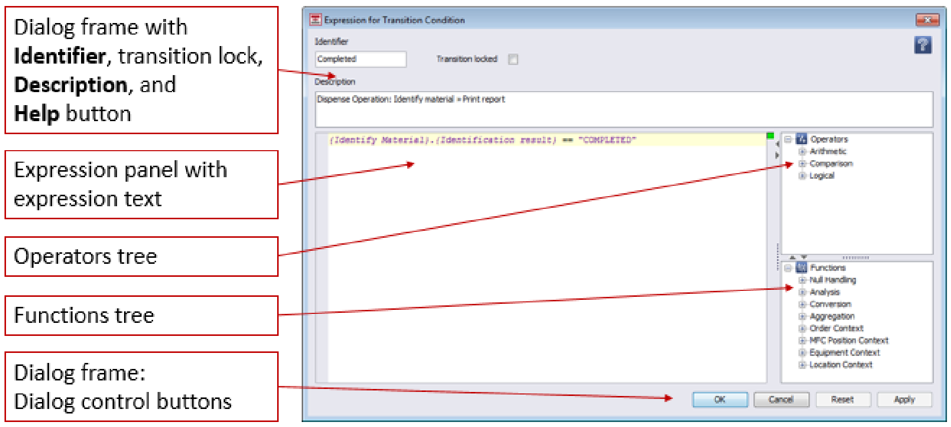
Above the expression panel, the frame provides a text box where you can type a textual description of the expression. The system shows the description in the respective attribute cell on the Parameter Panel.
The **Help** button opens the context-sensitive help system.
Below the expression panel, the dialog control buttons provide the following actions:

- To save your expression and close the Expression editor, click the **OK** button.
- To close the Expression editor without saving your changes, click the **Cancel** button.
- To reset your expression to its last saved form, click the **Reset** button.
- To save your expression without closing the Expression editor, click the **Apply** button.

Expression Editor for Transition Conditions

The Expression editor for transition conditions consists of

- the dialog frame for transition and expression meta data
- the expression panel (page 47) that holds the actual text of a condition expression
- two resizable tree panels that list available operators (page 60) and functions (page 67).



Identifier: Completed

Description: Dispense Operation: Identify material » Print report

```
{Identify Material}.{Identification result} == "COMPLETED"
```

Figure 25: Expression editor for transition conditions

Above the expression panel, the frame provides the following elements:

- The **Identifier** box is preset with a system-generated suggestion. You can edit the identifier or type a completely different one. The maximum number of characters for a transition identifier is 20 and it must be unique within its graph. Transition identifiers are only mandatory for transitions with specific conditions or descriptions. For transitions in Recipe Designer - Batch (see "Transition" in Vol. 2), Recipe Designer - Device (see "Transition" in Vol. 3), or Workflow Designer (see "Transition" in Vol. 4) that only hold the default condition, they are optional.

- The **Transition locked** option allows you to lock a transition for editing, thus preventing any of its properties or its condition from being changed.

  TIP
  When a locked transition is located in an **Approved** building block, which is then drawn as element into a higher-level structure (recipe, workflow, or building block), the transition is locked permanently and cannot be unlocked. In this case, the **Transition locked** option displays the **Frozen** icon (page 104) instead of a checkbox.

- In the **Description** box, you can type a textual description of the transition condition.

  TIP
  The system shows a transition's identifier, its locking state (**Unlocked**, **Locked**, **Frozen**), and its description as tooltip when you hover over it in the Graph Window.

- The **Help** button opens the context-sensitive help system.

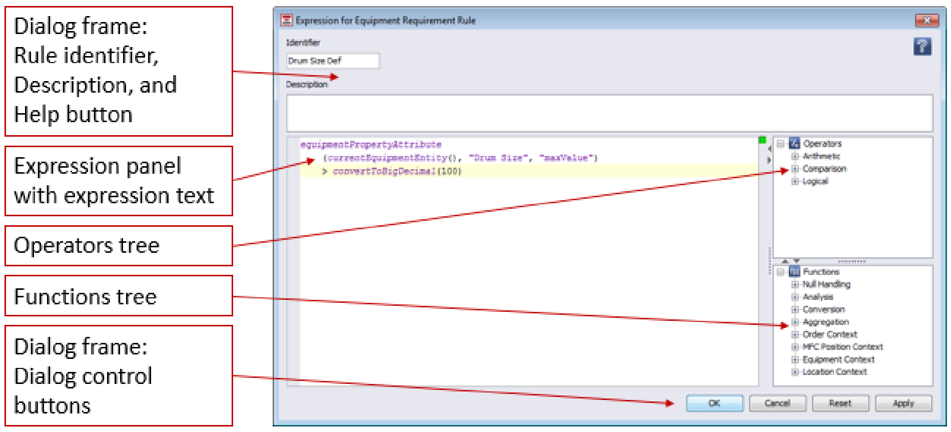
Below the expression panel, the dialog control buttons provide the following actions:

- To save your expression and close the Expression editor, click the **OK** button.
- To close the Expression editor without saving your changes, click the **Cancel** button.
- To reset your expression to its last saved form, click the **Reset** button.
- To save your expression without closing the Expression editor, click the **Apply** button.

Expression Editor for Equipment Requirement Rules

The Expression editor for equipment requirement rules consists of

- the dialog frame for equipment requirement and expression meta data
- the expression panel (page 47) that holds the actual text of a condition expression
- two resizable tree panels that list available operators (page 60) and functions (page 67).



```
equipmentPropertyAttribute
  (currentEquipmentEntity(), "Drum Size", "maxValue")
  > convertToBigDecimal(100)
```

Figure 26: Expression editor for equipment requirement rules

Above the expression panel, the frame provides the following elements:

- The **Identifier** box is preset with a system-generated suggestion. You can edit the identifier or type a completely different one. Rule identifiers must be unique per equipment requirement.
- In the **Description** box, you can type a textual description of the rule.
- The **Help** button opens the context-sensitive help system.

Below the expression panel, the dialog control buttons provide the following actions:

- To save your expression and close the Expression editor, click the **OK** button.
- To close the Expression editor without saving your changes, click the **Cancel** button.
- To reset your expression to its last saved form, click the **Reset** button.
- To save your expression without closing the Expression editor, click the **Apply** button.

Expression Panel

The expression panel represents the work area of the Expression editor and is basically an enhanced text editor that supports editing of expressions and references with auto-completion and concurrent syntax and semantics validation.

TIP

For specific purposes, such as migration or the implementation of highly complex expressions that are beyond the Expression editor's capabilities, there is a Pnuts mode available.

Type **@Pnuts** as first line of your expression to set the editor to this mode.

Please be aware that the Pnuts mode

- is an exclusive mode that applies to the entire expression. So once you have set the editor to Pnuts mode, expressions written in the regular Expression editor mode will not be interpreted. The Operators and Functions trees are grayed out.
- provides only a restricted set of syntax or semantics checks.
- expects you to explicitly include a default condition with *transitionUtil* when used for transition conditions.

Generally, we do not recommend to use Pnuts in expressions.

IMPORTANT

A thread pool manages the execution of procedures and unit procedures of all orders processed on the EBR server. Hence, the number of orders processed at the same time is not restricted by the maximum of threads on the EBR server.
As the threads of the thread pool can be used by several procedures and unit procedures, a transition, e.g. between two operations, must not block the processing of the corresponding thread, since thread pooling may block the execution of further orders by the EBR server.
Blocking transitions can easily be introduced accidentally by using Pnuts code in the expression editor of Recipe and Workflow Designer when it includes blocking code, such as an infinite wait condition for some event. Therefore, we strongly recommend to refrain from using Pnuts code. Instead, use PharmaSuite functions only, since they return their result in a finite time.
However, blocking transitions can also occur due to the fact that transitions are only evaluated exactly once after each phase completion (within an operation run). This means that you must not create conditions that wait for a certain time or an external event to take place. What could happen in this situation is that at evaluation time the condition does not apply and thus blocks the transition, which then remains blocked as there is no further evaluation of the condition. If you wish to model a delay (directly or by waiting for an external event), we recommend to use a dedicated phase that periodically checks for the expected change in the condition.

To format your expressions for better readability you can use blanks, tabs, and line breaks. Your formatting is retained when you save an expression.
The editor supports text manipulation by keyboard with common shortcuts (page 118).

TIP

Please note the following notation conventions for numbers in expressions:

- "." is the only permitted decimal separator as in **678.934**
- There are no digit grouping symbols, as in **69352990478.749**

You can build expressions by using the following elements:

CONSTANTS (PAGE 55)

TIP

Please note that some phases may provide outputs that have other data types than those listed above, e.g. Part (material). In parameter attributes, those outputs can only be referenced and thus passed on to a later phase, but not used with operators or functions for further evaluations. The same restriction applies to transition conditions, however, with the exception that you can evaluate if an output is **undefined** and thus use the **isNull**, **nvl**, and **coalesce** functions with suitable logical operators.

OPERATORS (PAGE 60)

FUNCTIONS (PAGE 67)

BRACKETS

- Only round brackets (parentheses) are allowed in expressions as well as the identifiers of unit procedures, operations, phases, and outputs. You can use them to override or simply to clarify the precedence of the operators used in an expression.
- Square brackets are not allowed in expressions, but within phase and output identifiers.
- Pointy brackets cannot be used in expressions, since they would be interpreted as inequality signs (greater than, smaller than). They are allowed within phase and output identifiers.
- Curly brackets (braces) are always interpreted as start or end delimiter of a reference's phase or output identifiers and are therefore not allowed within the identifiers.

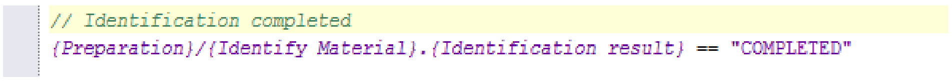
TIP

Please note that in Recipe and Workflow Designer you can only influence the identifiers of phases. Output identifiers come with the phases and are named by the phase designers during building block development.

COMMENTS

You can add comments to your expressions, which the system will disregard when it processes the expression:

- Use // to introduce a single-line comment.
- Use **/* ... */** to enclose a multi-line comment.



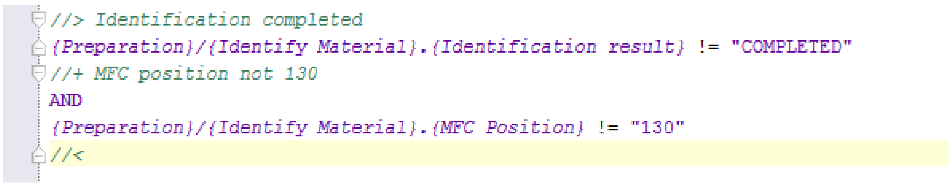
```
// Identification completed
{Preparation}/{Identify Material}.{Identification result} == "COMPLETED"
```

Figure 27: Expression with comment

SECTIONS

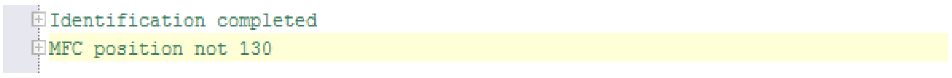
To add further structuring to your expressions, you can convert comments into sections:

- Use //> to start a section.
- Use //+ to continue a section.
- Use //< to close a section.



```
//> Identification completed
{Preparation}/{Identify Material}.{Identification result} != "COMPLETED"
//+ MFC position not 130
AND
{Preparation}/{Identify Material}.{MFC Position} != "130"
//<
```

Figure 28: Expression with sections (all open)



```
Identification completed
MFC position not 130
```

Figure 29: Expression with sections (all closed)

REFERENCES TO OUTPUTS FROM PREVIOUSLY PROCESSED PHASES

- Only if a phase has been processed does it provide an output that can be fed into another phase as attribute value. For this reason, you can never reference an output of a phase that is a strict successor of the phase in which you try to use the output.
- Branches and loops, however, require special notice in this context, since they are only potentially passed through and/or completed during processing, so their outputs are not reliably available. Thus you can reference any such potentially available outputs, but need to be aware of the fact that the provided value may be **Undefined** so that the phase to which you are feeding the output must be able to deal with such an **Undefined** input value.

The basic rule for outputs from operations with internal or external loops is that it is always and only the data from the last or current run of a loop that is considered for feeding into a successor component. Depending on the structure level on which a loop is located, however, the system takes different views as to what constitutes a last run.
So when an operation contains an internal loop around a selection branch, it provides the value recorded at the last time the specific path of the branch was passed through. An **Undefined** value can only occur if the path has not yet been passed through.

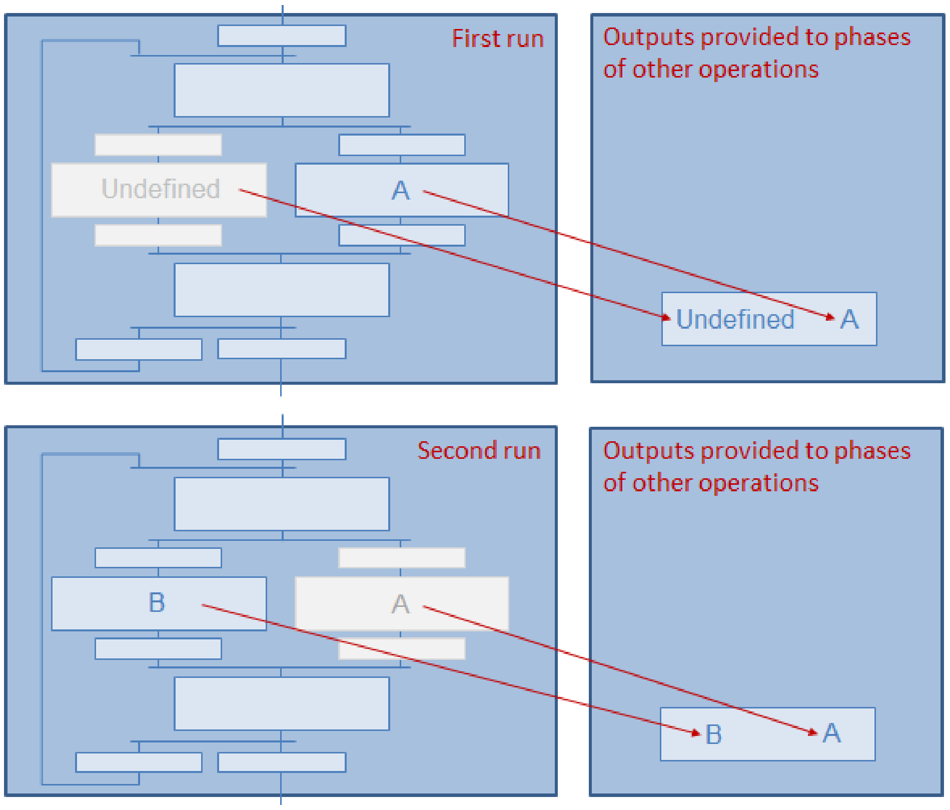

First run
Undefined
A
Outputs provided to phases of other operations
Undefined
A
Second run
B
A
Outputs provided to phases of other operations
B
A

Figure 30: Outputs from operation-internal loops

When an operation contains a selection branch but is also part of an external loop, so that the complete operation can be passed through several times, the system only considers the path or paths that are actually being taken during processing. Thus, only those outputs provide data that have been passed during the last or current run while all other outputs provide **Undefined** as values.

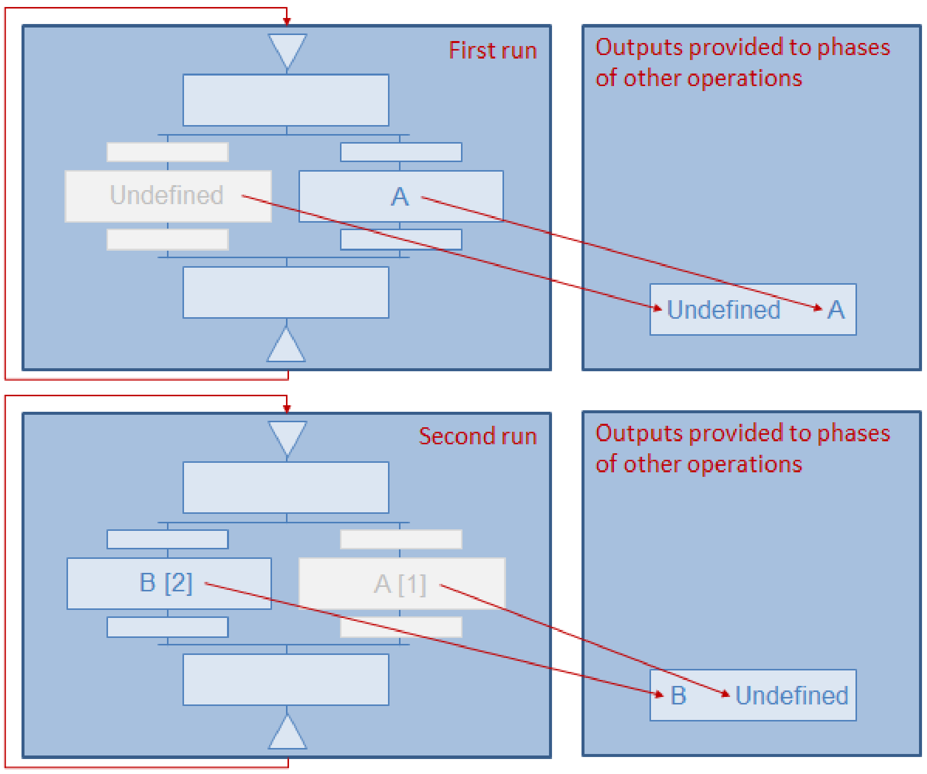

First run
Undefined
A
Outputs provided to phases of other operations
Undefined
A
Second run
B [2]
A [1]
Outputs provided to phases of other operations
B
Undefined

Figure 31: Outputs from operation-external loops

TIP

Please note that event-triggered operations are considered to have implicit external loops. The system uses the count of the individual runs to determine the last or current run. So it is the run with the highest count that provides its output value to successor operations.

- An output reference consists of a phase path and an output identifier. Depending on where in the graph structure the referenced phase is located, the phase path can contain up to three path elements, namely a unit procedure identifier, an operation identifier, and a phase identifier. The identifiers are enclosed in curly brackets (braces) and separated by slashes. A phase path must not be longer than necessary, so when you reference the output of a phase located in the same operation you only need the phase identifier as phase path. The Expression editor marks redundant path elements as errors.
  The output identifier is also enclosed in in curly brackets (braces), but uses a period to separate it from the phase path.
  Examples:
  - reference to an output of a local phase, i.e. a phase located in the same operation
    {Run Mixing Unit}.{Completion time}
  - reference to an output of a phase located in a different unit procedure
    {Pre-accounting}/{Dispensed Material Accounting}/{Account Materials}.{Completion time}

  TIP
  Please note that the outputs of phases that are located in a Dispense operation cannot be referenced in this manner.

COMMANDS

Commands are only suited for use in conditional equipment requirement rules. They represent the instructions that are necessary for defining the two sections of a conditional rule.

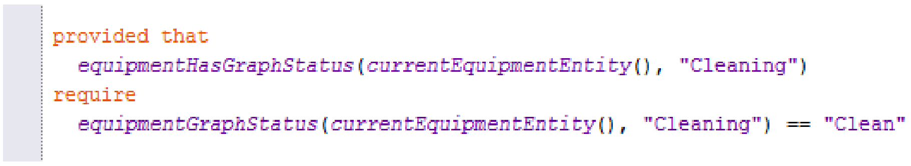
TIP
Please note that a conditional rule must contain both commands.

The following commands are available:

- `provided that`
  introduces the first section of the conditional rule. It defines the condition that an entity must fulfill so that the system evaluates the expression defined in the second section.
  Example:
  provided that
      equipmentHasGraph(currentEquipmentEntity(), "Cleaning")
  The condition defines that the following requirement only applies if the "Cleaning" equipment graph is assigned to the identified equipment. If not, the requirement is ignored.

TIP
If the equipment entity returned by the currentEquipmentEntity function is the parent of a group, the condition is evaluated for all members of the group. The following requirement only applies to those group members that fulfill the condition.

- `require`
  introduces the second section of the conditional rule. It defines the actual requirement which the entity must fulfill.
  Example:
  require
      equipmentGraphStatus(currentEquipmentEntity(), "Cleaning") == "Clean"
  The requirement states that the status of the "Cleaning" graph of the identified equipment must be "Clean".



```
provided that
  equipmentHasGraphStatus(currentEquipmentEntity(), "Cleaning")
require
  equipmentGraphStatus(currentEquipmentEntity(), "Cleaning") == "Clean"
```

Figure 32: Expression with conditional rule

For inserting expression elements, you can either type the texts yourself or make use of the intelligent auto-completion feature provided by the editor:

1. When you start typing, the system displays all available, suitable, and matching elements in a drop-down option list. The search is not case-sensitive and the available options are sorted by their element types.
2. To open the full list of available expression elements, place your cursor in the empty expression panel and press the CTRL+SPACEBAR keyboard shortcut.
3. Click the element you wish to insert or use the ARROW UP or ARROW DOWN keys to select it and then press the ENTER key to insert it.
   To close the drop-down list without inserting an option, click anywhere outside of the list or press the ESC key.
4. You can edit your expression manually at all times. Click anywhere in the expression panel to place your cursor and then type or delete characters as required.

TIPS
The auto-completion feature displays a warning marker to the left of all operations and phases whose outputs may return **Undefined** because they are not reliably available, such as outputs that originate from a parallel structure element.
Output identifiers are listed along with their respective data types. This information can help you to avoid using non-matching data types accidentally.

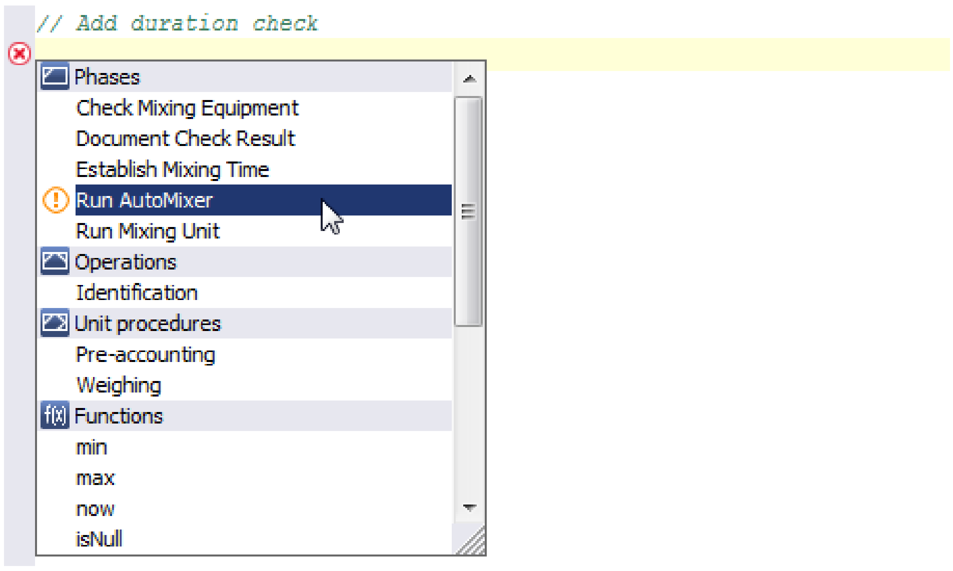

```
// Add duration check
```

Figure 33: Expression panel with comment and auto-completion

The editor validates your input as you type and marks errors with a wavy, red underline as soon as they occur. Additionally, the editor displays an error icon to the left of the affected line.
Another indicator is the global error status marker in the top-right corner of the expression panel that turns red and expands downward to mark individual error positions if the expression contains errors.

To see a more explicit error message, hover your mouse cursor over any of the error indicators. The editor will then display the error message in a tooltip.

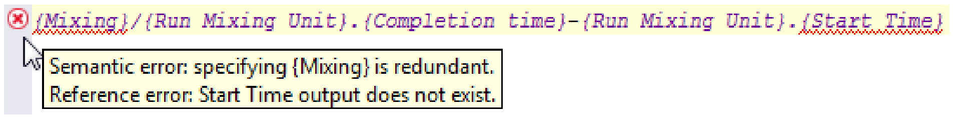

```
{Mixing}/{Run Mixing Unit}.{Completion time}-{Run Mixing Unit}.{Start Time}
```
Semantic error: specifying {Mixing} is redundant.
Reference error: Start Time output does not exist.

Figure 34: Expression panel with errors and message tooltip

Constants

Constants are differentiated by their data types:

- Long
  used for integral numbers:
  12345

- Float
  floating point number, can be used for fractional numbers, but needs a specific markup (**Float**) to be interpreted correctly by the system:
  123.45Float

- BigDecimal
  floating point number that allows calculating with greater precision than Float. By default, the system interprets fractional numbers as BigDecimal, so a markup (**BDecimal**) is only necessary when you use an integral number, but want it to be treated as BigDecimal:
  123.45 or **123BDecimal** or **12.3e42** (where **e** represents the exponential notation, and stands for "times ten raised to the power of").

  TIP
  BigDecimal values support up to 15 integral digits and are rounded to 9 fractional digits when stored to the database.

- Boolean
  with the values **true** and **false**.

- String
  used for displaying any sequence of characters, such as
  "**Read instruction text:**"

  TIP
  Please note that a string constant must be enclosed in quotes.

- IMESS88Equipment
  used for transporting an entire equipment data object with all its information and references to provide as process input.

- PhaseDataReference
  used for transporting all phase data collected during execution to provide as process input.

- Timestamp
  used for displaying dates and times and for time-related calculations.
  Its full format is **dd-MMM-yyyy@H:M:s.S**, but you can leave out the either entire time section, or the seconds/milliseconds section, or just the milliseconds:
  - 11-Jan-2013@12:13:54.232
  - 3-Mar-2015
  - 02-Sep-2013@12:13
  - 29-May-2012@12:13:17

  TIP
  Please note that timestamp constants always refer to UTC time (Coordinated Universal Time, which is the equivalent to GMT), to make timestamp constants independent of time zones.

- Duration
  used for displaying time spans and for time-related calculations.
  It supports the following time units: days (**d**), hours (**h**), minutes (**min**), seconds (**s**), and milliseconds (**ms**).
  The following specifics apply to using durations in your expressions:
  - Make sure to write the duration as consecutive string of characters without blanks, since otherwise, the system will not be able to interpret it correctly.
  - You can leave out any units you do not need for specifying your duration, so instead of **0d6h0min30s0ms** you can write **6h30s**.
  - You have to observe the order of the units, which means, for instance, that the system cannot interpret a duration that starts with seconds, followed by minutes and hours.
  - The values you specify for the various time units must not exceed the maximum number of each time unit that would move it to the next higher unit. Thus, for hours (**h**) the maximum is 23, for minutes (**min**) and seconds (**s)** it is 59, and for milliseconds (**ms**) it is 999.
    This restriction, however, does not apply to the highest unit you specify, which means that the system allows a duration such as **49h30min15s**, whereas typing **48h89min75s** in this context would not be valid.
- MeasuredValue
  used for displaying numeric values qualified by a unit of measure.

  TIP
  MeasuredValue values support up to 15 integral digits and are rounded to 9 fractional digits when stored to the database.

  The following specifics apply to using measured values in your expressions:
  - Make sure to write a measured value as string of characters without blanks as in
    280kg or **29gal**
  - If a unit of measure itself contains a blank, you have to replace it with an underscore "_" as in
    125fl_oz
  - You cannot use the MeasuredValue data type for time-related constants, use the Duration data type instead.
  - For calculations that have a measured value as result, the system returns the result in the unit of measure of the operand that is expected to provide greater accuracy, either because of the unit's finer granularity or due to a greater precision on account of the number of fractional digits:

1kg + 100g = 1100g
1.000kg + 100g = 1.100kg

- The precision of plus and minus operations between measured values depends on the number of fractional digits given with the operand that has the unit of measure with the greater accuracy. Thus, the result of **1kg + 100.00g = 1100.00g** whereas **1.00kg + 100g = 1100g**.
- The precision of division operations between a measured value and a number or another measured value is established by adding the second operand's (divisor) base-10 logarithm to the number of fractional digits of the first operand (dividend):
  <number of fractional digits of dividend> + log(<divisor>).
  This means that the system calculates **3 + log(100) = 5** to determine the number of fractional digits for **10.000g / 100**, and consequently returns **0.10000g**.
  The same calculation is used to establish the number of fractional digits in **10.000g / 100g = 0.10000**.
- Similarly, the precision of multiplication operations between a measured value and a number is established by subtracting the number's base-10 logarithm from the number of fractional digits of the measured value:
  <number of fractional digits of measured value> + log(<number>).
  So the system calculates **3 - log(100) = 1** to determine the number of fractional digits for **10.000g * 100 and returns 1000.0g**.

TIP
Please note that the system generally rounds logarithms up, so that **log(100) = 2**, whereas **log(101) = 3**.

- The calculation sequence of operators with the same precedence (page 60) is strictly left to right in pairs. Thus **a+b+c+d** is interpreted as **((a+b)+c)+d**.

TIP

It is generally recommended to avoid mixing metric and US customary units of measure in calculations, since the results of these calculation depend on the units of measure, the precision of the individual operands, and their sequence.

Specific to these types of implicit conversion calculations is that after the system has determined the result unit of measure (that of the operand with the greater accuracy) it converts the other operand to match the result unit of measure. For converting, it uses the system-defined conversion relation. After having converted the operand, the system rounds the converted operand to the same precision it had in its initial unit of measure.

Mixed unit examples:

1kg +1000g + 1000000mg + 1.000lb is interpreted as
((1kg +1000g) + 1000000mg) + 1.000lb and returns **3453600mg**.

- In this case, the first two calculations increase the precision to milligrams and return **3000000mg**.
- The third calculation thus is **3000000mg + 1.000lb** and the system performs the following steps:
  1. It determines the result unit of measure:
     mg (has the finer granularity).
  2. It performs the conversion, using the system-defined relation **1lb = 0.453592370kg**. This means a twofold conversion needs to take place: **lb** » **kg** and **kg** » **mg** (conversion relation **1mg = 0.000001000kg**)
     1.000 * 0.453592370 / 0.000001000 = 453592.370000000000, which is then rounded to **453600** to have the same (4-digit) precision as the **lb** value. So the result of the conversion is **453600mg**.
  3. It performs the calculation:
     3000000mg + 453600mg = 3453600mg
     No further changes to the precision are required.

1.000lb + 1kg +1000g + 1000000mg is interpreted as
((1.000lb + 1kg) +1000g) + 1000000mg and returns **3360900mg**.

- In this case, each of the three calculations also involves a conversion.
  The first calculation is **1.000lb + 1kg** and the system runs the three steps:
  1. Result unit of measure: **lb**
  2. Conversion: **lb** » **kg** with conversion relation: **1lb = 0.453592370kg**
     1 * 1.000000000 / 0.453592370 = 2.204622622, rounded to **2.** So the result of the conversion is **2lb**.
  3. Calculation:
     1.000lb + 2lb = 3.000lb
     The result has a precision of three fractional digits.
- The second calculation is **3.000lb +1000g**:
  1. Result unit of measure: **lb**
  2. Conversion: **g** » **kg** and **kg** » **lb** with conversion relations
     1g = 0.000010000kg and **1lb = 0.453592370kg**
     1000 * 0.001000000 / 0.453592370 = 2.204622622, rounded to **2.205**. So the result of the conversion is **2.205lb**.

3. Calculation:
   3.000lb + 2.205lb = 5.205lb
   The result has a precision of three fractional digits.

- The third calculation is **5.205lb + 1000000mg**:
  1. Result unit of measure: **mg**
  2. Conversion: twofold, **lb** » **kg** and **kg** » **mg** with conversion relations **1mg = 0.000001000kg 1lb = 0.453592370kg** and **1mg = 0.000001000kg 5.205 * 0.453592370 / 0.000001000 = 2360948.285850000000**, rounded to **2360900**. So the result of the conversion is **2360900mg**.
  3. Calculation:
     2360900mg + 1000000mg = 3360900mg
     No further changes to the precision are required.

Operators

Operators apply to operands, like in **A + B**, where **A** and **B** are the operands to which the + operator applies.
A **unary** operator applies to only one operand, whereas a **binary** operator applies to two operands.

When it comes to evaluating expressions, it is important to be aware of the precedence rule that applies to the operations and defines the order in which the operations of an expression are performed. The following list shows the order of operations, starting with the highest-precedence operation:

1. unary plus (+) (page 61) and minus (-) (page 62)
2. multiply (*) (page 62) and divide (/) (page 63)
3. binary plus (+) (page 61) and minus (-) (page 62)
4. all comparison operators:
   - equal to (==) (page 64)
   - not equal to (!=) (page 64)
   - less than (<) (page 64)
   - less than or equal to (<=) (page 64)
   - greater than (>) (page 64)
   - greater than or equal to (>=) (page 64)
5. logical NOT (page 67)
6. logical AND (page 66)
7. logical OR (page 66)

TIP

When your expression contains a sequence of operations that have the same precedence, it is recommended to use parentheses to disambiguate the order in which they are processed.

- To insert an operator at the current cursor position in the expression panel (page 47), double-click its node in the tree.

Arithmetic Operators

Arithmetic operators allow you to perform calculations between numeric values.

- Plus (+)
  binary, is used for additions or string concatenations:
  (**{Check ph value 01}.{instance count}** + **{Check ph value 02}.{instance count}**) <= 4
  "Date of execution: " + convertToDisplayString(now())

  Operations with binary plus for time-related operands, measured values, and strings and their results:

Left Operand	Right Operand	Operation Result
Duration	Duration	Duration
Timestamp	Duration	Timestamp
MeasuredValue	MeasuredValue	MeasuredValue (can return **Undefined**)
String	String	String

TIP
Please note that an operation with incompatible operands, such as lengths (e.g. meters) and weights (e.g. pounds) returns **Undefined** as result.

- Plus (+)
  unary, is rarely used, since numbers without a sign are assumed to be positive.

  Operations with unary plus for time-related operands and/or measured values and their results:

Left Operand	Right Operand	Operation Result
N/A	Duration	Duration
N/A	MeasuredValue	MeasuredValue

- Minus (-)
  binary, is used for subtractions:
  {Adjust ph value}.{instance count} - {Check ph value 01}.{instance count} >= -1

  Operations with binary minus for time-related operands and/or measured values and their results:

Left Operand	Right Operand	Operation Result
Duration	Duration	Duration
Timestamp	Duration	Timestamp
Timestamp	Timestamp	Duration
MeasuredValue	MeasuredValue	MeasuredValue (can return **Undefined**)

TIP
Please note that an operation with incompatible operands, such as lengths (e.g. meters) and weights (e.g. pounds) returns **Undefined** as result.

- Minus (-)
  unary, is used for negations:
  {Adjust ph value}.{instance count} **-** {Check ph value 01}.{instance count} >= **-1**

  Operations with unary minus for time-related operands and/or measured values and their results:

Left Operand	Right Operand	Operation Result
N/A	Duration	Duration
N/A	MeasuredValue	MeasuredValue

- Multiply (*)
  binary, is used for multiplications:
  {Adjust ph value}.{instance count} * 2 <= {Check ph value 01}.{instance count}

  Multiplication operations with time-related operands and/or measured values and their results:

Left Operand	Right Operand	Operation Result
Number (Long, Float, BigDecimal)	Duration	Duration
Duration	Number (Long, Float, BigDecimal)	Duration

Left Operand	Right Operand	Operation Result
MeasuredValue	Number (Long, Float, BigDecimal)	MeasuredValue
Number (Long, Float, BigDecimal)	MeasuredValue	MeasuredValue

TIP
Please note that you can turn a Number data type into a MeasuredValue by multiplying the numeric value with 1<unit of measure>.
Example:
20 * 1kg = 20kg

- Divide (/)
  binary, used for divisions:
  {Adjust ph value}.{instance count} / {Check ph value 01}.{instance count} > 1

TIP
Please note that when the right operand of your division is a reference that returns **0** or **undefined**, the result of the division is also **undefined**.

Division operations with time-related operands and/or measured values and their results:

Left Operand	Right Operand	Operation Result
Duration	Duration	BigDecimal
Duration	Number (Long, Float, BigDecimal)	Duration
MeasuredValue	MeasuredValue	BigDecimal (can return **Undefined**)
MeasuredValue	Number (Long, Float, BigDecimal)	MeasuredValue

TIP
Please note that an operation with incompatible operands, such as lengths (e.g. meters) and weights (e.g. pounds) returns **Undefined** as result.

COMPARISON OPERATORS

Comparison operators are necessarily all binary, since they compare two operands.

> **TIP**
> Please note that if one or both of the operands are **undefined**, the result of the comparison itself is **undefined**.

- Equal to (==)
  {Read instruction}.{out} == OK
- Not equal to (!=)
  {Read instruction}.{out} != OK
- Less than (<)
  {Adjust ph value}.{instance count} < 3
- Less than or equal to (<=)
  {Check ph value}.{value} <= 5.5
- Greater than (>)
  {Adjust ph value}.{instance count} > 3
- Greater than or equal to (>=)
  {Check ph value}.{value} >= 5.5

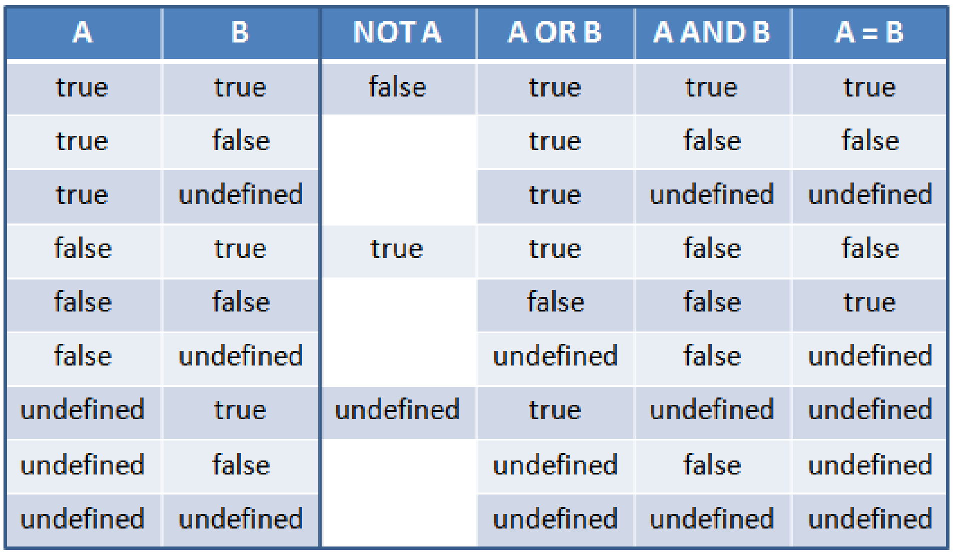
Logical Operators

Logical operators always return **true**, **false**, or **undefined** as result. **Undefined** (null) can happen, for instance, when one of the operands of the logical operation is **undefined**.

TIP

Please note the order of precedence that applies to logical operations, with NOT preceding AND and OR, which has the lowest precedence.
This means that **NOT A OR B** is interpreted as **(NOT A) OR B**.

The results of logical operations correspond to the three-valued logic used by SQL.



A	B	NOT A	A OR B	A AND B	A = B
true	true	false	true	true	true
true	false		true	false	false
true	undefined		true	undefined	undefined
false	true	true	true	false	false
false	false		false	false	true
false	undefined		undefined	false	undefined
undefined	true	undefined	true	undefined	undefined
undefined	false		undefined	false	undefined
undefined	undefined		undefined	undefined	undefined

Figure 35: Three-valued truth table

- Conjunction (AND)
  binary, an expression **A AND B** returns true only if both A and B are true.

A	B	A AND B
true	true	true
true	false	false
true	undefined	undefined
false	true	false
false	false	false
false	undefined	false
undefined	true	undefined
undefined	false	false
undefined	undefined	undefined

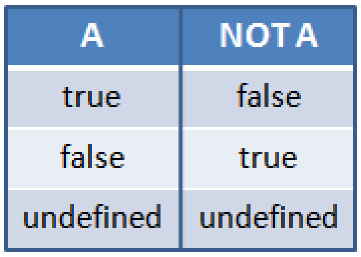
Figure 36: Truth table - A AND B

- Disjunction (OR)
  binary; an expression **A OR B** returns true if either A or B or both A and B are true.

A	B	A OR B
true	true	true
true	false	true
true	undefined	true
false	true	true
false	false	false
false	undefined	undefined
undefined	true	true
undefined	false	undefined
undefined	undefined	undefined

Figure 37: Truth table - A OR B

- Negation (NOT)
  unary, an expression **NOT A** returns true for any value (not undefined) other than A.

A	NOT A
true	false
false	true
undefined	undefined

  Figure 38: Truth table - NOT A

Functions

The Expression editor provides a number of functions whose implementation with the available operators would either lead to very complex expressions or would not be possible at all.
A function consists of its name typically followed by one or more comma-separated arguments in parentheses:
function_name(arg1, arg2, ...)

TIP

Please note that you can also use references as arguments. This may, however, cause an argument to be **undefined**.

- To insert a function with sample arguments at the current cursor position in the expression panel (page 47), double-click its node in the tree.
- To view a function with sample arguments, hover your mouse cursor over the function. The system displays the entire function as tooltip.

Null Handling Functions

Null handling functions provide the means for dealing with phases that return **undefined** as output.

- Undefined - one of two (nvl)
  inserted as **nvl(arg1, arg2)**
  It has the following characteristics:
  - It expects exactly two arguments of the same data type (page 55).
  - It evaluates the arguments from left to right and returns the first argument that is not **undefined** as result.
  - If all arguments are **undefined**, its result is also **undefined**.

- Undefined - one of many (coalesce)
  inserted as **coalesce(arg1, arg2, ...)**
  It has the following characteristics:
  - It expects at least two arguments of the same data type (page 55).
  - It evaluates the arguments from left to right and returns the first argument that is not **undefined** as result.
  - If all arguments are **undefined**, its result is also **undefined**.
- Is undefined (isNull)
  inserted as **isNull(arg1)**
  It has the following characteristics:
  - It expects exactly one argument of any data type (page 55).
  - It returns **true** as result if the argument is **undefined**. In all other cases it returns **false** as result.

Analysis Functions

Analysis functions are a collection of functions for retrieving and inserting external information, such as a timestamp, or for parsing external information to return a result.

- Current Date and Time (now)
  inserted as **now()**
  It has the following characteristics:
  - It does not expect any argument.
  - At evaluation time, i.e. during execution, it returns a Timestamp data type (page 56), which provides the then current time of the database server.
- Contains Item (containsItem)
  inserted as **containsItem(arg1, arg2 [, arg3])**
  It has the following characteristics:
  - It expects at least two arguments of the String data type (page 56). The third argument is optional.
  - It parses a separator-delimited list (**arg1**) for occurrences of a specified list item (**arg2**).
  - By default, it uses semicolon (;) as separator character. With **arg3**, however, you can declare another separator character.
  - If it locates an occurrence of **arg2**, it returns **true**, if not, it returns **false**.
  - If either of the three arguments is **undefined**, its result is also **undefined**.

TIP
This function is especially suited to deal with lists of tags returned from automated equipment. For these purposes, you would use the full list of potential return values of a FlexibleTagDefinition property as **arg1** and the identifier of the tag you wish to monitor as **arg2**.
Examples:
containsItem("AlarmHighTemp;AlarmLowPressure;AlarmEmergencyShutOff", "AlarmLowPressure") returns **true**
containsItem("AlarmHighTemp:AlarmLowPressure:AlarmEmergencyShutOff", "AlarmPressureLimit", ":") returns **false**

Conversion Functions

Conversion functions are a collection of functions that allow to convert the data type of a value to another data type.

- Convert to Unitless Number (convertTo)
  inserted as **convertTo(arg1, arg2)**
  It has the following characteristics:
  - It expects a measured value or a duration as first argument and as second argument a suitable, convertible unit of measure given as string. It returns a BigDecimal data type (page 56).

TIP
Please note that some phases may return measured values without a unit of measure. Even though these values look like a BigDecimal, their data type is still MeasuredValue, which you need to convert to BigDecimal in order to allow calculations. In this case, you have to provide an empty string as second argument.

  Examples:
    - convertTo(15m, "m") returns 15
    - convertTo(1h5min,"s") = 3900
    - convertTo(12min,"h") = 0.2
    - convertTo({<phase identifier>}.{<output>}, "") returns the unitless value from the phase output as BigDecimal value.
  - If the unit of measure defined for the second argument does not exist, the result is **undefined**.
  - If one of the arguments is **undefined**, its result is also **undefined**.

- Convert to BigDecimal (convertToBigDecimal)
  inserted as **convertToBigDecimal(arg1)**
  It has the following characteristics:
  - It expects an argument of a numeric data type (page 55), such as a Long or Float and converts it to a BigDecimal data type.
    Examples:
    - convertToBigDecimal(12345) returns 12345BDecimal
    - convertToBigDecimal(123.45Float) returns 123.45
  - If the argument is **undefined**, its result is also **undefined**.
- Convert to Long (convertToLong)
  inserted as **convertToLong(arg1, arg2)**
  It has the following characteristics:
  - It expects a fractional number, typically a BigDecimal, as first argument and as second argument a rounding mode, given as string.
  - The following rounding modes are available:
    - HALF_UP:
      rounds towards the nearest integral number, unless there are two integrals at exactly the same distance. In this case it rounds away from zero.
    - CEILING:
      always rounds up.
    - FLOOR:
      always rounds down.

    Examples:
    - convertToLong(2.5, "HALF_UP") returns 3
    - convertToLong(-1.6, "HALF_UP") returns -2
    - convertToLong(1.3, "CEILING") returns 2
    - convertToLong(-1.7, "CEILING") returns -1
    - convertToLong(2.6, "FLOOR") returns 2
    - convertToLong(-1.1, "FLOOR") returns -2
  - If either or both arguments are **undefined**, its result is also **undefined**.

- Convert to MeasuredValue (convertToMeasuredValue)
  inserted as **convertToMeasuredValue(arg1 [, arg2])**
  It has the following characteristics:
  - It expects at least one argument of a numeric data type (page 55) (Long, Float, or BigDecimal) as first argument and as second argument a unit of measure given as string. The second argument is optional.
  - It converts the numeric value into a MeasuredValue data type (page 57) with the unit of measure indicated as **arg2**. If you do not define the second argument, the system still converts the numeric value, so that it returns MeasuredValue without unit of measure.
    Examples:
    - convertToMeasuredValue(15, "m") returns 15 m
    - convertToMeasuredValue(34.8) returns 34.8
  - If the unit of measure defined for the second argument does not exist, the result is **undefined**.
  - If the numeric value argument (**arg1**) is **undefined**, its result is also **undefined**.

- Convert to String for Display (convertToDisplayString)
  inserted as **convertToDisplayString(arg1)**
  It has the following characteristics:
  - It expects exactly one argument of any data type (page 55).
    Examples:
    - convertToDisplayString("Mixer Unit") returns Mixer Unit
    - convertToDisplayString(42), returns 42
    - convertToDisplayString(10.5kg) returns 10.5 kg
    - convertToDisplayString(true) returns Yes
    - convertToDisplayString(73s) returns 1min 13s
    - convertToDisplayString(11-Jan-2013@12:13:54.232) returns 01/11/2013 12:13:54 PM CET
  - If the argument is **undefined**, its result is also **undefined**.

Aggregation Functions

Aggregation functions provide the means to evaluate two and more output values to determine the highest, lowest, or the average of all values.

- Average (average)
  inserted as **average(arg1, arg2, ...)**
  It has the following characteristics:
  - It expects at least two arguments of the same numeric data type (page 55).
  - It ignores arguments that are **undefined**.
  - It calculates the average of all usable (not **undefined**) arguments.
  - If only one argument is usable, its result equals the argument.
  - If all arguments are **undefined**, its result is also **undefined**.

  TIP
  When you use the **Average** function with arguments of the MeasuredValue data type (page 57), the system returns the result in the unit of measure of the first operand.

- Maximum (max)
  inserted as **max(arg1, arg2, ...)**
  It has the following characteristics:
  - It expects at least two arguments of the same, comparable data type (page 55).
  - It ignores arguments that are **undefined**.
  - It determines the largest of all usable (not **undefined**) arguments.
  - If only one argument is usable, its result equals the argument.
  - If all arguments are **undefined**, its result is also **undefined**.

- Minimum (min)
  inserted as **min(arg1, arg2, ...)**
  It has the following characteristics:
  - It expects at least two arguments of the same, comparable data type (page 55).
  - It ignores arguments that are **undefined**.
  - It determines the smallest of all usable (not **undefined**) arguments.
  - If only one argument is usable, its result equals the argument.
  - If all arguments are **undefined**, its result is also **undefined**.

Order Context Functions

Order context functions return values that are determined by the order that runs the expression during its execution. They are available for master recipes and building blocks on all levels of their hierarchical graph structure.

- Order ID (orderId)
  inserted as **orderId()**
  It returns the identifier of the order with which the master recipe is executed.

- Order Scaling Factor (orderScalingFactor)
  inserted as **orderScalingFactor()**
  It returns the factor (as BigDecimal), which has been used during order explosion for scaling the planned quantities of the order step inputs. The system determines the value by dividing the order's planned quantity by the planned quantity defined with the master recipe assigned to the order. The scaling factor is always displayed with nine fractional digits.
  It is only relevant to orders and thus not available in Workflow Designer.

- Produced Material ID (producedMaterialId)
  inserted as **producedMaterialId()**
  It returns the identifier of the material to be produced with the master recipe.

- Produced Material Short Description (producedMaterialShortDescription)
  inserted as **producedMaterialShortDescription()**
  It returns the short description of the material to be produced with the master recipe.

- Produced Material Type (producedMaterialType)
  inserted as **producedMaterialType()**
  It returns the type of the material to be produced with the master recipe.

- Batch ID (batchId)
  inserted as **batchId()**
  It returns the identifier of the batch to be produced with the master recipe.

- Batch Status (batchStatus)
  inserted as **batchStatus()**
  It returns the status of the batch to be produced with the master recipe.

MFC Position Context Functions

MFC position context functions return values that are determined by the **Position** defined with the material parameters in the Parameter Panel. They are available for master recipes, master workflows, and building blocks on all levels of their hierarchical graph structure.

TIP

Please note that when you use an **MFC position context** function to refer to a material input and the input is replaced in PharmaSuite for Production Management with the **New alternative order step input** action, the function uses the position of the alternative input to determine its return values.

- Material ID (materialId)
  inserted as **materialId(arg1)**
  It has the following characteristics:
  - It expects exactly one argument, namely the position of a material input or output, given as string.
  - It returns the identifier of the position's material, given as string data type (page 56).
    Examples:
    - materialId("20") returns "D005-01"
    - materialId({Preparation}/{Identify Material}.{MFC Position}) returns "D130-01"
  - If the argument is **undefined** or the position does not exist, its result is **undefined**.
- Material Short Description (materialShortDescription)
  inserted as **materialShortDescription(arg1)**
  It has the following characteristics:
  - It expects exactly one argument, namely the position of a material input or output, given as string.
  - It returns the short description of the position's material, given as string data type (page 56).
    Examples:
    - materialShortDescription("20") returns "Lactose"
    - materialShortDescription({Preparation}/{Identify Material}.{MFC Position}) returns "Sonolin 100mg premix"
  - If the argument is **undefined** or the position does not exist, its result is **undefined**.

- Material Type (materialType)
  inserted as **materialType(arg1)**
  It has the following characteristics:
  - It expects exactly one argument, namely the position of a material input or output, given as string.
  - It returns the material type of the position's material, given as string data type (page 56).
    Examples:
    - materialShortDescription("20") returns "RawMaterial"
    - materialShortDescription({Preparation}/{Identify Material}.{MFC Position}) returns "SemiFinishedGoods"
  - If the argument is **undefined** or the position does not exist, its result is **undefined**.

- Status (mfcPositionStatus)
  inserted as **mfcPositionStatus(arg1)**
  It has the following characteristics:
  - It expects exactly one argument, namely the position of a material input or output, given as string.
  - It returns the status of the material input or output, given as string data type (page 56).
    The following return values can occur:
    - "NOT_STARTED"
      The position has been created, but no source sublot has been identified for it yet.
    - "IN_PROCESS"
      The position is in process and at least one source sublot has been identified for it.
    - "COMPLETED_UNDERWEIGH"
      The position has been completed with an underweight exception.
    - "COMPLETED_INTOLERANCE"
      The position has been completed in tolerance.
    - "COMPLETED_OVERWEIGH"
      The position has been completed with an overweight exception.
    - "CANCELED"
      The position has been canceled.

- "ABORTED"
  The position has been aborted before weighing.

  Examples:

  - mfcPositionStatus("20") returns "NOT_STARTED"
  - mfcPositionStatus({Preparation}/{Identify Material}.{MFC Position}) returns "COMPLETED_INTOLERANCE".
- If the argument is **undefined** or the position does not exist, its result is **undefined**.

- Planned Quantity - Original (originalPlannedQuantity)
  inserted as **originalPlannedQuantity(arg1)**
  It has the following characteristics:
  - It expects exactly one argument, namely the position of a material input or output, given as string.
  - It returns the planned quantity of the material position as defined in the Parameter Panel and calculated during order explosion. It is given as MeasuredValue data type (page 57).
    Examples:
    - originalPlannedQuantity("20") returns 100.3 g
    - originalPlannedQuantity({Preparation}/{Identify Material}.{MFC Position}) returns 4.7 kg
  - If the argument is **undefined** or the position does not exist, its result is **undefined**.

- Planned Quantity - Execution (plannedQuantity)
  inserted as **plannedQuantity(arg1)**
  It has the following characteristics:
  - It expects exactly one argument, namely the position of a material input or output, given as string.
  - It returns the planned quantity of the material position as used during execution where it can have been adjusted due to the application of a prorate factor. It is given as MeasuredValue data type (page 57).
    Examples:
    - plannedQuantity("20") returns 17.4 ml
    - plannedQuantity({Preparation}/{Identify Material}.{MFC Position}) returns 1.502 g
  - If the argument is **undefined** or the position does not exist, its result is **undefined**.

- Actual Quantity - Last Split (actualSplitQuantity)
  inserted as **actualSplitQuantity(arg1)**

  > **TIP**
  > Please note that the function can only be used in the context of Dispense and Inline Weighing.

  It has the following characteristics:

  - It expects exactly one argument, namely the position of a material input, given as string.
  - It returns the actual quantity of all split positions that constituted the last target (e.g. a charging vessel in Inline Weighing) of a material input position. Thus it is only relevant to material input positions. For material outputs, which do not have split positions, the function always returns **undefined** as result. It is given as MeasuredValue data type (page 57).
    Examples:
    - actualSplitQuantity("20") returns 360.4 g
    - actualSplitQuantity({Preparation}/{Identify Material}.{MFC Position}) returns 59 l
  - If the argument is **undefined** or the position does not exist, its result is **undefined**.

- Actual Quantity - Total (totalActualQuantity)
  inserted as **totalActualQuantity(arg1)**

  > **TIP**
  > Please note that if applied to material inputs, the function can only be used in the context of Dispense and Inline Weighing.

  It has the following characteristics:

  - It expects exactly one argument, namely the position of a material input or output, given as string.
  - It returns the total actual quantity of a material position, excluding material that was replaced or declared as waste. For a material input, it calculates the sum of all of its split positions. In the case of material outputs, for which there are no split positions, it returns the sum of all produced sublots (excluding replaced sublots). It is given as MeasuredValue data type (page 57).
    Examples:
    - totalActualQuantity("20") returns 1600.4 g
    - totalActualQuantity({Preparation}/{Identify Material}.{MFC Position}) returns 5.8 kg

- If the argument is **undefined** or the position does not exist, its result is **undefined**.

- Sum of Actual Input Quantities - Total (sumTotalActualInputQuantity) inserted as **sumTotalActualInputQuantity(...)**

  TIP
  Please note that the function can only be used in the context of Dispense and Inline Weighing.

  It has the following characteristics:

  - It expects either
    - any number of arguments, namely the positions of material inputs, given as string or
    - no argument at all, in which case the system considers all material inputs of the unit procedure.
  - If you have specified a position or list of positions as arguments, it returns the sum of all actual quantities of the specified positions. You can specify any position of the entire procedure.
    If you have not specified an argument, it returns the sum of all actual quantities of the unit procedure.
    Material that was replaced or declared as waste is not included in the sum.
    The function's result is given as MeasuredValue data type (page 57).
    Examples:
    The procedure contains six positions (10, 20, ... 60), each of which has an actual input quantity of 40.3 g.
    - sumTotalActualInputQuantity("10", "30", "40") returns 120.9 g
    - sumTotalActualInputQuantity() returns 241.8 g
  - It ignores arguments that return an undefined value.
    However, if all arguments return undefined values, the function's result is also **undefined**.
    Additionally, if a specified position itself is **undefined** or does not exist as material input, the function's result is also **undefined**.

- Prorate Factor (prorateFactor)
  inserted as **prorateFactor(arg1)**
  It has the following characteristics:
  - It expects exactly one argument, namely the position of a material output, given as string.
  - It is only relevant to material outputs and thus not available in Recipe Designer - Device and Workflow Designer.
  - It returns the prorate factor, which is calculated by dividing the actual produced quantity of an output by its originally planned output quantity. It is given as MeasuredValue data type (page 57).
  - If the argument is **undefined** or the position of a material input, its result is **undefined**.
- Yield (yield)
  inserted as **yield(arg1)**
  It has the following characteristics:
  - It expects exactly one argument, namely the position of a material output, given as string.
  - It is only relevant to material outputs and thus not available in Recipe Designer - Device and Workflow Designer.
  - It returns the yield, which is calculated by dividing the actual produced quantity of an output by its planned (potentially prorated) output quantity. It is given as MeasuredValue data type (page 57).
  - If the argument is **undefined** or the position of a material input, its result is **undefined**.

TIP

Please note that the functions for MFC position context rely on their phases to provide the required data. The **Prorate Factor** and **Yield** functions, for example, cannot be used with the **Material Tracking Phases** package.

Equipment Context Functions

Equipment context functions return values that are determined by the physical equipment entities used during execution on the shop floor. The **Current Equipment Entity** function is restricted to being used in equipment requirement rules, while the other equipment context functions are available in all contexts.

- Current Equipment Entity (currentEquipmentEntity)
  inserted as **currentEquipmentEntity()**
  It has the following characteristics:
  - It does not expect any argument.
  - At evaluation time, i.e. during execution, it returns a database object of the IMESS88Equipment data type (page 56), which provides the equipment entity that has been identified and is in use for processing the phase.

  TIP
  When the function is used in a flexible rule and the identified entity belongs to a group, it refers to the group's parent entity, which is not necessarily the identified entity.
  When used in a conditional rule, which is group-enabled, the function refers to all members of the group and not only the group's parent entity.

- Equipment Is Member of Class (equipmentIsMemberOfClass)
  inserted as **equipmentIsMemberOfClass(arg1, arg2)**
  It has the following characteristics:
  - It expects an equipment object (IMESS88Equipment data type (page 56)) as first argument and as second argument an equipment class identifier given as string.
  - It returns **true** if the entity is a member of the indicated class. If the entity is not a member of the class or the class itself does not exist, it returns **false**.
    Examples:
    There is a **TabletPresses-A** equipment class that contains all tablet presses.
    - equipmentIsMemberOfClass({Identify Tableting Unit}.{Equipment object}, "TabletPresses-A") returns **true**
    - equipmentIsMemberOfClass(currentEquipmentEntity(), "TabletPresses-A") returns **true** if the identified equipment entity is a tablet press.
  - If one of the arguments is **undefined**, its result is also **undefined**.

- Equipment Property Attribute (equipmentPropertyAttribute)
  inserted as **equipmentPropertyAttribute(arg1, arg2 [, arg3, arg4])**
  It has the following characteristics:

  - It expects an equipment object (IMESS88Equipment data type (page 56)) as first argument, as second argument a property type identifier given as string, as optional third argument a suitable property type attribute identifier, given as string, and as optional fourth argument an index, given as long.

  TIP
  Please note that the property type identifier (arg2) and the property type attribute identifier (arg3) must be string constants and cannot be drawn via the information flow from preceding phases.

    - If you wish to retrieve the value of a property type, you can omit to define arg3, which the system interprets as "value" unless specifically defined otherwise.
    - For property types of the FlexibleAttributeDefinition data type, however, you always have to define arg3, since they may not even have an attribute with "value" as identifier.
    - For specification property types of the BigDecimal data type, you can retrieve the minimum, maximum, or current values. You can omit arg3 if you wish to retrieve the value, but need to define it to retrieve the minimum or maximum values.
    - If the property type attribute given as arg3 has an index, you need to provide its value as fourth argument. If the attribute is not indexed, you have to omit arg4.

  TIP
  Please note that for runtime property types other than FlexibleAttributeDefinition you can only retrieve their value.

  - It returns a value in the data type of the property type attribute given as third argument.
    Examples:

    - equipmentPropertyAttribute({Identify Coating Unit}.{Equipment object}, "Coater Prepared", "Unit Assembled") returns the current attribute value of the identified entity's property, e.g. either "Yes" or "No".
    - equipmentPropertyAttribute({Identify Coating Unit}.{Equipment object}, "Vessel Cleaning Status", "value")
      or
      equipmentPropertyAttribute({Identify Coating Unit}.{Equipment

object}, "Vessel Cleaning Status") return the current status of the identified entity, e.g. "Clean".

    - equipmentPropertyAttribute({Identify Coating Unit}.{Equipment object}, "Drum Size", "minValue") returns the minimum drum size of the identified entity.
    - equipmentPropertyAttribute({Identify Scale}.{Equipment object}, "Scale Test", "position", 14) returns the intended positioning of test weight #14 defined in the Scale Test property type.
  - If either of the four arguments is **undefined**, its result is also **undefined**.

- Equipment Has Property Attribute (equipmentHasPropertyAttribute)
  inserted as **equipmentHasPropertyAttribute(arg1, arg2 [, arg3, arg4])**
  It has the following characteristics:
  - It expects an equipment object (IMESS88Equipment data type (page 56)) as first argument, as second argument a property type identifier given as string, as optional third argument a suitable property type attribute identifier, given as string, and as optional fourth argument an index, given as long.

TIP
Please note that the property type identifier (arg2) and the property type attribute identifier (arg3) must be string constants and cannot be drawn via the information flow from preceding phases.

    - If you wish to check for the **value** attribute of a property type, you can omit to define arg3, which the system interprets as "value" unless specifically defined otherwise.
    - For property types of the FlexibleAttributeDefinition data type, however, you always have to define arg3, since they may not even have an attribute with "value" as identifier.
    - For specification property types of the BigDecimal data type, you can check for the minimum, maximum, or current values. You can omit arg3 if you wish to check for the current value, but need to define it to check for the minimum or maximum values.
    - If the property type attribute given as arg3 has an index, you need to provide its value as fourth argument. If the attribute is not indexed, you have to omit arg4.

TIP
Please note that for runtime property types other than FlexibleAttributeDefinition you can only retrieve their value.

- It returns **true** if the entity contains the indicated property or property attribute, otherwise it returns **false**.
  Examples:
  - equipmentHasPropertyAttribute({Identify Coating Unit}.{Equipment object}, "Coater Prepared", "Unit Assembled") returns **true** if the identified equipment entity contains a **Coater Prepared** property with a **Unit Assembled** attribute.
  - equipmentHasPropertyAttribute({Identify Coating Unit}.{Equipment object}, "Vessel Cleaning Status", "value")
    or
    equipmentHasPropertyAttribute({Identify Coating Unit}.{Equipment object}, "Vessel Cleaning Status") returns **true** if the identified equipment entity contains the **Vessel Cleaning Status** property, which has the **value** attribute by default.
  - equipmentHasPropertyAttribute({Identify Coating Unit}.{Equipment object}, "Drum Size", "minValue") returns **true** if the identified equipment entity has a **Drum Size** property with a **minValue** attribute.
  - equipmentHasPropertyAttribute({Identify Scale}.{Equipment object}, "Scale Test", "position", 14) returns **true** if the identified equipment entity has a **Scale Test** property with a **position** attribute for test weight **#14**.
- If either of the four arguments is **undefined**, its result is also **undefined**.

- Equipment Graph Status (equipmentGraphStatus)
  inserted as **equipmentGraphStatus(arg1, arg2 [, arg3])**
  It has the following characteristics:
  - It expects an equipment object (IMESS88Equipment data type (page 56)) as first argument, as second argument a purpose given as non-localized string, and as optional third argument either the status attribute **Key** or its **Display text**, also given as non-localized string ("KEY" or "DISPLAY").

  TIP
  The auto-completion feature supports you with inserting the non-localized string constants.

  - If you do not specify the third argument, it defaults to KEY.
  - Depending on the third argument, it returns the **Key** or the **Display text** attribute of the entity's current status, which is defined by the graph of the given purpose that is assigned to the entity.
    Examples:

        - equipmentGraphStatus({Identify Tableting Unit}.{Equipment object}, "Cleaning" , "KEY") returns "REUSABLE"
        - equipmentGraphStatus({Identify Tableting Unit}.{Equipment object}, "Cleaning") returns "REUSABLE"
        - equipmentGraphStatus({Identify Tableting Unit}.{Equipment object}, "Cleaning" , "DISPLAY") returns "Reusable"
    - If either of the three arguments is **undefined**, its result is also **undefined**.

- Equipment Has Graph (equipmentHasGraph)
  inserted as **equipmentHasGraph(arg1, arg2)**
  It has the following characteristics:
    - It expects an equipment object (IMESS88Equipment data type (page 56)) as first argument and as second argument a purpose given as non-localized string.

    TIP
    The auto-completion feature supports you with inserting the non-localized string constants.

    - It returns **true** if the entity contains an equipment graph with the indicated purpose. Otherwise it returns **false**.
    - If one of the arguments is **undefined**, its result is also **undefined**.

- Equipment Graph Expiry Date (equipmentGraphExpiryDate)
  inserted as **equipmentGraphExpiryDate(arg1, arg2)**
  It has the following characteristics:
    - It expects an equipment object (IMESS88Equipment data type (page 56)) as first argument and as second argument a purpose given as non-localized string.

    TIP
    The auto-completion feature supports you with inserting the non-localized string constants.

    - It returns the expiry date of the entity's current status, which is defined by the graph of the given purpose that is assigned to the entity. If the status cannot expire, the result is **undefined**.
    - If one of the arguments is **undefined**, its result is also **undefined**.

- Equipment FSM Status (equipmentFSMStatus)
  inserted as **equipmentFSMStatus(arg1, arg2 [, arg3])**
  It has the following characteristics:
  - It expects an equipment object (IMESS88Equipment data type (page 56)) as first argument, as second argument the identifier of a FlexibleStateModel property type, given as string, and as optional third argument either the "KEY" string or the "DISPLAY" string.

  TIP
  The auto-completion feature supports you with inserting the non-localized string constants.

  - If you do not specify the third argument, it defaults to DISPLAY.
  - Depending on the third argument, it returns the key, which is the non-localized state name of the attribute or its display text, which is its localized status.
    Examples:
    The system includes a Flexible State Model for Coater Cleaning with three states (CLEAN, UNCLEAN, IN USE), which are displayed as **Clean**, **Cleaning required**, and **In use**. The FSM is referenced in the **CoaterCleaning** FlexibleStateModel property type.
    - equipmentFSMStatus({Identify Coater}.{Equipment object}, "CoaterCleaning" , "KEY") returns "CLEAN"
    - equipmentFSMStatus({Identify Coater}.{Equipment object}, "CoaterCleaning") returns "Clean"
    - equipmentFSMStatus({Identify Coater}.{Equipment object}, "CoaterCleaning" , "DISPLAY") returns "Clean"
  - If either of the three arguments is **undefined**, its result is also **undefined**.

- Date and Time of Last Equipment FSM Status Change (lastEqTransitionTimestamp)
  inserted as **lastEqTransitionTimestamp(arg1, arg2, arg3 [, arg4])**
  It has the following characteristics:
  - It expects an equipment object (IMESS88Equipment data type (page 56)) as first argument, as second argument the identifier of a FlexibleStateModel property type, given as string, as third argument the non-localized state name or the localized status of the entity, both given as string, and as optional fourth argument either the "KEY" string or the "DISPLAY" string.
  - The value given for **arg3** determines the string expected for the optional fourth argument (**arg4**). If **arg3** is a non-localized state name, **arg4** must be "KEY", if **arg3** is a localized status, **arg4** must be "DISPLAY".

TIP
The auto-completion feature supports you with inserting the non-localized string constants.

- If you do not specify the fourth argument, it defaults to DISPLAY and thus requires **arg3** to be a localized status.
- It returns a timestamp of the last time when the status of the indicated equipment entity (**arg1**) changed to the indicated status (**arg3**). The function makes use of the equipment logbook to determine this information.
  Examples:
  The system includes a Flexible State Model for Coater Cleaning with three states (CLEAN, UNCLEAN, IN USE), which are displayed as **Clean**, **Cleaning required**, and **In use**. The FSM is referenced in the **CoaterCleaning** FlexibleStateModel property type.
  - lastEqTransitionTimestamp({Identify Coater}.{Equipment object}, "CoaterCleaning" , "CLEAN" , "KEY") returns 09/17/2016 03:25 PM CEST
  - lastEqTransitionTimestamp({Identify Coater}.{Equipment object}, "CoaterCleaning" , "Clean") returns 09/17/2016 03:25 PM CEST
  - lastEqTransitionTimestamp({Identify Coater}.{Equipment object}, "CoaterCleaning" , "Clean" , "DISPLAY") returns 09/17/2016 03:25 PM CEST
- If either of the four arguments is **undefined**, its result is also **undefined**.

TIP
Please note that accessing the equipment logbook during execution to determine the required information may take some time and may thus cause a slight processing delay when used in a transition condition or a rule.

- Last Product ID (lastProduct)
  inserted as **lastProduct(arg1)**
  It has the following characteristics:
  - It expects exactly one argument of the IMESS88Equipment data type (page 56).
  - It returns the identifier of the last material in whose production the equipment entity was used. The function makes use of the equipment logbook to determine this information.
  - If the argument is **undefined**, its result is also **undefined**.

TIP
Please note that accessing the equipment logbook during execution to determine the required information may take some time and may thus cause a slight processing delay when used in a transition condition or a rule.

Location Context Functions

Location context functions return values that are determined by the execution station of the phase to which the expression refers. They are available for master recipes, master workflows, and building blocks only on the phase level.

- Work Center ID (workCenterId)
  inserted as **workCenterId()**
  It returns the identifier of the work center that is connected to the station on which the phase is executed.

- Work Center Description (workCenterDescription)
  inserted as **workCenterDescription()**
  It returns the description of the work center that is connected to the station on which the phase is executed.

- Room ID (roomId)
  inserted as **roomId()**
  It returns the identifier of the room that is connected to the station on which the phase is executed.

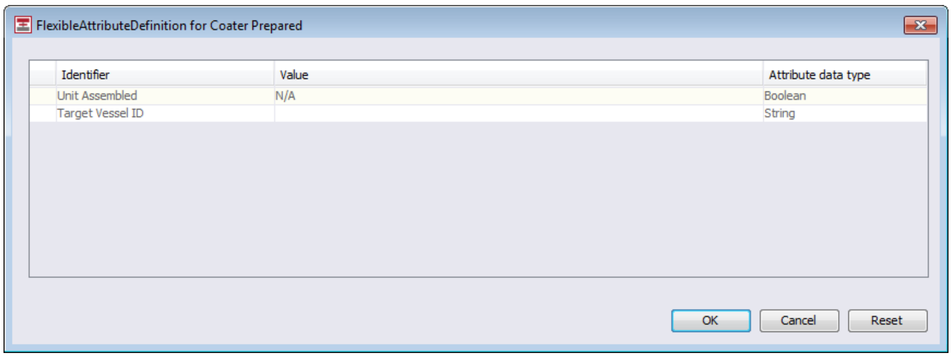
- Room Status (roomStatus)
  inserted as **roomStatus()**
  It returns the cleaning status of the room that is connected to the station on which the phase is executed.

- Storage Area ID (storageAreaId)
  inserted as **storageAreaId()**
  It returns the identifier of the storage area that is connected to the station on which the phase is executed.

- Storage Area Description (storageAreaDescription)
  inserted as **storageAreaDescription()**
  It returns the description of the storage area that is connected to the station on which the phase is executed.

FlexibleAttributeDefinition Editor

The FlexibleAttributeDefinition editor supports you with creating and configuring a bundle of runtime attributes.


FlexibleAttributeDefinition for Coater Prepared

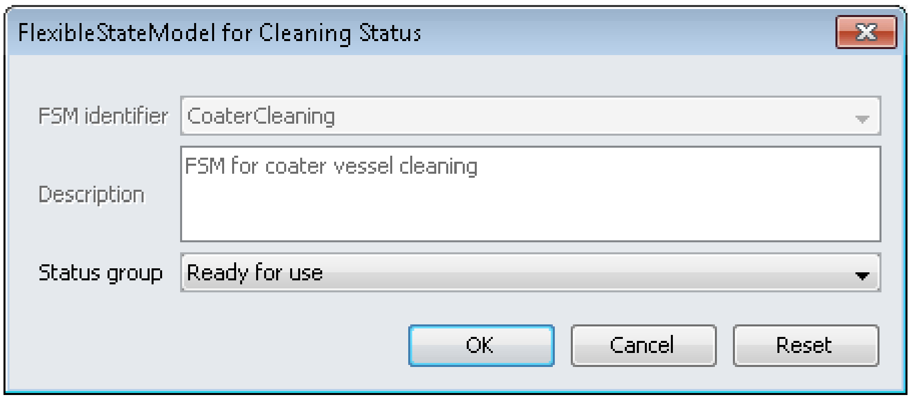
Identifier	Value	Attribute data type
Unit Assembled	N/A	Boolean
Target Vessel ID		String

Figure 39: FlexibleAttributeDefinition editor for property requirement

- When you define a property requirement, you can only view the list of attributes and their values defined for the property.
- To save your data and close the FlexibleAttributeDefinition editor, click the **OK** button.
- To close the FlexibleAttributeDefinition editor without saving your changes, click the **Cancel** button.
- To reset your data to its last saved form, click the **Reset** button.

FlexibleStateModel Editor

The FlexibleStateModel editor supports you with defining a status graph to govern an equipment entity and with setting the required status the equipment entity must have to be suitable for use.


FlexibleStateModel for Cleaning Status

FSM identifier: CoaterCleaning

Description: FSM for coater vessel cleaning

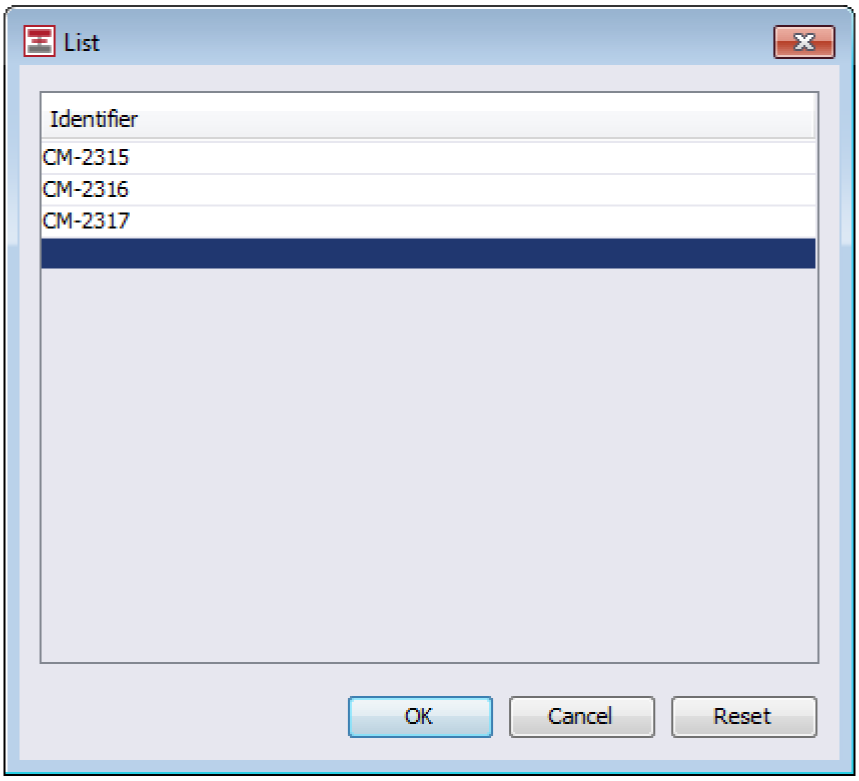
Status group: Ready for use

Figure 40: FlexibleStateModel editor for property requirement

- When you define a property requirement you can specify which status or statuses of the FlexibleStateModel an equipment entity must have to be suitable for use. For this purpose, select an option from the **Status group** list, which represents a bundle of semantically connected statuses.
- To save your data and close the FlexibleStateModel editor, click the **OK** button.
- To close the FlexibleStateModel editor without saving your changes, click the **Cancel** button.
- To reset the FlexibleStateModel definition to its last saved form, click the **Reset** button.

List Editor

The List editor supports you with defining the items of a list. The identifiers have to be unique within the list.



Identifier
CM-2315
CM-2316
CM-2317

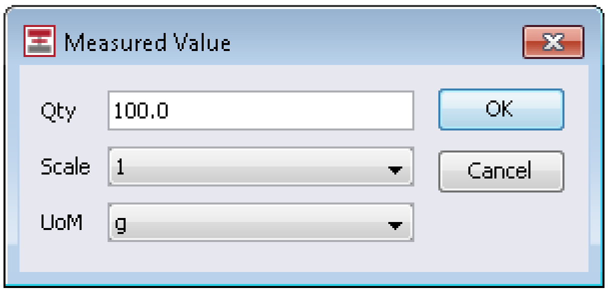
Figure 41: List editor

- To add a list item, type in the empty row at the bottom of the list.
  The system automatically adds a new empty list row after you have completed the entry in the cell.
- To delete a list item, delete its identifier or select the row and press the DEL key.
- To save your list items and close the List editor, click the **OK** button.

- To close the List editor without saving your changes, click the **Cancel** button.
- To reset your list items to their last saved form, click the **Reset** button.

Measured Value Editor

The Measured Value editor supports you when you enter a value with a unit of measure. **Scale** indicates the number of fractional digits.



Qty: 100.0
Scale: 1
UoM: g

Figure 42: Measured Value editor

- To save your value and close the Measured Value editor, click the **OK** button.
- To close the Measured Value editor without saving your changes, click the **Cancel** button.

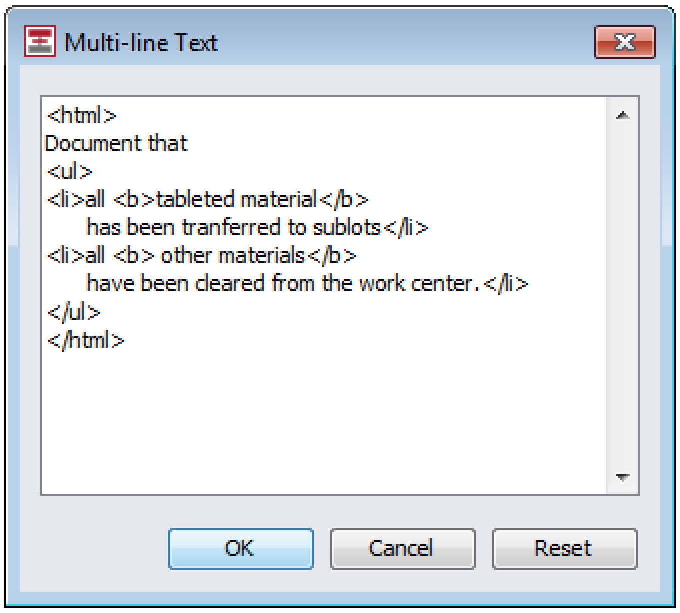
Multi-line Text Editor

The Multi-line Text editor supports you when you need to write text that is too long for a simple input box. You can use the HTML syntax to apply formatting to your text. The system supports the following HTML tags: <b>, <br>, <font>, <u>, <i>, <sup>, <sub>, <ol>, <ul>, and <li>.

TIPS

Please keep in mind that by using list formatting with HTML you will change the indents of your text lines. Thus you may disturb the text alignment of your phases when you mix HTML-formatted phases with plain-text phases.

Please note that text formatted with HTML is displayed as plain text including the HTML tags in Master Recipe Reports or Master Workflow Reports. In batch reports, DHR reports, and workflow reports, however, formatting with the supported tags is interpreted and applied.



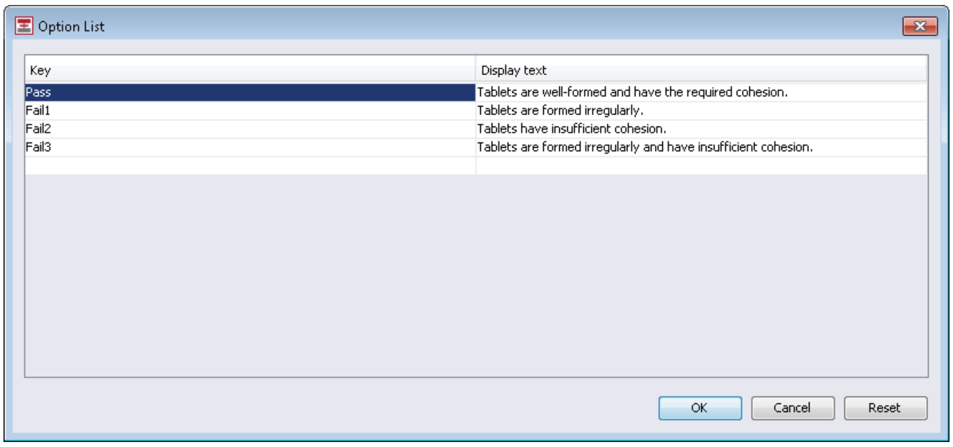
```
<html>
Document that
<ul>
<li>all <b>tableted material</b>
    has been tranferred to sublots</li>
<li>all <b> other materials</b>
    have been cleared from the work center.</li>
</ul>
</html>
```

Figure 43: Multi-line Text editor

- To save your text and close the Multi-line Text editor, click the **OK** button.
- To close the Multi-line Text editor without saving your changes, click the **Cancel** button.
- To reset your text to its last saved form, click the **Reset** button.

Option List Editor

The Option List editor supports you with defining the key and display text pairs of option lists. Both keys and display texts have to be unique within the option list.



Key	Display text
Pass	Tablets are well-formed and have the required cohesion.
Fail1	Tablets are formed irregularly.
Fail2	Tablets have insufficient cohesion.
Fail3	Tablets are formed irregularly and have insufficient cohesion.

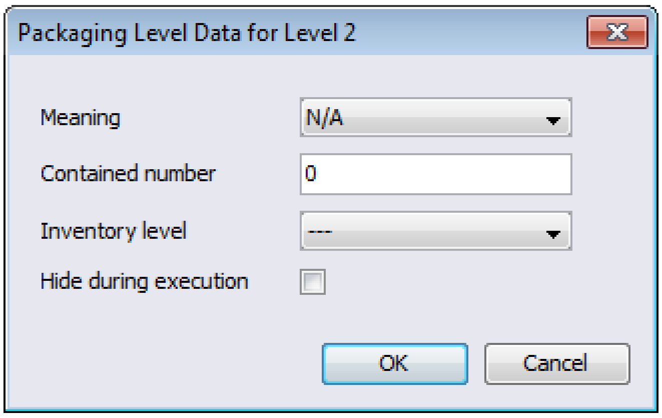
Figure 44: Option List editor

- To add an option, type in the empty row at the bottom of the list.
  The system automatically adds a new empty list row after you have completed the first cell.
- To delete an option, delete its key and text or select the row and press the DEL key.
- To save your options and close the Option List editor, click the **OK** button.
- To close the Option List editor without saving your changes, click the **Cancel** button.
- To reset your options to their last saved form, click the **Reset** button.

Packaging Level Data Editor

The Packaging Level Data editor supports you when you define the attributes of packaging levels:

- **Meaning**, to indicate the physical purpose of a packaging unit (**Tablet**, **Vial**, **Folding carton**, etc.).
- **Contained number**, to set the number of sub-units contained in a unit of the given **Meaning**.
- **Inventory level**, to define for a warehouse context if the given unit represents a **Sublot**, **Logistic unit**, or both.
- **Hide during execution**, to specify if the level information is displayed during phase execution on the shop floor.



Meaning	N/A
Contained number	0
Inventory level	---
Hide during execution	☐

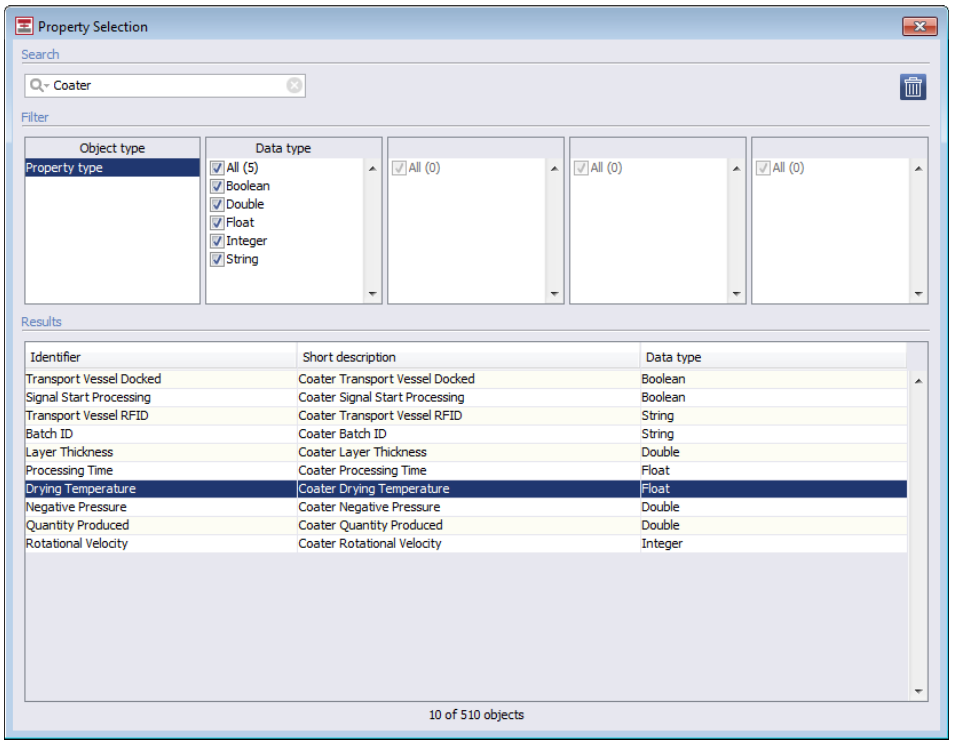
Figure 45: Packaging Level Data editor

- To save your data and close the Packaging Level Data editor, click the **OK** button.
- To close the Packaging Level Data editor without saving your changes, click the **Cancel** button.

Property Selection Editor

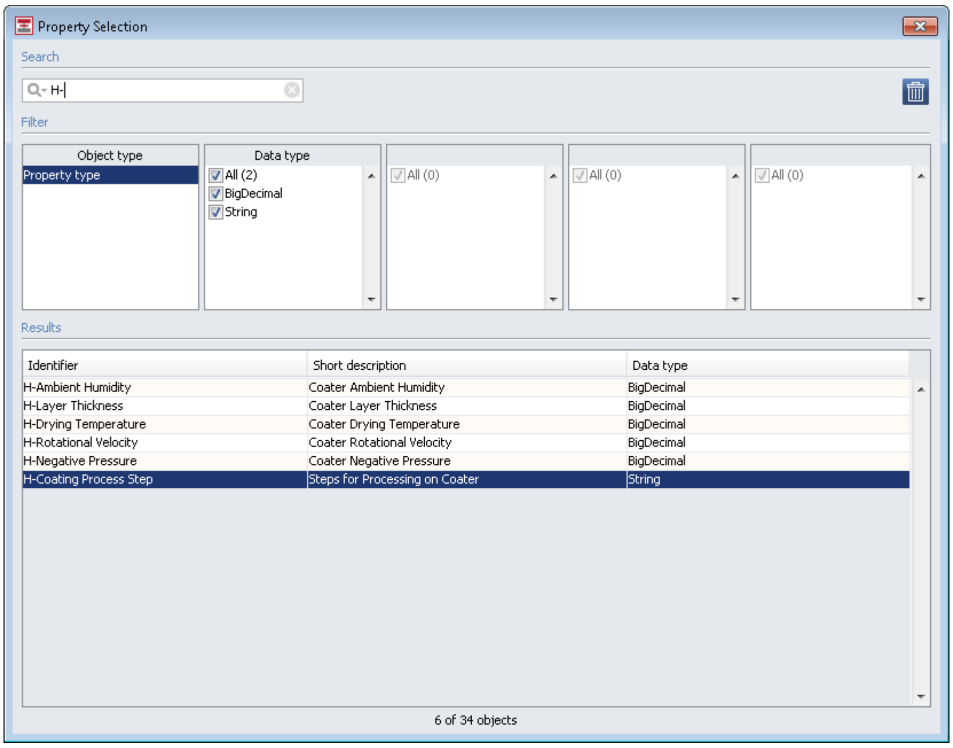
The Property Selection editor gives you direct access to all property types relevant to the data type of the property process parameter that is being configured in the Parameter Panel. Select the property type you wish to add as property to the process parameter. Use the Search (page 130) and Filter (page 130) tools to restrict the number of items shown in the **Results** list.

The **Results** list displays all search and filter results and is updated along with each change to the search or filter criteria.
By default, the list is sorted alphabetically with respect to the **Identifier** column of the object, but you can adjust both the sort and the column order (page 131).



Identifier	Short description	Data type
Transport Vessel Docked	Coater Transport Vessel Docked	Boolean
Signal Start Processing	Coater Signal Start Processing	Boolean
Transport Vessel RFID	Coater Transport Vessel RFID	String
Batch ID	Coater Batch ID	String
Layer Thickness	Coater Layer Thickness	Double
Processing Time	Coater Processing Time	Float
Drying Temperature	Coater Drying Temperature	Float
Negative Pressure	Coater Negative Pressure	Double
Quantity Produced	Coater Quantity Produced	Double
Rotational Velocity	Coater Rotational Velocity	Integer

Figure 46: Property Selection editor for Automation properties



Identifier	Short description	Data type
H-Ambient Humidity	Coater Ambient Humidity	BigDecimal
H-Layer Thickness	Coater Layer Thickness	BigDecimal
H-Drying Temperature	Coater Drying Temperature	BigDecimal
H-Rotational Velocity	Coater Rotational Velocity	BigDecimal
H-Negative Pressure	Coater Negative Pressure	BigDecimal
H-Coating Process Step	Steps for Processing on Coater	String

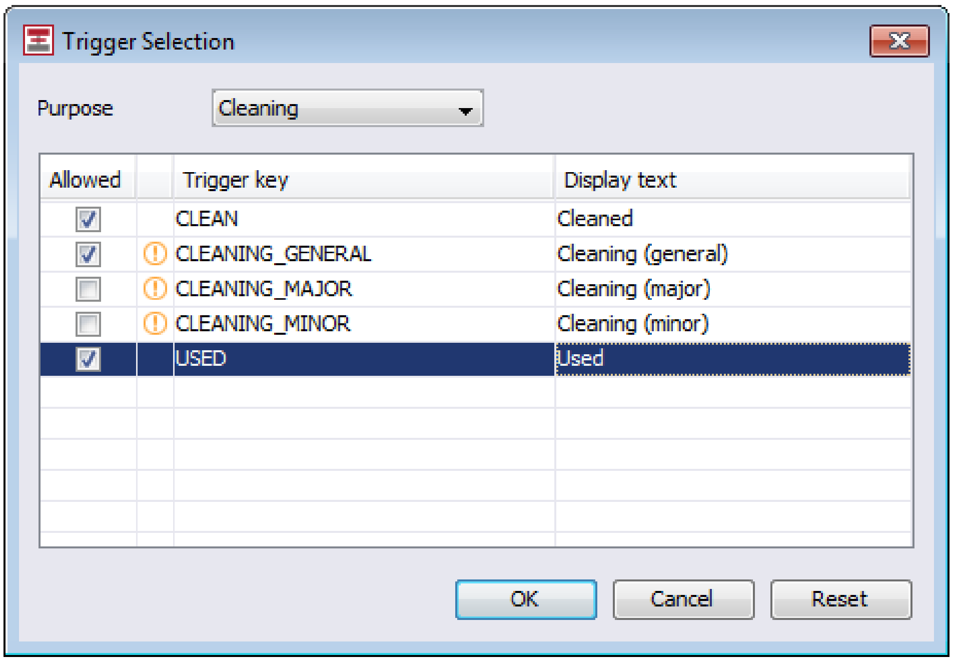
6 of 34 objects

Figure 47: Property Selection editor for Historian properties

- To add a property type as property to the process parameter, double-click it in the **Results** list. The system assigns it to the process parameter and closes the Property Selection editor.
- To remove a property from a process parameter, click the **Remove property** button in the upper right corner of the Property Selection editor. The system removes it from the process parameter and closes the Property Selection editor.

Trigger Selection Editor

The Trigger Selection editor gives you direct access to all triggers available for a specific purpose. Select the triggers you wish to display as option on the phase. By default, the list of triggers is sorted alphabetically with respect to the **Trigger key** column of the object, but you can adjust both the sort and the column order (page 131).
When there are triggers listed that are not contained in all graphs of the selected purpose, they are marked with a warning icon (page 105). If you select one of the marked triggers, make sure that it is actually contained in the graph assigned to the equipment entities whose statuses you intend to have changed.



Purpose: Cleaning

Allowed		Trigger key	Display text
✓		CLEAN	Cleaned
✓	ⓘ	CLEANING_GENERAL	Cleaning (general)
	ⓘ	CLEANING_MAJOR	Cleaning (major)
	ⓘ	CLEANING_MINOR	Cleaning (minor)
✓		USED	Used

Figure 48: Trigger Selection editor

- To add a trigger as option to the phase, select it as **Allowed**.
- To remove a trigger from the options displayed on the phase, unselect it.

Graphical Elements

The following lists contain all buttons, marker icons, and cursors used in Recipe and Workflow Designer.

Toolbar Buttons

Action buttons available in the toolbars:

Change status
Located on the Status toolbar, it triggers a status change for the component that is currently active in the upper tab bar (batch master recipe (see "Changing the Status of Master Recipes" in Vol. 2), device master recipe (see "Changing the Status of Master Recipes" in Vol. 3), master workflow (see "Changing the Status of Master Workflows" in Vol. 4), batch change request (see "Changing the Status of Change Requests" in Vol. 2), device change request (see "Changing the Status of Change Requests" in Vol. 3), workflow change request (see "Changing the Status of Change Requests" in Vol. 4), building block (page 29)).

Close
Located on the File toolbar, it closes the main component tab that is currently active in the upper tab bar. This action also closes all of its subordinate components tabs that may be open in the lower tab bar.

Help
Located on the Help toolbar as well as on some dialogs, it opens the context-sensitive help system of Recipe Designer - Batch (see "Help Access" in Vol. 2), Recipe Designer - Device (see "Help Access" in Vol. 3), or Workflow Designer (see "Help Access" in Vol. 4).

Input
Located on the Setlist toolbar, it inserts a local input material for the selected component.

Invert drawing direction
Located on the Setlist toolbar, it inverts the drawing direction of the graph for the next component. The graph will now grow up and leftwards instead of down and rightwards.

Join

Located on both the Setlist and the process workflow toolbars, it joins the open branches of the graph at the selected step. If there is no unique join operation possible, the system does not join any branches.

Loop

Located on the process workflow toolbar, it inserts a loop from the currently selected step to the next step you select.

New master recipe

Located on the File toolbar of Recipe Designer, it creates a new master recipe, thus first opening the **Select Material** dialog (batch (see "Select Dialog" in Vol. 2) or device (see "Select Dialog" in Vol. 3)) then the **New Master Recipe** dialog to define the recipe's identifier, and afterwards a new blue tab in the upper tab bar.

New master workflow

Located on the File toolbar of Workflow Designer, it creates a new master workflow thus first opening the **New Master Workflow** dialog and afterwards a new blue tab in the upper tab bar.

New operation

Located on the File toolbar, it creates a new operation, thus first opening the **New Operation** dialog (page 15) and afterwards a new yellow tab in the upper tab bar.

New phase

Located on the File toolbar, it creates a new phase, thus first opening the **New Phase** dialog (page 15) and afterwards a new yellow tab in the upper tab bar.

New procedure

Located on the File toolbar, it creates a new procedure, thus first opening the **New Procedure** dialog (page 15) and afterwards a new yellow tab in the upper tab bar.

New unit procedure

Located on the File toolbar, it creates a new unit procedure, thus first opening the **New Unit Procedure** dialog (page 15) and afterwards a new yellow tab in the upper tab bar.

Open

Located on the File toolbar, it opens the **Open** dialog of Recipe Designer - Batch (see "Open Dialog" in Vol. 2), Recipe Designer - Device (see "Open Dialog" in Vol. 3), or Workflow Designer (see "Open Dialog" in Vol. 4) to select a component for opening it in the Graph Window.

Open graph pagination

Located on the File toolbar, it opens the **Graph Pagination** dialog to display the tiled print preview for the component graph that is currently active in the Graph Window. Only available for master recipes and master workflows.

Open universe

Located on the main toolbar, it opens the Universe of Recipe Designer - Batch (see "Universe" in Vol. 2), Recipe Designer - Device (see "Universe" in Vol. 3), or Workflow Designer (see "Universe" in Vol. 4) to select a component for loading it into the Setlist.

Output

Located on the Setlist toolbar, it inserts a local output material for the selected component.

Redo

Located on the Edit toolbar, it redoes the last action revoked with the **Undo** action. You can redo up to 100 actions, thus you can step by step redo the last 100 actions you have revoked before.

Replace

Located on the Setlist toolbar, it replaces the component assigned to the currently selected graph step with the component you select from the Setlist. It performs a full replacement, which means that all configurations made with the process parameters of the replaced graph component are overwritten and thus lost.

Save

Located on the File toolbar, it saves all changes made to the main component that is currently active in the upper tab bar. This action saves all of its subordinate components that may be open in the lower tab bar.

Selection branch

Located on both the Setlist and the process workflow toolbars, it inserts a selection branch at the currently selected step.

Sequence
Located on both the Setlist and the process workflow toolbars, it inserts the next step in sequence at the currently selected step.

Simultaneous branch
Located on both the Setlist and the process workflow toolbars, it inserts a simultaneous branch at the currently selected step.

Smart replace
Located on the Setlist toolbar, it replaces the component assigned to the currently selected graph step with the component you select from the Setlist. Contrary to the full replacement of the **Replace** action, the **Smart replace** action in Recipe Designer - Batch (see "Setlist" in Vol. 2), Recipe Designer - Device (see "Setlist" in Vol. 3), and Workflow Designer (see "Setlist" in Vol. 4) merges the data configured with the parameters of the new replacing building block into the data of the replaced graph component.

Undo
Located on the Edit toolbar, it revokes the last action you have performed. You can undo up to 100 actions, thus you can step by step revoke the last 100 actions you have performed.

View status history
Located on the Status toolbar, it opens the **Status History** window to display a list of all status changes that have been performed on the component that is currently active in the upper tab bar.
Only available for batch master recipes (see "Status History of Master Recipes" in Vol. 2), device master recipes (see "Status History of Master Recipes" in Vol. 3), master workflows (see "Status History of Master Workflows" in Vol. 3), and change requests (batch (see "Status History of Change Requests" in Vol. 2), device (see "Status History of Change Requests" in Vol. 3), or workflow (see "Status History of Change Requests" in Vol. 4)).

Zoom in
Located on the View toolbar, it zooms in on the graph in the currently active tab, doubling its display size.

Zoom out
Located on the View toolbar, it zooms out from the graph in the currently active tab, reducing its size to half of its previous display size.

Zoom to 100%
Located on the View toolbar, it resets the zoom factor of the currently active graph to its default value.

Dialog Buttons

Action buttons available on the dialogs of Recipe and Workflow Designer:

Add ...
Located on an equipment requirement parameter in the Parameter Panel, it opens an option list that holds the property types contained in the Setlist, to refine existing or add further properties.
Located on the **Trigger-enabled** capability in the Parameter Panel of an operation, it adds a new row to the list of triggers, to reference another trigger phase that triggers new runs of the operation.

Add parameter bundle
Only available for specific phases.
Located on the Parameter Panel for process parameters with parameter bundles, it opens an option list that holds all bundle types available for the phase. Once you have selected the type, it opens the **Add <Bundle Type>** dialog to define the bundle's identifier. Afterwards the system adds all process parameters of the bundle to the list of parameters.

Compile usage list
Located on the **Usage List** tab of a change request, it determines all occurrences of the old building block and displays them in list form.

Delete cell content
Located in a focused table cell of a Property Window, it deletes the cell content. It is only available for cells that can only be filled by way of a cell editor and are thus not directly editable.

Execute change request
Located on the **Action List** tab of a change request, it starts the execution of the change request.

Execute comparison
Located in the Comparison window, it executes the comparison of the current component with its baseline.

Lock
Located on the Parameter Panels for material parameters, equipment requirements, work center assignments, privilege parameters, capabilities, process parameters, or transitions, it locks the selected or multi-selected parameters or transitions, respectively.

Open cell editor
Located in a focused table cell of the Parameter Panel or a Property Window, it opens a suitable cell editor for editing the content of a cell. If a cell contains an expression, the **fx** button remains visible as marker at all times.

Refresh from database
Located on the **Search** panel of the data tools (page 130), it synchronizes the database snapshot taken at the beginning of the search and filter operation with the central database.

Remove parameter
Located on the Parameter Panels for material parameters, equipment requirements, work center assignments, privilege parameters, capabilities, or process parameters with parameter bundles, it removes the selected or multi-selected parameters from the list.

Toggle added objects
Located in the Comparison window, it toggles the display of difference entries that indicate that objects were added.

Toggle changed objects
Located in the Comparison window, it toggles the display of difference entries that indicate that objects were changed.

Toggle error messages
Located in the Messages window, it toggles the display of error messages.

Toggle information messages
Located in the Messages window, it toggles the display of information messages.

Toggle removed objects
Located in the Comparison window, it toggles the display of difference entries that indicate that objects were removed.

Toggle warning messages
Located in the Messages window, it toggles the display of warning messages.

Unlock
Located on the Parameter Panels for material parameters, equipment requirements, work center assignments, privilege parameters, capabilities, process parameters, or transitions, it unlocks the selected or multi-selected parameters or transitions, respectively.

Marker Icons

Marker icons displayed in Recipe and Workflow Designer:

Abort-and-reactivate-enabled
Displayed on unit procedure building block images in the Graph Window, it indicates that the operation holds the **Abort-and-reactivate-enabled** capability.
Displayed in the **Capability** table of the Setlist, to the left of the **Abort-and-reactivate-enabled** capability.

Added
Displayed in the list panel of the Comparison window, it indicates that the comparison has detected that an object was added.

Approved
Displayed in the tree view of the Explorer and on the individual building block images in the Graph Window, it indicates that the source of a master recipe, master workflow, or building block element is an **Approved** building block. Thus the element's structure cannot be modified, only its unlocked parameters and transitions, along with its properties, such as identifier and description.

Auto-startable
Displayed on operation building block images in the Graph Window, it indicates that the operation holds the **Auto-startable** capability.
Displayed in the **Capability** table of the Setlist, to the left of the **Auto-startable** capability.

Cancelation
Displayed in the status information on the **Action List** tab of a change request, it indicates that the marked action was canceled and no changes actions were performed.

Changed
Displayed in the list panel of the Comparison window, it indicates that the comparison has detected that an object was changed.

Dependent property
Displayed in the list of property types available for adding to an equipment requirement parameter, it indicates that the property type is contained in the requirement's equipment class and is thus available for further refining.

Detachable
Displayed on operation or unit procedure building block images in the Graph Window, it indicates that the operation holds the **Detachable** capability.
Displayed in the **Capability** table of the Setlist, to the left of the **Detachable** capability.

Direct resolution available for error, warning, or information

Displayed in the list panel of the Messages window, it indicates that for an error, warning, or information message a direct solution is available by right-clicking the message to open a corresponding shortcut menu in Recipe Designer - Batch (see "Shortcut Menus" in Vol. 2), Recipe Designer - Device (see "Shortcut Menus" in Vol. 3), or Workflow Designer (see "Shortcut Menus" in Vol. 4).

Error

- Displayed in the list panel of the Messages window, it indicates that the concurrent checks run on the active master recipe, master workflow, or building block have returned an error.
- Displayed in the status information on the **Action List** tab of a change request, it indicates that the marked action has encountered an error during its execution.

Frozen
Displayed in the **Lock** column of the Parameter Panel and on the Expression editor for transition conditions, it indicates that a parameter or transition is locked and located in an **Approved** building block which has been drawn as element into a higher-level structure (recipe, workflow, or building block). The parameter or transition is locked permanently and cannot be unlocked.

Has condition
Displayed on the individual transition images in the Graph Window, it indicates that a specific condition has been defined for the transition. Double-click the transition image to view or edit the condition in the Expression editor (page 45).

Information
Displayed in the list panel of the Messages window, it indicates that the concurrent checks run on the active master recipe, master workflow, or building block have returned an information message.

Operation hierarchy level
Displayed in the **Capability** table of the Setlist, in the **Levels** column, it indicates that the capability can be assigned to operations.

Progress
Displayed in the result column on the **Action List** tab of a change request, it indicates that the execution of the marked action is in progress.

Removed
Displayed in the list panel of the Comparison window, it indicates that the comparison has detected that an object was removed.

Success
Displayed in the status information on the **Action List** tab of a change request, it indicates that the marked action completed successfully.

Unit procedure hierarchy level
Displayed in the **Capability** table of the Setlist, in the **Levels** column, it indicates that the capability can be assigned to unit procedures.

Warning

- Displayed in the list panel of the Messages window, it indicates that the concurrent checks run on the active master recipe, master workflow, or building block have returned a warning.
- Displayed in the status information on the **Action List** tab of a change request, it indicates that the marked action has encountered a warning during its execution.
- Displayed on the Trigger Selection editor (page 95), it indicates that the marked trigger does not occur in all graphs of the selected purpose.

Cursors

Cursors displayed while working in the Graph Window:

Draw
Indicates that

- you can click and drag to draw a link
- you are in the loop drawing mode.

Draw endpoint
Indicates that you can click to select a component as endpoint of a link or loop.

Draw endpoint unallowed
Indicates that you cannot select a component as endpoint of a link or loop.

Move
Indicates that you are moving selected components.

Pan
Indicates that you can click and drag to pan the Graph Window.

Select
Indicates that you can

- click to select the component over which the cursor hovers
- SHIFT-click and drag to draw a marquee for selecting a group of components.

Select parameter
Indicates that you are hovering over a parameter button of a step.

Wait
Indicates that the system is currently busy.

Expression Editor Icons

Icons displayed in the Expression editor (page 44):

Constants
Displayed as section heading in the auto-completion feature of the expression panel (page 47), it indicates that the listed options are constants.

Error
Displayed in the expression panel (page 47), it indicates that the expression contains one or more errors. To open a tooltip that lists the errors, hover over the icon.

Functions
Displayed at the root node of the functions tree (page 67) or as section heading in the

auto-completion feature of the expression panel (page 47), it indicates that the listed options are functions.

Operations
Displayed as section heading in the auto-completion feature of the expression panel (page 47), it indicates that the listed options are operations.

Operators
Displayed at the root node of the operators tree (page 60) or as section heading in the auto-completion feature of the expression panel (page 47), it indicates that the listed options are operators.

Outputs
Displayed as section heading in the auto-completion feature of the expression panel (page 47), it indicates that the listed options are outputs.

Phases
Displayed as section heading in the auto-completion feature of the expression panel (page 47), it indicates that the listed options are phases.

Separators
Displayed as section heading in the auto-completion feature of the expression panel (page 47), it indicates that the listed options are separators.

Unit procedures
Displayed as section heading in the auto-completion feature of the expression panel (page 47), it indicates that the listed options are unit procedures.

Warning
Displayed in the auto-completion feature of the expression panel (page 47), it indicates that the unit procedure, operation, or phase listed as possible reference source is located on a parallel branch of the graph. Thus it may still be in process when the system tries to retrieve its output and would consequently return **Undefined**.

Input Boxes

Input boxes can have different background colors that assist you with filling in the data of a form. This behavior applies to forms and signature dialogs.

The different background colors indicate the following:

⬜

White
An unfilled input box. A blinking cursor indicates that it has the focus and you can start to type your entry.

🟨

Yellow
A mandatory box that must be filled before the form can be saved or closed with the **OK** button. A blinking cursor indicates that it has the focus and you can start to type your entry.

Chapter 5

Keyboard Operation

Control by keyboard is primarily necessary for navigation and editing purposes.

Screen Area Shortcuts

Use the following keys and keyboard shortcuts to navigate the screen areas of Recipe and Workflow Designer:

Explorer/Messages Tree:

- ENTER/SPACEBAR
  Triggers a single-click action that updates connected screen areas, Graph Window for the Explorer tree, breadcrumbs and list panel for the Messages Window tree.

- LEFT ARROW/NUMPAD -
  Collapses the selected tree node or, if the node is collapsed, moves the focus to its next parent object.

- RIGHT ARROW/NUMPAD +
  Expands the selected tree node or, if the node is expanded, moves the focus to its first child object.

- UP/DOWN ARROW
  Navigates between the tree nodes.

Graph Window:

- ALT+LEFT ARROW
  Opens or moves to the previously active tab within the currently active master recipe. Step by step, you can backtrack up to the first tab you have opened.

- ALT+RIGHT ARROW
  Becomes active after you have used the ALT+LEFT ARROW shortcut. Step by step, it revokes the action of backtracking to the previously active tab, thus opening or moving to the respective tabs. Once you have reached the tab from which you started the initial backtracking, the shortcut becomes inactive.

- ALT+UP ARROW
  Opens or moves to the tab that is one hierarchy level up from the active tab.

- CTRL+1
  Resets the zoom factor of the currently active graph to its default value.
- CTRL+MINUS
  Zooms out from the graph in the currently active tab, reducing its size to half of its previous display size.
- CTRL+PERIOD
  Sets the zoom factor of the currently active graph to fit the graph on the available screen space of the Graph Window.
- CTRL+PLUS
  Zooms in on the graph in the currently active tab, doubling its display size.
- CTRL+LEFT ARROW
  In the lower tab bar, moves to the next tab to the left of the active tab. If there is no tab to the left, the selection wraps and starts at the rightmost tab of the tab bar.
- CTRL+RIGHT ARROW
  In the lower tab bar, moves to the next tab to the right of the active tab. If there is no tab to the right, the selection wraps and starts at the leftmost tab of the tab bar.

Parameter Panel:

- CTRL+MINUS
  In the **Results** list, collapses all process parameter sub-tables.
- CTRL+PLUS
  In the **Results** list, expands all process parameter sub-tables.

Framework Navigation Shortcuts

Use the following keyboard shortcuts to navigate the general framework of Recipe and Workflow Designer:

- ALT+B
  Toggles the display of the **Source** property window of the currently active component in Recipe Designer - Batch (see "Source Property Window" in Vol. 2), Recipe Designer - Device (see "Source Property Window" in Vol. 3), or Workflow Designer (see "Source Property Window" in Vol. 4).
- ALT+C
  Opens the **Material Flow Control** tab for the currently active batch (see "MFC Data" in Vol. 2) or device (see "MFC Data" in Vol. 3) recipe, workflow (see "MFC Data" in Vol. 4), or procedure building block.

- ALT+E
  Toggles the display of the Explorer for the currently active component in Recipe Designer - Batch (see "Explorer" in Vol. 2), Recipe Designer - Device (see "Explorer" in Vol. 3), or Workflow Designer (see "Explorer" in Vol. 4).

- ALT+F1
  Opens the start page of the help system of Recipe Designer - Batch (see "Help Access" in Vol. 2), Recipe Designer - Device (see "Help Access" in Vol. 3), or Workflow Designer (see "Help Access" in Vol. 4).

- ALT+F4
  Closes the application window.

- ALT+H
  Toggles the display of the **Header** property window of the currently active component in Recipe Designer - Batch (see "Header Property Window" in Vol. 2), Recipe Designer - Device (see "Header Property Window" in Vol. 3), or Workflow Designer (see "Header Property Window" in Vol. 4).

- ALT+M
  Toggles the display of the Map for the currently active component in Recipe Designer - Batch (see "Map" in Vol. 2), Recipe Designer - Device (see "Map" in Vol. 3), or Workflow Designer (see "Map" in Vol. 4).

- ALT+O
  Toggles the display of the Comparison window for the currently active component in Recipe Designer - Batch, Recipe Designer - Device, or Workflow Designer.

- ALT+P
  Toggles the display of the Phase Preview for the currently active component in Recipe Designer - Batch, Recipe Designer - Device, or Workflow Designer.

- ALT+R
  Toggles the display of the **Element** property window of the currently active component in Recipe Designer - Batch (see "Element Property Window" in Vol. 2), Recipe Designer - Device (see "Element Property Window" in Vol. 3), or Workflow Designer (see "Element Property Window" in Vol. 4).

- ALT+S
  Toggles the display of the Setlist for your current session of Recipe Designer - Batch (see "Setlist" in Vol. 2), Recipe Designer - Device (see "Setlist" in Vol. 3), or Workflow Designer (see "Setlist" in Vol. 4).

- ALT+U
  Opens the Universe for your current session of Recipe Designer - Batch (see "Universe" in Vol. 2), Recipe Designer - Device (see "Universe" in Vol. 3), or Workflow Designer (see "Universe" in Vol. 4) to select a component for loading it into the Setlist.

- ALT+V
  Toggles the display of the Messages window for the currently active component in Recipe Designer - Batch, Recipe Designer - Device, or Workflow Designer.

Action Shortcuts

Use the following keyboard shortcuts to handle the actions available for objects in Recipe and Workflow Designer:

- CTRL+A
  Selects all components of the currently active SFC or MFC graph.

- CTRL+ALT+F4
  Closes all main component tabs except for the main component tab that is currently active in the upper tab bar.

- CTRL+D
  Unselects all current selections made in the currently active SFC or MFC graph.

- CTRL+E
  In an MFC graph tab, merges all graph nodes whose merge targets can be determined unambiguously by the system.

- CTRL+F
  In an SFC graph tab, opens the **Find Component** dialog to type a sequence of characters to search for in the components of the currently open graph.

- CTRL+F4
  Closes the main component tab that is currently active in the upper tab bar. This action also closes all of its subordinate components tabs that may be open in the lower tab bar.

- CTRL+F12
  Opens the **Save <Component Type> as** dialog to save the main component that is currently active in the upper tab bar under a new identifier. Along with saving the main component, this action also saves all of its subordinate components that may be open in the lower tab bar.

- CTRL+G
  Toggles the display of grid lines in the Graph Window.

- CTRL+H
  - For batch master recipes, opens the **Change Status** dialog to perform a status change (see "Changing the Status of Master Recipes" in Vol. 2) on the master recipe that is currently active in the upper tab bar.
  - For device master recipes, opens the **Change Status** dialog to perform a status change (see "Changing the Status of Master Recipes" in Vol. 3) on the master recipe that is currently active in the upper tab bar.
  - For master workflows, opens the **Change Status** dialog to perform a status change (see "Changing the Status of Master Workflows" in Vol. 4) on the master workflow that is currently active in the upper tab bar.
  - For batch change requests, opens the **Change Status** dialog to perform a status change (see "Changing the Status of Change Requests" in Vol. 2) on the change request that is currently active in the upper tab bar.
  - For device change requests, opens the **Change Status** dialog to perform a status change (see "Changing the Status of Change Requests" in Vol. 3) on the change request that is currently active in the upper tab bar.
  - For workflow change requests, opens the **Change Status** dialog to perform a status change (see "Changing the Status of Change Requests" in Vol. 4) on the change request that is currently active in the upper tab bar.
  - For building blocks, opens an **Electronic Signature** dialog to perform and sign a status change (page 29) on the building block that is currently active in the upper tab bar.

- CTRL+I
  Inverts all selections made in the currently active SFC or MFC graph. This action unselects all selected components and selects all unselected components.

- CTRL+L
  - In an SFC graph tab, selects all links of the currently active graph.
  - In the list of parameters in a Parameter Panel in Recipe Designer - Batch (see "Parameter Panel" in Vol. 2), Recipe Designer - Device (see "Parameter Panel" in Vol. 3), or Workflow Designer (see "Parameter Panel" in Vol. 4), locks the selected parameters or transitions, respectively.

- CTRL+M
  In an MFC graph tab, merges the selected graph nodes to form a transfer or a confluence.

- CTRL+O
  - In an SFC graph tab of Recipe Designer - Batch (see "Open Dialog" in Vol. 2), Recipe Designer - Device (see "Open Dialog" in Vol. 3), or Workflow Designer (see "Open Dialog" in Vol. 4), opens the **Open** dialog to select a component to be loaded into the Graph Window.
  - In an MFC graph tab, merges the selected graph node with the final output of the graph.
- CTRL+P
  - For master recipes, opens the **Master Recipe Report** of the master recipe that is currently active in the upper tab bar.
    Only available for saved master recipes.
  - For master workflows, opens the **Master Workflow Report** of the master workflow that is currently active in the upper tab bar.
    Only available for saved master workflows.
- CTRL+Q
  Opens the **Status History** window to display a list of all status changes that have been performed on the component that is currently active in the upper tab bar.
  Only available for batch master recipes (see "Status History of Master Recipes" in Vol. 2), device master recipes (see "Status History of Master Recipes" in Vol. 3), master workflows (see "Status History of Master Workflows" in Vol. 3), and change requests (batch (see "Status History of Change Requests" in Vol. 2), device (see "Status History of Change Requests" in Vol. 3), or workflow (see "Status History of Change Requests" in Vol. 4)).
- CTRL+R
  - In an SFC graph tab, opens the **Graph Pagination** dialog to display the tiled print preview for the currently active graph.
    Only available for master recipes and master workflows.
  - In an MFC graph tab, resets the MFC graph to its initial, unmerged state.
- CTRL+S
  Saves all changes made to the main component that is currently active in the upper tab bar. This action also saves all of its subordinate components that may be open in the lower tab bar.
- CTRL+SHIFT+C
  Creates a new change request, thus first opening the **New Change Request** dialog and afterwards a new orange tab in the upper tab bar.

- CTRL+SHIFT+F
  Creates a new phase, thus first opening the **New Phase** dialog (page 15) and afterwards a new yellow tab in the upper tab bar.

- CTRL+SHIFT+F4
  Closes all main components tabs that are currently open in the upper tab bar. This action also closes all of their subordinate components tabs that may be open in the lower tab bar.

- CTRL+SHIFT+H
  Only available for saved master recipes, master workflows, change requests, or building blocks, performs an internal preparation on a complete component that has passed all checks. The preparation is necessary to make the component available for status changes.

- CTRL+SHIFT+L
  In an SFC graph tab, unselects all links of the currently active graph.

- CTRL+SHIFT+M
  Creates a new master recipe, thus first opening the **Select material** (batch (see "Select Dialog" in Vol. 2) or device (see "Select Dialog" in Vol. 3)), then the **New Master Recipe** dialog to define the recipe's identifier, and afterwards a new blue tab in the upper tab bar.

- CTRL+SHIFT+O
  Creates a new operation, thus first opening the **New Operation** dialog (page 15) and afterwards a new yellow tab in the upper tab bar.

- CTRL+SHIFT+P
  - For master recipes, opens the **Master Recipe Report** of the master recipe that is currently active in the upper tab bar. The report does not include the **Comparison with Baseline** section.
    Only available for saved master recipes.
  - For master workflows, opens the **Master Workflow Report** of the master workflow that is currently active in the upper tab bar. The report does not include the **Comparison with Baseline** section.
    Only available for saved master workflows.

- CTRL+SHIFT+R
  Creates a new procedure, thus first opening the **New Procedure** dialog (page 15) and afterwards a new yellow tab in the upper tab bar.

- CTRL+SHIFT+S
  Saves all changes made to all main components that are currently open in the upper tab bar. This action also saves all of their subordinate components that may be open in the lower tab bar.
- CTRL+SHIFT+T
  In an SFC graph tab, unselects all steps and transitions of the currently active graph.
- CTRL+SHIFT+U
  Creates a new unit procedure, thus first opening the **New Unit Procedure** dialog (page 15) and afterwards a new yellow tab in the upper tab bar.
- CTRL+SHIFT+W
  Creates a new master workflow, thus first opening the **New Master Workflow** dialog and afterwards a new blue tab in the upper tab bar.
- CTRL+T
  - In an SFC graph tab, selects all steps and transitions of the currently active graph.
  - In an MFC graph tab, splits the selected graph component.
- CTRL+U
  In the list of parameters in a Parameter Panel in Recipe Designer - Batch (see "Parameter Panel" in Vol. 2), Recipe Designer - Device (see "Parameter Panel" in Vol. 3), or Workflow Designer (see "Parameter Panel" in Vol. 4), unlocks the selected parameters or transitions, respectively.
- CTRL+W
  In the Graph Window, closes the tab that is currently active in the lower tab bar.
- CTRL+Y
  In an SFC graph tab, redoes the last action revoked with the **Undo** action. You can redo up to 100 actions, thus you can step by step redo the last 100 actions you have revoked.
- CTRL+Z
  In an SFC graph tab, revokes the last action you have performed. You can undo up to 100 actions, thus you can step by step revoke the last 100 action you have performed.

TIP
Please note that the stepwise **Undo** function does not work on changes you have performed on parameters through the Parameter Panel.

- DEL
  - In an SFC graph tab, deletes all currently selected components.
  - In the list of parameters in a Parameter Panel in Recipe Designer - Batch (see "Parameter Panel" in Vol. 2), Recipe Designer - Device (see "Parameter Panel" in Vol. 3), or Workflow Designer (see "Parameter Panel" in Vol. 4), deletes the selected parameter.
  - In the building block panel of a change request in Recipe Designer - Batch (see "Change Requests for Mass Changes" in Vol. 2), Recipe Designer - Device (see "Change Requests for Mass Changes" in Vol. 3), or Workflow Designer (see "Change Requests for Mass Changes" in Vol. 4), deletes the selected building block.
- F1
  Opens the context-sensitive help system of Recipe Designer - Batch (see "Help Access" in Vol. 2), Recipe Designer - Device (see "Help Access" in Vol. 3), or Workflow Designer (see "Help Access" in Vol. 4).
- F2
  In an SFC graph tab, makes the identifier of the currently selected component available for editing. Press the ESC key to cancel this action.
- F3
  In an SFC graph tab, finds the next occurrence in a component of the characters you typed in the **Find Component** dialog box.
- F4
  - In the Parameter Panel, for attributes that provide an editor for input support, opens the suitable editor.
  - In the Property Windows, for properties that provide an editor for input support, opens the suitable editor.
- F12
  In an SFC graph tab, opens the **Create <Object Type>** dialog where you can save your parameterized building block to make it available for selection from the Universe.

- INS
  - In an SFC graph tab, inserts a new unconnected step at the current cursor position.
  - In a Parameter Panel for process parameters with parameter bundles in Recipe Designer - Batch (see "Parameter Panel" in Vol. 2), Recipe Designer - Device (see "Parameter Panel" in Vol. 3), or Workflow Designer (see "Parameter Panel" in Vol. 4), opens the option list for selecting a new process parameter bundle to be inserted.

Expression Editor Shortcuts

Use the following keyboard shortcuts to manipulate text in the expression panel:

- ALT+F3
  In the expression panel, opens the quick search component.
- BACKSPACE
  In the expression panel, deletes the character to the left of the cursor.
- CTRL+A
  In the expression panel, selects all characters.
- CTRL+B
  In the expression panel, selects the current expression segment between the nearest brackets.
- CTRL+BACKSPACE
  In the expression panel, deletes the characters to the left of the cursor up to the next word.
- CTRL+C
  In the expression panel, copies the selected characters to the clipboard.
- CTRL+D
  In the expression panel, duplicates the selected characters.
- CTRL+DEL
  In the expression panel, deletes the characters to the right of the cursor up to the next word.
- CTRL+END
  In the expression panel, moves the cursor to the end of the text.

- CTRL+ENTER
  In the expression panel, inserts a line break after the current cursor position.
- CTRL+F
  In the expression panel, opens the **Find Text** dialog.
- CTRL+G
  In the expression panel, opens the **Go to line number** dialog to move the cursor to a specific line.
- CTRL+H
  In the expression panel, opens the **Replace Text** dialog.
- CTRL+HOME
  In the expression panel, moves the cursor to the beginning of the text.
- CTRL+LEFT ARROW
  In the expression panel, moves the cursor to the preceding word to the left.
- CTRL+MINUS
  In the expression panel, collapses all sections.
- CTRL+PLUS
  In the expression panel, expands all sections.
- CTRL+R
  In the expression panel, deletes the line in which the cursor is currently positioned.
- CTRL+RIGHT ARROW
  In the expression panel, moves the cursor to the next word to the right.
- CTRL+SHIFT+END
  In the expression panel, selects all characters from the current cursor position to the end of the text.
- CTRL+SHIFT+HOME
  In the expression panel, selects all characters from the current cursor position to the beginning of the text.
- CTRL+SHIFT+J
  In the expression panel, joins the current line with the following line.
- CTRL+SHIFT+LEFT ARROW
  In the expression panel, selects the word preceding the cursor position to the left.

- CTRL+SHIFT MINUS
  In the expression panel, collapses the section in which the cursor is currently positioned.
- CTRL+SHIFT+PLUS
  In the expression panel, expands the section in which the cursor is currently positioned.
- CTRL+SHIFT+RIGHT ARROW
  In the expression panel, selects the word following the cursor position to the right.
- CTRL+SHIFT+U
  In the expression panel, toggles the case of the currently selected letters to all uppercase or all lowercase.
- CTRL+SHIFT+V
  In the expression panel, opens a dialog to select one of the previous clipboards.
- CTRL+SPACEBAR
  In the expression panel, opens the intelligent auto-completion feature to list potential references or operators to insert.
- CTRL+V
  In the expression panel, pastes characters from the clipboard.
- CTRL+W
  In the expression panel, selects the word at the current cursor position.
- CTRL+X
  In the expression panel, cuts the selected characters and writes them to the clipboard.
- CTRL+Y
  In the expression panel, redoes the last action you have revoked with CTRL+Z. You can redo up to 100 actions, thus you can step by step redo the last 100 actions you have revoked.
- CTRL+Z
  In the expression panel, revokes the last action you have performed. You can undo up to 100 actions, thus you can step by step revoke the last 100 action you have performed.
- DEL
  In the expression panel, deletes the character to the right of the cursor.

- DOWN
  In the expression panel, moves the cursor down to the next lower line.
- END
  In the expression panel, moves the cursor to the end of the line.
- ENTER
  In the expression panel, inserts a line break.
  If the auto-completion feature is active, it inserts the selected option.
- F3
  In the expression panel, searches the next occurrence of the text specified in the **Find Text** dialog.
- HOME
  In the expression panel, moves the cursor to the beginning of the line.
- INSERT
  In the expression panel, toggles the insert/overwrite input modes.
- LEFT ARROW
  In the expression panel, moves the cursor to the preceding character to the left.
- PAGE DOWN
  In the expression panel, moves the cursor one page down.
- PAGE UP
  In the expression panel, moves the cursor one page up.
- RIGHT ARROW
  In the expression panel, moves the cursor to the next character to the right.
- SHIFT+DOWN
  In the expression panel, selects all characters from the current cursor position to the next line.
- SHIFT+END
  In the expression panel, selects all characters from the current cursor position to the end of the line.
- SHIFT+ENTER
  In the expression panel, inserts a new line after the current line.

- SHIFT+F3
  In the expression panel, searches the previous occurrence of the text specified in the **Find Text** dialog.
- SHIFT+HOME
  In the expression panel, selects all characters from the current cursor position to the beginning of the line.
- SHIFT+LEFT ARROW
  In the expression panel, selects the character preceding the current cursor position to the left.
- SHIFT+PAGE DOWN
  In the expression panel, selects all characters from the current cursor position one page down.
- SHIFT+PAGE UP
  In the expression panel, selects all characters from the current cursor position one page up.
- SHIFT+RIGHT ARROW
  In the expression panel, selects the character following the current cursor position to the right.
- SHIFT+TAB
  In the expression panel, resets the indent of indented lines.
- SHIFT+UP
  In the expression panel, selects all characters from the current cursor position to the preceding line.
- TAB
  In the expression panel, inserts a tab character. The system provides tab stops at approximately every fifth character.
  If you have text selected, the entire line or lines containing the selected text is/are indented.
- UP
  In the expression panel, moves the cursor up to the next higher line.

Dialog Boxes

The system displays information messages, warnings, error messages, and some types of signature requests as modal dialog boxes on top of the application. The following keys are available for navigating and operating dialog boxes:

- ENTER
  For buttons, triggers a single-click action on the default button.
- ESC
  Cancels the dialog and closes the dialog box.
- SPACEBAR
  For buttons, triggers a single-click action on the focused button.
- TAB
  Navigates along the defined navigation path, moving the focus between input boxes and buttons.

Chapter 6

Basic Operations

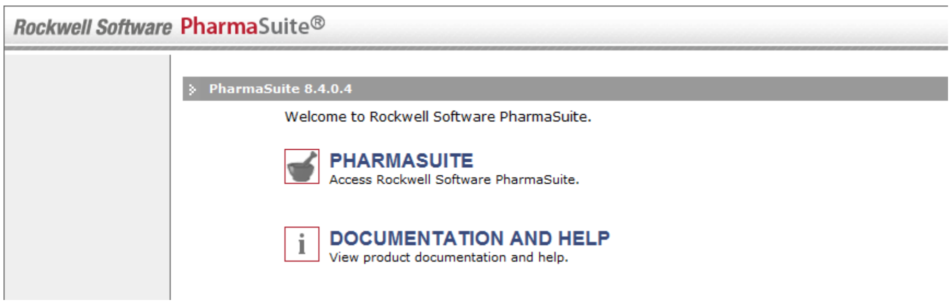
The following sections describe basic and recurring operations and functions in Recipe and Workflow Designer.

Start, Login, Logout, and Password Change

Before you can start working with PharmaSuite your system administrator must have created a user account for you. The PharmaSuite administrator will inform you of your login name and initial password.
Depending on your company policy you may be forced to change your password when you log in for the first time. In this case the system will display a message that indicates that your password has expired. Then the system will prompt you to change it (page 128).

Start PharmaSuite

To start PharmaSuite double-click the respective icon on the user interface or select it from the start menu. The system displays the webstart page in a browser window, from which you can select to start the application or view either the help system or the documentation.



Welcome to Rockwell Software PharmaSuite.

Access Rockwell Software PharmaSuite.

View product documentation and help.

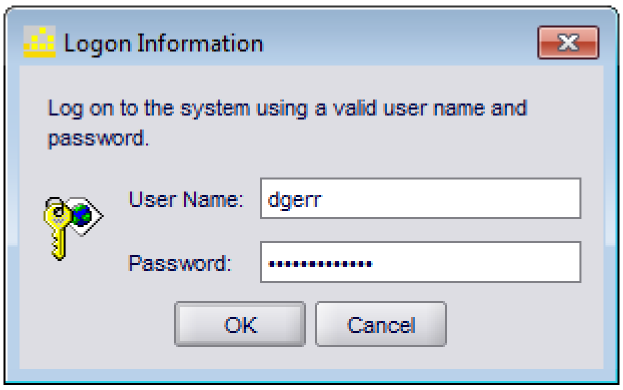
Figure 49: PharmaSuite webstart page

Login

When you select to start PharmaSuite it runs through an initialization phase in the course of which you will also see the splash screen of Shop Operations, which is the internal platform of PharmaSuite. As soon as the initialization phase has been completed, the login form for user login appears.

The login form contains two mandatory fields, one for the login name and one for the password. Your login name and your password are unique for all PharmaSuite applications and are linked to your role and user privileges.

Type your login name and password in the respective boxes. Please note that your password is masked by asterisks (*). Click the **OK** button to complete the login procedure. If your login attempt is not successful, a message appears and you have to repeat the procedure.


Logon Information

Log on to the system using a valid user name and password.

User Name: dgerr

Password: ••••••••••••••

Figure 50: Login form

After you have successfully logged in, the system displays the PharmaSuite welcome page. From here you can start the Production Execution, Production Management, Data Manager, Recipe and Workflow Designer, and Production Responses applications, change your password and work station, or access the system documentation and help.

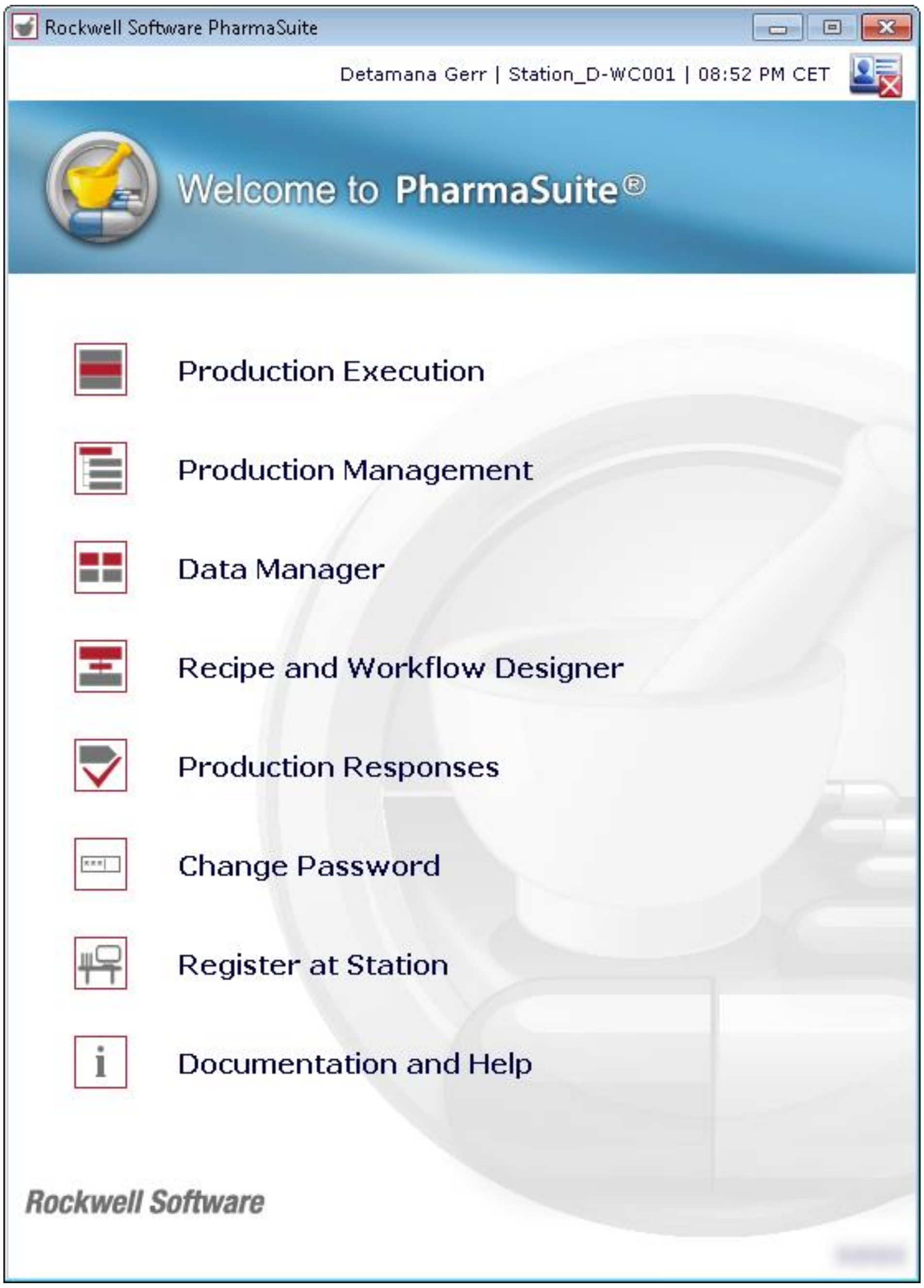


Figure 51: PharmaSuite welcome page

TIP

Please note that logins can be linked to access rights, which means that you can only start an application if your system administrator has assigned the suitable access privileges to you.
Some logins, especially in the production execution environment, are directly connected to an application and work station. This means that the welcome page will be skipped and the application will start directly after you have successfully logged in.

Logout

In Recipe and Workflow Designer, from the **File** menu, select the **Exit** function or use the ALT+F4 keyboard shortcut to quit the application.

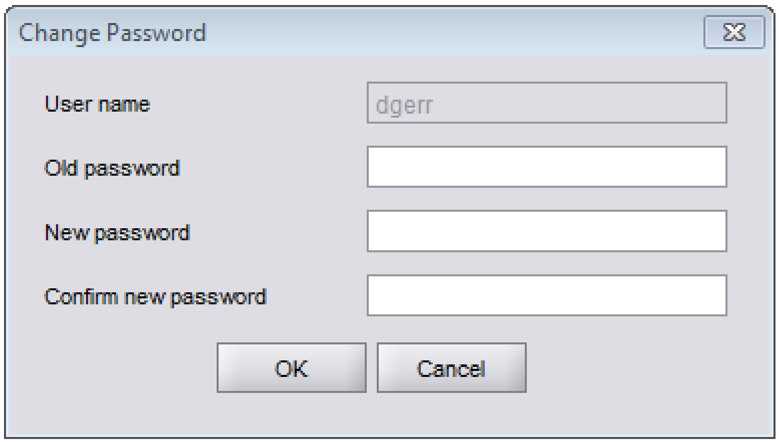
On the PharmaSuite welcome page, the **Logout** button is also located in the top right corner. Click it to return to the webstart page.

If you decide to log out from a running application, the system will request you to confirm the decision and also warn you if there is any unsaved data you may want to save before you log out.

Password Change

You can access the function for changing your password from the PharmaSuite welcome page.

1. Click the **Change Password** link to open the **Change Password** form.
   When your password expires the system will open the form automatically. This can also happen when you log in for the first time to force you to change the initial password, which your system administrator defined for you.
2. On the **Change Password** form, the **User Name** box is output-only and contains your login name.
3. Type your current password in the **Old Password** box.
4. Type your new password first in the **New Password** box and then in the **Confirm New Password** box.
   For security reasons, passwords are masked by asterisks (*).
5. Click the **OK** button to close the form.
   From now on, use the new password to log in.



User name	dgerr
Old password	
New password	
Confirm new password	

Figure 52: Change password

REQUIRED SERVERS

For providing its full functional scope, PharmaSuite relies on the following servers that are responsible for communication to external systems or between its applications.

- Electronic Batch Recording (EBR) server
  It controls the execution of EBR recipes and workflows and can process incoming messages from a Distributed Control System.
- Triggered Operation Management (TOM) server
  It manages event-triggered operations.
- Operation Execution (OE) server
  It controls the execution of server-run operations.
- Automation Integration (AI) server
  It controls the communication with automation-related systems.
- Transition server
  It performs automatic, system-triggered status changes on objects, such as master recipes, master workflows, batches, orders, or workflows and can process incoming messages from an external Quality Management System.

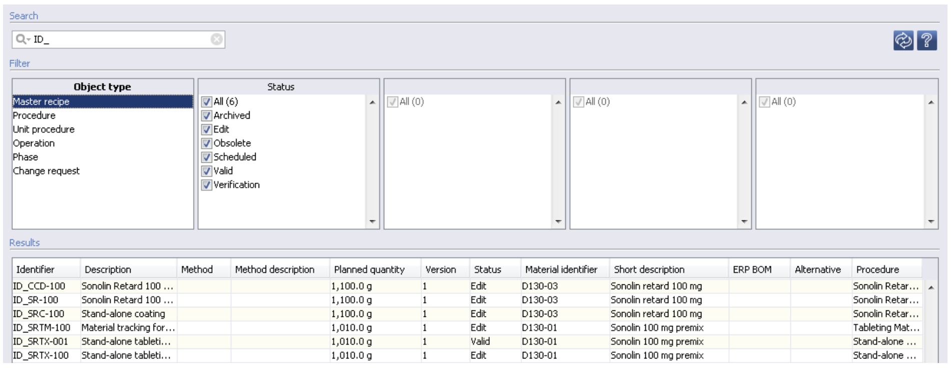
PharmaSuite runs a heartbeat check on the servers to monitor their availability. To see if there are any issues, open the **About PharmaSuite** dialog of Recipe Designer - Batch (see "About PharmaSuite" in Vol. 2), Recipe Designer - Device (see "About PharmaSuite" in Vol. 3), or Workflow Designer (see "About PharmaSuite" in Vol. 4), which shows the status of the EBR server. For information on the other servers, open the **Details** dialog and refer to the section that indicates servers with heartbeat issues.

TIP

Of the servers listed above you only need the Transition server for running Recipe and Workflow Designer.

Data Tools

When you design recipes, workflows, or building blocks, you are working on individual data records that are stored in the database, which you need to access. For these situations, Recipe and Workflow Designer supports you with tools for searching (page 130) and filtering (page 130) lists of data objects it has retrieved from the database and provides sorting (page 131) functions you can apply to all tabular displays of data objects.



Identifier	Description	Method	Method description	Planned quantity	Version	Status	Material identifier	Short description	ERP BOM	Alternative	Procedure
ID_CCD-100	Sonolin Retard 100 ...			1,100.0 g	1	Edit	D130-03	Sonolin retard 100 mg			Sonolin Retar...
ID_SR-100	Sonolin Retard 100 ...			1,100.0 g	1	Edit	D130-03	Sonolin retard 100 mg			Sonolin Retar...
ID_SRC-100	Stand-alone coating			1,100.0 g	1	Edit	D130-03	Sonolin retard 100 mg			Sonolin Retar...
ID_SRTM-100	Material tracking for...			1,010.0 g	1	Edit	D130-01	Sonolin 100 mg premix			Tableting Mat...
ID_SRTX-001	Stand-alone tableti...			1,010.0 g	1	Valid	D130-01	Sonolin 100 mg premix			Stand-alone ...
ID_SRTX-100	Stand-alone tableti...			1,010.0 g	1	Edit	D130-01	Sonolin 100 mg premix			Stand-alone ...

Figure 53: Search and Filter tools with sortable Results list

Searching

The **Search** tool provides you with a fast and efficient way to locate items with specific content in the pre-filtered **Results** list:

- In the input box, type the string of characters that will be applied to the **Results** list. The search becomes effective with the second character.
- To clear the input box, click the **Cancel** icon that appears when you have typed the first character in the box.

TIP

By default, the search is not case-sensitive and runs over all columns of the **Results** list. Click the magnifying glass icon to restrict the search to specific columns, set the search criteria to be case-sensitive, or allow the use of wildcards.

Filtering

The **Filter** tool is hierarchically organized into search categories that make refining your search fast and easy.

1. In the first column, select the type of the database objects you are searching. The system displays all database objects of this type in the **Results** list.

   TIP
   If the **Object type** is defined by the context from which you have opened the form, the column only provides corresponding values.

2. Depending on the selected object type, the system provides further search categories.
   Select or unselect one, several, or all of the options available in the first column to the right of the **Object type** to reduce the number of objects displayed in the **Results** list.

3. To further refine your search result, move to the next category column and select or unselect its options as required. The system applies your choices as additional filter and displays the subset of remaining objects in the **Results** list.

TIP

Please note that Recipe Designer takes a snapshot of the database when you access an object type. This means that the filter tool will remain unaware of changes made to database objects or new objects added by other users while you browse through the data of this object type. To synchronize the snapshot with the central database, click the **Refresh from database** button.

Sorting

By default, the sort order of a data table depends on the type of data objects it displays. It is, however, possible to change the sort order, sort by another column, and sort by two or more columns as primary, secondary, and further levels. Additionally you can reorder the table columns themselves.

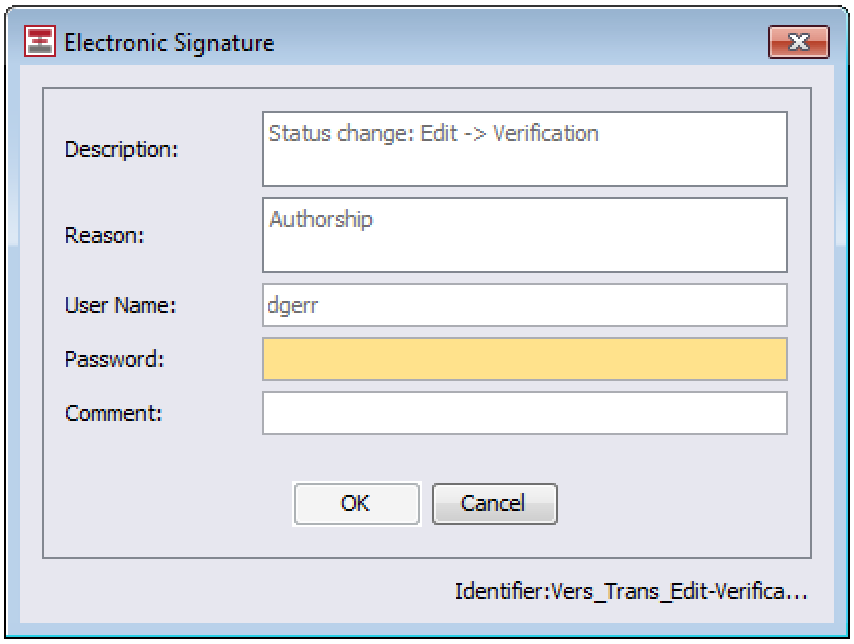
- To adjust the sorting, proceed as follows:
  1. Click any column header to sort the table by this column in ascending order. The system indicates the sort order with a triangle pointing up.
  2. Re-click the same column header to switch the sort order to descending. The system indicates the sort order with a triangle pointing down.
  3. CTRL-click a yet unmarked column header to add this column as further sort level in ascending sort order, indicated by the triangle pointing up and the count number indicating the sort level.
  4. Re-CTRL-click the same column header to switch the sort order to descending without changing its sort level.
- To reorder the table columns, click a column header and drag it to the desired position in the table.

Signature Requests

When performing safety-sensitive or GxP-relevant functions the system may request you to enter an electronic signature, for example during a status change. Signatures are linked to user groups and access privileges, which means that the system will only accept the signature of a user who is qualified to perform the task in question. Unless the required signature data has been entered correctly, subsequent functions cannot be executed.

For situations requiring a witness, the system will ask not only for a single but for a double signature. In these cases two different users, typically with different qualifications, have to complete the signature form before task processing can continue.

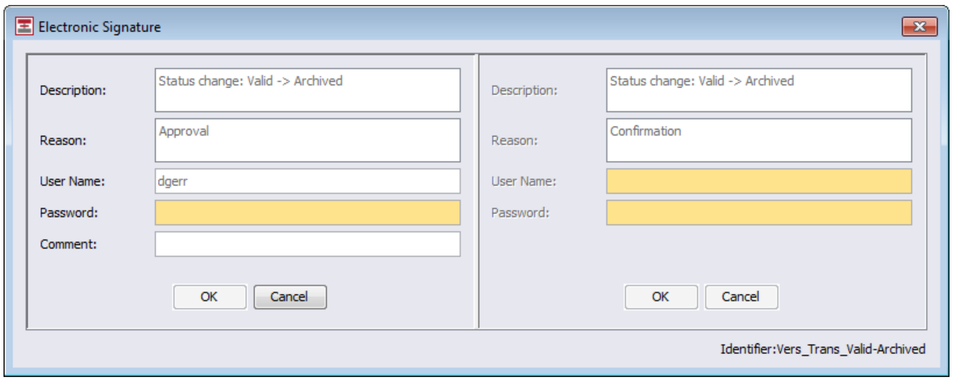
To perform an electronic signature, type your login name and password and click the **OK** button. Comments can be optional or mandatory and may consist of up to 255 characters.



Description:	Status change: Edit -> Verification
Reason:	Authorship
User Name:	dgerr
Password:	
Comment:	

Identifier:Vers_Trans_Edit-Verifica...

Figure 54: Single electronic signature



Description:	Status change: Valid -> Archived	Description:	Status change: Valid -> Archived
Reason:	Approval	Reason:	Confirmation
User Name:	dgerr	User Name:	
Password:		Password:	
Comment:			

Identifier:Vers_Trans_Valid-Archived

Figure 55: Double electronic signature to support witness role

Index