LISTEN.
THINK.
SOLVE.®

PharmaSuite®

IPC PHASES

RELEASE 8.4

USER MANUAL

PUBLICATION PSIP-UM003D-EN-E-DECEMBER-2017
Supersedes publication PSIP-UM003C-EN-E

Allen-Bradley • Rockwell Software

Rockwell Automation

Contact Rockwell See contact information provided in your maintenance contract.

Copyright Notice © 2017 Rockwell Automation Technologies, Inc. All rights reserved.
This document and any accompanying Rockwell Software products are copyrighted by Rockwell Automation Technologies, Inc. Any reproduction and/or distribution without prior written consent from Rockwell Automation Technologies, Inc. is strictly prohibited. Please refer to the license agreement for details.

Trademark Notices FactoryTalk, PharmaSuite, Rockwell Automation, Rockwell Software, and the Rockwell Software logo are registered trademarks of Rockwell Automation, Inc.

The following logos and products are trademarks of Rockwell Automation, Inc.:

FactoryTalk Shop Operations Server, FactoryTalk ProductionCentre, FactoryTalk Administration Console, FactoryTalk Automation Platform, and FactoryTalk Security.
Operational Data Store, ODS, Plant Operations, Process Designer, Shop Operations, Rockwell Software CPGSuite, and Rockwell Software AutoSuite.

Other Trademarks ActiveX, Microsoft, Microsoft Access, SQL Server, Visual Basic, Visual C++, Visual SourceSafe, Windows, Windows 7 Professional, Windows Server 2008, Windows Server 2012, and Windows Server 2016 are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Adobe, Acrobat, and Reader are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.

ControlNet is a registered trademark of ControlNet International.

DeviceNet is a trademark of the Open DeviceNet Vendor Association, Inc. (ODVA).

Ethernet is a registered trademark of Digital Equipment Corporation, Intel, and Xerox Corporation.

OLE for Process Control (OPC) is a registered trademark of the OPC Foundation.

Oracle, SQL*Net, and SQL*Plus are registered trademarks of Oracle Corporation.

All other trademarks are the property of their respective holders and are hereby acknowledged.

Warranty This product is warranted in accordance with the product license. The product's performance may be affected by system configuration, the application being performed, operator control, maintenance, and other related factors. Rockwell Automation is not responsible for these intervening factors. The instructions in this document do not cover all the details or variations in the equipment, procedure, or process described, nor do they provide directions for meeting every possible contingency during installation, operation, or maintenance. This product's implementation may vary among users.

This document is current as of the time of release of the product; however, the accompanying software may have changed since the release. Rockwell Automation, Inc. reserves the right to change any information contained in this document or the software at any time without prior notice. It is your responsibility to obtain the most current information available from Rockwell when installing or using this product.

Contents

Figures

Chapter 1

IPC Phases

The IPC phases of PharmaSuite represent a set of phases that are designed to cover the needs of in-process controls (IPC) by providing

- triggers to initiate IPC runs at defined intervals and
- the means to collect and evaluate the data recorded during IPC runs.

TIP

Please note that even though the IPC phases are designed for modeling in-process controls, they are not specifically restricted to being used in this context.
They can, however, only display their full functional scope if they are located in recipes with a suitable structure (page 3).

The following phases are available:

- Time-based Trigger (page 17)
- Counter-based Trigger (page 23)
- Get Values (page 31)
- Show Values (page 51)

This section contains important information about using the IPC phases in master recipes. Please read this section carefully, because it provides a solid background for all operations you may wish to perform with your system.

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.

Structural Context

Whether or not a recipe's unit procedure is suitable for holding IPC phases does not only depend on its graph structure, but is also controlled by the capabilities assigned to the operations of the unit procedure:

- Capability prerequisites:
  - Only operations that hold both the **Event-triggered** capability and the **Trigger-enabled** capability can interpret the trigger events sent by a trigger phase.
  - Trigger phases become active and complete automatically and do not require user interaction. For this reason, they have no user interface and need to be located in an operation that runs on a server, invisible to operators who process orders with PharmaSuite for Production Execution. Thus an operation that holds trigger phases needs to have the **Server-run** capability.
- Graph structure prerequisites:
  - A trigger phase can only send trigger events to an **Event-triggered** operation if the operation is active to process the triggers. Consequently, the **Event-triggered** operation must run at the same time as the phase from which it receives trigger events. This means that an operation holding trigger phases must be located after the same simultaneous branch on a parallel track (page 4) to the **Event-triggered** operation whose runs it controls.
  - **Get values** and **Show values** phases, on the other hand, need to be processed sequentially (page 13), as the **Get values** phase collects the processing data, which the **Show values** phase displays and evaluates. The order of the phases, however, does not necessarily have to be strictly sequential within one operation.
  - The **Show values** phase can be configured to either process data collected from a **Get values** phase within the same operation or it can retrieve the collected data of a **Get values** phase located outside of its own operation (page 15). In both cases the **Get values** phase must have run at least once to have collected the data to be shown.

Trigger Phases

The placement of **Time-based trigger** and **Counter-based trigger** phases in a recipe is determined by the capabilities their operations need to have and by the structural requirement to ensure their simultaneous activity with the **Event-triggered** operation which they send trigger events.

TIP

Please note that

- an **Event-triggered** operation can reference several trigger phases for receiving trigger events from them.
- a trigger phase can be referenced by several **Event-triggered** operations for sending them trigger events.

The following rules apply with respect to the start and completion of trigger processing of each of the two phases:

- A trigger phase becomes active automatically as soon as processing reaches its operation, but trigger processing does not start until at least one **Event-triggered** operation that references the phase has become active as well. During execution this means that only when the template of an **Event-triggered** operation becomes visible in the Cockpit of PharmaSuite for Production Execution, does the trigger phase start processing and can send the trigger events, which create the runs of the operation.
- If no **Event-triggered** operation becomes active, the trigger phase completes automatically after its defined timeout period has elapsed.
- If trigger processing has started along with one of its **Event-triggered** operations, the phase continues to send trigger events until the last of its **Event-triggered** operations is completed, which happens when an operator removes the template of the **Event-triggered** operation from the Cockpit of PharmaSuite for Production Execution. Without a target to which it can send its trigger events, the trigger phase completes automatically.

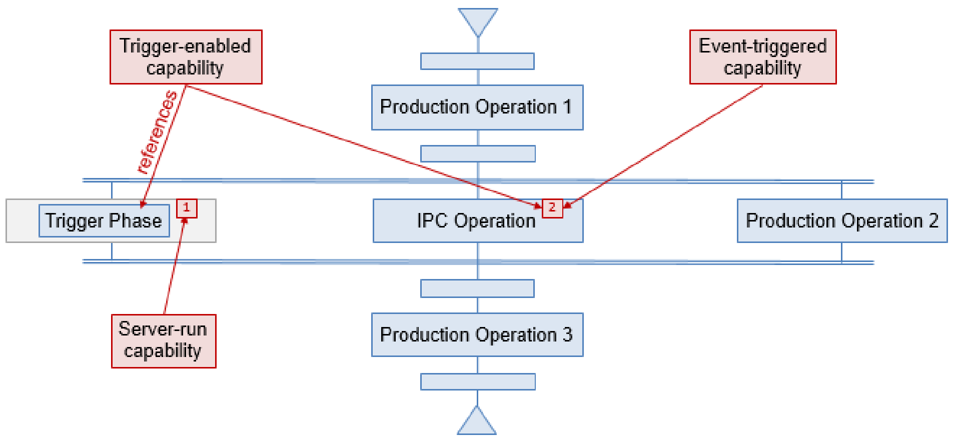


Figure 1: Unit procedure with trigger phase and Event-triggered operation

IMPORTANT

The general rules for building recipe structures as SFC graphs also govern unit procedures that hold trigger phases and **Event-triggered** operations. Thus it is possible to build recipes that are valid from an SFC graph perspective, but do not meet the functional requirements made by processing use cases such as IPC operations. A recipe with functionally invalid structures is bound to cause serious issues during execution.

ISSUE: OPERATIONS NOT STRICTLY PARALLEL

Having operations precede either the **Server-run** operation with the trigger phases or the **Event-triggered** operation may lead to timing issues during execution.

- If the **Server-run** operation has a direct predecessor operation within the simultaneous branch, the system shows the following behavior during execution:
  - The template of the **Event-triggered** operation is visible in the Cockpit, but its trigger phase is not yet active to provide it with trigger events.
  - The **Server-run** operation holding the trigger phase can only become active when its preceding operation has been completed, thus causing an indeterminate delay of trigger processing.

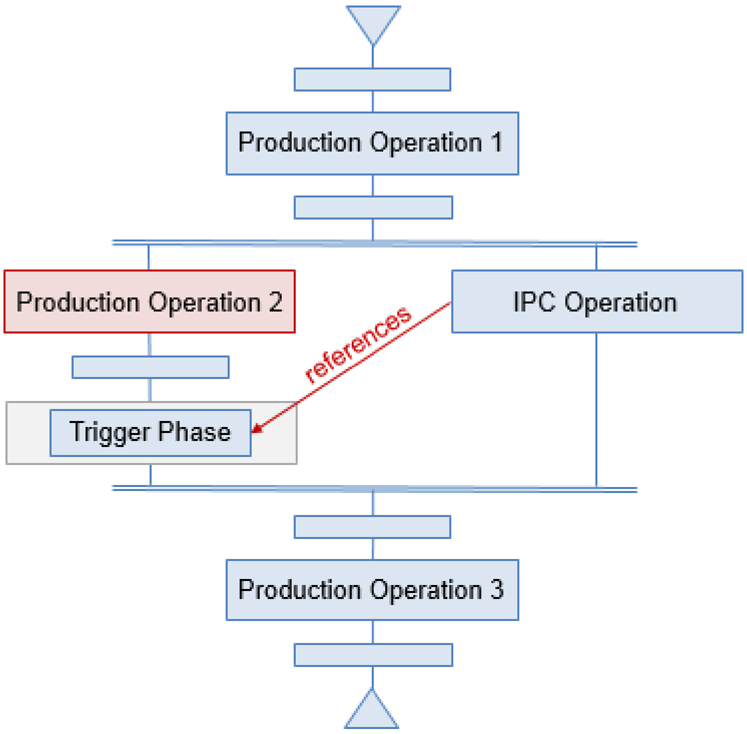


Figure 2: Functionally invalid - Server-run operation with preceding operation

- If the **Event-triggered** operation has a direct predecessor operation within the simultaneous branch, the system shows the following behavior during execution:
  - The trigger phase becomes active, but trigger processing does not start unless the operation preceding the **Event-triggered** operation has been completed and its template has become active.
  - The active trigger phase waits for its targeted **Event-triggered** operation to become active, but only until its defined timeout period has elapsed.
  - After the timeout period has elapsed, the trigger phase is completed automatically.
  - Thus the indeterminate delay of the targeted **Event-triggered** operation can cause its entire trigger schedule to fail, since its referenced trigger phase has timed out before the **Event-triggered** operation has become active at all.

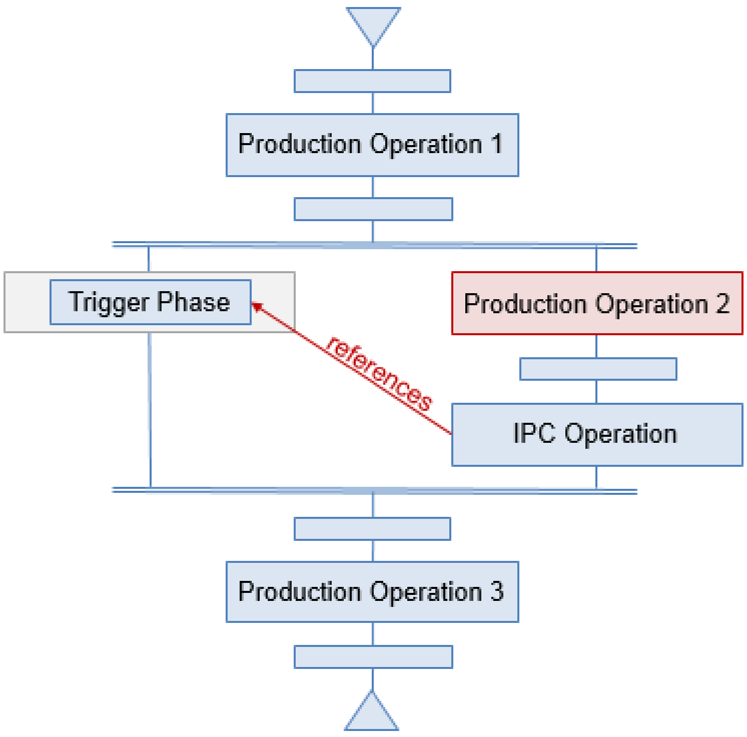


Figure 3: Functionally invalid - Event-triggered operation with preceding operation

TIP

To avoid issues of this type, place the predecessor operation before the simultaneous branch to establish a simultaneous activation of the **Server-run** operation holding the trigger phases and its targeted **Event-triggered** operation.

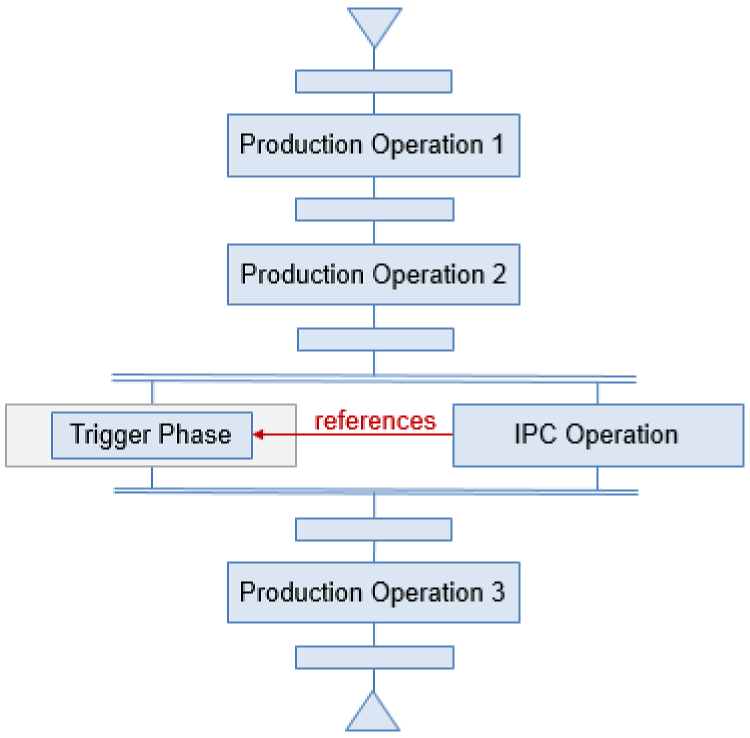


Figure 4: Functionally valid - strictly parallel Server-run and Event-triggered operations

ISSUE: SEQUENTIAL EVENT-TRIGGERED OPERATIONS WITH IDENTICAL PHASE REFERENCE

If your unit procedure has two **Event-triggered** operations that are located on the same track after the simultaneous branch and that both reference the same trigger phase, the system shows the following behavior during execution:

- The **Server-run** operation holding the trigger phase becomes active together with the first **Event-triggered** operation and the trigger phase begins to send trigger events as scheduled.
- Once the first **Event-triggered** operation has completed, the trigger phase is informed of this fact and completes automatically.
- As a consequence, when the second **Event-triggered** operation becomes active, its trigger phase is not available and its trigger schedule is bound to fail.

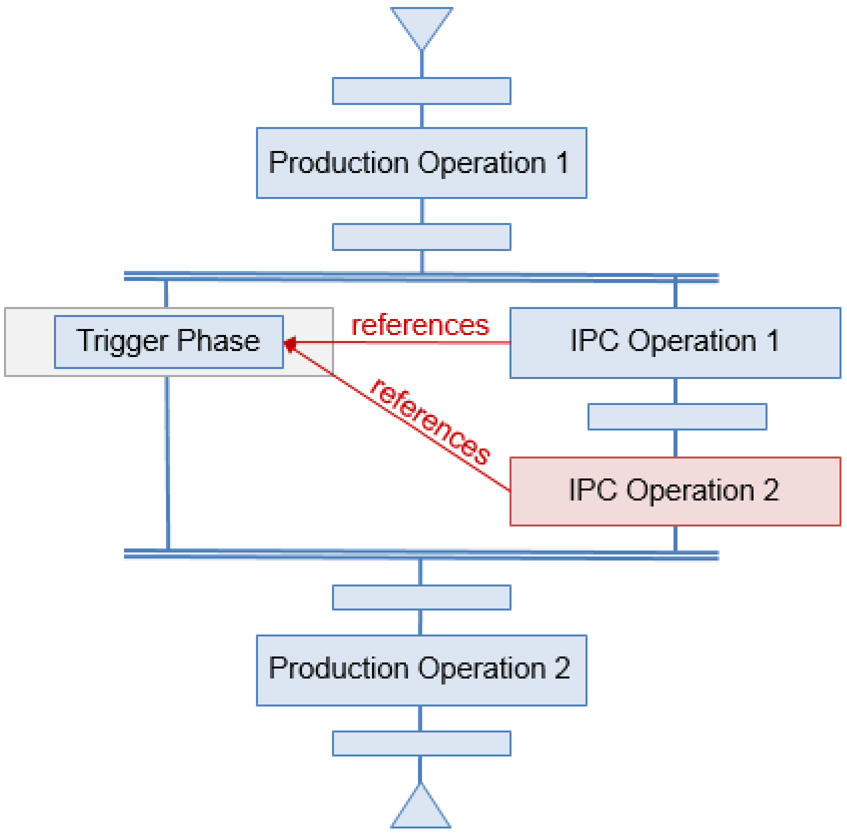


Figure 5: Functionally invalid - sequential Event-triggered operations with identical phase reference

TIP

To avoid issues of this type, place sequential trigger phases into the **Server-run** operation, one for each **Event-triggered** operation.

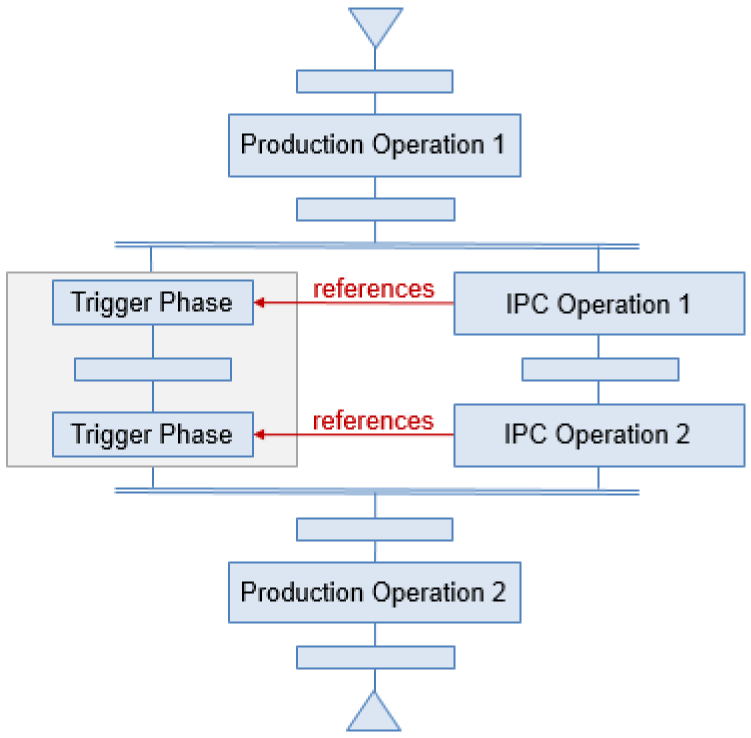


Figure 6: Functionally valid - sequential Event-triggered operations with sequential phases to reference

Issue: Looped Event-triggered Operation after Simultaneous Branch

If your unit procedure has a loop only around its single **Event-triggered** operation, the system shows the following behavior during execution:

- The **Server-run** operation holding the trigger phase becomes active together with the **Event-triggered** operation and the trigger phase begins to send trigger events as scheduled.
- Once the **Event-triggered** operation has completed for the first time and reaches the loop's decision transitions, the trigger phase is informed of this fact and completes automatically.
- As a consequence, if the transition decides that the loop has to be passed through and the **Event-triggered** operation needs to be run again, its trigger phase is not available and its trigger schedule is bound to fail.

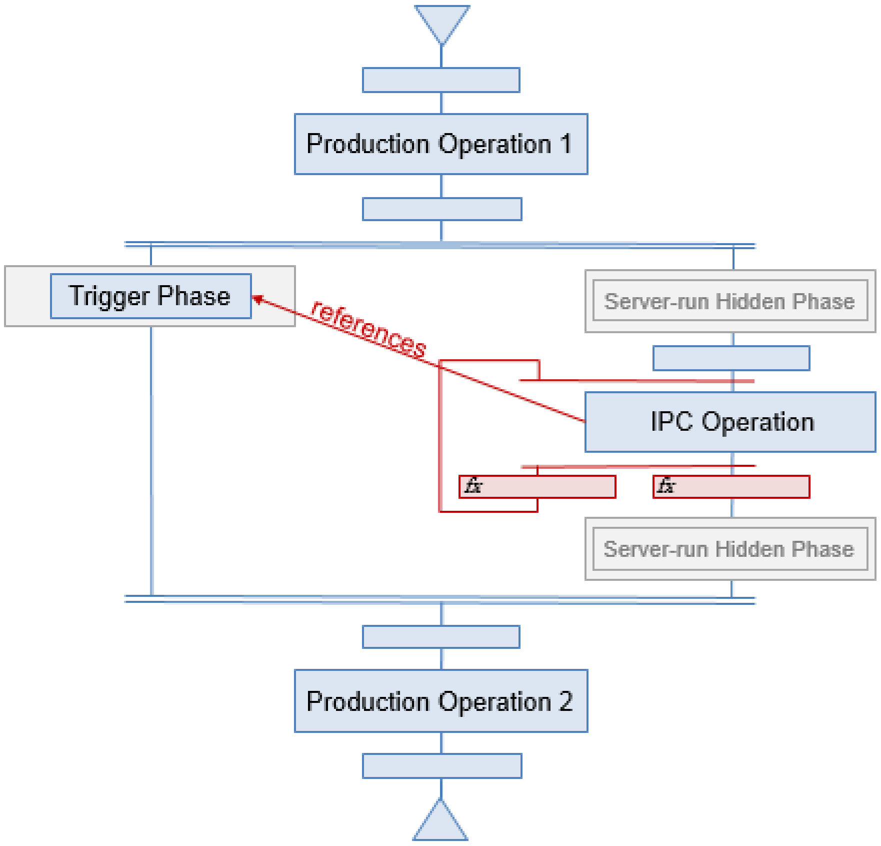


Figure 7: Functionally invalid - looped Event-triggered operation after simultaneous branch

TIP

To avoid issues of this type, enclose the entire simultaneous branch in the loop.

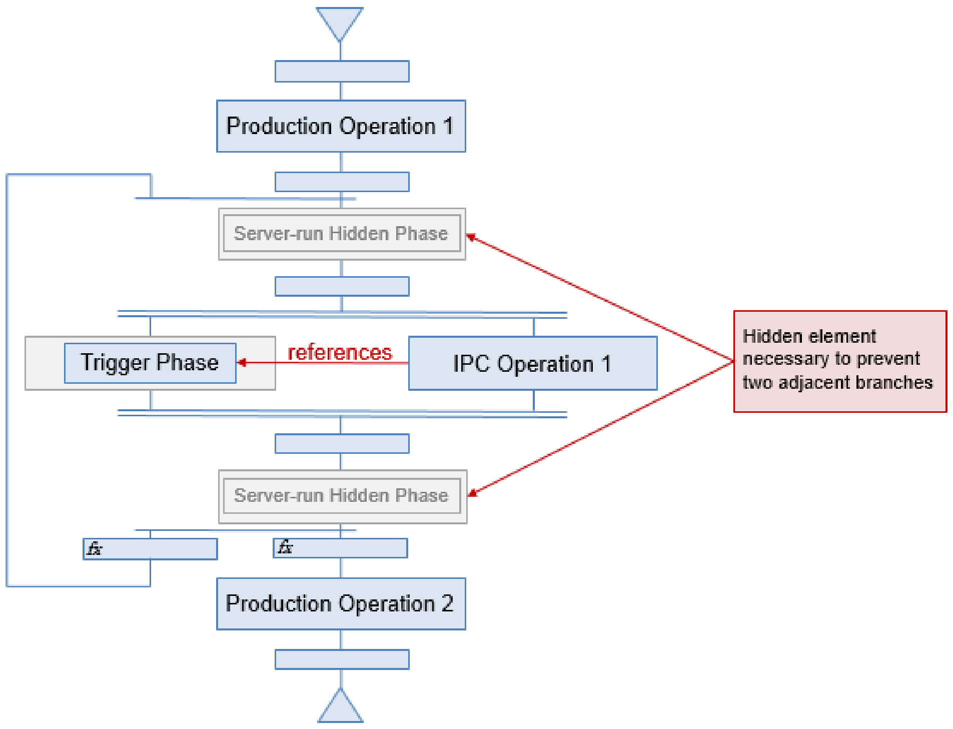


Figure 8: Functionally valid - loop around Server-run and Event-triggered operations

Data Collection and Evaluation Phases

The following rules apply to the usage of the **Get values** and **Show values** phases in recipes:

- A **Show values** phase always needs to reference a **Get values** phase.
- For performing statistical data evaluations, the **Show values** phase relies on the parameter bundle identifiers defined with its referenced **Get Values** phase.
- A **Show values** phase can be placed within or outside the operation that contains its referenced **Get values** phase.

Phase Placement

When you place the **Get values** and **Show values** phases in the same unit procedure you need to make sure that they will be processed one after another.

- For displaying the data of exactly one run of a **Get values** phase, you can place the two phases in the same operation in sequential order.

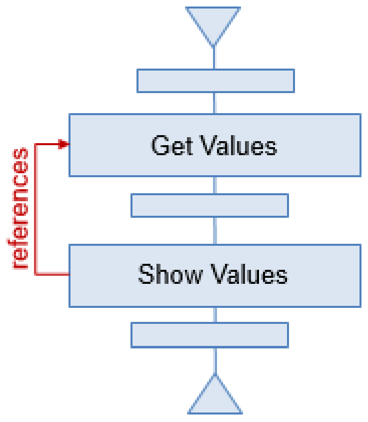


Figure 9: Sequential placement in one operation

- For displaying and evaluating the data of several runs of your **Get Values** phase, you can place them after a simultaneous branch with a loop around the **Get values** phase. For this structure you need two Hidden Phases enclosing the loop to prevent adjacent branches and another Hidden Phase preceding the **Show values** phase enabling it to reference the **Get values** phase.

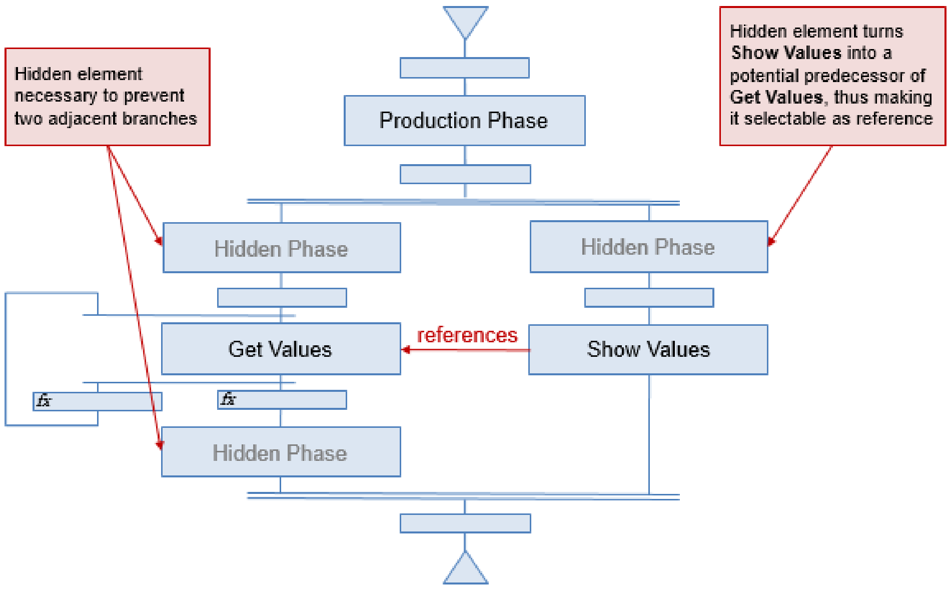

Hidden element necessary to prevent two adjacent branches
Production Phase
Hidden element turns Show Values into a potential predecessor of Get Values, thus making it selectable as reference
Hidden Phase
Hidden Phase
Get Values
references
Show Values
Hidden Phase

Figure 10: Parallel placement in one operation

- For displaying and evaluating the data of the individual runs of your **Get Values** phase as they are generated by the implicit loop of an **Event-triggered** operation, you can place the **Get values** phase inside the **Event-triggered** operation and the **Show values** phase into another operation outside of the **Event-triggered** operation.

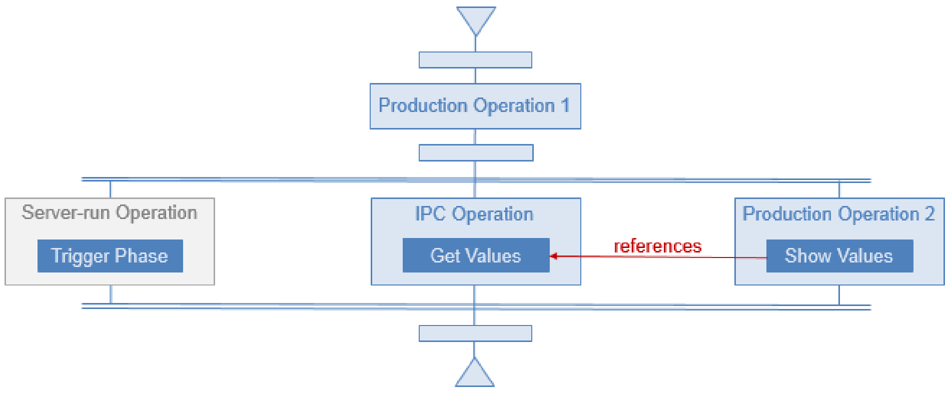


Figure 11: Placement in different operations

Scope of Data Display and Evaluation

Regarding the source **Get values** phases whose values the **Show values** phase displays and evaluates, the system provides two different configurations:

- **Within operation run**
  Only the data that has been collected by a **Get values** phase within the current run of the operation in which the **Show values** phase is located is displayed and considered for evaluation. Each re-run of the **Get values** phase, caused by a loop provides one set of collected data per run of the phase.

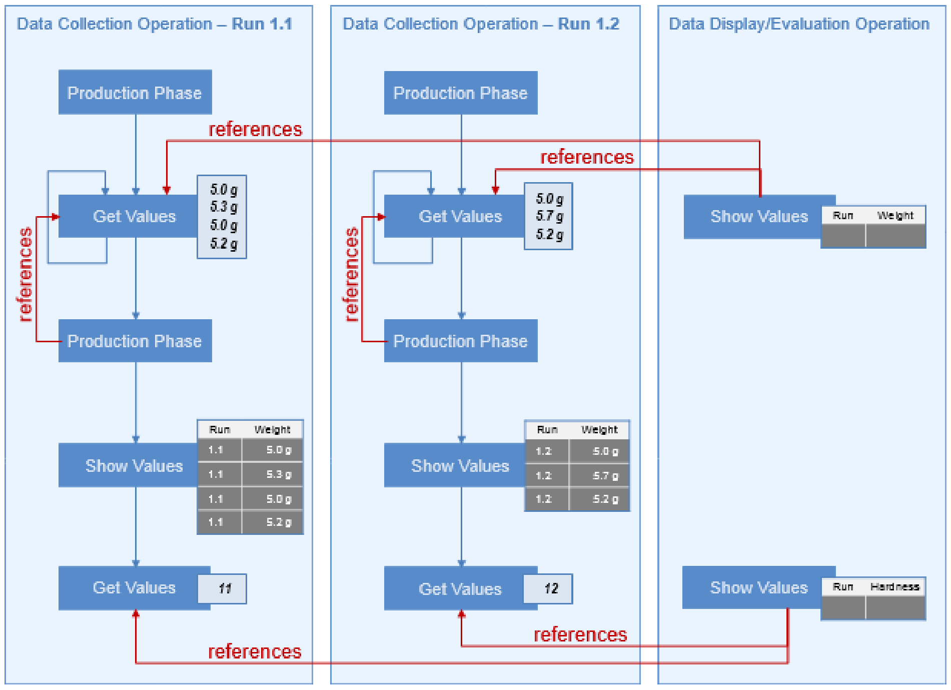
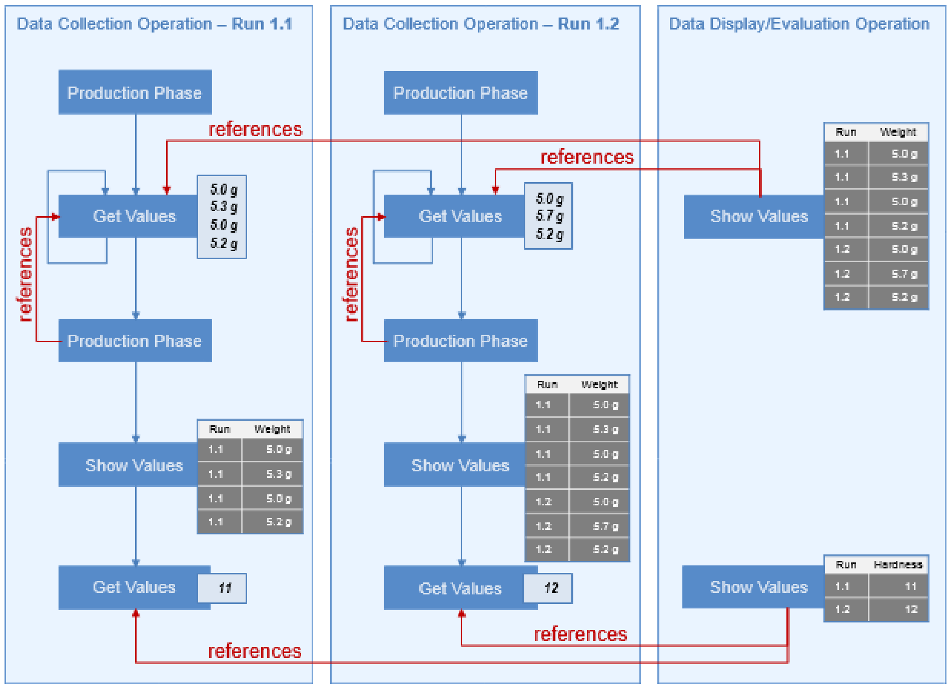


Figure 12: Display of data within operation run

- **Across operation runs**
  All runs of the operation holding the **Get values** phase are considered for display and evaluation. This is the default configuration for use with **Event-triggered** operations in the IPC scenario, where each run of an **Event-triggered** operation for in-process-controls is collected for display and evaluation.



Run	Weight
1.1	5.0 g
1.1	5.3 g
1.1	5.0 g
1.1	5.2 g

Run	Weight
1.1	5.0 g
1.1	5.3 g
1.1	5.0 g
1.1	5.2 g
1.2	5.0 g
1.2	5.7 g
1.2	5.2 g

Run	Weight
1.1	5.0 g
1.1	5.3 g
1.1	5.0 g
1.1	5.2 g
1.2	5.0 g
1.2	5.7 g
1.2	5.2 g

Run	Hardness
1.1	11
1.2	12

Figure 13: Display of data across all operation runs

Time-based Trigger

The **Time-based trigger** phase allows to automatically create runs of an event-triggered operation (ETO) based on a time-related cycle.

It can be used for processing requirements, such as:

- In manufacturing, average tablet weight and hardness have to be recorded every 30 minutes.
- In packaging, visual checks of filled folding cartons have to be executed every hour.

Execution

The **Time-based Trigger** phase is intended for being run on the Operation Execution (OE) server. It generates events to trigger the runs of event-triggered operations, such as IPC operations.
As a server-run phase it starts automatically and becomes active when an operator starts an operation that references the phase as its trigger phase. Once active it generates trigger events according to its configured schedule.
The phase completes automatically when the last operation whose runs it triggers is completed by an operator.

Phase Design

The characteristics of the **Time-based Trigger** phase are defined via process parameters and their attributes.
Since it runs on the OE server and does not require operator interaction, it is not visible to operators during processing.
When the phase is completed, the batch report shows the values configured for its delay and cycle times as well as how many triggers it sent while it was active.

Process Parameters

The following process parameters are available to configure the phase's behavior during execution:

Delay time

Defines how long after its first event-triggered operation has been started the phase triggers the first run of the operation.

Attribute	Type	Comment
Duration	Duration	If left blank, the system interprets this as 0 and immediately triggers a run when a respective event-triggered operation is started.

Cycle time

Defines the interval between any two trigger events.

Attribute	Type	Comment
Duration	Duration	The minimum cycle time is 30 seconds. So if you set a value of less than 30 seconds or leave the cell blank, the system interprets this as 30 seconds.

Timeout period

Defines how long the phase waits for its first event-triggered operation to be started before it completes automatically. This prevents deadlocks that might occur if an event-triggered operation fails to start entirely.

Attribute	Type	Comment
Duration	Duration	If left blank, the system interprets this as 30 minutes.

Timeout exception

Represents a system-triggered exception that is displayed in the overview Exception Window and in the batch report.
The system records a timeout incident as exception, since the quality of a product may be compromised if scheduled events, such as in-process controls, do not take place.

Attribute	Type	Comment
Risk assessment	Choice list	Defines the risk level of the exception and thus controls the related signature privilege. Available settings: **None**, **Low**, **Low (mandatory comment)**, **Medium**, **Medium (mandatory comment)**, **High**, **High (mandatory comment)**. Default setting: **High**.
Exception text	Text	Defines the exception description used during exception handling and within the batch record. Maximum length is 250 characters.

Output Variables

Instead of specifying a fixed value to be displayed or used during execution, you can also use an expression created in the Expression editor to draw the output of another phase or the calculated result of several phase outputs as value into a parameter attribute. When you reference phase outputs in this manner you need to be aware of the following restrictions:

- 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 **Time-based trigger** phase provides the following output variables:

Identifier

- Data type: String, used for displaying a pre-defined sequence of characters, such as "**Read Instruction**".
- Usage: The output variable provides the identifier of the phase.

Instance count

- Data type: Long, used for integral numbers:
  12345
- Usage: The output variable provides the count of the number of instances the phase has been processed, for example in a loop. The count is also increased when the phase is skipped from an operator's perspective, since the phase is still executed, but as a hidden phase.
  The count variable of a phase that has not been executed provides 0 as output value.

Start time

- Data type: Timestamp, used for displaying dates and times and for time-related calculations.
  To use a timestamp in a phase attribute, you have to make sure it has a matching data type, so to display it in an instruction text, you have to convert it into a string.
- Usage: The output variable provides the start time of the phase.

Completion time

- Data type: Timestamp, used for displaying dates and times and for time-related calculations.
  To use a timestamp in a phase attribute, you have to make sure it has a matching data type, so to display it in an instruction text, you have to convert it into a string.
- Usage: The output variable provides the completion time of the phase.

TIP

To calculate a duration from two timestamps and display it in a specific format, you need to use two conversion functions on the calculation:

- **Convert to Unitless Number** (**convertTo**) takes the calculated duration and converts it into the duration's value for one of its units (e.g. minutes or seconds).
- **Convert to String for Display** (**convertToDisplayString)** takes the converted value and displays it as string to which you can add the unit, also as string.

Example:

Sample Phase with Start time = 14-Nov-2014@10:15
Sample Phase with Completion time = 14-Nov-2014@11:47
The duration is to be displayed in minutes.

```
convertToDisplayString
  (convertTo
      ({Sample Phase}.{Completion time}-{Sample Phase}.{Start time},
"min")
  )
 + " min"
```

As result of the expression, the system displays "**92 min**".

Chapter 3

Counter-based Trigger

The **Counter-based trigger** phase allows to automatically create runs of an event-triggered operation (ETO) based on a counter-based cycle.

It can be used for processing requirements, such as:

- In manufacturing, average tablet weight and hardness have to be recorded every 5,000 tablets.
- In packaging, visual checks of filled folding cartons have to be executed every 200 cartons.

TIP

Please note that the **Counter-based trigger** phase is intended for use with automation integration.

Execution

The **Counter-based Trigger** phase is intended for being run on the Operation Execution (OE) server. It generates events to trigger the runs of event-triggered operations, such as IPC operations.
As a server-run phase it starts automatically and becomes active when an operator starts an operation that references the phase as its trigger phase. Once active it generates trigger events according to its configured schedule.
The phase completes automatically when the last operation whose runs it triggers is completed by an operator.

Phase Design

The characteristics of the **Counter-based Trigger** phase are defined via process parameters and their attributes.
Since it runs on the OE server and does not require operator interaction, it is not visible to operators during processing.
When the phase is completed, the batch report shows the values configured for its delay and cycle counts as well as how many triggers it sent while it was active.

Process Parameters

The following process parameters are available to configure the phase's behavior during execution:

Identified equipment entity

Indicates the equipment entity to be processed with the phase. You cannot define a specific equipment identifier in the box, but have to use the Expression editor to select the equipment object output of a preceding **Identify equipment** phase.

Attribute	Type	Comment
Equipment object	Reference	Reference to the output of a preceding phase that provides an identified equipment entity.

Interval definition

Defines on the one hand how many counts after its first event-triggered operation has been started the phase triggers the first run of the operation. On the other hand it defines the interval between any two trigger events. The system expects values to count up, so the count attributes do not support negative values.

Attribute	Type	Comment
Delay count	Long	If left blank, the system interprets this as 0 and immediately triggers a run when a respective event-triggered operation is started.
Cycle count	Long	The minimum cycle count is 1 second. So if you set a count value of less than 1 or leave the cell blank, the system interprets this as 1 and triggers a run after each counted object.

Timeout period

Defines how long the phase waits for its first event-triggered operation to be started before it completes automatically. This prevents deadlocks that might occur if an event-triggered operation fails to start entirely.

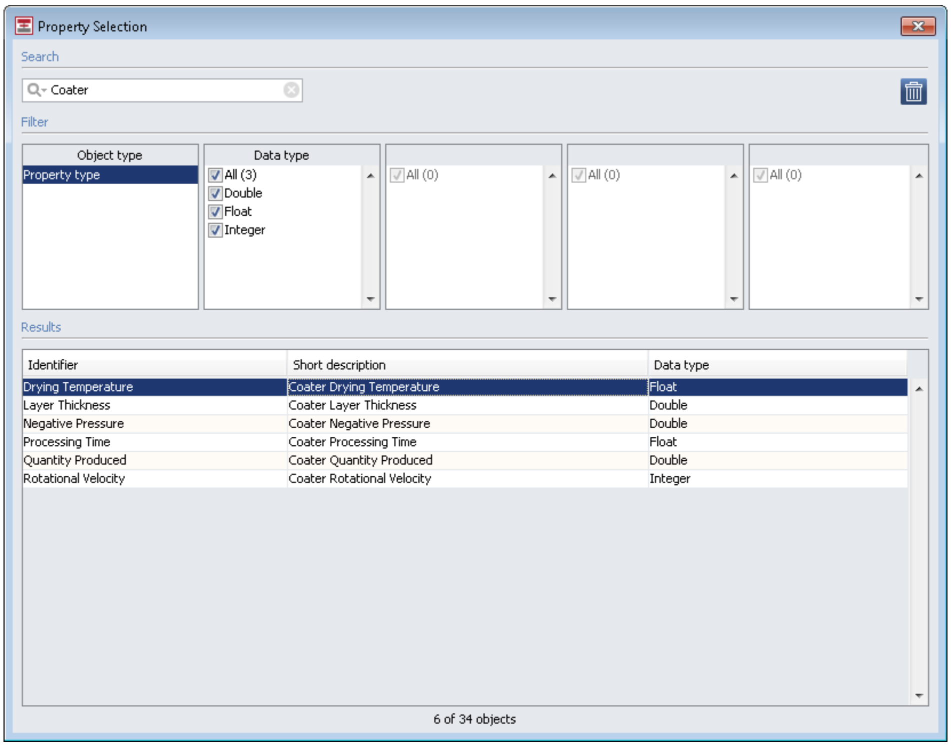
Attribute	Type	Comment
Duration	Duration	If left blank, the system interprets this as 30 minutes.

Numeric property

Indicates the equipment property of the **Numeric** data type whose tag is accessed and read from the automation layer.

Attribute	Type	Comment
Property	String	Equipment property to be read.

For choosing an equipment property of a suitable data type, the system provides a Property Selection editor.



Identifier	Short description	Data type
Drying Temperature	Coater Drying Temperature	Float
Layer Thickness	Coater Layer Thickness	Double
Negative Pressure	Coater Negative Pressure	Double
Processing Time	Coater Processing Time	Float
Quantity Produced	Coater Quantity Produced	Double
Rotational Velocity	Coater Rotational Velocity	Integer

6 of 34 objects

Figure 14: Property Selection editor for Automation properties (Numeric)

Reading cycle

Defines how often the system reads the value of the numeric property from the automation layer.

Attribute	Type	Comment
Duration	Duration	Defines the interval in seconds between two consecutive reading actions. If you set a duration of less than 2 seconds or leave the cell blank, the system interprets this as 2 seconds and accordingly reads the value every 2 seconds.

Timeout exception

Represents a system-triggered exception that is displayed in the overview Exception Window and in the batch report.
The system records a timeout incident as exception, since the quality of a product may be compromised if scheduled events, such as in-process controls, do not take place.

Attribute	Type	Comment
Risk assessment	Choice list	Defines the risk level of the exception and thus controls the related signature privilege. Available settings: **None**, **Low**, **Low (mandatory comment)**, **Medium**, **Medium (mandatory comment)**, **High**, **High (mandatory comment)**. Default setting: **High**.
Exception text	Text	Defines the exception description used during exception handling and within the batch record. Maximum length is 250 characters.

Automation error exception

Represents a system-triggered exception that is displayed in the overview Exception Window and in the batch report.
The system records issues with data retrieval from the automation layer as exception, since they may interfere with the scheduling of the runs of event-triggered operation, such as in-process controls, and thus compromise product quality.

Attribute	Type	Comment
Risk assessment	Choice list	Defines the risk level of the exception. Since there is no operator interaction for the exception, it is not linked to a signature privilege. Available settings: **None**, **Low**, **Low (mandatory comment)**, **Medium**, **Medium (mandatory comment)**, **High**, **High (mandatory comment)**. Default setting: **High**.
Exception text	Text	Defines the exception description used during exception handling and within the batch record. Maximum length is 250 characters.

Counter reset exception

Represents a system-triggered exception that is displayed in the overview Exception Window and in the batch report.
The system records an exception if the counter to which the phase refers is reset to another value, since this may interfere with the scheduling of the runs of event-triggered operation, such as in-process controls, and thus compromise product quality.

Attribute	Type	Comment
Risk assessment	Choice list	Defines the risk level of the exception. Since there is no operator interaction for the exception, it is not linked to a signature privilege. Available settings: **None**, **Low**, **Low (mandatory comment)**, **Medium**, **Medium (mandatory comment)**, **High**, **High (mandatory comment)**. Default setting: **High**.

Attribute	Type	Comment
Exception text	Text	Defines the exception description used during exception handling and within the batch record. Maximum length is 250 characters.

Output Variables

Instead of specifying a fixed value to be displayed or used during execution, you can also use an expression created in the Expression editor to draw the output of another phase or the calculated result of several phase outputs as value into a parameter attribute. When you reference phase outputs in this manner you need to be aware of the following restrictions:

- 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 **Counter-based trigger** phase provides the following output variables:

Identifier

- Data type: String, used for displaying a pre-defined sequence of characters, such as "**Read Instruction**".
- Usage: The output variable provides the identifier of the phase.

Instance count

- Data type: Long, used for integral numbers:
  12345
- Usage: The output variable provides the count of the number of instances the phase has been processed, for example in a loop. The count is also increased when the phase is skipped from an operator's perspective, since the phase is still executed, but as a hidden phase.
  The count variable of a phase that has not been executed provides 0 as output value.

Start time

- Data type: Timestamp, used for displaying dates and times and for time-related calculations.
  To use a timestamp in a phase attribute, you have to make sure it has a matching data type, so to display it in an instruction text, you have to convert it into a string.
- Usage: The output variable provides the start time of the phase.

Completion time

- Data type: Timestamp, used for displaying dates and times and for time-related calculations.
  To use a timestamp in a phase attribute, you have to make sure it has a matching data type, so to display it in an instruction text, you have to convert it into a string.
- Usage: The output variable provides the completion time of the phase.

TIP

To calculate a duration from two timestamps and display it in a specific format, you need to use two conversion functions on the calculation:

- **Convert to Unitless Number** (**convertTo**) takes the calculated duration and converts it into the duration's value for one of its units (e.g. minutes or seconds).
- **Convert to String for Display** (**convertToDisplayString**) takes the converted value and displays it as string to which you can add the unit, also as string.

Example:

Sample Phase with Start time = 14-Nov-2014@10:15
Sample Phase with Completion time = 14-Nov-2014@11:47
The duration is to be displayed in minutes.

```
convertToDisplayString
  (convertTo
      ({Sample Phase}.{Completion time}-{Sample Phase}.{Start time},
"min")
  )
 + " min"
```

As result of the expression, the system displays "**92 min**".

Chapter 4

Get Values

The **Get values** phase allows an operator to collect up to five values of the following data types:

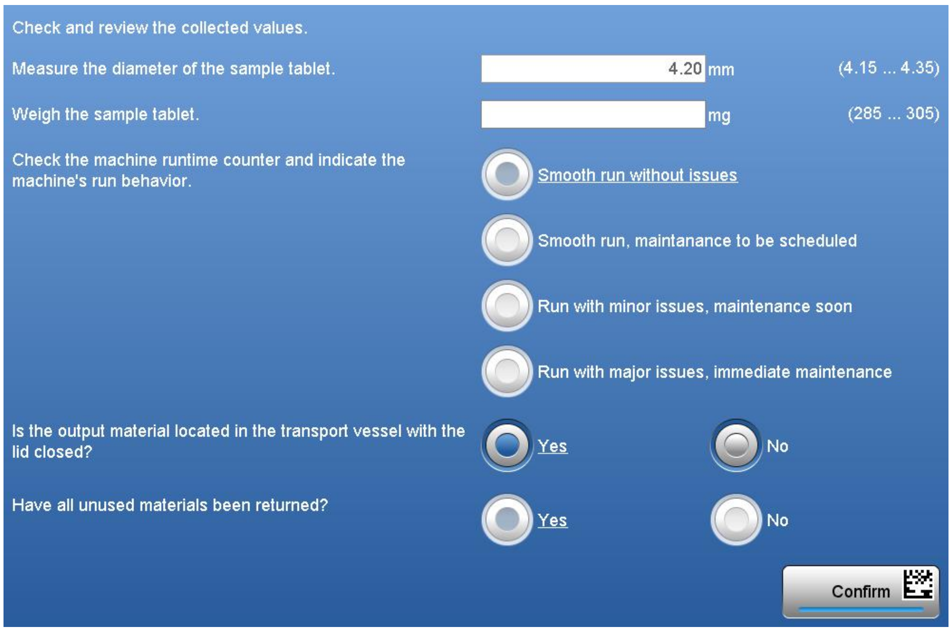
- Measured Value (measured value): value and unit of measure,
- Option Value (choice list): choice from a pre-defined list of options, and
- Boolean Value: choice between Yes and No (`true` and `false`).

It can be used for processing requirements, such as:

- Recording of tablet weights during IPC
  The tablet weight must range between 5 g and 6 g. These boundary values can be defined as limits and corresponding limit violations can be tracked as exceptions.
- Recording of visual appearance during IPC
  During the inspection of a liquid product sample, the visual appearance of the sample can be selected from a pre-defined list (e.g. Transparent, Cloudy, Dark).
- Recording of the execution of mandatory checks during IPC
  The execution of a visual check needs to be recorded by confirming with **Yes**.

Execution

The **Get values** phase records process values of configurable data types entered during execution and can match them against definable limits and expected values. It supports up to five Measured, Boolean, or Option Values.
As long as the phase is active, it provides user-triggered exceptions to override any value that was inserted automatically with its input box configured to be not editable.
After phase completion, it provides post-completion exceptions to correct the values recorded during processing.
After completion the phase displays the recorded values in the Execution Window.
The Navigator displays the phase completion timestamp and provides access to the post-completion exceptions.

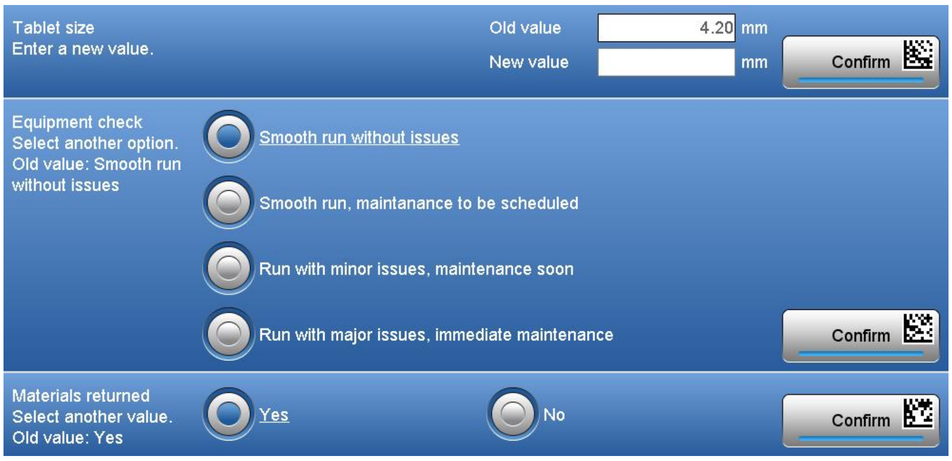


Check and review the collected values.

Measure the diameter of the sample tablet.	4.20 mm	(4.15 ... 4.35)
Weigh the sample tablet.	mg	(285 ... 305)
Check the machine runtime counter and indicate the machine's run behavior.	Smooth run without issues / Smooth run, maintanance to be scheduled / Run with minor issues, maintenance soon / Run with major issues, immediate maintenance	
Is the output material located in the transport vessel with the lid closed?	[✓ Yes] No	
Have all unused materials been returned?	Yes No	

Confirm

Figure 15: Get values during execution



Tablet size
Enter a new value.

Old value 4.20 mm
New value mm
Confirm

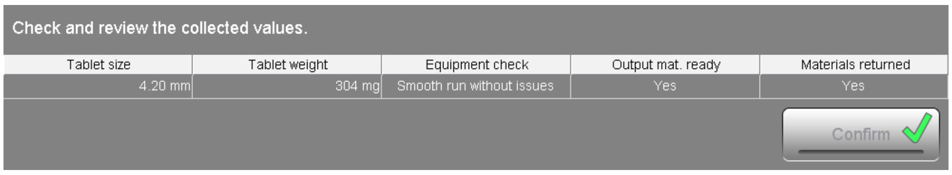
Equipment check
Select another option.
Old value: Smooth run without issues

[✓ Smooth run without issues]
Smooth run, maintanance to be scheduled
Run with minor issues, maintenance soon
Run with major issues, immediate maintenance
Confirm

Materials returned
Select another value.
Old value: Yes

[✓ Yes] No
Confirm

Figure 16: User-triggered exceptions of Get values

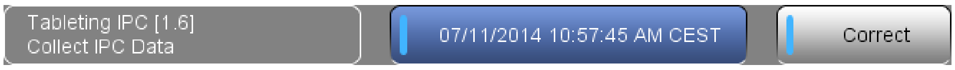


Check and review the collected values.

Tablet size	Tablet weight	Equipment check	Output mat. ready	Materials returned
4.20 mm	304 mg	Smooth run without issues	Yes	Yes

Confirm ✓

Figure 17: Get values after phase completion

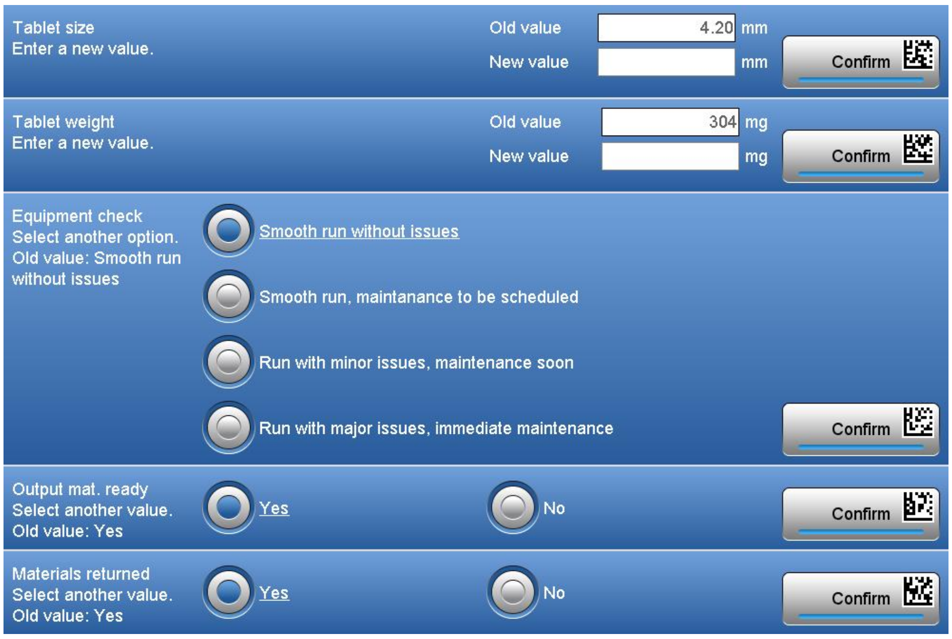


Tableting IPC [1.6]
Collect IPC Data

07/11/2014 10:57:45 AM CEST

Correct

Figure 18: Get values in the Navigator



Tablet size
Enter a new value.
Old value 4.20 mm
New value mm
Confirm

Tablet weight
Enter a new value.
Old value 304 mg
New value mg
Confirm

Equipment check
Select another option.
Old value: Smooth run without issues
[✓ Smooth run without issues]
Smooth run, maintanance to be scheduled
Run with minor issues, maintenance soon
Run with major issues, immediate maintenance
Confirm

Output mat. ready
Select another value.
Old value: Yes
[✓ Yes] No
Confirm

Materials returned
Select another value.
Old value: Yes
[✓ Yes] No
Confirm

Figure 19: Post-completion exceptions of Get values

Phase Design

The characteristics of the **Get values** phase are defined via process parameters and their attributes.
Its user interface is designed in two columns of varying widths that span several rows, depending on the number of values to be collected. When the phase is active, the merged columns of the first row provide space for textual instructions. Each of the following up to five rows is dedicated to recording one value by providing suitable layouts for the three different types of collectible values:

- For Measured Values, it shows a description text and an input box with unit of measure, and limit range, if defined.
- For Option Values, it shows a vertical list of options, each of them followed by a description text.
- For Boolean Values, it shows the two available options with adjacent description text in a horizontal arrangement.

The right column of the bottom row contains the **Confirm** button.
When the phase is completed it shows the same layout for the instruction text in the first row and the **Confirm** button in the bottom row. The recorded values, however, are presented in a table, with one column per value and the value's **Short description** as column header.
Exception handling during execution is controlled by a risk assessment classification and an exception message that are both defined by the recipe author in the exception's process parameter.

Process Parameters

In addition to the permanent process parameters that are always present, the **Get values** phase also provides optional process parameter bundles, which you can insert if required. A process parameter bundle is a group of one or more semantically connected process parameters, all of which are needed to produce a specific behavior or function when the phase is executed.

You can add up to five process parameters bundles of three different value types (Boolean Value, Measured Value, Option Value) to the **Get values** phase, one for each value you wish to record.

ADDING PARAMETER BUNDLES

1. Click the **Add parameter bundle** button.
   The system opens an option list that holds all bundle types available for the phase.
2. Select the type.
   The system opens the **Add <Bundle Type>** dialog to define the bundle's identifier.
3. Type an identifier and click the **OK** button.
   The system adds all process parameters of the bundle to the list of parameters.

TIP
Please note that the identifier of the parameter bundle is used as identifier of the bundle's master parameter and as prefix to all other parameters of the bundle.

REMOVING PARAMETER BUNDLES

1. In the list of parameters, select any one of the parameters of the bundle you wish to remove.
   All parameter identifiers of a bundle start with the bundle's identifier.
2. Click the **Remove parameter** button.
   The system asks you to confirm the action and then removes all parameters of the bundle.

TIP
Please note that you cannot remove individual parameters of a bundle.

The following process parameters are available to configure the phase's behavior during execution:

Instruction

Represents the instruction text that is visible on the preview, the active, and the completed view of the phase.

Attribute	Type	Comment
Column 1	HTML text	Instruction text to be displayed. Maximum length is 2000 characters (including HTML tags).
Column 2	HTML text	Not used
Column 3	HTML text	Not used.

Boolean Value - Master (bundle identifier)

During execution the value appears in two different contexts, prior to recording it may require an explanation or instruction, while after recording its table display requires a header or summary.

Attribute	Type	Comment
Description	Text	Descriptive text or question describing the situation that needs to be answered with TRUE (e.g. Yes) or FALSE (e.g. No). If left blank, the system displays the short description instead. If the **Short description** is also left blank, it reverts to the **Bundle identifier**. Maximum length is 250 characters.
Short description	Text	Defines the column header of the table in the **Completed** view and, if applicable, of a linked **Show values** phase. To avoid line breaks in the column headers, keep the **Short description** as short as possible.

Boolean Value - Boolean options

Defines the texts for the two options available for selection during execution.

Attribute	Type	Comment
Display text for TRUE	String	Defines the string displayed as **TRUE** option. Maximum length is 8 characters. Default setting: **Yes**
Display text for FALSE	String	Defines the string displayed as **FALSE** option. Maximum length is 8 characters. Default setting: **No**

Boolean Value - Expected value configuration

Defines if the actual option selected during execution must be checked against an expected value.

Attribute	Type	Comment
Enabled	Flag	Controls if a check is performed. If so, ensure that the **Expected value** attribute of the **Expected value definition** process parameter (page 37) is set.
Display	Flag	Controls if an expected value is displayed during execution. The expected option is underlined. Ensure that the **Expected value** attribute of the **Expected value definition** process parameter (page 37) is set.
Risk assessment	Choice list	Defines the risk level of the exception and thus controls the related signature privilege. Available settings: **None**, **Low**, **Low (mandatory comment)**, **Medium**, **Medium (mandatory comment)**, **High**, **High (mandatory comment)**. Default setting: **High**.
Exception text	Text	Defines the exception description used during exception handling and within the batch record. Maximum length is 250 characters.

Boolean Value - Expected value definition

Defines the option (by its display text (page 36)) required as expected value if the respective check is enabled.

Attribute	Type	Comment
Expected value	String	Defines the expected value.
Default value	String	Defines the pre-selected item in the list of options.
Value editable	Flag	Controls if the displayed value is editable during execution. Default setting: **Yes**

Boolean Value - Override value

Represents a user-triggered exception that is accessible from the Exception Window. The exception allows an operator to override the value even if it is set to read-only for regular execution, which you achieve by unselecting the **Value editable** attribute of the **Expected value definition** process parameter (page 37).
It covers incidents when a reading error causes a calculated process value to fail the expected value check, but the actual value corresponds to the expected value so that the process can be continued.

Attribute	Type	Comment
Risk assessment	Choice list	Defines the risk level of the exception and thus controls the related signature privilege. Available settings: **None**, **Low**, **Low (mandatory comment)**, **Medium**, **Medium (mandatory comment)**, **High**, **High (mandatory comment)**. Default setting: **High**.
Exception text	Text	Defines the exception description used during exception handling and within the batch record. Maximum length is 250 characters.

Boolean Value - Correct value

Represents a post-completion exception that is accessible from the Navigator.
The exception allows an operator to correct any of the values entered while the phase was active.
It covers incidents when the operator has entered an incorrect value on account of a reading error, but has confirmed and completed the phase before detecting the error.

Attribute	Type	Comment
Risk assessment	Choice list	Defines the risk level of the exception and thus controls the related signature privilege. Available settings: **None**, **Low**, **Low (mandatory comment)**, **Medium**, **Medium (mandatory comment)**, **High**, **High (mandatory comment)**. Default setting: **High**.
Exception text	Text	Defines the exception description used during exception handling and within the batch record. Maximum length is 250 characters.

Measured value - Master (bundle identifier)

During execution the value appears in two different contexts, prior to recording it may require an explanation or instruction, while after recording its table display requires a header or summary.

Attribute	Type	Comment
Description	Text	Instruction describing the measured value to be entered in the input box. If left blank, the system displays the short description instead. If the **Short description** is also left blank, it reverts to the **Bundle identifier**. Maximum length is 250 characters.
Short description	Text	Defines the column header of the table in the **Completed** view and, if applicable, of a linked **Show values** phase. To avoid line breaks in the column headers, keep the **Short description** as short as possible.

Measured Value - Unit of measure

Defines the unit of measure that qualifies the measured value.

Attribute	Type	Comment
UoM	Unit of measure	Must match a unit of measure available within PharmaSuite. See also attributes of the **Limit definition** process parameter (page 40).

Measured Value - Limit configuration

During execution, the actual process value entered in the input box is checked against the configured limits when the operator moves the focus away from the box that holds the value.

Attribute	Type	Comment
Enabled	Flag	Controls if a check is performed. If so, ensure that the **Lower limit** and **Upper limit** attributes of the **Limit definition** process parameter (page 40) are set.
Display	Flag	Controls if the limit range is displayed during execution.
Risk assessment	Choice list	Defines the risk level of the exception and thus controls the related signature privilege. Available settings: **None**, **Low**, **Low (mandatory comment)**, **Medium**, **Medium (mandatory comment)**, **High**, **High (mandatory comment)**. Default setting: **High**.
Exception text	Text	Defines the exception description used during exception handling and within the batch record. Maximum length is 250 characters.

Measured Value - Limit definition

Limits are defined as absolute values. When defining the limit values you need to make sure that the unit of measure must be of the same system of measurement as the one used for the **Limit configuration** process parameter (page 39) (e.g. weight: mg, kg, pound; length: mm, m, inch).

You can define a default value to be shown in the value box and configure if the default value is editable during execution. This way you can use an expression to draw the output of another phase into the value box and even record an exception if an operator needs to edit it.

Attribute	**Type**	**Comment**
Lower limit	MeasuredValue	Defines the value of the lower limit (including the values themselves). Limit values with more than 7 digits are truncated at the end in the Phase Preview.
Upper limit	MeasuredValue	Defines the value of the upper limit (including the values themselves). Limit values with more than 7 digits are truncated at the end in the Phase Preview.
Default value	MeasuredValue	Defines the default value.
Value editable	Flag	Controls if the displayed value is editable during execution. Default setting: **Yes**

Measured Value - Override value

Represents a user-triggered exception that is accessible from the Exception Window. The exception allows an operator to override the value even if it is set to read-only for regular execution, which you achieve by unselecting the **Value editable** attribute of the **Limit definition** process parameter (page 40).
It covers incidents when a reading error causes a calculated process value to fail the defined limits, but the actual value is within the required range so that the process can be continued.

Attribute	Type	Comment
Risk assessment	Choice list	Defines the risk level of the exception and thus controls the related signature privilege. Available settings: **None**, **Low**, **Low (mandatory comment)**, **Medium**, **Medium (mandatory comment)**, **High**, **High (mandatory comment)**. Default setting: **High**.
Exception text	Text	Defines the exception description used during exception handling and within the batch record. Maximum length is 250 characters.

Measured Value - Correct value

Represents a post-completion exception that is accessible from the Navigator.
The exception allows an operator to correct any of the values entered while the phase was active.
It covers incidents when the operator has entered an incorrect value on account of a reading error, but has confirmed and completed the phase before detecting the error.

Attribute	Type	Comment
Risk assessment	Choice list	Defines the risk level of the exception and thus controls the related signature privilege. Available settings: **None**, **Low**, **Low (mandatory comment)**, **Medium**, **Medium (mandatory comment)**, **High**, **High (mandatory comment)**. Default setting: **High**.

Attribute	Type	Comment
Exception text	Text	Defines the exception description used during exception handling and within the batch record. Maximum length is 250 characters.

Option value - Master (bundle identifier)

During execution the value appears in two different contexts, prior to recording it may require an explanation or instruction, while after recording its table display requires a header or summary.

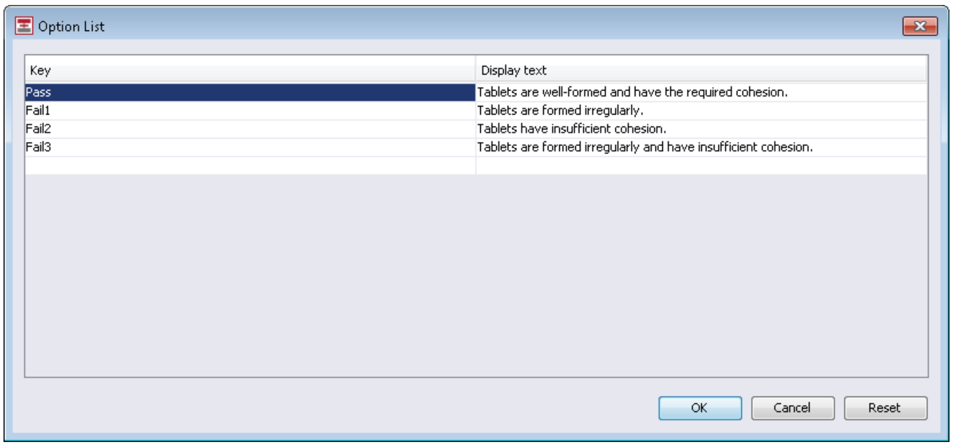
Attribute	Type	Comment
Description	Text	Descriptive text or statement describing the situation that requires one of the available options to be selected. If left blank, the system displays the short description instead. If the **Short description** is also left blank, it reverts to the **Bundle identifier**. Maximum length is 250 characters.
Short description	Text	Defines the column header of the table in the **Completed** view and, if applicable, of a linked **Show values** phase. To avoid line breaks in the column headers, keep the **Short description** as short as possible.

Option Value - List of options

Defines the options available for selection during execution.

Attribute	Type	Comment
Options	Text (structured)	Defines the available options as key/display text value pairs. Both keys and display texts are unique within a phase.

For entering the key/display text pairs the system provides an Option List editor.



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 20: Option List editor

Option Value - Expected value configuration

Defines if the actual option selected during execution must be checked against an expected value.

Attribute	Type	Comment
Enabled	Flag	Controls if a check is performed. If so, ensure that the **Expected value** attribute of the **Expected value definition** process parameter (page 44) is set.
Display	Flag	Controls if an expected value is displayed during execution. The expected option is underlined. Ensure that the **Expected value** attribute of the **Expected value definition** process parameter (page 44) is set.

Attribute	Type	Comment
Risk assessment	Choice list	Defines the risk level of the exception and thus controls the related signature privilege. Available settings: **None**, **Low**, **Low (mandatory comment)**, **Medium**, **Medium (mandatory comment)**, **High**, **High (mandatory comment)**. Default setting: **High**.
Exception text	Text	Defines the exception description used during exception handling and within the batch record. Maximum length is 250 characters.

Option Value - Expected value definition

Defines the option (by its key (page 43)) required as expected value if the respective check is enabled.

Attribute	Type	Comment
Expected value	String	Defines the expected value.
Default value	String	Defines the pre-selected item in the list of options.
Value editable	Flag	Controls if the displayed value is editable during execution. Default setting: **Yes**

Option Value - Override value

Represents a user-triggered exception that is accessible from the Exception Window. The exception allows an operator to override the value even if it is set to read-only for regular execution, which you achieve by unselecting the **Value editable** attribute of the **Expected value definition** process parameter (page 44).
It covers incidents when a reading error causes a calculated process value to fail the expected value check, but the actual value corresponds to the expected value so that the process can be continued.

Attribute	Type	Comment
Risk assessment	Choice list	Defines the risk level of the exception and thus controls the related signature privilege. Available settings: **None**, **Low**, **Low (mandatory comment)**, **Medium**, **Medium (mandatory comment)**, **High**, **High (mandatory comment)**. Default setting: **High**.
Exception text	Text	Defines the exception description used during exception handling and within the batch record. Maximum length is 250 characters.

Option Value - Correct value

Represents a post-completion exception that is accessible from the Navigator.
The exception allows an operator to correct any of the values entered while the phase was active.
It covers incidents when the operator has entered an incorrect value on account of a reading error, but has confirmed and completed the phase before detecting the error.

Attribute	Type	Comment
Risk assessment	Choice list	Defines the risk level of the exception and thus controls the related signature privilege. Available settings: **None**, **Low**, **Low (mandatory comment)**, **Medium**, **Medium (mandatory comment)**, **High**, **High (mandatory comment)**. Default setting: **High**.
Exception text	Text	Defines the exception description used during exception handling and within the batch record. Maximum length is 250 characters.

Output Variables

Instead of specifying a fixed value to be displayed or used during execution, you can also use an expression created in the Expression editor to draw the output of another phase or the calculated result of several phase outputs as value into a parameter attribute. When you reference phase outputs in this manner you need to be aware of the following restrictions:

- 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 **Get values** phase provides the following output variables:

Data reference

- Data type: PhaseDataReference, used for transporting all phase data collected during execution to provide as process input.
- Usage: The output variable provides a reference to a data collection phase.

Boolean Value

If you have added Boolean Value process parameter bundles to the list of parameters, the system provides output variables for each of the bundles.

TIP
Similar to the convention used for naming a bundle's process parameters, its output variables are prefixed with its bundle identifier.

The following output variables are available for Boolean Value bundles:

Boolean Value - Option key

- Data type: Boolean, with the values **true** and **false**
- Usage: The output variable provides the selected boolean option.

Boolean Value - Option text

- Data type: String, used for displaying a pre-defined sequence of characters, such as "**Yes**" or "**No**".
- Usage: The output variable provides the display text of the selected option.

Measured Value

If you have added Measured Value process parameter bundles to the list of parameters, the system provides output variables for each of the bundles.

> **TIP**
> Similar to the convention used for naming a bundle's process parameters, its output variables are prefixed with its bundle identifier.

The following output variables are available for Measured Value bundles:

Measured Value - Unit of measure

- Data type: String, used for displaying a pre-defined sequence of characters, such as "**mm**" or "**ea**".
- Usage: The output variable provides the unit of measure of the process value.

Measured Value - Value

- Data type: MeasuredValue, used for displaying numeric values qualified by a unit of measure.
- Usage: The output variable provides the complete process value as a **MeasuredValue** object.

Option Value

If you have added Option Value process parameter bundles to the list of parameters, the system provides output variables for each of the bundles.

> **TIP**
> Similar to the convention used for naming a bundle's process parameters, its output variables are prefixed with its bundle identifier.

The following output variables are available for Option Value bundles:

Option Value - Option key

- Data type: String, used for displaying a pre-defined sequence of characters, such as "**NEXT_ITEM**" or "**COMPLETED**".
- Usage: The output variable provides the key value of the selected option.

Option Value - Option text

- Data type: String, used for displaying a pre-defined sequence of characters, such as "**Passed without issues**" or "**Passed with issues**".
- Usage: The output variable provides the display text of the selected option.

Identifier

- Data type: String, used for displaying a pre-defined sequence of characters, such as "**Read Instruction**".
- Usage: The output variable provides the identifier of the phase.

Instance count

- Data type: Long, used for integral numbers:
  12345
- Usage: The output variable provides the count of the number of instances the phase has been processed, for example in a loop. The count is also increased when the phase is skipped from an operator's perspective, since the phase is still executed, but as a hidden phase.
  The count variable of a phase that has not been executed provides 0 as output value.

Start time

- Data type: Timestamp, used for displaying dates and times and for time-related calculations.
  To use a timestamp in a phase attribute, you have to make sure it has a matching data type, so to display it in an instruction text, you have to convert it into a string.
- Usage: The output variable provides the start time of the phase.

Completion time

- Data type: Timestamp, used for displaying dates and times and for time-related calculations.
  To use a timestamp in a phase attribute, you have to make sure it has a matching data type, so to display it in an instruction text, you have to convert it into a string.
- Usage: The output variable provides the completion time of the phase.

TIP

To calculate a duration from two timestamps and display it in a specific format, you need to use two conversion functions on the calculation:

- **Convert to Unitless Number** (**convertTo**) takes the calculated duration and converts it into the duration's value for one of its units (e.g. minutes or seconds).
- **Convert to String for Display** (**convertToDisplayString)** takes the converted value and displays it as string to which you can add the unit, also as string.

Example:

Sample Phase with Start time = 14-Nov-2014@10:15
Sample Phase with Completion time = 14-Nov-2014@11:47
The duration is to be displayed in minutes.

```
convertToDisplayString
   (convertTo
        ({Sample Phase}.{Completion time}-{Sample Phase}.{Start time},
"min")
   )
 + " min"
```

As result of the expression, the system displays "**92 min**".

Chapter 5

Show Values

The **Show values** phase allows an operator to view the data that has been collected along with the **Get values** phase across multiple runs, which are based on loops or ETO templates. The data is represented in a table format and supports up to five columns with values of the following data types:

- Measured Value (measured value): value and unit of measure,
- Option Value (choice list): choice from a pre-defined list of options, and
- Boolean Value: choice between Yes and No (`true` and `false`).

It can be used for processing requirements, such as:

- Recording of tablet weights across multiple IPC runs.
- Recording of visual appearance across multiple IPC runs (e.g. Transparent, Cloudy, Dark).
- Recording of the execution of mandatory checks across multiple IPC runs.

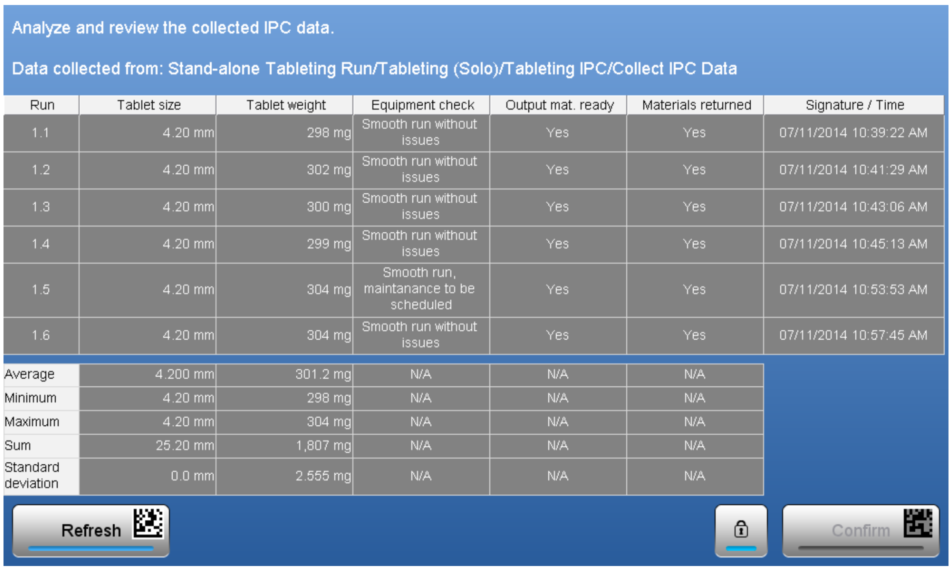
Execution

The **Show values** phase presents and evaluates the data collected during the runs of its linked **Get values** phase in tabular representations. The specified scope of data collection determines if the phase considers only the data collected during the current run of the phase's operation or during all of its runs. It supports up to five Measured, Boolean, or Option Values.
As the phase can run simultaneously with its data collection phase, it provides a **Refresh** button an operator can tap to update the displayed data. When the operator has completed the last run of the data collection phase and has refreshed the displayed data, he can unlock the **Confirm** button and complete the phase.

TIP

Please note that the **Unlock** button that enables phase confirmation is not present when the phase has been configured to request a phase completion signature.

After completion the phase displays the data in the Execution Window.
The Navigator displays the phase completion timestamp.



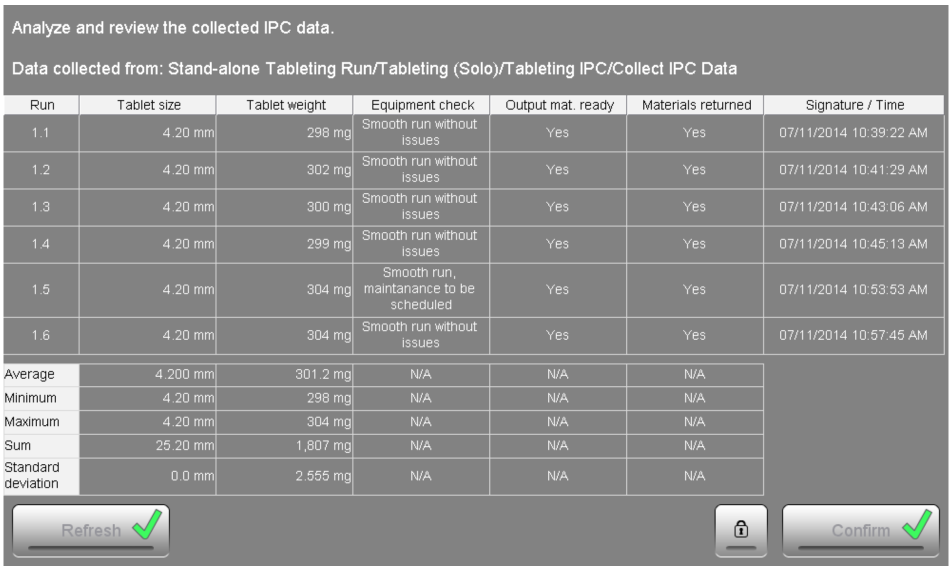
Analyze and review the collected IPC data.

Data collected from: Stand-alone Tableting Run/Tableting (Solo)/Tableting IPC/Collect IPC Data

Run	Tablet size	Tablet weight	Equipment check	Output mat. ready	Materials returned	Signature / Time
1.1	4.20 mm	298 mg	Smooth run without issues	Yes	Yes	07/11/2014 10:39:22 AM
1.2	4.20 mm	302 mg	Smooth run without issues	Yes	Yes	07/11/2014 10:41:29 AM
1.3	4.20 mm	300 mg	Smooth run without issues	Yes	Yes	07/11/2014 10:43:06 AM
1.4	4.20 mm	299 mg	Smooth run without issues	Yes	Yes	07/11/2014 10:45:13 AM
1.5	4.20 mm	304 mg	Smooth run, maintanance to be scheduled	Yes	Yes	07/11/2014 10:53:53 AM
1.6	4.20 mm	304 mg	Smooth run without issues	Yes	Yes	07/11/2014 10:57:45 AM

Average	4.200 mm	301.2 mg	N/A	N/A	N/A
Minimum	4.20 mm	298 mg	N/A	N/A	N/A
Maximum	4.20 mm	304 mg	N/A	N/A	N/A
Sum	25.20 mm	1,807 mg	N/A	N/A	N/A
Standard deviation	0.0 mm	2.555 mg	N/A	N/A	N/A

Figure 21: Show values during execution



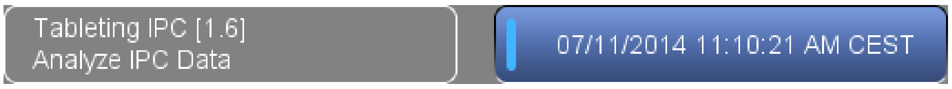
Analyze and review the collected IPC data.

Data collected from: Stand-alone Tableting Run/Tableting (Solo)/Tableting IPC/Collect IPC Data

Run	Tablet size	Tablet weight	Equipment check	Output mat. ready	Materials returned	Signature / Time
1.1	4.20 mm	298 mg	Smooth run without issues	Yes	Yes	07/11/2014 10:39:22 AM
1.2	4.20 mm	302 mg	Smooth run without issues	Yes	Yes	07/11/2014 10:41:29 AM
1.3	4.20 mm	300 mg	Smooth run without issues	Yes	Yes	07/11/2014 10:43:06 AM
1.4	4.20 mm	299 mg	Smooth run without issues	Yes	Yes	07/11/2014 10:45:13 AM
1.5	4.20 mm	304 mg	Smooth run, maintanance to be scheduled	Yes	Yes	07/11/2014 10:53:53 AM
1.6	4.20 mm	304 mg	Smooth run without issues	Yes	Yes	07/11/2014 10:57:45 AM

Average	4.200 mm	301.2 mg	N/A	N/A	N/A
Minimum	4.20 mm	298 mg	N/A	N/A	N/A
Maximum	4.20 mm	304 mg	N/A	N/A	N/A
Sum	25.20 mm	1,807 mg	N/A	N/A	N/A
Standard deviation	0.0 mm	2.555 mg	N/A	N/A	N/A

Figure 22: Show values after phase completion



Tableting IPC [1.6]
Analyze IPC Data

07/11/2014 11:10:21 AM CEST

Figure 23: Show values in the Navigator

Phase Design

The characteristics of the **Show values** phase are defined via process parameters and their attributes.
Its user interface is designed in two columns that span several rows. When the phase is active, the merged columns of the first row provide space for textual instructions. In the second row, the phase displays the path of the **Get values** phase whose data is displayed and evaluated. The next row holds the table of collected data, spanning all columns, which grows with each collection run. In the row below the data table the phase displays a configurable statistics table that can provide statistical evaluations over all collected values of the MeasuredValue data type. The left column of the bottom row contains the left-aligned **Refresh** button and at its right margin the **Unlock** button. The **Confirm** button is located in its right column.
When the phase is completed it shows the same layout with the data of the collected values and their evaluations listed in the two tables.
Exception handling during execution is controlled by a risk assessment classification and an exception message that are both defined by the recipe author in the exception's process parameter.

Process Parameters

In addition to the permanent process parameters that are always present, the **Show values** phase also provides optional process parameter bundles, which you can insert if required. A process parameter bundle is a group of one or more semantically connected process parameters, all of which are needed to produce a specific behavior or function when the phase is executed.

You can add up to five process parameters bundles of the **Statistics** value type to the **Show values** phase, one for each value whose data you wish to evaluate.

ADDING PARAMETER BUNDLES

1. Click the **Add parameter bundle** button.
   The system opens an option list that holds all bundle types available for the phase.
2. Select the type.
   The system opens the **Add <Bundle Type>** dialog to define the bundle's identifier.
3. Type an identifier and click the **OK** button.
   The system adds all process parameters of the bundle to the list of parameters.

TIP
Please note that the identifier of the parameter bundle is used as identifier of the bundle's master parameter and as prefix to all other parameters of the bundle.

Removing parameter bundles

1. In the list of parameters, select any one of the parameters of the bundle you wish to remove.
   All parameter identifiers of a bundle start with the bundle's identifier.
2. Click the **Remove parameter** button.
   The system asks you to confirm the action and then removes all parameters of the bundle.

   TIP
   Please note that you cannot remove individual parameters of a bundle.

The following process parameters are available to configure the phase's behavior during execution:

Instruction

Represents the instruction text that is visible on the preview, the active, and the completed view of the phase.

Attribute	Type	Comment
Column 1	HTML text	Instruction text to be displayed. Maximum length is 2000 characters (including HTML tags).
Column 2	HTML text	Not used
Column 3	HTML text	Not used.

Definition

Defines the phase whose collected data are displayed and whose values of the Measured Value type are available for statistical evaluations. You can choose to either consider only one run of the operation in which the data collection phase is located or all runs of the operation, which can be looped explicitly in the recipe or implicitly as event-triggered operation.

Attribute	Type	Comment
Data collection reference	Reference	Reference to a preceding data collection phase.
Scope of data collection	Choice list	Defines the scope of collected data. Available settings: **Within operation run**, **Across operation runs**. Default setting: **Across operation runs**.

Attribute	Type	Comment
Show table in batch report	Flag	Defines if the table of values is displayed in the batch report. Default settings: **Yes** Note: The table of statistics, if applicable, is always displayed in the batch report.

Statistics - Master (bundle identifier)

Defines the statistical evaluations to be performed on the data of the Measured Value type collected in a referenced **Get values** phase.

Attribute	Type	Comment
Show average	Flag	Defines if the average value is displayed during execution.
Show minimum	Flag	Defines if the minimum value is displayed during execution.
Show maximum	Flag	Defines if the maximum value is displayed during execution.
Show sum	Flag	Defines if the sum value is displayed during execution.
Show standard deviation	Flag	Defines if the standard deviation value is displayed during execution.

The bundle identifier of the statistics bundle must match the bundle identifier of the respective value bundle of the referenced **Get values** phase.
During execution, values can only be calculated and displayed for the Measured Value data type.

Output Variables

Instead of specifying a fixed value to be displayed or used during execution, you can also use an expression created in the Expression editor to draw the output of another phase or the calculated result of several phase outputs as value into a parameter attribute. When you reference phase outputs in this manner you need to be aware of the following restrictions:

- 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 **Show values** phase provides the following output variables:

Statistics

If you have added Statistics process parameter bundles to the list of parameters, the system provides output variables for each of the bundles. The individual output variables are always available, even if the **Master (bundle identifier)** process parameter (page 55) does not configure the evaluations to be displayed.

TIP
Similar to the convention used for naming a bundle's process parameters, its output variables are prefixed with its bundle identifier.

The following output variables are available for Statistics bundles:

Statistics - Average

- Data type: MeasuredValue, used for displaying numeric values qualified by a unit of measure.
- Usage: The output variable provides the average value as a **MeasuredValue** object.

Statistics - Maximum

- Data type: MeasuredValue, used for displaying numeric values qualified by a unit of measure.
- Usage: The output variable provides the maximum value as a **MeasuredValue** object.

Statistics - Minimum

- Data type: MeasuredValue, used for displaying numeric values qualified by a unit of measure.
- Usage: The output variable provides the minimum value as a **MeasuredValue** object.

Statistics - Standard deviation

- Data type: MeasuredValue, used for displaying numeric values qualified by a unit of measure.
- Usage: The output variable provides the standard deviation value as a **MeasuredValue** object.

Statistics - Sum

- Data type: MeasuredValue, used for displaying numeric values qualified by a unit of measure.
- Usage: The output variable provides the sum value as a **MeasuredValue** object.

Identifier

- Data type: String, used for displaying a pre-defined sequence of characters, such as "**Read Instruction**".
- Usage: The output variable provides the identifier of the phase.

Instance count

- Data type: Long, used for integral numbers:
  12345
- Usage: The output variable provides the count of the number of instances the phase has been processed, for example in a loop. The count is also increased when the phase is skipped from an operator's perspective, since the phase is still executed, but as a hidden phase.
  The count variable of a phase that has not been executed provides 0 as output value.

Start time

- Data type: Timestamp, used for displaying dates and times and for time-related calculations.
  To use a timestamp in a phase attribute, you have to make sure it has a matching data type, so to display it in an instruction text, you have to convert it into a string.
- Usage: The output variable provides the start time of the phase.

Completion time

- Data type: Timestamp, used for displaying dates and times and for time-related calculations.
  To use a timestamp in a phase attribute, you have to make sure it has a matching data type, so to display it in an instruction text, you have to convert it into a string.
- Usage: The output variable provides the completion time of the phase.

TIP

To calculate a duration from two timestamps and display it in a specific format, you need to use two conversion functions on the calculation:

- **Convert to Unitless Number** (**convertTo**) takes the calculated duration and converts it into the duration's value for one of its units (e.g. minutes or seconds).
- **Convert to String for Display** (**convertToDisplayString)** takes the converted value and displays it as string to which you can add the unit, also as string.

Example:

Sample Phase with Start time = 14-Nov-2014@10:15
Sample Phase with Completion time = 14-Nov-2014@11:47
The duration is to be displayed in minutes.

```
convertToDisplayString
   (convertTo
        ({Sample Phase}.{Completion time}-{Sample Phase}.{Start time},
"min")
   )
 + " min"
```

As result of the expression, the system displays "**92 min**".

Index