



Recipe Editor User Guide

Rockwell Automation Publication BATCH-UM006C-EN-D - November 2023
Supersedes Publication BATCH-UM006B-EN-D - April 2022



Important User Information

Read this document and the documents listed in the additional resources section about installation, configuration, and operation of this equipment before you install, configure, operate, or maintain this product. Users are required to familiarize themselves with installation and wiring instructions in addition to requirements of all applicable codes, laws, and standards.

Activities including installation, adjustments, putting into service, use, assembly, disassembly, and maintenance are required to be carried out by suitably trained personnel in accordance with applicable code of practice.

If this equipment is used in a manner not specified by the manufacturer, the protection provided by the equipment may be impaired.

In no event will Rockwell Automation, Inc. be responsible or liable for indirect or consequential damages resulting from the use or application of this equipment.

The examples and diagrams in this manual are included solely for illustrative purposes. Because of the many variables and requirements associated with any particular installation, Rockwell Automation, Inc. cannot assume responsibility or liability for actual use based on the examples and diagrams.

No patent liability is assumed by Rockwell Automation, Inc. with respect to use of information, circuits, equipment, or software described in this manual.

Reproduction of the contents of this manual, in whole or in part, without written permission of Rockwell Automation, Inc., is prohibited.

Throughout this manual, when necessary, we use notes to make you aware of safety considerations.



WARNING: Identifies information about practices or circumstances that can cause an explosion in a hazardous environment, which may lead to personal injury or death, property damage, or economic loss.



ATTENTION: Identifies information about practices or circumstances that can lead to personal injury or death, property damage, or economic loss. Attentions help you identify a hazard, avoid a hazard, and recognize the consequence.

IMPORTANT Identifies information that is critical for successful application and understanding of the product.

Labels may also be on or inside the equipment to provide specific precautions.



SHOCK HAZARD: Labels may be on or inside the equipment, for example, a drive or motor, to alert people that dangerous voltage may be present.



BURN HAZARD: Labels may be on or inside the equipment, for example, a drive or motor, to alert people that surfaces may reach dangerous temperatures.



ARC FLASH HAZARD: Labels may be on or inside the equipment, for example, a motor control center, to alert people to potential Arc Flash. Arc Flash will cause severe injury or death. Wear proper Personal Protective Equipment (PPE). Follow ALL Regulatory requirements for safe work practices and for Personal Protective Equipment (PPE).

Rockwell Automation recognizes that some of the terms that are currently used in our industry and in this publication are not in alignment with the movement toward inclusive language in technology. We are proactively collaborating with industry peers to find alternatives to such terms and making changes to our products and content. Please excuse the use of such terms in our content while we implement these changes.

Preface	About this manual.....	9
	Legal Notices	9
	Additional resources	10
	Recipe Editor	12
	Open Recipe Editor	14
	Set Recipe Editor options	15
	Recipe Editor interface	15
	Recipe Editor menu bar	17
	Recipe Editor toolbar	17
	Recipe construction toolbox.....	19
	Recipe Editor example	19
	Recipe creation overview	20
	Chapter 1	
Sequential Function Chart	SFC components	23
	SFC steps	23
	Transition expressions.....	24
	Data types.....	25
	Transition operators	26
	Parameter expression functions	26
	Linked elements	27
	SFC limitations.....	28
	SFC execution	29
	Null procedures	30
	Synchronize execution of steps.....	31
	Arbitration	31
	Subarbitration	33
	Add a null procedure	36
	Timer steps	37
	Timer step hold behavior	37
	Recipe comments	38
	Chapter 2	
Table-based recipes overview	Recipe structure icons	40
	Table view transition expressions	40
	Parallel structures.....	41
	Complex parallelism example	41
	Simple parallelism example.....	42
	Chapter 3	
Recipe creation	Unit-based recipes	45

Unit class-based recipes.....	45
Material-based recipes.....	46
Material class-based recipes.....	46
Dynamic Unit Allocation	46
Unit requirement name.....	47
Downstream unit requirements	48
Enable Dynamic Unit Allocation.....	48
Dynamic Unit Allocation affects binding.....	49
Binding methods.....	49
Criteria for unit selection	49
Open recipe.....	50
Cannot open recipe error message	52
Create operations and unit procedures.....	52
Edit unit procedure requirements	55
Create a procedure	55
Unit requirement name example.....	58
Change procedure unit requirements	59
Smart binding.....	59
Binding requirements.....	60
Configure binding requirements.....	60
Binding preferences.....	61
Configure binding preferences.....	62
Create binding expressions	63
Recipe header data	64
Add recipe header data	66
Recipe storage.....	67
Copy recipe options.....	67
Copy recipe with backup directory	68
Recipe unit requirements	69
Recipe procedure levels.....	70

Chapter 4

Recipe formulations

Formulation parameters	73
Add a recipe formulation.....	74
Delete a recipe formulation	75

Chapter 5

Build a Sequential Function Chart

Add a branch structure to an SFC.....	77
Add transitions to an SFC	78
Define a step.....	78
Link SFC components	79
Remove a link from an SFC	79

	Assign a transition expression	80
	Remove a step from an SFC	80
	Create material loops	81
	Recipe comments	82
	Add comments to an SFC.....	82
	Associate recipe comments	83
	Disassociate recipe comments	84
	Delete recipe comments	84
	Add a parallel step to an SFC	85
	Parallel steps use same resource	86
	Insert steps into an SFC.....	87
	Insert timer steps	88
	Manually add steps to an SFC	88
	 Chapter 6	
Build a table	Insert steps into a table	91
	Add a parallel step to a table.....	91
	Add step before and after parallel	92
	Define a step	93
	Remove a step from a table.....	93
	 Chapter 7	
Bind Expressions	Expression operators expanded	96
	Expression data types	96
	Operands.....	96
	Expression validation	97
	 Chapter 8	
Parameter types in a recipe	Recipe properties	99
	Recipe parameters	99
	Assign recipe formula parameters	99
	Report parameters	100
	Assign report parameters	100
	Aggregate report values	101
	Phase parameters	101
	Operation sequence parameters	101
	Recipe parameters.....	102
	Defer parameters.....	102
	Formula values and report limits	103
	Material recipe parameters	104
	Material report parameters	105
	Formula Value Entry Report.....	105

	Assign values and phase limits.....	108
	Assign formula values to Timer.....	109
Parameter expressions and reports	Chapter 9	
	Evaluate parameter expressions.....	112
	Validate parameter expressions	112
	Parameter expression override.....	112
	Report parameter expressions	112
	Override a report expression.....	113
	Expression value configurations.....	113
	Configure expression values	113
	Parameter expression operators	114
	Parameter expression functions	115
Phase link groups	Chapter 10	
	Message partner phases	117
	Message partner application.....	118
	Link group rules	118
	Phase communication	119
	Create a phase link group	119
	Delete a phase link group.....	120
Recipe approval process overview	Chapter 11	
	Configure recipe approval.....	124
	Approve a recipe.....	125
	Revert a recipe approval	126
	Automatic system signoff.....	128
Recipe versioning overview	Chapter 12	
	Recipe versioning.....	131
	How recipe versions are named.....	131
	Restrictions for recipe versions.....	133
	Enable recipe versioning option	133
	Disable recipe versioning option	134
	Create a recipe version.....	134
	Recipe version control.....	135
	Recipe version history.....	137
	Obsoleted recipe versions	137
	Enable recipe versioning or approval	138
Security authority overview	Chapter 13	
	Security authority configuration.....	141

	Secure recipe.....	142
	Chapter 14	
Complete and maintain recipe	Verify recipe	145
	What gets verified?	146
	SFC validation.....	146
	Run SFC validation.....	147
	Set allowable SFC permutations	148
	SFC validation error types	148
	Parallel activation of segment	149
	Unreachable terminal step	150
	Parallelism with terminal step	150
	Token cannot reach terminal step	151
	Unreachable linear segment.....	153
	Release Recipe as Step option	154
	Release recipe to production	155
	Rebuild the recipe directory	155
	Recipe maintenance	155
	Find Recipe References overview.....	156
	Find recipe references.....	156
	Page setup	157
	Generate reports option.....	157
	Print a working set of recipes.....	158
	Print a single recipe.....	159
	Remove a recipe.....	160
	Translation.....	160
	Chapter 15	
Import and export recipes	Import recipes	163
	Select a directory.....	164
	Select a SQL Server database.....	164
	Export recipes	165
	Import conflicts	166
	Resolve duplicate name conflict.....	166
	Resolve recipe basename conflict.....	167
	Resolve recipe version conflict	167
	Resolve conflicts rebuilding directory	167
	Import and export recipe restrictions	167
	Appendix A	
Recipe formats	RDB format	169
	Add a Windows user to the MasterRecipeAuthor and	

MasterRecipeViewer local groups.....170
Create the MasterRecipes database170
XML format171
XML recipe schema171
Set recipe file format..... 172

Appendix B

Import/Export error issues

Improperly secured recipes.....173
Verification results173
Remove area model conflict 173
Reinstate an obsolete procedure 174
Verification warnings and errors..... 175
Invalid recipe folder or path..... 175

Index

About this manual

This manual provides usage instructions for the FactoryTalk Batch Recipe Editor. It is one of a set of related manuals that describe installing, programming, and operating the FactoryTalk Batch system.

To review FactoryTalk Batch release notes and latest information regarding product compatibility refer to the [Product Compatibility and Download Center \(PCDC\)](#).

Legal Notices

Rockwell Automation publishes legal notices, such as privacy policies, license agreements, trademark disclosures, and other terms and conditions on the [Legal Notices](#) page of the Rockwell Automation website.

Software and Cloud Services Agreement

Review and accept the Rockwell Automation Software and Cloud Services Agreement [here](#).

Open Source Software Licenses

The software included in this product contains copyrighted software that is licensed under one or more open-source licenses.

You can view a full list of all open-source software used in this product and their corresponding licenses by opening the `oss_license.txt` file located your product's `OPENSOURCE` folder on your hard drive. This file is divided into these sections:

- **Components**
Includes the name of the open-source component, its version number, and the type of license.
- **Copyright Text**
Includes the name of the open-source component, its version number, and the copyright declaration.
- **Licenses**
Includes the name of the license, the list of open-source components citing the license, and the terms of the license.

The default location of this file is:

```
C:\program files (x86)\Rockwell Software\Batch View  
Server\docs\relnotes\OPENSOURCE
```

You may obtain Corresponding Source code for open-source packages included in this product from their respective project web site(s).

Alternatively, you may obtain complete Corresponding Source code by contacting Rockwell Automation via the **Contact** form on the Rockwell Automation website:

<http://www.rockwellautomation.com/global/about-us/contact/contact.page>.

Please include "Open Source" as part of the request text.

Additional resources

This table is a comprehensive documentation list for the FactoryTalk® Batch products from Rockwell Automation.

Installation, Quick Start, and Getting Results Guides

Resource	Description
FactoryTalk Batch Components Installation and Upgrade Guide (BATCH-IN002)	Provides information and procedures for FactoryTalk Batch system installation. Includes information for FactoryTalk Batch Material Manager, FactoryTalk Event Archiver, and associated FactoryTalk Batch Client and Server components.
FactoryTalk Batch View Quick Start Guide (FTBVS-QS001)	Provides information about using FactoryTalk Batch View to create, view, and command control recipes, acknowledge prompts and signatures, view equipment phases and diagnostic information, and view profile information.
FactoryTalk Batch View HMI Controls Quick Start Guide (BATCH-QS001D)	Provides a general overview of FactoryTalk Batch View HMI Controls.
FactoryTalk Batch eProcedure® Getting Results Guide (BWEPRO-GR011)	Explains the basics of FactoryTalk Batch eProcedure.
FactoryTalk Batch Getting Results Guide (BATCH-GR011)	Introduces the basics of automated batch manufacturing and the FactoryTalk Batch product components.
FactoryTalk Batch Material Manager Getting Results Guide (BWMTR-GR011)	Introduces the basics of FactoryTalk Batch Material Manager.

User Guides

Resource	Description
FactoryTalk Batch Material Editor User Guide (BWMTR-UM001)	Provides access to information and procedural instructions required to configure materials and the containers to hold them. The material data is stored in the material database, which is used to create material-based recipes. This information is intended as a reference for formulators.
FactoryTalk Batch Equipment Editor User Guide (BATCH-UM004)	Provides information on creating and maintaining an equipment database (area model). The area model is available to all other FactoryTalk Batch programs, including the Recipe Editor, Batch View, and Phase Simulator.

Resource	Description
FactoryTalk Batch PhaseManager™ User Guide (BATCHX-UM011)	Describes the integration of the FactoryTalk Batch software with the Studio 5000 Logix Designer® application and the Logix 5000™ family of controllers. The integration simplifies the configuration and maintenance of the FactoryTalk Batch automation system, provides better communication between the FactoryTalk Batch Server and the Logix 5000 controller, and significantly reduces the programming effort required to develop the phase logic code that resides in your Logix 5000 controller.
FactoryTalk Batch Recipe Editor User Guide (BATCH-UM006)	Provides instructions on using FactoryTalk Batch Recipe Editor to create and configure master recipes for use in batch automation. The interface is based on IEC 61131-3 sequential function charts to organize recipes graphically into procedures, unit procedures, operations, and phases. Build recipes using either the SFC format or a table-based format.
FactoryTalk Batch View HMI Controls User Manual (FTBVS-UM003)	Provides details about using FactoryTalk Batch View HMI Controls to monitor and interact with the production process within a FactoryTalk View SE Display Client.
FactoryTalk Batch View User Manual (FTBVS-UM002)	Provides information and procedural instructions for using FactoryTalk Batch View in a modern and intuitive portal into a comprehensive batching solution for effective operations, leveraging its own web server using HTML5 technology to provide connectivity into a FactoryTalk Batch Server.
FactoryTalk Event Archiver User Guide (BATCH-UM012)	Provides information and instructions specific to the FactoryTalk Event Archiver. Intended for use by system administrators and production supervisors.

Administrator Guides

Resource	Description
FactoryTalk Batch Administrator Guide (BATCH-UM003)	Provides instructions for configuring security and services, and implementation and use of components not typically accessed or used by batch operators, such as the FactoryTalk Batch Server.
FactoryTalk Batch eProcedure Administrator Guide (BWEPRO-UM011)	Provides procedures specific to FactoryTalk Batch eProcedure, such as implementing security. Included are instructions for tasks specific to FactoryTalk Batch, such as configuring security and services to support FactoryTalk Batch eProcedure. Provides instructions on the implementation and use of components not typically accessed or used by batch operators, such as the FactoryTalk Batch Server.
FactoryTalk Batch Material Manager Administrator Guide (BWEPRO-UM011)	Provides information and instructions specific to FactoryTalk Batch Material Manager. Intended for use by system administrators and database administrators.

Reference Guides

Resource	Description
FactoryTalk Batch Material Server API Reference Manual (BWMTR-RM001)	Provides access to information regarding the interface between the FactoryTalk Batch Material Server and the FactoryTalk Batch Material Editor and FactoryTalk Batch. It is intended to be used as a reference information by custom interface developers.
FactoryTalk Batch PCD Programming Reference Manual (BATCH-RM004)	Provides information and instructions about the FactoryTalk Batch PCD interface design. It is intended to be used as a reference guide for PCD programmers.

Resource	Description
FactoryTalk Batch Server API Reference Manual (BATCH-RM003)	Provides information regarding the interface between the FactoryTalk Batch Server and FactoryTalk Batch View — the Server Application Programming Interface (API). It is intended to be used as a reference guide by custom interface developers.
FactoryTalk Batch System Files Reference Manual (BATCH-RM005)	Provides the technical information for configuration and maintenance of a FactoryTalk Batch system. It can be used as a reference information for implementation engineers and system administrators.
FactoryTalk Batch eProcedure Instruction File Design Reference Manual (BWEPRO-RM001)	Includes information about the building of manual nstruction files for manual phases in the equipment database This information is intended to be used as a reference by instruction file authors.

View or download publications at <http://www.rockwellautomation.com/literature>. To order paper copies of technical documentation, contact your local Allen-Bradley® distributor or sales representative.

Rockwell Automation recognizes that some of the terms that are currently used in our industry and in this publication are not in alignment with the movement toward inclusive language in technology. We are proactively collaborating with industry peers to find alternatives to such terms and making changes to our products and content. Please excuse the use of such terms in our content while we implement these changes.

Recipe Editor

Use FactoryTalk Batch Recipe Editor to create and configure master recipes for use in batch automation. A master recipe is a type of recipe that accounts for equipment capabilities and may include process cell-specific information.

The interface contains IEC 61131-3 sequential function charts to graphically organize recipes into procedures, unit procedures, operations, and phases. Build recipes using either the SFC format or a table-based format.

Procedural levels:

Recipes use the ISA S88.01 Batch Control Standards for configuration and display, which define these levels for the procedural model:

- **Batch control:** Consists of a sequence of one or more steps (phases) that must be performed in a defined order for a finite period of time to process finite quantities of input material to produce finished product.
- **Procedure:** The strategy for carrying out a process. In general, it refers to the strategy for making a batch within a process cell. It may refer to a process that does not result in the production of a product.

Examples:	Make Product A	Make Product B
------------------	----------------	----------------

- **Unit Procedure:** A strategy for carrying out a contiguous process within a unit. It consists of contiguous operations and the algorithm necessary for the initiation, organization, and control of those operations.

Examples:	Emulsification	Dehydrogenation
------------------	----------------	-----------------

- **Operation:** A procedural element defining an independent processing activity consisting of the algorithm necessary for the initiation, organization, and control of phases.

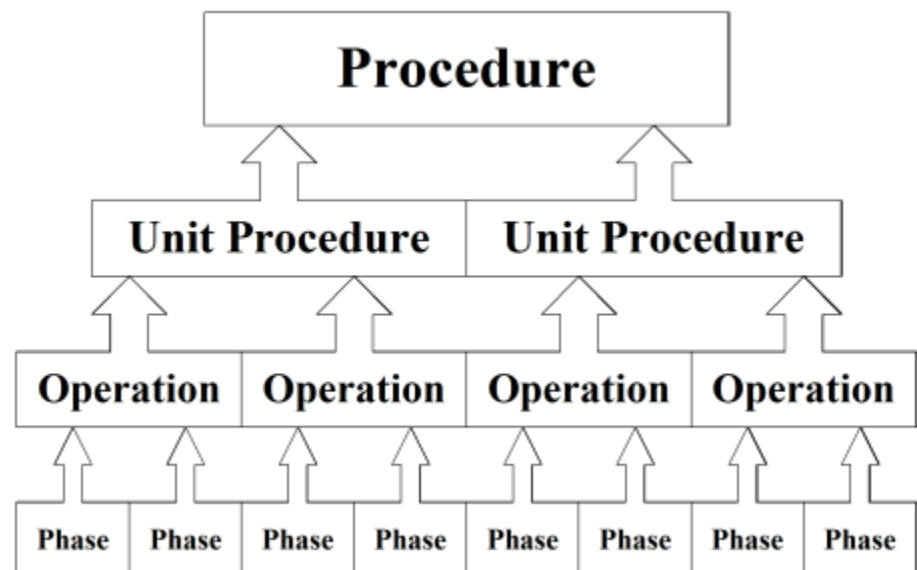
Examples:	Filtration	Reaction
------------------	------------	----------

- **Phase:** The lowest level of the procedural element in the procedural control model. A phase consists of the control steps and the algorithm necessary for the initiation, organization, and control of those steps.

Examples:	Agitate	Heat
------------------	---------	------

The recipe phase is the lowest level within a recipe. The phase provides an interface to basic control. The recipe phase maps directly to the engineered logic on the plant floor. Base an added phase on a phase already defined in the area model using the FactoryTalk Batch Equipment Editor. Based on the unit requirements specified in the recipe, a phase in the FactoryTalk Batch Equipment Editor maps to a specific phase (an instance of a phase class). This phase is mapped (using tags) to equipment phase defined in the engineered logic.

An operation consists of one or more recipe phases, and it must run within a single unit in the area model. Combine multiple operations into a single unit procedure if each operation runs in the same unit in the area model. Unit procedures combine to create a procedure, the highest recipe level. Procedures can run across multiple units, allowing for unit-to-unit transfers.



Open Recipe Editor

Follow these instructions to open the FactoryTalk Batch Recipe Editor.

1. Select **Start > Rockwell Software > Recipe Editor**.
2. If the **Log on to FactoryTalk** dialog box opens, enter the user name and password and select **OK**.

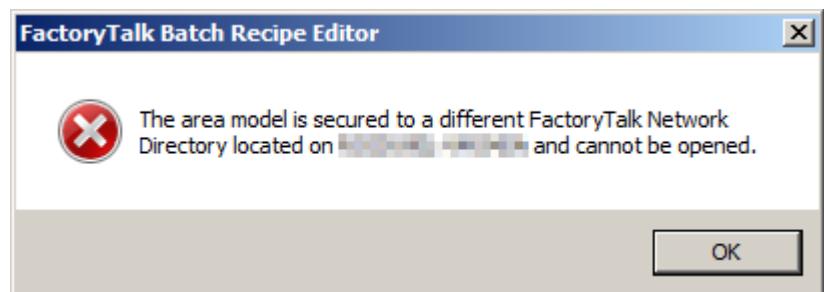


Tip: If **SingleSignOn** is configured, the **Log On to FactoryTalk** dialog box does not open. Proceed to step 5.

3. If the FactoryTalk user name and password is not configured to run the FactoryTalk Batch Recipe Editor, this message opens.



4. If the user is not found, a **Logon Message** opens. Select **OK** to close this message, check the user credentials, and retry or enter other credentials.
5. If Security Authority is applied to the area model, and there is a mismatch between the security authority identifier (SAI) in the area model and the (SAI) in the current FactoryTalk Network Directory, this error message opens:



Make note of the FactoryTalk Network Directory and host computer information, then select **OK** to close the error message and exit the FactoryTalk Batch Recipe Editor.

6. If the recipe directory defined in the FactoryTalk Batch Equipment Editor contains binary recipes from a previous version of FactoryTalk Batch, a dialog box opens to translate the recipes to the new schema. After translation, verify the recipes.



Tip: If upgrading from a FactoryTalk Batch version older than the previous major release (such as from version 10.x or earlier to version 12.x), contact the Rockwell Customer Support Representative to have recipes upgraded to the new recipe schema.

7. The **Recipes Requiring Verification** dialog box may open when the FactoryTalk Batch Recipe Editor opens. Select **Verify All** to verify any unverified recipes. Select **Verify and Validate All** to verify and validate the SFCs of any unverified recipes.

IMPORTANT FactoryTalk users with ViewOnly permissions to the FactoryTalk Batch Recipe Editor cannot open the FactoryTalk Batch Recipe Editor to verify recipes. The FactoryTalk Batch Recipe Editor closes.

The FactoryTalk Batch Recipe Editor displays the **Procedure View** and **Recipe Construction** panes.

Set Recipe Editor options

Configure the general FactoryTalk Batch Recipe Editor options, such as enabling/disabling support for dynamic unit allocation and parallel operations, as well as configuring the pane size in the FactoryTalk Batch Recipe Editor window. Selected options apply to all recipes saved after the options are applied.



Tip: FactoryTalk Batch procedures can have parallel unit procedures and operations can have parallel equipment phases, regardless of this options setting.

To refer to the area model, open the FactoryTalk Batch Equipment Editor.

1. Select **View > Options**. The **Options** dialog box opens.

These options are available in the **Options** dialog box:

Item	Description
Support Dynamic Unit Allocation	Dynamic unit allocation enables the creation of Unit Class-based recipes with specific unit requirements (such as downstream unit requirements) and allows the choice of unit binding type to apply. If selected, the Procedure - Unit Requirements dialog box displays. The Unit Requirement menu selection becomes active in a procedure, and the dialog for defining a step includes a unit requirement selection.
Allow Parallel Operations	Select this option to allow the creation of AND convergences and divergences, which allow operations to run simultaneously within unit procedures.
Font Size	Set the font size for text displayed in the procedure view outline (left frame). The default is 7 points.
Number of Decimal Places for the Display Parameter	This option adjusts the number of decimal places displayed for real numbers in the SFC View . This value defaults to 2. The actual value of the parameter is unaltered. Only the number displayed in the FactoryTalk Batch Recipe Editor in the SFC View is affected.
Maximum SFC Permutations	The maximum number of SFC permutations (caused by OR Divergences within a recipe) that the SFC Validation Tool allows. Recipes structures exceeding this limit are too complex to verify. Please note that the higher this setting, the longer SFC validation takes.


2. Set the FactoryTalk Batch Recipe Editor options as required.
3. Select **Apply** to save selections.

Recipe Editor interface


The FactoryTalk Batch Recipe Editor interface provides a workspace and tools to graphically construct recipes and specify a sequence of phases in an

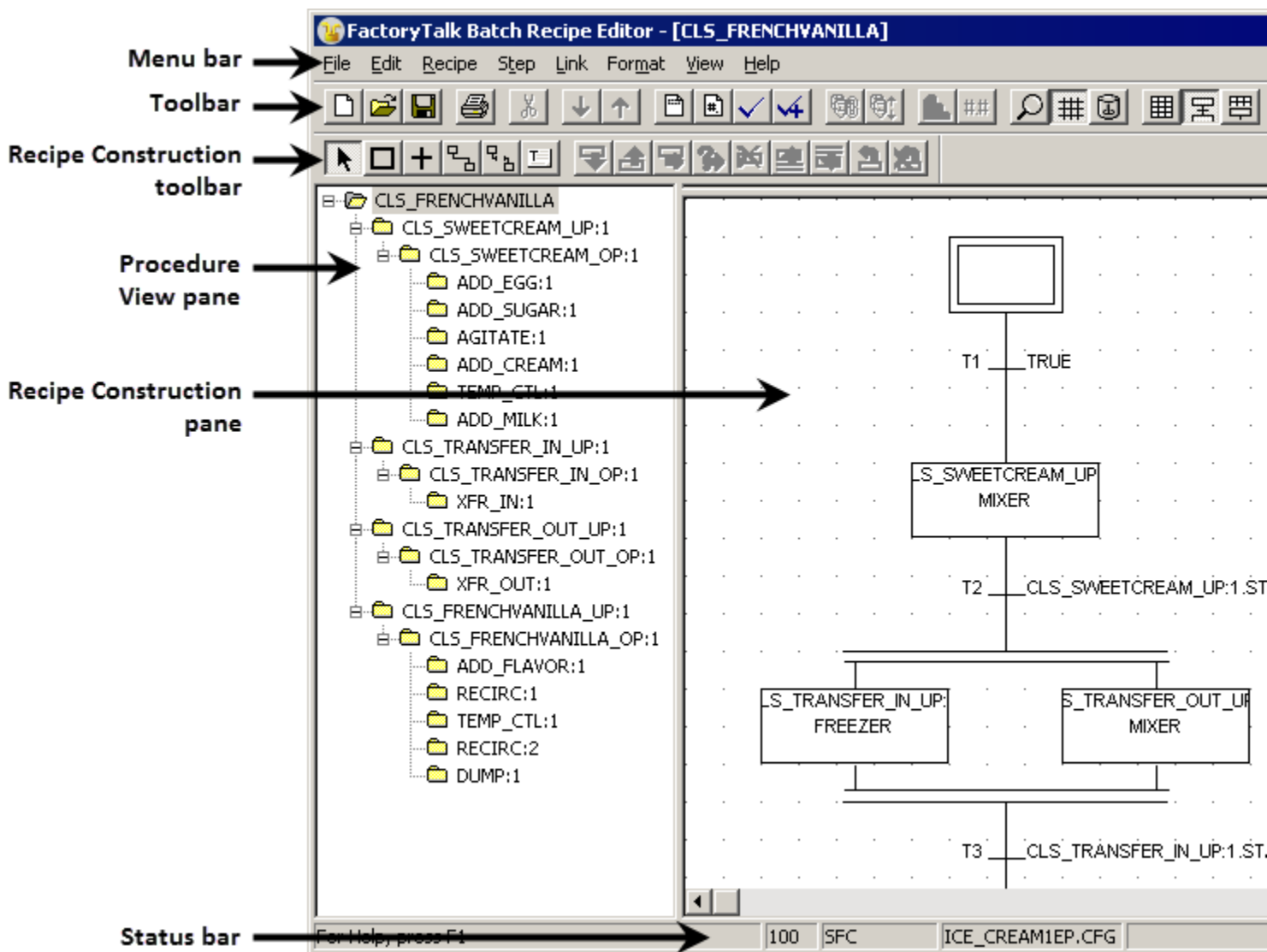
operation. Configure the actual phase logic in the process-connected device (PCD). Configure the interface to the PCD in the FactoryTalk Batch Equipment Editor.

The Procedure View Pane contains a hierarchical list of the current recipe components. Selecting a component from the list highlights the corresponding step in the **Recipe Construction** pane.

 Tip: To resize the panes in the FactoryTalk Batch Recipe Editor, select and drag the splitter bar between the two panes to the desired size.

The **Recipe Construction** pane provides a way to construct and view recipe structures using a sequential function chart (SFC) or a table. The **SFC** view and the **Table** view display exclusively. Tile the **Recipe Construction** pane to display both views at the same time. Selecting a component within either view highlights the corresponding item in the **Procedure View** pane.

 Tip: When using tiled views, selecting a recipe step in one view highlights the corresponding item in the other view. The selected step in the active view highlights in dark blue. The step in the inactive view highlights in light blue. Additionally, the status bar displays the word **Table** or **SFC** accordingly.



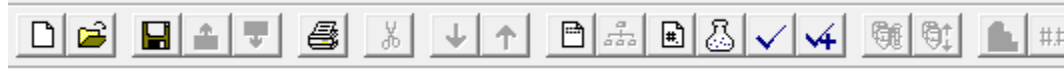
Recipe Editor menu bar

Opened recipe menu bar commands:




Item	Description
File	Lists options for creating new recipes (New Top Level), open existing recipes (Open Top Level), close recipes, and save recipes. Use the File menu to create recipe versions (Check In), work-in-progress (WIP) copies (Check Out), remove recipes, rebuild the recipe directory, verify recipes, validate SFCs, import recipes, and export recipes. Secure an open recipe using Security Authority . The Page Setup (printer/paper choices), Generate Reports (print options), and Exit commands available from the File menu.
Edit	Select, add, or remove a recipe step within the SFC View and Table View . Select, add, or remove transitions, links, and material loops within the SFC View . The Edit menu contains the option to add, link, or delete recipe comments (Text Box). These commands are also available in the Recipe Construction Toolbox .
Recipe	Enter or view recipe header information, view recipe version parent information, approve or revert recipe approvals, add/delete recipe formula parameters and reports, edit unit/bind requirements, set bind preferences, and verify recipes.
Step	Allows redefine, enter formula values, parameter values, and report limits for a selected step.
Link	Create, edit, and delete phase link groups.
Format	Modify the magnification of the SFC view and align SFC elements to a grid.
View	Switch between the SFC View , Table View , or to tile the Recipe Construction Pane for displaying both views. Customize the FactoryTalk Batch Recipe Editor by hiding/showing the Toolbar , Status Bar , and recipe construction Toolbox . Additionally, it allows show/hide the page boundaries and set the FactoryTalk Batch Recipe Editor options: <ul style="list-style-type: none"> • Supporting dynamic unit allocation • Parallel operations • Formula parameter decimal places • Maximum number of SFC validation OR permutations
Help	Provides online help and information about FactoryTalk Batch.
























Recipe Editor toolbar

The FactoryTalk Batch Recipe Editor toolbar contains buttons used to perform FactoryTalk Batch Recipe Editor commands. You can reposition and detach the toolbar from the window by dragging it to a new position. When the toolbar is detached from the window, moved or hide by right-clicking it and selecting **Hide or Move** from the shortcut menu.



This table describes the FactoryTalk Batch Recipe Editor toolbar buttons and their corresponding keyboard shortcuts:
















Graphic	Name	Description	Keyboard Shortcut
	New Top Level	Creates a new procedure, unit procedure, or operation.	Ctrl+N
	Open Top Level	Opens an existing recipe file.	Ctrl+O
	Save	Saves the current recipe.	Ctrl+S

Graphic	Name	Description	Keyboard Shortcut
	Check In	Check in and version the recipe element (create a recipe version).	None
	Check Out	Check out the versioned recipe to edit (create a work-in-progress (WIP) copy).	None
	Generate Reports	Opens the Generate Reports dialog box for printing and reporting options.	Ctrl+T
	Delete Selection	Deletes the currently selected steps, transitions, or text boxes.	Delete
	Go Down	Moves down through the recipe levels and displays the related recipe structure for the step.	None
	Go Up	Moves up through the recipe levels and displays the related recipe structure for the step.	None
	Header Data	Enter or change header data for the viewed recipe level, including the name.	Ctrl+H
	Version History	View information for the currently open recipe and its previous version.	None
	Recipe Formula Parameters	Create, edit, and delete recipe formula parameters.	Ctrl+R
	Recipe Formulations	Add and delete recipe formulations.	Ctrl+W
	Verify	Verifies recipes for proper construction and displays a list of unresolved problems with a recipe. Also saves the recipe in the process.	Ctrl+L
	Verify and Validate	Verifies recipes based on the current unit requirements and validates the SFC structure. Also saves the recipe in the process.	Ctrl+N
	Bind Requirements	Add or delete binding requirements assigned to a unit requirement. Available only with enabled Dynamic Unit Allocation.	Ctrl+M
	Bind Preferences	Add, delete, and prioritize unit requirement binding requirements. Available only with enabled Dynamic Unit Allocation.	Ctrl+B
	Redefine Step	Change the unit procedure, operation, or phase for the selected step.	Ctrl+I
	Value Entry	Enter formula values and modify the phase report limits for the selected step.	Ctrl+A
	Zoom	Specify the magnification of the current SFC display.	None
	Grid Snap	Displays a grid in the Recipe Construction pane, and aligns elements of the SFC to the grid when moved in the Recipe Construction pane.	Ctrl+G
	Invoke Equipment Editor	Opens the FactoryTalk Batch Equipment Editor.	None
	View as Table	Displays the Table View within the Recipe Construction pane.	Shift+Q
	View a SFC	Displays the SFC View within the Recipe Construction pane.	Shift+C
	Tile Views	Displays the Table View and the SFC View within the Recipe Construction pane.	Shift+Z
	Show/Hide Page Boundaries	Displays the recipe print layout in the SFC View .	None

Recipe construction toolbox

The Recipe Construction toolbox contains the command buttons for creating and editing steps, transitions, text boxes, and links in the recipe construction pane. Reposition or detach the toolbox from the window dragging it to a new position. When the toolbox is detached from the window, you can move or hide it. Right-click the toolbox and select the appropriate option from the shortcut menu. Press **Ctrl+Q** to add or remove the toolbox from the FactoryTalk Batch Recipe Editor window.

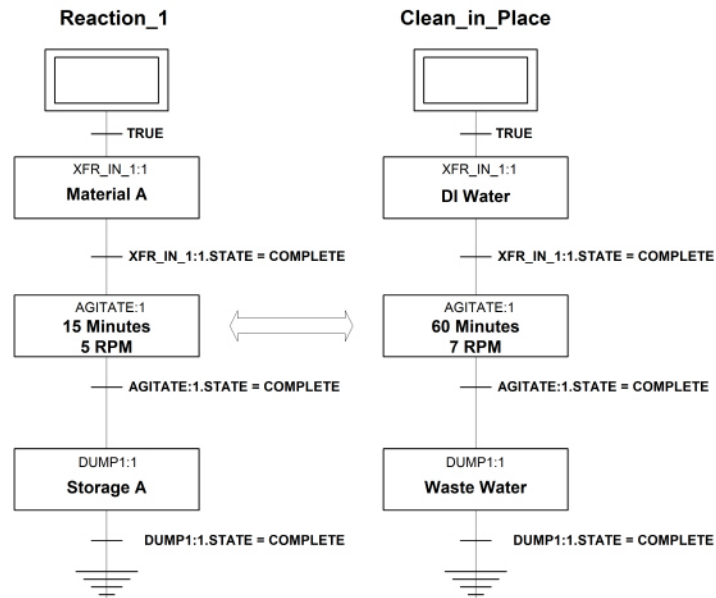
The toolbox command buttons and the keyboard shortcut keys:

Graphic	Name	Description	Keyboard Shortcut
	Selection Tool	Selects a step in the Procedure View pane, Table view, or SFC view. Select transitions in the SFC view. Select and drag the selection pointer over several SFC elements to select a group of elements.	Shift+P
	Step Tool	Adds a new step to the SFC.	Shift+S
	Transition Tool	Adds a new transition to the SFC.	Shift+T
	Link Tool	Links recipe components in the SFC view. Select a step and drag the cursor from the step to a transition to link the two components.	Shift+L
	Remove Link Tool	Unlinks recipe components in the SFC view. Select a step and drag the cursor from the step to a transition to unlink the two components.	Shift+R
	Text Box Tool	Adds a text box for recipe comments.	Shift+D
	Add Step	Adds a step and transition after the selected recipe element. (Valid for initial, sequential, and parallel steps, as well as transitions.)	Shift+A
	Insert Step	Inserts a step and transition before the selected recipe element. (Valid for sequential, parallel, and terminal steps, as well as transitions.)	Shift+I Shift+Insert
	Add Parallel	Adds a step in parallel with the selected step. (Valid for sequential and parallel steps only.)	Shift+E
	Add Branch	Adds a step and additional transitions to form a branch structure. (Valid for regular steps only.)	Shift+B
	Remove Step	Removes a step from the SFC structure, automatically removing the transitions and rearranging the SFC. (Valid for sequential and parallel steps only.)	Shift+M
	Insert Step Before Parallel	Inserts a step before a parallel structure. Select a step within the parallel structure to enable the tool.	Shift+O
	Add Step After Parallel	Adds a step after a parallel structure. Select a step within the parallel structure to enable the tool.	Shift+F
	Create Material Loop	Creates a material loop for the selected material enabled step. The step must have the preceding and following elements attached before creating the material loop.	Shift+N
	Undo Material Loop	Removes a material loop just created, before any other editing takes place.	Shift+X

Recipe Editor example

To continue the example in the FactoryTalk Batch Recipe Editor, you create two operations, Reaction_1 and Clean_In_Place, which are shown below. The Reaction_1 operation transfers Material A into the reactor, agitates the contents for 15 minutes at 5 RPM, and then dumps the contents into Storage Unit A. The Clean_in_Place operation transfers water into the reactor, agitates the contents for 60 minutes at 7 RPM, and then dumps the contents as Waste Water.

Both operations use the same phases but in a different way. You control how the phase operates through the use of parameters, such as speed and time as shown in the AGITATE:1 phase.



For more information regarding how FactoryTalk Batch applies the ISA S88.01 concepts, contact your Rockwell Sales Representative. For a copy of the S88.01 document, write to: **ISA**, 67 Alexander Drive, P.O. Box 12277, Research Triangle Park, NC 27709

See also

Start the sample FactoryTalk Batch Phase Simulator

Recipe creation overview

A common best practice is to design a recipe top-down and build it bottom-up, focusing on high-level procedures that accurately depict the process used to make the product. In determining where to segment the process into Unit Procedures and Operations, consider breaking the process into reusable modules.

Each recipe level consists of descriptive information, formula information, unit requirements, and the required processes to make the batch. A batch is a running control recipe. The material that is being produced or that has been produced by a single execution of a recipe is also considered a batch.

A parameter is used to allow for flexibility in the recipe creation process. Parameter values are used in transition conditions or for substitution of phase parameters. A parameter is created for these recipe levels: operation, unit procedure or procedure. Values are assigned when a unit procedure or operation step is added to a recipe. A recipe parameter is specific to a recipe and is on the recipe phase and downloaded to the equipment phase at runtime.

A phase link group identifies phases that may communicate and work together.

The processes involved in building a recipe:

Recipe Creation

For recipe creation:

- Determine the procedure levels.
- Specify unit requirements.
- Enter recipe header data.
- Define recipe parameters.
- Save the recipe.



Tip: After creating a new blank recipe, build the recipe structure by adding and configuring recipe steps and transitions. Represent the recipe structure using a SFC (sequential function chart), a table, or both. (See **Build a Sequential Function Chart** or **Build a Table** for more information.)

SFC or Table Creation and Configuration

For SFC or Table creation and configuration:

- Create a sequential function chart or table.
- Define the steps.
- Configure the step parameters.
- Define the transitions (SFC only).
- Create phase link groups.

Recipe Completion

To complete recipe creation:

- Verify the recipe and correct errors.
- (optional) Create a version of the recipe to set it to read-only and protect it from further modification.
- (optional) Complete the Recipe Approvals process, either in conjunction with the recipe version process, or on its own.
 - Set the recipe property **Release recipe as step** to true, or signoff on it using the approval process.
 - Set the recipe property **Release recipe to production** to true, or signoff on it by using the approval process.
- (optional) Secure the recipe using Security Authority. When secured, the recipe is bound to the Security Authority Identifier in the current FactoryTalk Network Directory. Afterwards, the same FactoryTalk Network Directory opens the recipe.

Recipe Maintenance

For recipe maintenance:

- Find recipe references.
- Create additional recipe versions, or work-in-progress (WIP) copies of a versioned recipe, to handle modifications to the area model, or changes to process and formula requirements.
- Print the recipe documentation.
- Remove a recipe file.
- Translate recipes.

Sequential Function Chart






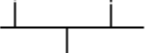

In the FactoryTalk Batch Recipe Editor, create a sequential function chart (SFC) to represent the recipe logic flow. SFC provides a more flexible alternative to creating recipes. SFC methodologies define each recipe procedure and equipment.

Benefits of SFC:

- Unambiguous representation of logic.
- Support for all important logic structures: decisions, loops, and parallelism.
- Standardized in IEC 61131-3 specification.

SFC components

Sequential function charts are composed of six graphical structures linked to form a chart. Each of the graphical structures represents an element in the recipe. The rules for execution of an SFC describe how the procedure executes. The table describes each recipe element and the graphical structure used in the SFC.

Step	Purpose	Example
Initial Step	The logical start of the SFC.	
Step	A reference to a subordinate recipe element.	
Final Step	The logical end of the SFC.	
Transition	Defines how recipe control moves from step to step.	
OR Divergence	Represents a decision, where recipe control passes to only one of the subsequent steps.	
AND Divergence	Represents multiple procedures processed concurrently.	
OR Convergence	Indicates where two optional execution paths converge back into one execution path.	
AND Convergence	Indicates where two simultaneous execution paths converge back into one execution path.	

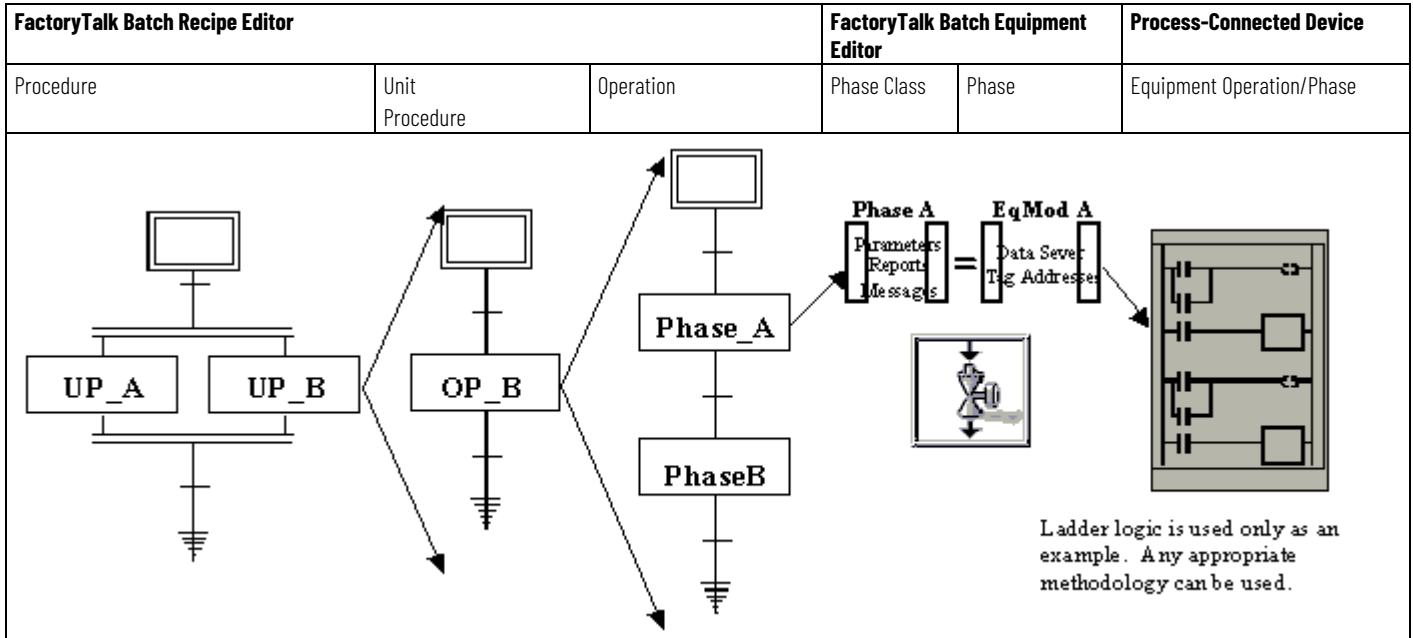
SFC steps

A recipe can contain up to four procedure levels:

- Procedure
- Unit Procedure
- Operation
- Phase

Each of the first three procedure levels can contain one or more steps, each referencing a lower level procedure. A step in a procedure references a unit procedure. A step in a unit procedure references an operation. And a step in an operation references a phase. Define phases in the area model using the FactoryTalk Batch Equipment Editor. A phase is a specific instance of a phase class. The phase maps to the equipment phase, which is the engineered logic residing in the process-connected device (PCD).

This diagram depicts the relationships between the recipe procedure levels and the phase logic in the PCD. Equipment phases and phases are the links between the recipe and the engineered logic.



Transition expressions

Transitions are the criteria that must be met before the recipe continues to the next logical step. Transition expressions are evaluated for syntax at the time they are created and are designated in the SFC with the name Tn, where n represents a unique number. The expression associated with each transition follows the traditional expression evaluation for the arithmetic operators (+, -, *, and /), the logical operators (AND, OR, and NOT), and the functions (ABS, MOD, TRUN, RND, and RDUP). The expression must always evaluate to either TRUE or FALSE.

With transfer of control, a running equipment phase receives notifications as expressions become TRUE that new input parameter values are available. This is modeled by a series of steps referencing the phase to run continuously. Each step contains the new set of inputs. The transitions between the steps test the process conditions. The equipment phase does not stop running until the transfer of control series is COMPLETE.



Tip: The maximum length of a transition expression is 1023 characters. If the transition expression exceeds this limit, reorganize the parallel steps in the recipe.

Transition Identifiers	Example
-------------------------------	----------------

TRUE Evaluates to non-zero	
FALSE Evaluates to zero (0)	
<unit>.<unit tag> <unit> represents one of the pre-configured units, and <unit tag> is a legal tag for the <unit>	PREMIX.AL101
<unit class>.<tag class> For class-based recipes, specify the name of the <unit class> and an associated <tag class>	PREMIX.LEVEL
<step name>.<step attribute> <step name> is the name of one of the steps in the same level of the recipe <step attribute> is one of the following:	
FAILURE The FAILURE attribute is zero if no error has been encountered. Otherwise, its value is greater than zero.	XFR_OUT_1:1.FAILURE = 0
OWNERID The OWNERID attribute can be either PROGRAM or EXTERNAL, where PROGRAM signifies the batch is controlling the step and EXTERNAL means that the batch is not controlling the step.	XFR_OUT_1:1.OWNERID = PROGRAM
PAUSE Set by the FactoryTalk Batch Server. Used to command the phase to pause at the next programmed pause transition. Can have the value of TRUE or FALSE.	XFR_OUT_1:1.PAUSE = TRUE
PAUSED Set by phase logic when the phase reaches a pause location. Can have the value of TRUE or FALSE	XFR_OUT_1:1.PAUSED = FALSE
SINGLESTEP Set by the FactoryTalk Batch Server. Allows the phase to pause at each location. Can have the value of TRUE or FALSE	XFR_OUT_1:1.SINGLESTEP= TRUE
STATE Legal STATE identifiers are: ABORTING, HOLDING, STOPPING, STARTING, RESTARTING, RUNNING, HELD, COMPLETE, STOPPED, ABORTED, IDLE, and READY	XFR_OUT_1:1.STATE = COMPLETE
STEPINDEX An integer which represents the current step index of the active phase.	XFR_OUT_1:1.STEPIINDEX= 10
<step name>.<report parameter> If the step represents a recipe level, phase, or operation sequence, it may have report parameters. Report parameters contain values of tags written from a PCD/phase to FactoryTalk Batch.	XFR_OUT_1:1.AMOUNT_TRANSFERRED
<step name>.<recipe formula parameter> Recipe formula parameters for a step contain the values for the recipe level, phase, or operation sequence represented by the step.	XFR_OUT_1:1.AMOUNT_TO_TRANSFER
<recipe formula parameter> Each level of the recipe may have recipe formula parameters. The values for these parameters may be found in higher levels of the recipe.	AMOUNT_TO_TRANSFER

Data types

The data types supported are integer, real, string, and enumeration.



Tip: Unsigned data types are not supported.

The following are data type examples:

Integer: 423

Real: 423.123456789012

String: The string constant must be in quotes: "READY".

Enumeration: As a string, the enumeration constant must be in quotes: "BUTTER_PECAN". As an integer, the ordinal for the enumeration may be: 4.

IMPORTANT If you want the result of the expression to be an Integer, the values used to build the expression must be Integers – Real is not compatible with an Integer. However, using division in an expression always results in the value being presented as a Real number.

Transition operators

Transition expressions support these operators. The precedence of the execution depicts from highest to lowest. An operator with a higher precedence executes before an operator of lower precedence.

Transition Operator	Description
()	Expressions within parentheses evaluates before expressions outside of parentheses.
NOT, unary minus	Logical NOT user, and "-".
*, /, AND	Multiplication, division, and logical AND.
+, -, OR	Addition, subtraction, and logical OR.
<, <=, >, >=	Less than, less than or equal to, greater than, and greater than or equal to.
=, <>	Equal to, and not equal to.

Parameter expression functions

Functions determine how the expression parser handles Real and Integer data types used in a parameter expression. This table lists available functions and their behavior on positive and negative values:

Function	Description	Behavior: Value	Behavior: Result
RND()	Round – Numeric values round to the nearest integer.	6.7	7
		6.5	7
		6.3	6
		-6.7	-7
		-6.5	-6
		-6.3	-6
RDUP()	Round up – Numeric values round to the next larger integer.	6.7	7
		6.5	7
		6.3	7
		-6.7	-6
		-6.5	-6
		-6.3	-6

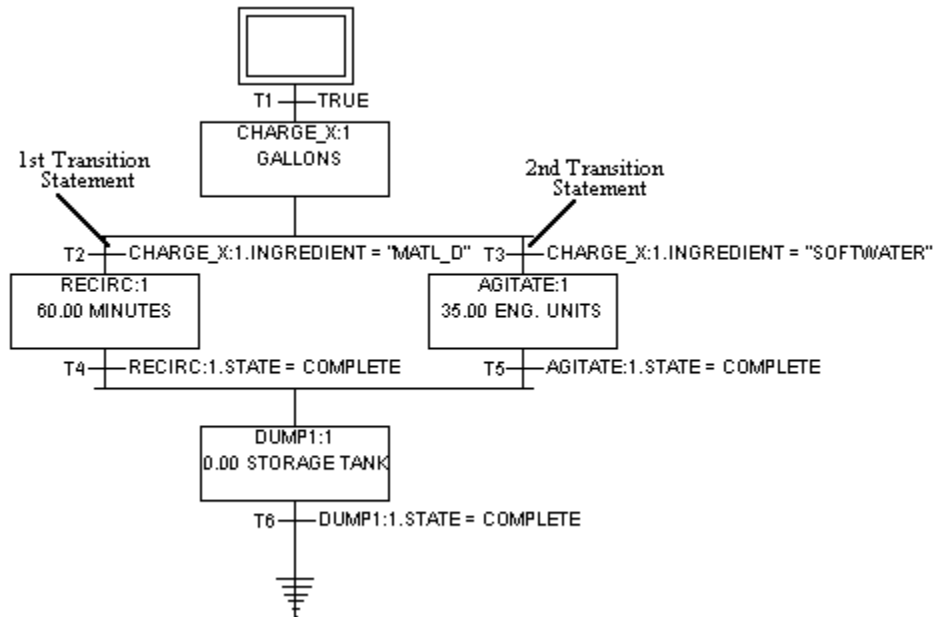
Function	Description	Behavior: Value	Behavior: Result
TRNC ()	Truncate - Retains only the integer portion of the numeric value.	6.7 6.5 6.3 -6.7 -6.5 -6.3	6 6 6 -6 -6 -6
ABS ()	Absolute - Numeric values are positive values. For example, if the Real or Integer is 6.7, there is no effect. If the Real or Integer is -6.7, it multiplied by -1 and is 6.7.	6 6.7 -6 -6.7	 6.7 6 6.7
MOD ()	Modulo - Returns the modulo, or remainder, of a division. (Integers only)	7 MOD 6	1

Linked elements

Sequential function charts must follow a step-transition-step pattern to be valid; a step links to a transition and a transition links to a step. Recipe elements link to form a simple sequence (step-transition-step), or take any of these paths:

Examples

SFC Example #1

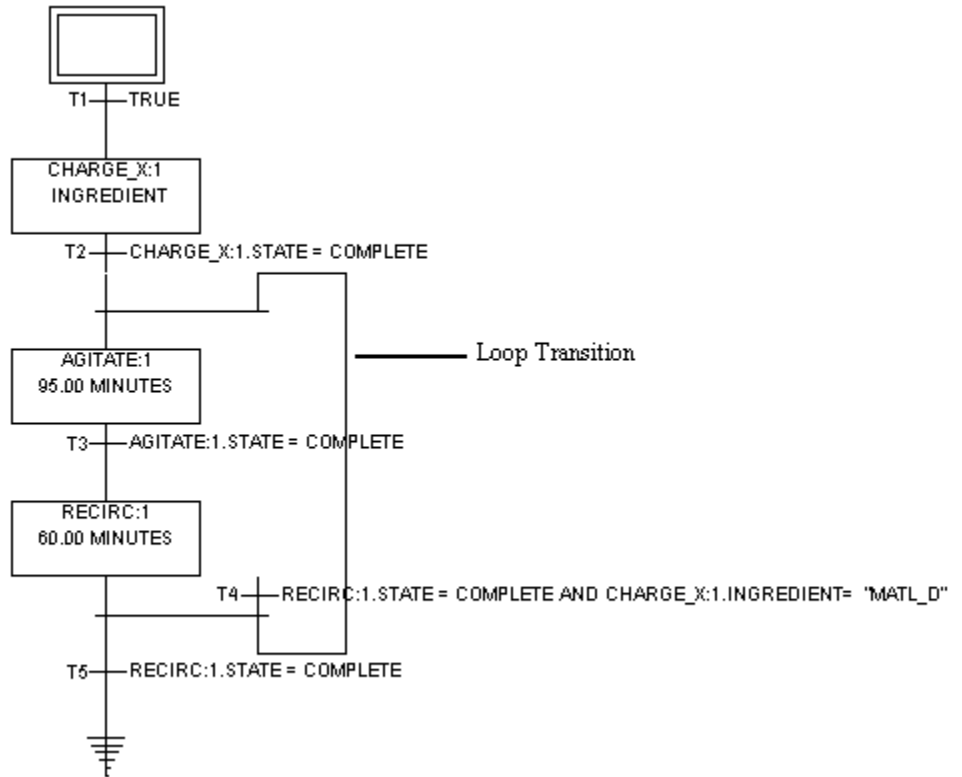


Decision logic (OR path) allows an SFC to flow in one direction or another depending on the choice made by the transition statement. A single horizontal line connects the two statements. Create the decision construct of an SFC by selecting two parallel steps and connecting them to the step above them with two separate transitions.

Parallel logic (AND path) allows more than one step to be active concurrently. A double horizontal line connects the two statements. Create the parallel construction of an SFC by selecting one transition and connecting it to two parallel steps.

The double lines of a parallel structure do not count as either a step or a transition.

SFC Example #2



Loop logic (REPEAT path) allows a process to flow from beginning to end and back again to a previous step where it repeats the loop. Create the loop construction of an SFC by joining a transition to a previous step.

The single lines of a decision branch do not count as either a step or a transition.



Tip: To create a material rebinding loop, select a material-enabled step and then select the **Create**



Material Loop tool. Loop logic allows a process to flow back to a previous step where it repeats the loop. The system automatically includes a null procedure in the material loop for rebinding.

SFC limitations

Use the FactoryTalk Batch Recipe Editor to build an SFC of any size. For very large procedures, divide into multiple procedures each with fewer steps.

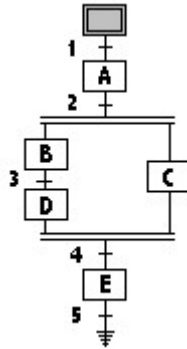
The maximum length of a transition expression is 1023 characters. If the transition expression exceeds this limit, reorganize the parallel steps in the recipe.

IMPORTANT If a batch has more than nine material-enabled, class-based phases running in parallel, the FactoryTalk Batch Server will run out of memory and will not write any messages to the Server log.

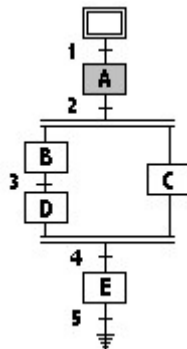
SFC execution

This is an overview of SFC execution:

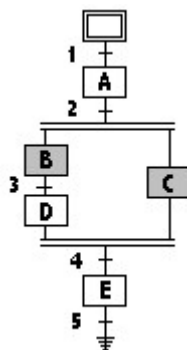
- Step 1: Upon activation of this recipe, the initial step is active (indicated by the step color).



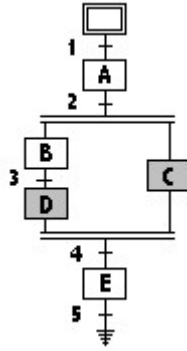
- Step 2: Control passes to Step A after Transition 1 becomes TRUE.



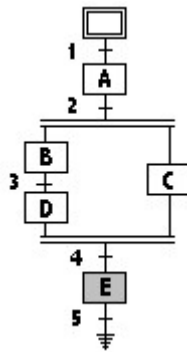
- Step 3: After Transition 2 becomes TRUE, Step A deactivates and Steps B and C become active.



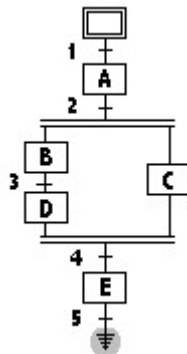
- Step 4: After Transition 3 becomes TRUE, Step B deactivates and Step D activates.



- Step 5: When Transition 4 becomes TRUE, Steps C and D deactivate and Step E activates.



- Step 6: When Transition 5 becomes TRUE, Step E deactivates and the final step activates. This indicates that the entire recipe is complete.



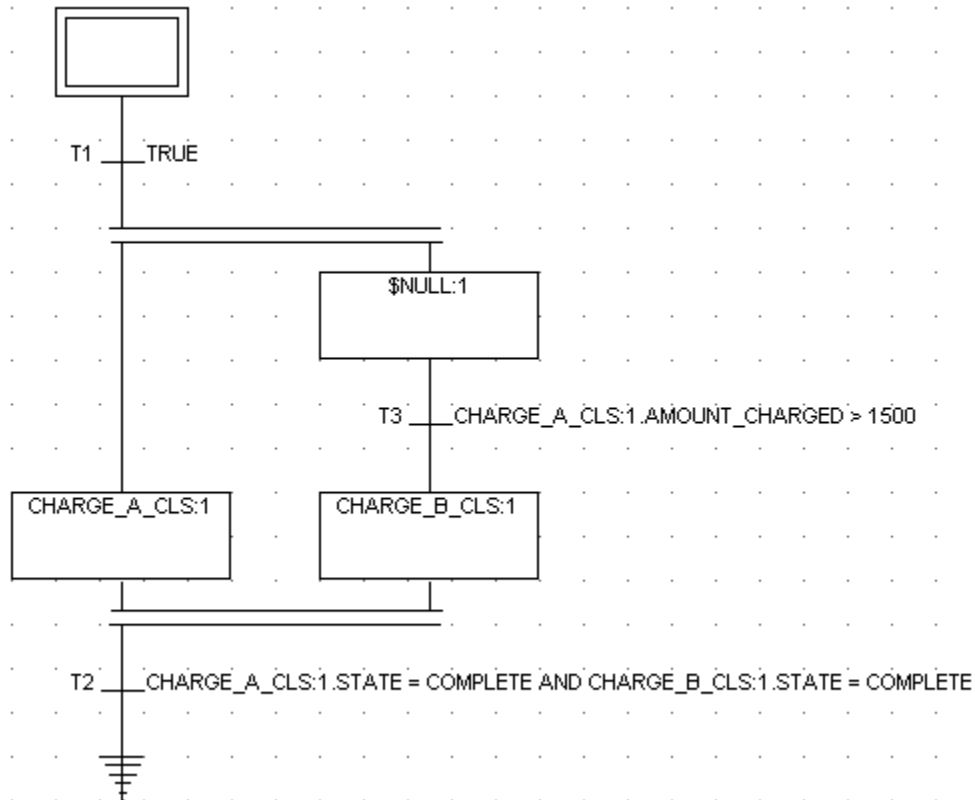
Null procedures

A null procedure contains no parameters and is not associated with a phase on the plant floor. Use null procedures to follow the step-transition-step pattern in a sequential function chart (SFC) and enable the SFC to do arbitration that is more flexible. This creates otherwise impossible SFC structures, such as separating consecutive transitions.

Add null procedures at the unit procedure, operation, and phase levels. Null procedures automatically provided in the **Unit Procedure Select**, **Operation Select**, and **Select Phase** dialog boxes; there is no need to create null phases in the FactoryTalk Batch Equipment Editor.

Synchronize execution of steps

One powerful application of null procedures is to synchronize the execution of steps. In this picture, the CHARGE_A_CLS:1 step and the CHARGE_B_CLS:1 step execute in parallel. The execution of the CHARGE_B_CLS:1 step does not occur until the CHARGE_A_CLS:1 step has already charged 1500 liters. To accomplish this timing, place a null procedure at the beginning of the parallel structure. A new transition creates before the CHARGE_B_CLS:1 step. This new transition contains a transition condition that allows the CHARGE_B_CLS:1 step to execute only after the CHARGE_A_CLS:1 step has charged 1500 liters.



See also

[Arbitration](#) on [page 31](#)

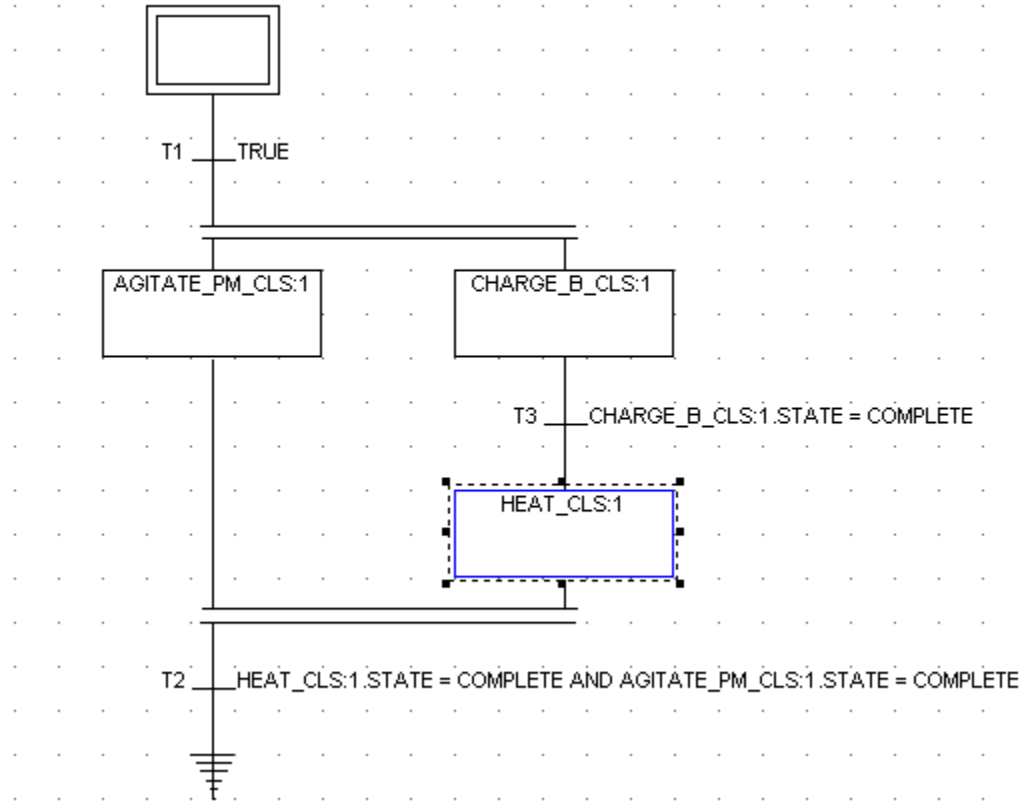
[Subarbitration](#) on [page 33](#)

[Add a null procedure](#) on [page 36](#)

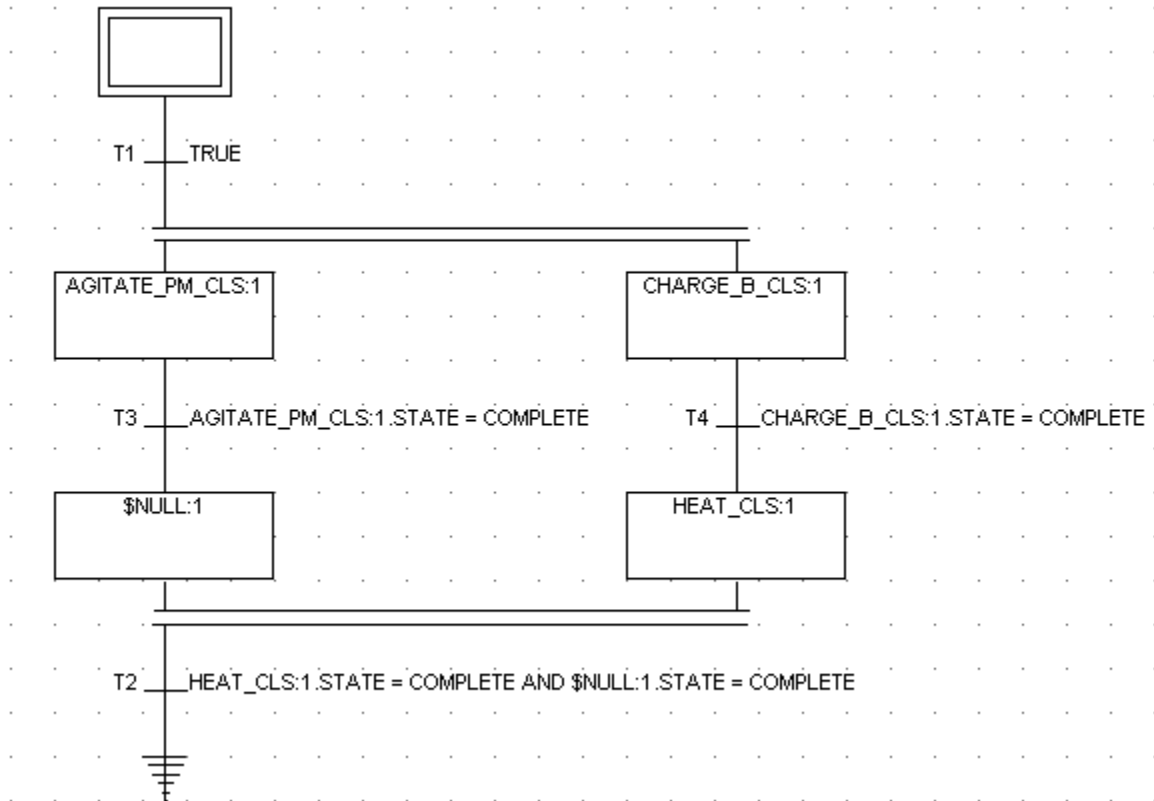
Arbitration

Use Null procedures to regulate the arbitration of resources. A step cannot release any allocated resources it owns until the transition after the step becomes true. When this step is part of a parallel structure, a delay in releasing allocated resources may occur while the step waits for the other steps within the parallel structure to complete. Similarly, if the step is a shared phase, the phase is not be released until the other parallel steps complete.

This picture illustrates the arbitration problem that can occur in a parallel SFC structure. If the AGITATE_PM_CLS:1 step is a shared phase, it is not released until CHARGE_B_CLS:1 and HEAT_CLS:1 have completed. Similarly, if AGITATE_PM_CLS:1 has acquired any shared resources, those resources are not released until all steps within the parallel structure are complete. A shared resource is a resource used in parallel by an unlimited number of steps at a time inside a recipe structure. Any other procedures waiting for resources owned by AGITATE_PM_CLS:1 are forced to wait for those resources, even though the resources may be available.



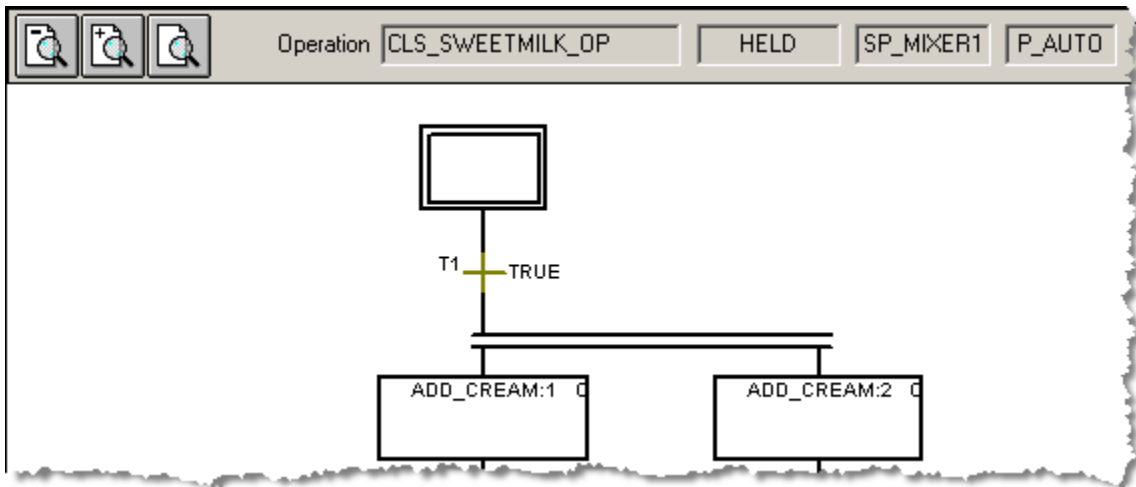
This picture illustrates the same parallel structure, using a null procedure to correct the delay. The null phase inserts below the AGITATE_PM_CLS:1 step, allowing a transition above the null phase. This transition allows the AGITATE_PM_CLS:1 step to complete, even if other phases within the parallel structure have not, thereby releasing the phase if shared, and releasing any shared resources.



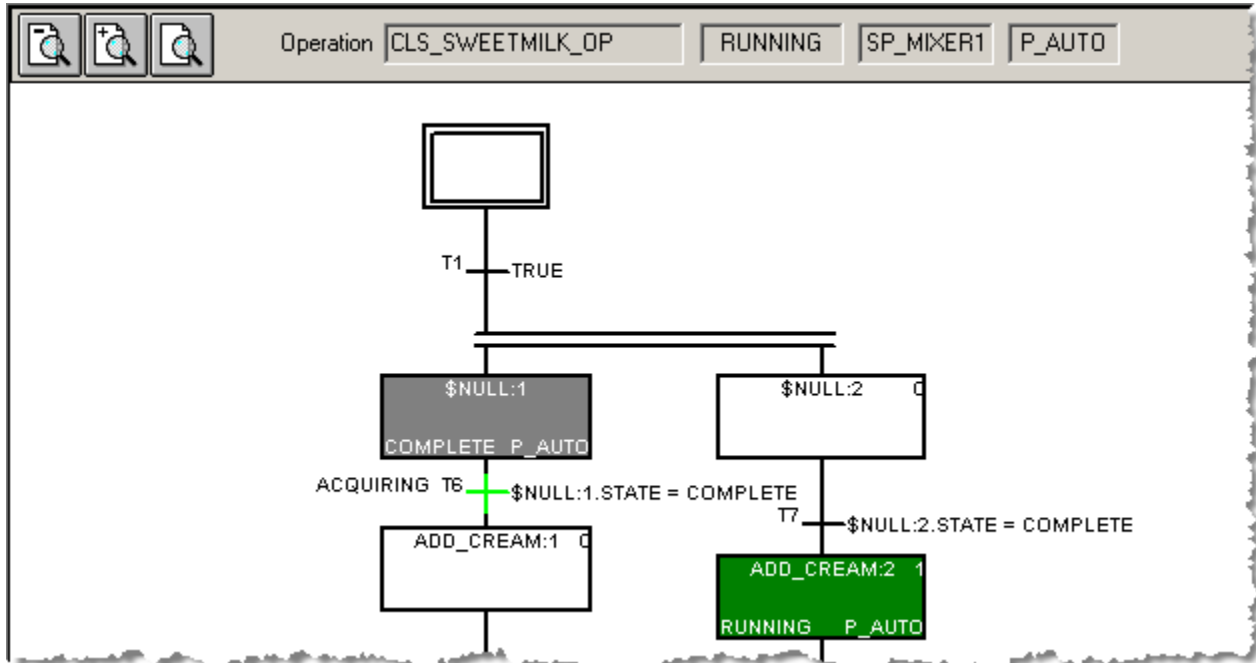
Subarbitration

Through the process of subarbitration, the FactoryTalk Batch Server determines how dedicated resources within a recipe procedure allocate when there are common resources requested at the same time. This can occur when a recipe with a parallel structure has parallel steps that require the same dedicated resource. If the parallel steps occur just after an AND Divergence or just before an AND Convergence, must add null procedures in order to run batches successfully.

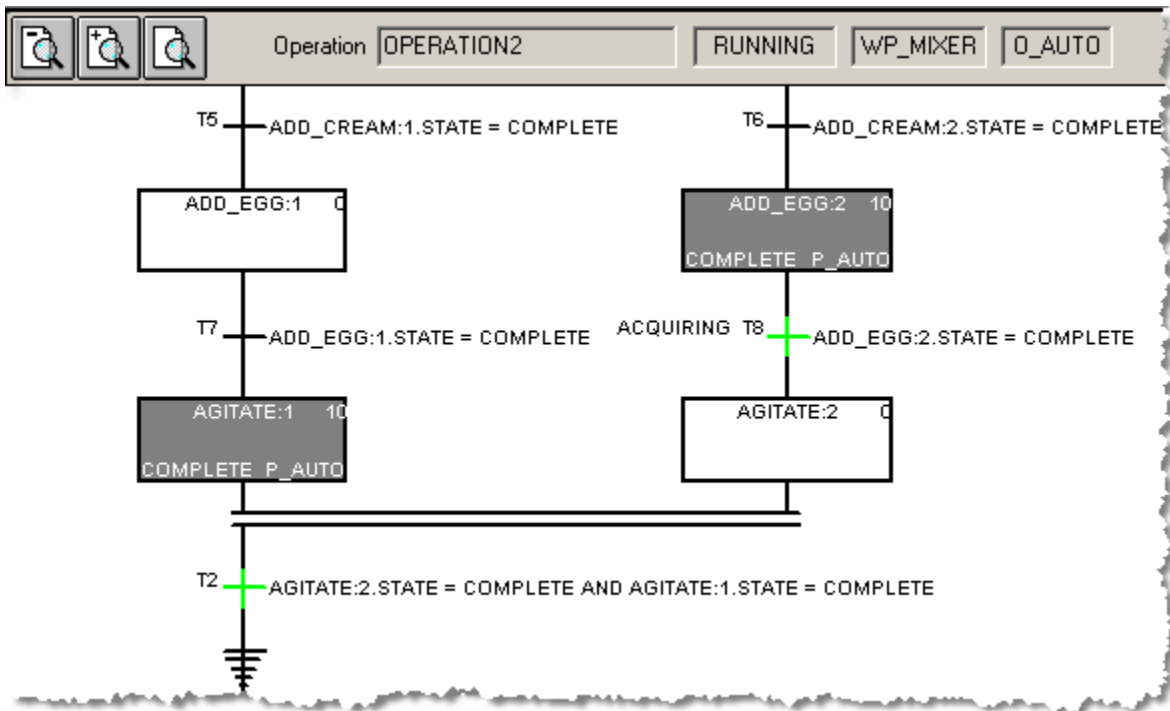
This image shows an AND Divergence of two parallel steps requesting the same resource. This structure results in the batch HELD until the conflict is resolved and the errors clear. Because both steps are requesting the same dedicated resource at exactly the same time, the resource does not release to either one of them.



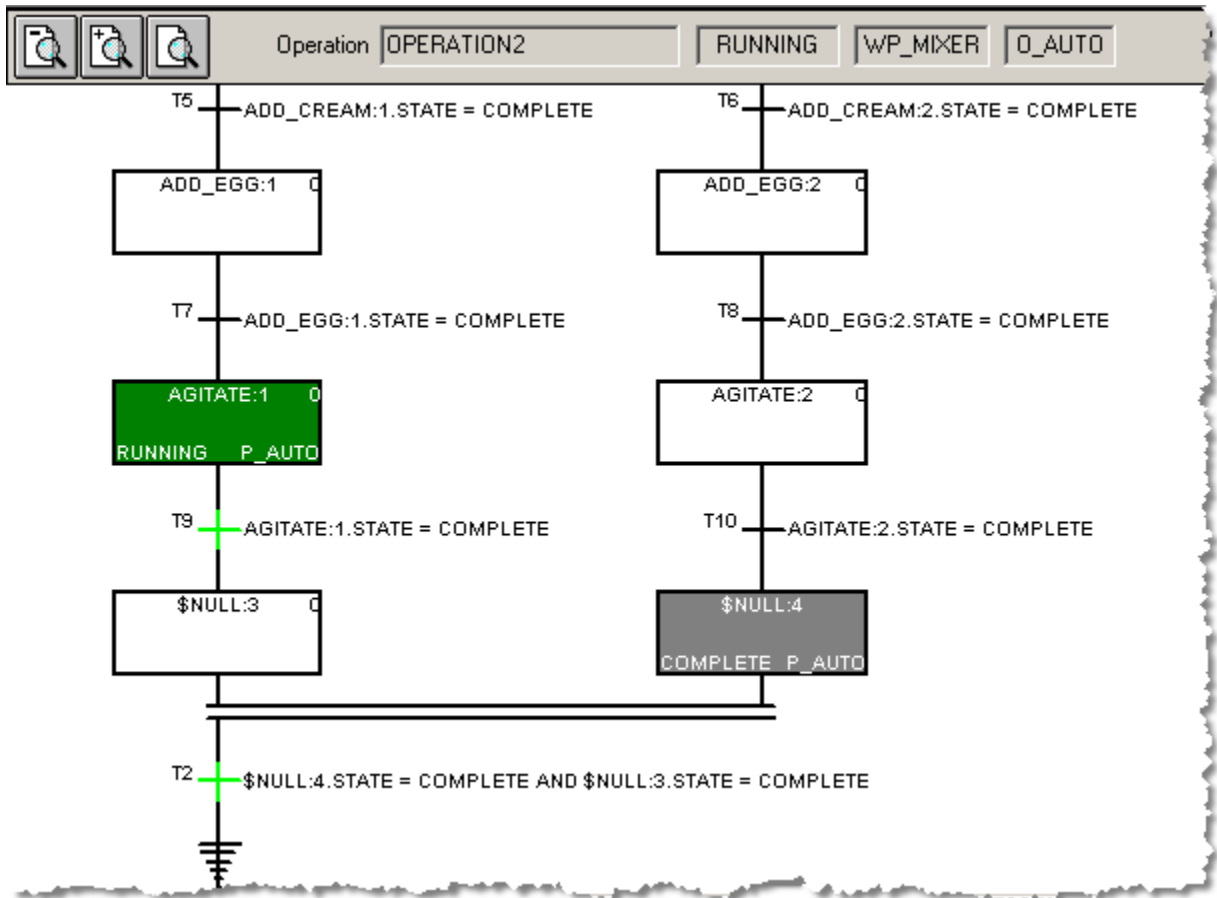
This image shows the same parallel structure with the addition of two null phases after the AND Divergence. This changes the timing just enough to allow one side to process before the other. The FactoryTalk Batch Server can then sub-arbitrate the resources required for ADD_CREAM:1 and ADD_CREAM:2.



This image shows an AND Convergence of two parallel steps requesting the same dedicated resource. In this case, the recipe appears to continue running, but the AGITATE:1 step will not release its resource. This prevents the AGITATE:2 step from acquiring the resource. The batch will not progress to the convergence transition.



This image shows the same parallel structure with the addition of two null procedures before the AND Convergence. The Server can then release the resources and transition past the convergence.



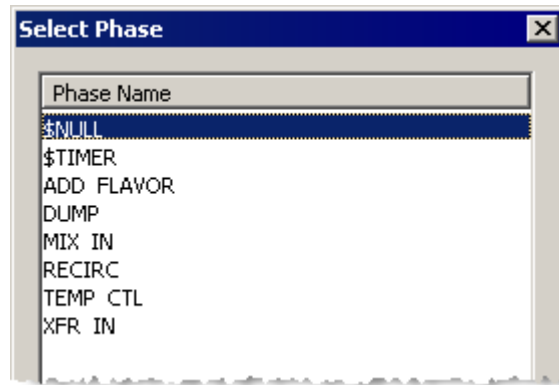
Add a null procedure

Add a null procedure instructions are similar to insert a step. The **NULL** option is always available in the **Unit Procedure Select**, **Operation Select**, and **Select Phase** dialog boxes.

To add a null procedure:

1. Using the **Selection Tool**, select a step or transition that either precedes or follows the new step.
2. Select **Insert Step** to insert a new step before the selected element, or select **Add Step** to insert a new step after the selected element. A new UNDEFINED step displays on the SFC with a blue box outlined in a black dashed line. Additionally, the SFC structure automatically rearranges to make room for the new elements. One of these dialog boxes opens, based on the recipe level: Select Phase, Operation Select, or Unit Procedure Select.

3. Select the **\$Null** option.



4. Select **OK**.

See also

[Insert steps into an SFC](#) on [page 87](#)

[Define a step](#) on [page 78](#)

Timer steps

Use timer steps to create a step that causes a branch in a recipe to pause for a specified period or to create a step that monitors the amount of time a parallel step executes. The **TIMER** option is always available in the **Unit Procedure Select**, **Operation Select**, and **Select Phase** dialog boxes.

The two **TIMER** types have these parameters and reports:

Timer Type	Parameters	Reports
COUNT_DOWN	<ul style="list-style-type: none"> • TIMER_TYPE • HOLD_BEHAVIOR (CONTINUE, RETENTIVE, and RESET) • SETPOINT 	<ul style="list-style-type: none"> • ELAPSED_TIME • REMAINING_TIME
COUNT_UP	<ul style="list-style-type: none"> • TIMER_TYPE • HOLD_BEHAVIOR (CONTINUE, RETENTIVE, and RESET) 	ELAPSED_TIME

Timer step hold behavior

When a Timer step restarts:

- If configured the Timer as **CONTINUE**, the Timer step will continue timing using the initial start time while in the **HELD** state. The Timer procedure accumulates time in the **RUNNING** and **HELD** states based on the initial start time.
- If configured the Timer as **RESET**, the Timer step is set to restart timing from 0.
- If configured the Timer as **RETENTIVE**, upon entering the **RUNNING** state, the timer will continue timing from the last known **ELAPSED_TIME** value from when the timer went into **HELD**.

Once these actions are completed, the Timer procedure transitions to the **RUNNING** state. If the Timer procedure cannot determine the type of hold

behavior configured, an error generates, causing the recipe to hold based on the configured hold propagation.

See also

[Timer steps](#) on [page 37](#)

Recipe comments

Use recipe comments to give written instructions to operators, share notes with other engineers, or access a reference. Recipe commenting associates data with a step, transition, or the entire recipe. The comment is viewable at design and at run time.

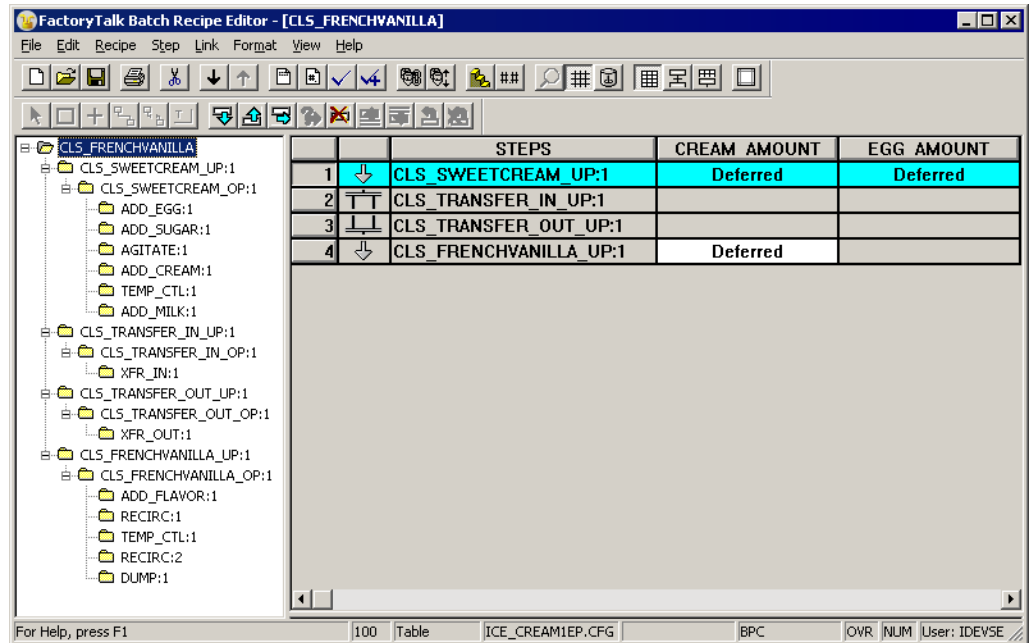
The comment, and its associated information, is a text box. Above each text box, the assigned reference name displays as a C followed by a number. View added SFC text boxes in the SFC views of FactoryTalk Batch View and FactoryTalk Batch View HMI Controls.

Table-based recipes overview

When you build a recipe in the FactoryTalk Batch Recipe Editor you can represent the logic flow of the recipe using a sequential function chart (SFC) or a table. Table-based recipes provide a mechanism for creating simple recipes that do not require a complex recipe structure or complex transition expressions. Additionally, they allow you to view and edit all recipe parameters without having to navigate between steps. The logic flow of the recipe is represented using a sequential function chart (SFC) or a table. Table-based recipes provide a mechanism for creating simple recipes that do not require a complex recipe structure or complex transition expressions. Additionally, they allow viewing and editing of all recipe parameters without having to navigate between steps.

Recipe structures created in the **Table** view provide a list-based representation of recipe logic flow. Steps execute in order, starting with the top row, and proceeding downward through the table. Each row consists of an individual phase and its associated parameters. When selecting a phase row, the column headings show the parameter names for that phase. When selecting a different phase, the column headings change to reflect the selected phase parameters.

Steps are executed starting with row one, and proceeding downward until the end of the recipe. Steps located within a single parallel structure execute at the same time. Every step within the **Table** view displays the name of the Procedure, Unit Procedure, Operation, or Phase represented by the step. All parameters associated with the selected step display as the column header. The parameter value displays below the parameter name and to the right of the step name. To edit parameter values, select the table cell where the parameter displays and enter a new value (if enumerations are created for the selected parameter, a list containing the enumerations is provided).



Recipe structure icons

The **Table** view provides a graphical representation of a recipe structure in the form of a structure icon. The **Table** view is only capable of displaying the graphical recipe structure for simple recipes. Simple recipes are recipes that do not have complex recipe structures such as loops and branches. Simple recipes only have simple parallel structures. No structure icon displays for complex recipe structures viewed in the **Table** view.

Every step within the **Table** view displays the name of the Procedure, Unit Procedure, Operation, or Phase represented by the step.

The Table view can contain any of these recipe structure icons:

Graphic	Name	Description
	Simple Step	Represents a single step within the recipe path.
	Begin Parallel Step	Represents the first step in a parallel structure containing multiple steps.
	Parallel Step	Represents the step located within a parallel structure containing multiple steps.
	End Parallel Step	Represents the final step in a parallel structure containing multiple steps.

Table view transition expressions

Each recipe step has an associated transition. Transitions designate criteria that must be true before the recipe continues to the next logical step. The transition expression must always evaluate to either TRUE or FALSE. Transitions generate automatically for all table-based recipe steps when

created. By default, all table-based recipe transitions contain the transition expression **<STEP>.STATE = COMPLETE**, where **<STEP>** is the step associated with the transition.



Tip: Transition expressions cannot be edited using the **Table** view. To edit a transition expression, use the **Transition Expression Builder** dialog box in the **SFC** view.

Parallel structures

The FactoryTalk Batch Recipe Editor allows creation of simple parallel structures using the **Table** view. These structures can have multiple steps in parallel, but cannot have more than one step located on the same path. That is, all steps located within the parallel structure must be preceded by the same AND divergence and followed by the same AND convergence.

More complex parallel structures created using the **SFC** view are considered undetermined when viewed in the **Table** view. Recipe structure icons do not display for the entire recipe level in which the parallel structure is located.

Complex parallelism example

This table contains an example of a complex parallel structure created in the **SFC** view and how that same structure displays in the **Table** view. Notice that there are no recipe structure icons displayed for the entire recipe level in the **Table** view.

SFC View:

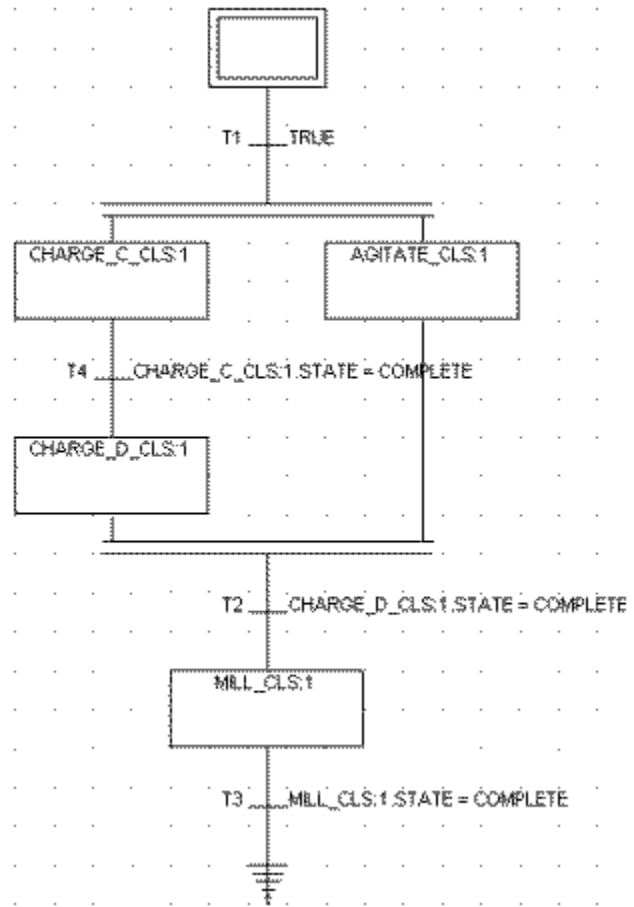


Table View:

STEPS	
1	CHARGE_C_CLS:1
2	AGITATE_CLS:1
3	CHARGE_D_CLS:2
4	MILL_CLS:1

Simple parallelism example

This table contains an example of a simple parallel structure displayed in the **SFC** view and in the **Table** view.

SFC View:

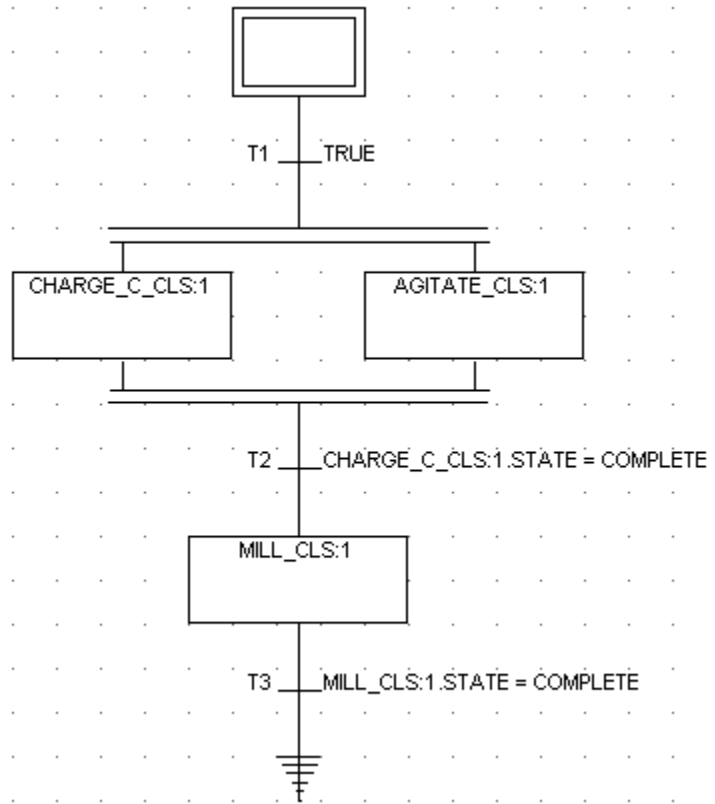


Table View:

STEPS		
1		CHARGE_C_CLS:1
2		AGITATE_CLS:1
3		MILL_CLS:1



Tip: If parallel steps require the same dedicated resource, the FactoryTalk Batch Server automatically determines how the resources allocate among steps when the batch runs (called subarbitration). If parallel steps occur just after an AND divergence or just before an AND convergence, add null procedures to the recipe in order for batches to run successfully.

Recipe creation

Before building an SFC or Table structure to define the logical flow of a recipe, first create the recipe in FactoryTalk. The recipe creation process includes:

- Determine the recipe level
- Specify unit requirements
- Define recipe formula parameters
- Enter recipe header data

See also

[Recipe procedure levels](#) on [page 70](#)

[Recipe unit requirements](#) on [page 69](#)

[Dynamic unit allocation](#) on [page 46](#)

[Open recipe](#) on [page 50](#)

[Create operations and unit procedures](#) on [page 52](#)

Unit-based recipes

A unit-based recipe includes the specific units used for the recipe. Recipe creation requires a unit-based recipe. Unit-based recipes use static binding and bind to a specific unit instance defined in the area model.

Unit-based recipes also include the specific unit instances used for the recipe. Define unit instances in the area model. During recipe creation, the recipe author specifies the required unit instance. As a result, unit-based recipes use a binding method called static binding. Static binding means that the recipe runs in only one unit instance. At batch creation time, unit-based recipes automatically bind to the specified unit instance.

Unit class-based recipes

Create class-based recipes for a unit class rather than for a specific unit instance. Class-based recipes are useful if a unit class contains many unit instances and a particular recipe runs in most of those unit instances. The use of class-based recipes reduces the number of recipes created and maintained. Create only one recipe per unit class rather than one recipe for each unit instance. At batch creation time, the operator assigns the specific unit instance in which the class-based recipe runs, or FactoryTalk Batch Server availability determines assignment of the unit.

When creating a recipe, the phases that are not common to all the unit instances built from a particular unit class are available for selection. When adding a non-common phase to the recipe, the FactoryTalk Batch Recipe

Editor automatically adds a Require Phase bind requirement to the unit requirements so that only the units having the phase are selectable at run time.

Enabling Dynamic Unit Allocation on a procedure level class-based recipe allows either the FactoryTalk Batch Server or the operator to select which unit instance of the unit class to use for each class-based unit procedure contained in the procedure. Select the unit instance either at batch creation or after the batch has started, depending on the selected unit allocation method.

Material-based recipes

Material-based recipes contain steps that specify the materials needed to run the recipe without requiring a definition of the equipment that is required to supply those materials. Each phase in the FactoryTalk Batch Equipment Editor contains configurations for material additions and distributions. Material-based recipes, unit classes, and the FactoryTalk Batch Material Editor drastically reduced the number of different recipes needed.

Material class-based recipes

Material class-based recipes contain steps configured to utilize any of several similar materials from a common class of materials. Configure material classes and specific materials in FactoryTalk Batch Material Editor.

A material-enabled phase is a phase configured by FactoryTalk Batch Equipment Editor and stored in the area model, which is enabled to support the specification of a material as a means to find appropriate equipment and bind to that equipment in a control recipe.

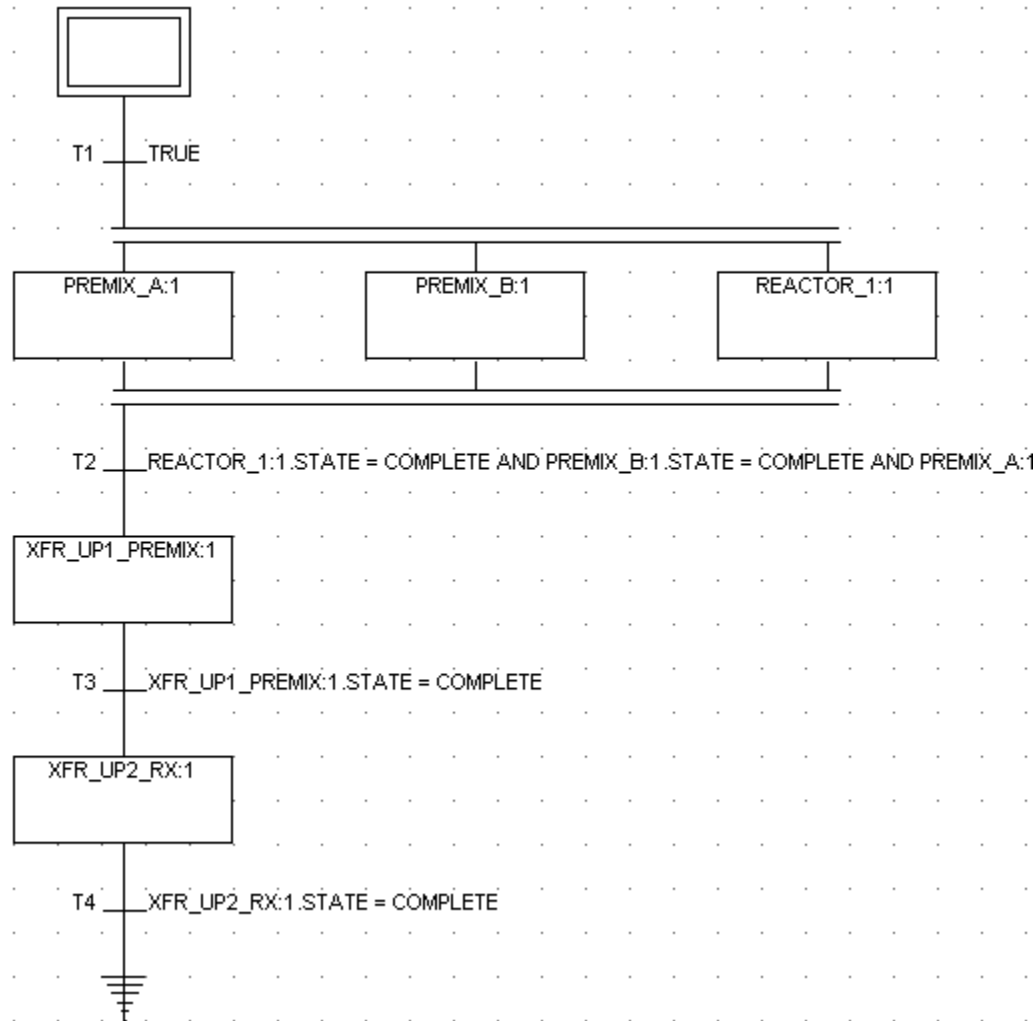
Configure material-enabled phases to be material class-based in FactoryTalk Batch Equipment Editor. In FactoryTalk Batch Recipe Editor, insert these material class-based phases into a recipe. When adding the recipe to the batch list in FactoryTalk Batch View, the operator selects a specific material from the class of materials. Using material class-based recipes reduces the number of recipes needed. In FactoryTalk Batch Recipe Editor, create one recipe and change the configuration of that recipe to be material class-based or material-specific.

Dynamic Unit Allocation

Dynamic Unit Allocation is a method for binding specific units to class-based unit procedure-level recipes that are contained within a procedure-level recipe. Dynamic Unit Allocation allows different unit procedures with the same unit requirements within the procedure. Define any downstream units required by unit instances and unit classes.

Downstream unit requirements may be defined for class-based and instance-based unit procedures. Downstream unit requirements are used to control batch flow and to make sure that the appropriate unit is selected during unit-binding.

This is an example procedure-level PREMIX recipe:



Unit requirement name

A unit requirement name is a user-defined name assigned to a unit class with a defined binding method. When creating the procedure-level recipe, the recipe author assigns the appropriate unit requirement name to each unit procedure step in the recipe. Assigning a unit requirement name solves potential complications when using Dynamic Unit Allocation.

In the example PREMIX Procedure-level recipe, assume that both of these class-based unit procedures have the PREMIX_A_CLS unit class assigned:

- PREMIX_A
- XFR_UP1_PREMIX

Additionally, there are five unit instances of the PREMIX_A_CLS unit class and unit procedures, PREMIX_A and XFR_UP1_PREMIX that can bind to any of the five unit instances. But they must bind to the same unit instance.

If the selected binding method is **First Available**, define a unit requirement name for both unit procedures. This is so the FactoryTalk Batch Server can bind both unit procedures to the same unit instance.

For example, in the sample recipe, the recipe author must assign the same unit requirement name to both unit procedures PREMIX_A and XFR_UP1_PREMIX. This same unit requirement identifies to the Batch server that whatever unit instance it binds to unit procedure PREMIX_A, it must also bind to unit procedure XFR_UP1_PREMIX.

If implementing Unit Attribute Binding, the recipe author can further define the binding requirements and preferences. The author can configure binding to units meeting specific criteria, for example, warmest available unit, capacity restrictions, or non-agitator units.

A procedure-level recipe can contain a mix of class-based unit procedures and unit-specific unit procedures. If Dynamic Unit Allocation is enabled, all unit procedures use a unit requirement name, even those that are unit-specific. Therefore, assign a unit requirement name to a unit instance (rather than to a unit class) and define the binding method as Static. Static binding means that any unit procedure steps must always be bound to the specific unit identified for this unit requirement name.

Downstream unit requirements

A downstream unit is a unit that follows another specified unit during batch execution. Downstream unit requirements mirror the linking of units represented in the area model. Specifying required downstream units allows control of batch flow and to make sure to select the appropriate unit during unit-binding. Configure the area model and recipes as follows to enforce these flow paths:

- Link the units in the area model using the FactoryTalk Batch Equipment Editor.
- Define downstream unit requirements in the recipe using the FactoryTalk Batch Recipe Editor.

When these configurations are in place, the FactoryTalk Batch Server enforces these flow paths when binding units to a class-based unit procedure or when displaying a unit selection list to the operator for unit binding.

Enable Dynamic Unit Allocation

Dynamic Unit Allocation is enabled in FactoryTalk Batch Recipe Editor.

To enable Dynamic Unit Allocation

1. From the **View** menu, select **Options** to display the **Options** dialog box.
2. Select **Support Dynamic Unit Allocation**.
3. Select **Apply**.



Tip: The allocation setting applies to ALL procedure-level recipes created, edited, or saved.

Dynamic Unit Allocation affects binding

If Dynamic Unit Allocation is not enabled, you must allocate a unit to every unit procedure contained within the procedure-level recipe during batch creation (a batch creates when it is on the batch list in the FactoryTalk Batch View or custom HMI application). This presents complications if this procedure-level recipe takes 30 days to complete from the moment the first unit procedure is initiated until the last unit procedure runs to completion. If some or all of the unit procedures contained within the procedure are class-based, it may be difficult to predict, at batch creation time, which units are available when needed.

Dynamic Unit Allocation solves this problem because it allows the bound unit to be defined when the unit procedure is ready to run, rather than when the batch is created. Late binding postpones the unit definition until the unit procedure requires it to run.

With late binding, a step is bound to equipment just before use.

- Unit procedure steps (also called dynamic unit allocation) support two types of late unit binding: First Available and Prompt binding.
- Material phase steps support two types of late phase binding: Automatic and Prompt binding.

Binding methods

These four binding methods are options for class-based unit procedures:

- **At Batch Creation** - The operator defines the units at batch creation. This is similar to what occurs with disabled Dynamic Unit Allocation.
- **First Available** - The FactoryTalk Batch Server assigns the unit for binding when the unit procedure is ready to run (FactoryTalk Batch Server-defined late binding).



Tip: Once the unit binds using the First Available binding method, the FactoryTalk Batch Server sorts through Equipment ID numbers in ascending order and selects the Equipment ID with the smallest ID number.

- **Prompt** - The operator responds to a prompt to assign the unit for binding when the unit procedure is ready to run (operator-defined late binding).
- **Operator Choice** - When the batch creates, the operator can choose the unit binding methods: At Batch Creation, First Available, or Prompt.

Criteria for unit selection

When automatically selecting a unit for binding, the FactoryTalk Batch Server tries to use the unit that the recipe can acquire first. The unit selected must meet these criteria:


- The acquired unit must belong to the unit class of the unit procedure step.
- Recipes can configure upstream, downstream, or both upstream and downstream dependencies that define a series of unit classes that a recipe requires as a recipe executes. The acquired unit supports the flow path to other units.

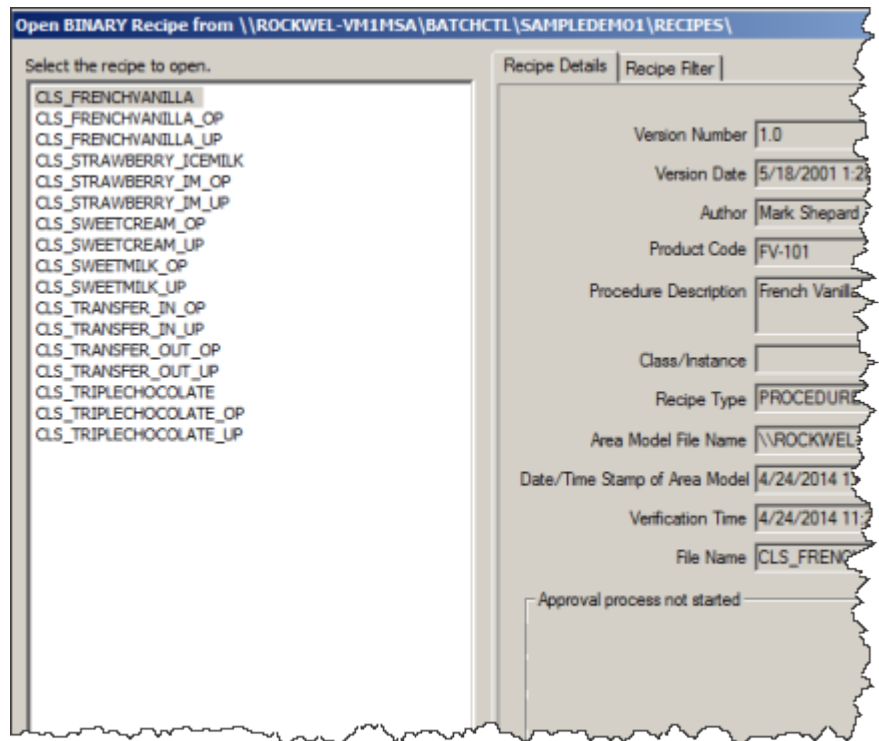
- Recipes containing material phase steps impose additional requirements when selecting a unit for binding:
 - The acquired unit must be capable of fulfilling the material requirements of all material phase steps in the recipe to be run in that unit.
 - Support any recipe to be run upstream or downstream flow path.

Open recipe

Open existing recipes for viewing and editing from within the FactoryTalk Batch Recipe Editor. Open entire recipe procedures, as well as individual operations or unit procedures.

To open a recipe:

1. Select **Open**  to display the **Open [Type] Recipe** dialog box, where [Type] is the recipe storage type (binary, XML, or RDB, configured in the FactoryTalk Batch Equipment Editor **Server Options** dialog box).
2. From the **Recipe Name** list, double-click a recipe to open it, or select the appropriate recipe and select **Open**.



Recipe header information for a selected recipe displays in the **Recipe Details** tab.

In the **Approvals Process** area, examine the approval process state for each selected recipe to see which recipes may require approval attention and action.

In the **Recipe Filter** tab, set filters as desired to narrow the number of recipe names. Multiple filters have an additive effect in narrowing the list shown. For example, effectively search for recipes that require

particular approval step signoffs by setting the **Step Name** and the **Step State** filters.

The screenshot shows a 'Recipe Filter' dialog box with three main sections:

- Recipe Properties:**
 - Recipe Type: All Procedures
 - Equipment Type: ALL
 - Class/Instance: (empty)
- Approval Process Data:**
 - Approval Process Type: Expedited Approval
 - Step Name: Release Recipe to Production
 - Step State: Not Started
- Versioning Data:**
 - Version Type: Versioned & Unversioned
 - Base Name: ALL

A 'Reset' button is located at the bottom right of the dialog.

Item	Available Selections
Recipe Type	All Procedures, Operation, Unit Procedure, or Procedure.
Equipment Type	Select ALL to view recipes regardless of equipment associations in the area model. To limit the displayed recipes to those that use particular equipment, select Unit Class or Unit . Then select from the corresponding classes or instances in the Class/Instance drop-down list.
Class/Instance	Unit classes defined in the area model display when Equipment Type is Unit Class . Similarly, unit instances display when Equipment Type is Unit .
Approval Process Type	ALL, Primary Approval Process, or Expedited Approval Process. If Recipe Approval Process is disabled, this list along with Step Name and Step State are unavailable.
Step Name	ALL, any defined step in the Primary approval process, Release Recipe as Step, and Release Recipe to Production (the latter two are the only steps used for Expedited Approval).

Item	Available Selections
Step State	ALL, Not Started, In Progress, Complete, Reverting, and \$\$System Signoff Pending (for translated recipes).
Version Type	Versioned & Unversioned, ALL, and Obsolete.
Basename	ALL, or any basename (name of the recipe element without the ~V# or _WIP extension). Obsolete recipes display in the list, but the obsolete status is not visible from the recipe name.

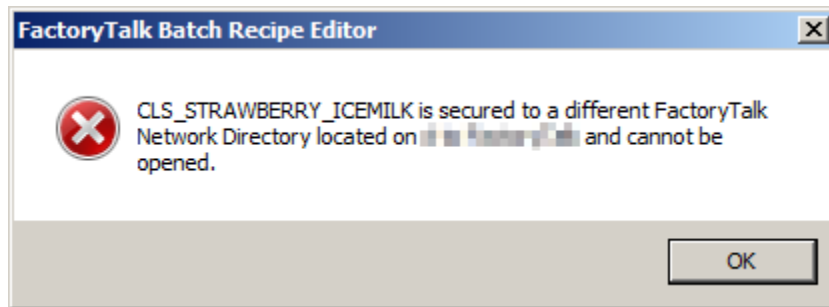
See also

[Recipe header data](#) on [page 64](#)

[Cannot open recipe error message](#) on [page 52](#)

Cannot open recipe error message

If Security Authority is enabled for a recipe, and there is a mismatch between the Security Authority Identifier (SAI) it contains and the SAI in the current FactoryTalk Network Directory, this error message opens:



Make note of the FactoryTalk Network Directory host computer information, then select **OK** to close the error message.

To recover the recipe, saved either as a backup of the FactoryTalk Network Directory SAI or of the recipe in unsecured form:

- Restore the FactoryTalk Network Directory SAI which secures the area model. The name of the computer that hosts that SAI provides in the dialog box. Use the FactoryTalk Administration Console to restore a saved backup of the SAI. Then open the secured recipe.
- Open an unsecured copy of the recipe if saved a copy of the recipe prior to applying Security Authority.

Create operations and unit procedures

Define unit requirements for unit procedures and operations when creating the recipe. Only indicate the specific unit instance or unit class used for this particular recipe.

To create operations and unit procedures:

1. Select **New** . The **New** dialog box opens.

2. Select **Unit Procedure** or **Operation**, and select **OK**. The **Create Unit Procedure** or **Create Operation** dialog box opens.
3. Enter recipe information and select **OK**. **Procedure Identifier** is the only required box. The **Unit Requirement** dialog box opens.

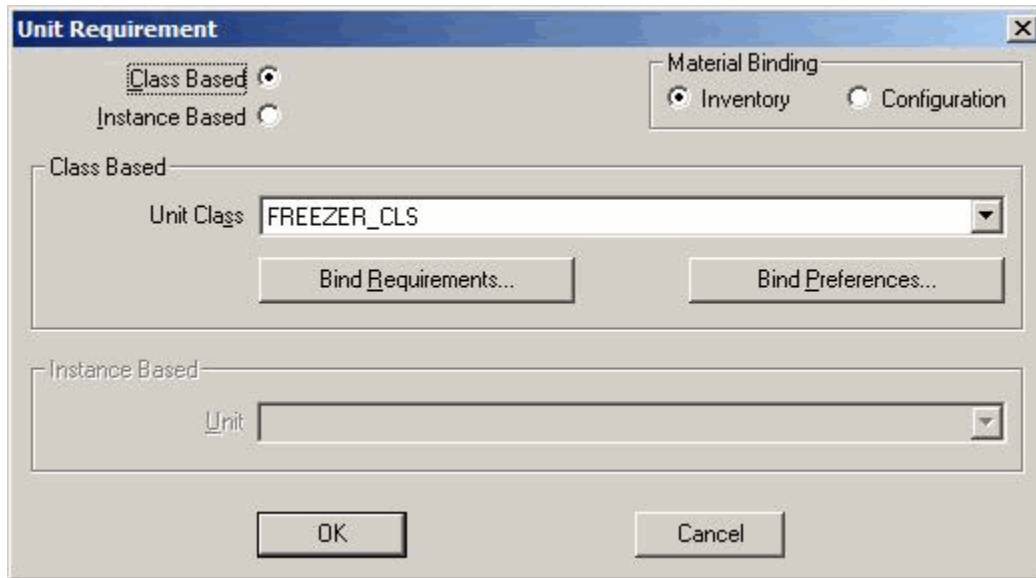
Item	Description
Procedure Identifier	The unique name that displays when working with a recipe in the FactoryTalk Batch View. It is also used as the file name when the recipe is saved. Follow methodologies for assigning product identification codes. The Procedure Identifier can contain letters (A-Z), numbers (0-9), and underscores (_). For Procedures, the Procedure Identifier can begin with either a letter or a number. For Unit Procedures and Operations, it must begin with a letter.
Version Number	Legacy, user-set version number (not related to the Recipe Versioning feature introduced in FactoryTalk Batch version 12). Manually update this version number as desired.
Version Data	Defaults to today's date, cannot be changed.
Author	The name of the individual that created the recipe.
Product Code	A short label that uniquely identifies the product.
Procedure Description	Product or process description at the procedure level. Indicate whether this is a class-based recipe, or any other distinguishing information required.
Procedure Abstract	A summary or outline of the steps of the procedure.
Version Description	Legacy, version description (not related to the Recipe Versioning feature introduced in FactoryTalk Batch version 12).

4. Select either the **Class Based** or **Instance Based** unit requirement. The **Class Based** group box shows the unit classes defined in the area model as class-based. The **Instance Based** group box shows the unit instances defined in the area model as unit-based.

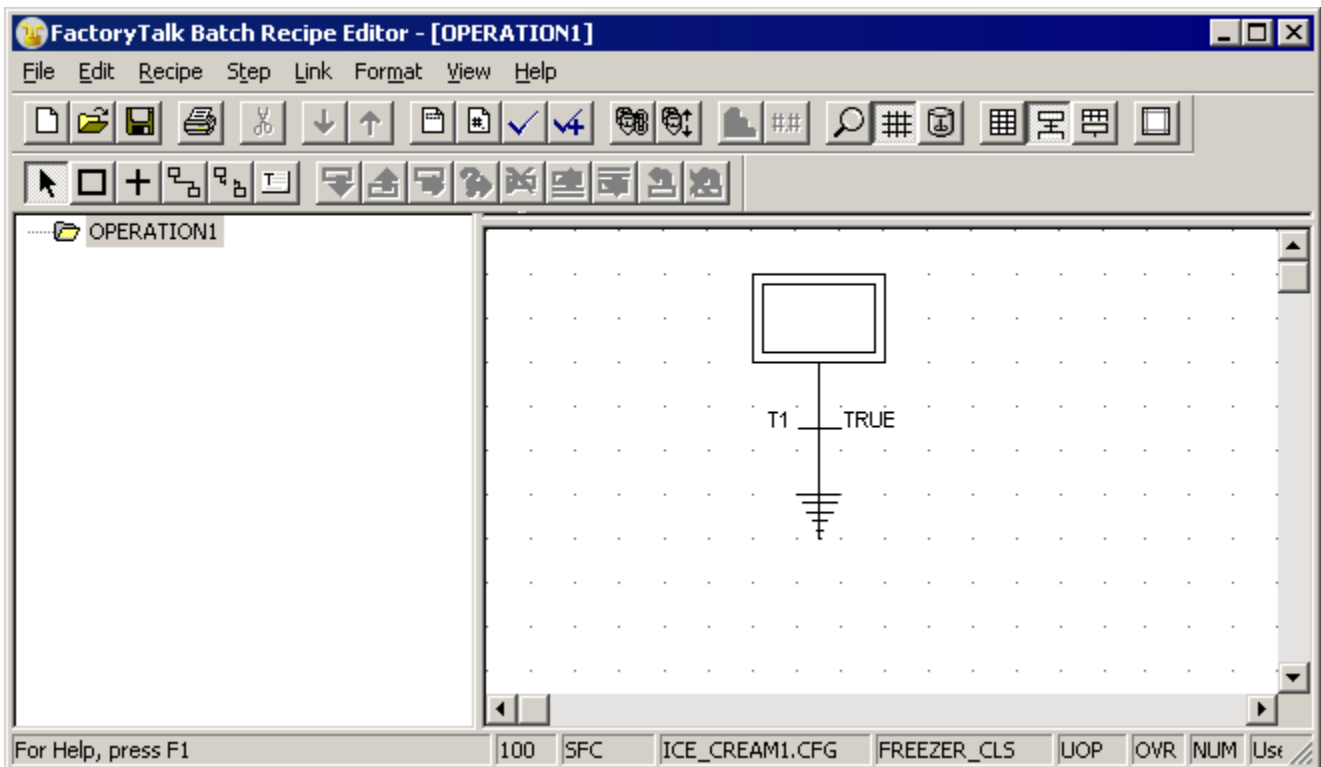
If supporting Dynamic Unit Allocation, **Bind Requirements** and **Bind Preferences** are enabled.

5. When binding materials, select either **Inventory** or **Configuration**. The Factory Talk FactoryTalk Batch Server and Material Server determines the appropriate set of binding candidates at runtime.
 - **By Inventory:** Containers assigned to a unit must have material in them.
 - **By Configuration:** Containers assigned to a unit can have zero quantity of material in them.

6. Choose either a **Unit Class** or **Unit** from the appropriate list.



7. Select **OK** to return to the **Procedure View** pane, which shows the default procedure name (**UNIT_PROCEDURE#** or **OPERATION#**) on the title bar. The **SFC** view contains the initial and final steps of the sequential function chart, and the **Table** view contains a blank table header.



See also

[Binding requirements](#) on [page 60](#)

[Insert steps into an SFC](#) on [page 87](#)

[Change operation/unit procedure unit requirements](#) on [page 55](#)

Edit unit procedure requirements

Edit the unit requirements within a unit procedure or operation at any time.

To change operation/unit procedure unit requirements:

1. Select the appropriate recipe level. Double-click the unit procedure or operation to edit. Make sure it displays in the **Recipe Construction** pane.
2. From the **Recipe** menu, select **Unit Requirements**. If not using the unit procedure or operation in a higher level recipe, the appropriate **Unit Requirement** dialog box opens.
3. Select **Class Based** or **Instance Based** and then determine the unit class or unit instance from the appropriate list. If the unit procedure or operation has been used in a higher level recipe, a warning message displays indicating what higher level recipes needs to be modified the unit requirements are changed. Select **Proceed** to update affected recipes.
4. Select **OK** to return to the **Procedure View** pane.




Tip: If modifying the unit requirements for a unit procedure or operation used in a higher level recipe, then make the required adjustments to the unit requirements of higher level recipes. If not making corrections, the **Released to Production** check boxes clears for higher level recipes.

Create a procedure

When creating a procedure, indicate all unit instances and unit classes used within the procedure. If the **Support Dynamic Unit Allocation** option is enabled in the **View > Options** dialog box, each unit instance or unit class must have a unit requirement name assigned when used in a procedure.

To create a procedure:

1. Select **New** . The **New** dialog box opens.
2. Select **Procedure** and select **OK**. The **Create Procedure** dialog box opens.
3. Enter recipe information and select **OK**. **Procedure Identifier** is the only required box.
 - **Procedure Identifier**. The unique name that displays when working with a recipe in the FactoryTalk Batch View. It is also used as the file name when the recipe is saved. Follow methodologies for assigning product identification codes. The Procedure Identifier can contain letters (A-Z), numbers (0-9), and underscores (_). For Procedures, the Procedure Identifier can begin with either a letter or a number. For Unit Procedures and Operations, it must begin with a letter.

- Version Number. Legacy, user-set version number (not related to the Recipe Versioning feature introduced in FactoryTalk Batch version 12). Manually update this version number as desired.
- Version Date. Defaults to today's date. Cannot be changed.
- Author. The name of the individual that created the recipe.
- Product Code. A short label that uniquely identifies the product.
- Procedure Description. Product or process description at the procedure level. Indicate whether this is a class-based recipe, or any other distinguishing information required.
- Procedure Abstract. A summary or outline of the steps of the procedure.
- Version Description. Legacy, version description (not related to the Recipe Versioning feature introduced in FactoryTalk Batch version 12).



Tip: If **Support Dynamic Unit Allocation** is enabled, the **Procedure-Unit Requirements** dialog box opens.

Procedure - Unit Requirements

	Unit Requirement Name	Class/Instance	Class	Binding Method	Downstre
1	Alias_MIXER_CLS	MIXER_CLS	<input checked="" type="checkbox"/>	Batch Creation	

Add Unit Requirement... Edit Unit Requirement... Delete Unit Requirement

- Select **Add Unit Requirement** to display the **Add Unit Requirement** dialog box.

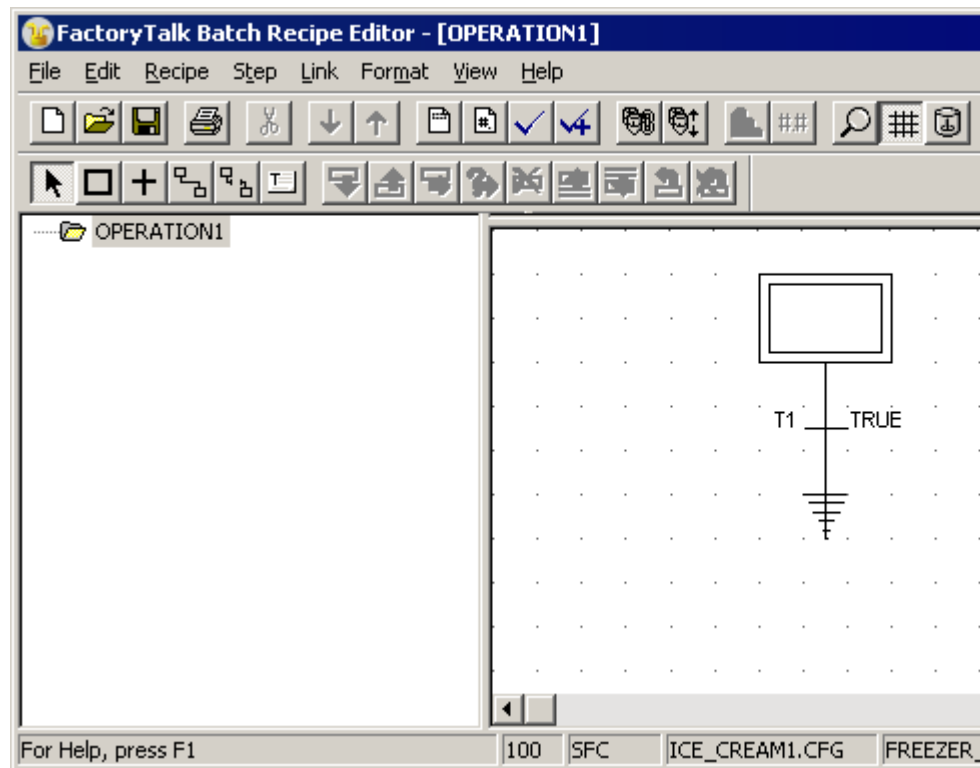
- Type a unique name in the **Name** box. The name can contain alphanumeric characters (A through Z, 0 through 9), an underscore (_), or a colon (:), but it **must** begin with an alphanumeric character.
Any unit procedure mapped to this unit requirement name uses the same unit to which this unit requirement name is bound. Bind the unit requirement name to the unit class or to a unit instance.
- Select **Class Based** or **Instance Based** to enable the corresponding group box.
- Select the required unit class or unit from the appropriate list.
- If adding Class Based requirements, select the appropriate binding method from the **Binding Method** list.
- If needed, add applicable **Bind Requirements** and **Bind Preferences**.
- In the **Available Downstream Unit Requirements** list, select the unit requirement name for the unit that is downstream from the unit requirement being defined and select the right arrow. The unit requirement is added to the **Selected Downstream Unit Requirements** list.



Tip: Specify **Downstream Unit Requirements** only after all additional unit requirements are specified. The list box contains the unit requirement names for all available units defined in the **Procedure-Unit Requirements** dialog box (except the one currently being defined).

To return all selected downstream units to the available downstream units, select **Remove All**.

11. Select **OK** to return to the **Procedure-Unit Requirements** dialog box.
12. Repeat steps 4 through 10 to define additional unit requirements for this procedure.
13. Select **Close** in the **Procedure-Unit Requirements** dialog box to return to the **Procedure View** pane, which now shows the default procedure name on the FactoryTalk Batch Recipe Editor title bar. The **SFC** view contains the initial and the final steps of the sequential function chart, and the **Table** view contains a blank table header.



Unit requirement name example

The unit requirement name is a label and acts as a lookup table for the FactoryTalk Batch Server. When a recipe is on the Batch List and the server encounters the unit requirement name, it looks up all the unit classes (or unit instances) mapped to the unit requirement name.

For example, a plant has two mixers that used to run one recipe. When the recipe runs, the operator or the FactoryTalk Batch Server (depending on the binding method) must pick which mixer to use. In the area model, these mixers are each assigned a unit instance (MIXER_1 and MIXER_2) from the same unit class (MIXER). In the recipe, create a unit requirement name (MIXERS) configured to the MIXER unit class, with the binding method at batch creation. When the recipe is on the Batch List, the FactoryTalk Batch Server encounters the label MIXERS, which tells it that piece of equipment is required for this recipe. The server looks for all the unit instances of the unit

class defined with this unit requirement name and prompts the operator to select either MIXER_1 or MIXER_2.

Any unit procedure, including its subordinate operations, mapped to this unit requirement name uses the specified unit class and the associated binding method. Because recipes using dynamic unit allocation require that all unit procedures use a unit requirement name, assign a unit requirement name to all instance-based unit procedures and define the binding method used as Static.

Change procedure unit requirements

If **Support Dynamic Unit Allocation** is enabled, edit unit requirements within a procedure at any time. If disabled, the **Unit Requirement** dialog box is not available at the procedure level. Remove a unit requirement from a procedure at any time, as long as no steps within this procedure define this unit requirement.

To change procedure unit requirements:

1. Open an existing procedure.
2. Select the procedure level of the recipe.
3. From the **Recipe** menu, select **Unit Requirements** to display the **Procedure-Unit Requirements** dialog box.
4. To edit a unit requirement, select the row containing the unit requirement, and then select **Edit Unit Requirement** to display the **Edit Unit Requirement** dialog box.
5. Make the required modifications to the unit requirement. Select **OK** to return to the **Procedure-Unit Requirements** dialog box.



Tip: Modifying from a class-based requirement to an instance-based requirement deletes all of the binding requirements and preferences.

6. To delete an existing unit requirement, select the row containing the unit requirement and select **Delete Unit Requirement**. If the procedure has, any steps defined using this unit requirement, a message displays stating that the requirement cannot be deleted. In this case, delete all steps defined against the unit requirement first. Deletion of a unit requirement removes the unit from any referenced downstream lists.

Smart binding

Smart binding, also referred to as unit attribute binding, enhances Dynamic Unit Allocation by allowing custom, user-defined binding requirements and binding preferences. With smart binding, specify:

- A binding requirement, such as "The unit must be in service."
- A binding preference, such as "I want the mixer cleaned less than 3 days ago."

Configure custom unit attributes such as **VESSEL_STATUS** in the FactoryTalk Batch Equipment Editor. The recipe author then configures binding

requirements and binding preferences based on those custom attributes in the FactoryTalk Batch Recipe Editor.



Tip: Binding requirements and binding preference expressions do not support parameters with deferred values or expression values.

Binding requirements

Bind requirements define subsets of class-based unit procedures and operations. Configure subsets as one of these options:

- **Expression** – Used to reference Unit Attributes assigned to the Unit Class associated with the Unit Requirement. Expression objects can also reference Recipe Header values such as **BATCH_SIZE**, output parameters on the parent recipe, and report and formula parameters on the peer steps within the recipe.
- **Require Phase** – Used when a Unit Requirement must be bound to a Unit that supports a specified recipe phase.
- **Reject Phase** – Used when a Unit Requirement must be bound to a Unit that **does not** support a specified recipe phase. This requirement prevents recipes from using units with unneeded capabilities.
- **Require Attribute** – Used when a Unit Requirement must be bound to a Unit that supports a specified Unit Attribute. It is a **demand** for a Unit that provides an attribute tag for support of the specified unit attribute.
- **Reject Attribute** – Used when a Unit Requirement must be bound to a unit that **does not** support a specified Unit Attribute. This requirement prevents recipes from using units with unneeded attributes.

Source	Type	Description
Global	Expression	VESSEL_STATUS <=> "OUT_OF_SERVICE"
BLEND_OP	Require Phase	TRI_BLENDER
MIX_UP	Require Attribute	TEMPERATURE
PROC_NAME	Expression	MATERIALS_OF_CONSTRUCTION = GLASS_LINED
PROC_NAME	Expression	CAPACITY >= PROC_NAME \ BATCH_SIZE
PROC_NAME	Reject Attribute	PRESSURE
PROC_NAME	Reject Phase	AGITATE

See also

[Binding preferences](#) on page 61

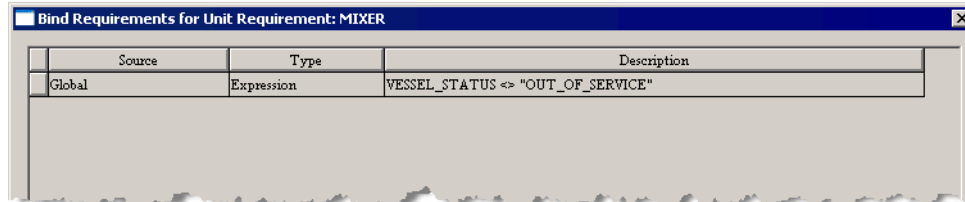
[Create binding expressions](#) on page 63

Configure binding requirements

Once the appropriate custom Unit Attribute definitions, assignments, and configurations are complete in the FactoryTalk Batch Equipment Editor, define the bind requirements in the FactoryTalk Batch Recipe Editor.

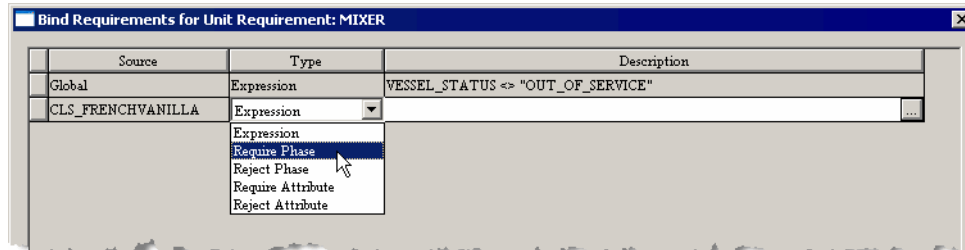
To configure binding requirements:

1. Open a new or existing unit procedure or operation recipe.
2. From the **Recipe** menu, select **Unit Requirements** to display the **Unit Requirement** dialog box.
3. Select **Bind Requirements** to display the **Bind Requirements for Unit Requirement: [CLASS_NAME]** dialog box.



Tip: Any global unit attributes list automatically and are view-only. Configure global unit attributes in the FactoryTalk Batch Equipment Editor.

4. Select **Add Requirement**. A new row displays containing the unit procedure/operation name and boxes for the bind Type and Description.
5. Configure the **Type** and **Description** columns with the applicable bind requirements.



Tip: The **Description** information changes based on the Type selected. If the Type is an Expression, select Browse ... to open the **Unit Binding Expression Builder**.

6. Select **OK** twice to return to the **Procedure View Pane**.

See also

[Binding preferences](#) on [page 61](#)

[Create binding expressions](#) on [page 63](#)

Binding preferences

A bind preference is an object that evaluates against an instance of a Unit Class in order to sort the legal bind targets for a Unit Requirement into a most preferred order. A bind preference can specify a preferred Phase or Unit Attribute, an expression to minimize or maximize, or a Phase or Attribute to avoid. The priority number indicates the order of evaluation for the FactoryTalk Batch Server.

Configure these preferences in recipes as using one of these options:

- **Expression** – A boolean expression used to reference unit attributes assigned to the unit class associated with the Unit Requirement.

Expression objects can also reference Recipe Header values such as **BATCH_SIZE**.

- **Minimize Expression** – An expression that evaluates to either an integer or real value. Legal bind targets for which the expression evaluates to a **lower** value are more **preferred** bind targets than those for which the expression evaluates to a higher value.
- **Maximize Expression** – An expression that evaluates to either an integer or real value. Legal bind targets for which the expression evaluates to a **higher** value are more **preferred** bind targets than those for which the expression evaluates to a lower value.
- **Prefer Phase** – Used as a recipe phase inclusion for bind preferences. The preferred unit supports the specified phase.
- **Avoid Phase** – Used as a recipe phase exclusion for bind preferences. The preferred unit **does not** support the specified phase.
- **Prefer Attribute** – Used as a recipe phase inclusion, it is a **request** for a Unit that provides an attribute tag for support of the specified unit attribute.
- **Avoid Attribute** – Used as a recipe phase exclusion, it is a **request** for a Unit that **lacks** support for the specified recipe phase.

Priority	Type	Description
1	Expression	EFFICIENCY > 80
2	Prefer Attribute	TEMPERATURE
3	Minimize Expression	DAYS_SINCE_LAST_CIP
4	Avoid Phase	XFR_OUT
5	Avoid Attribute	PRESSURE
6	Maximize Expression	TEMPERATURE_CAPACITY
7	Prefer Phase	AGITATE

See also

[Smart binding](#) on page 59

[Configure binding preferences](#) on page 62

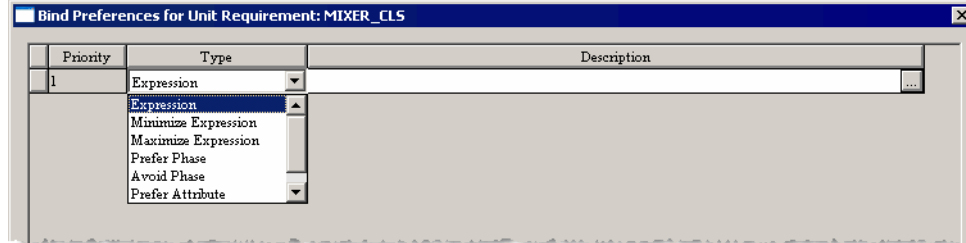
Configure binding preferences



Use this procedure to configure binding preferences.

To configure binding preferences:

1. Open a new or existing unit procedure or operation recipe.
2. From the **Recipe** menu, select **Unit Requirements** to display the **Unit Requirement** dialog box.
3. Select **Bind Preferences** to display the **Bind Preferences for Unit Requirement: [CLASS_NAME]** dialog box.
4. Select **Add Preference**. A new row displays containing the unit procedure/operation name and drop-down boxes for the Bind Type and Description.

- Configure the **Type** and **Description** columns with the applicable bind requirements.



 Tip: The **Description** information changes based on the Type selected. If the Type is an Expression, select browse  to open the **Unit Binding Expression Builder**.

- Select **OK** twice to return to the **Procedure View Pane**.

See also

[Create binding expressions](#) on [page 63](#)

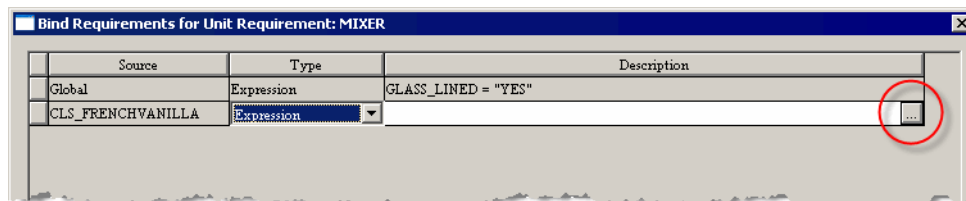
Create binding expressions


When selecting one of the Expression types as the **Type** of bind preference or requirement, define that expression in the **Unit Binding Expression Builder** dialog box.

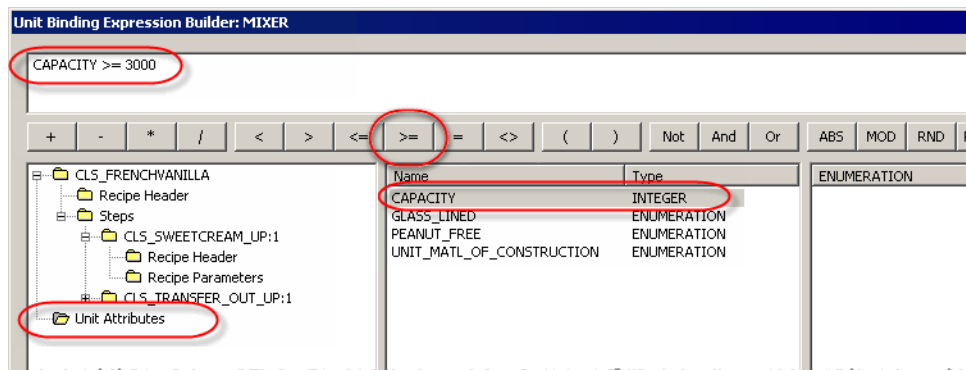
 Tip: Binding requirements and binding preference expressions do not support parameters with deferred values or expression values.

To create binding expressions

- From the **Bind Requirements** or **Bind Preferences** dialog box, select **Add Preference** or **Add Requirement**.



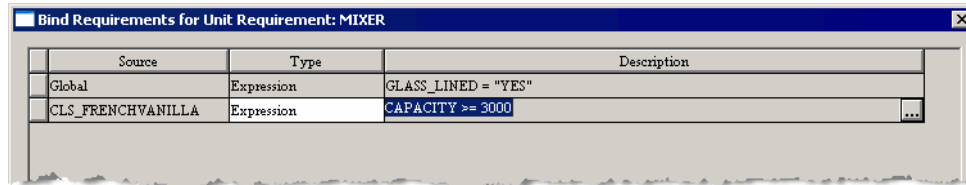
- From the **Type** list, select the type of expression, and then select browse  to the right of the **Description** to display the **Unit Binding Expression Builder** dialog box.





Tip: Bind requirement and preference expressions can reference items that are unit class attributes or a recipe procedure's recipe header, and step parameters.

3. From the tree view in the left pane, select the value type for building the expression, unit attributes or recipe header, or step parameters.
4. From the middle pane, select the operand used for the expression. The example shown lists operands like **CAPACITY** and **GLASS_LINED**.
5. Select an operator button to enter the desired operation into the expression. Type the operation directly into the Expression text box. The example uses the greater than or equal to operator.
6. If the operand is an Enumeration, double-click the value in the right pane to add the value to the expression in the text box; otherwise, type the desired value directly into the expression.
7. Select **OK** to save and validate the expression, and then return to the dialog box from which the **Expression Builder** was opened. The new bind requirement or preference displays.



8. Select **OK** to close the **Bind Requirement** or **Bind Preference** dialog box.

See also

[Bind Expressions](#) on [page 95](#)

Recipe header data


Header data is general information about the recipe. The **Header Data** dialog box contains:

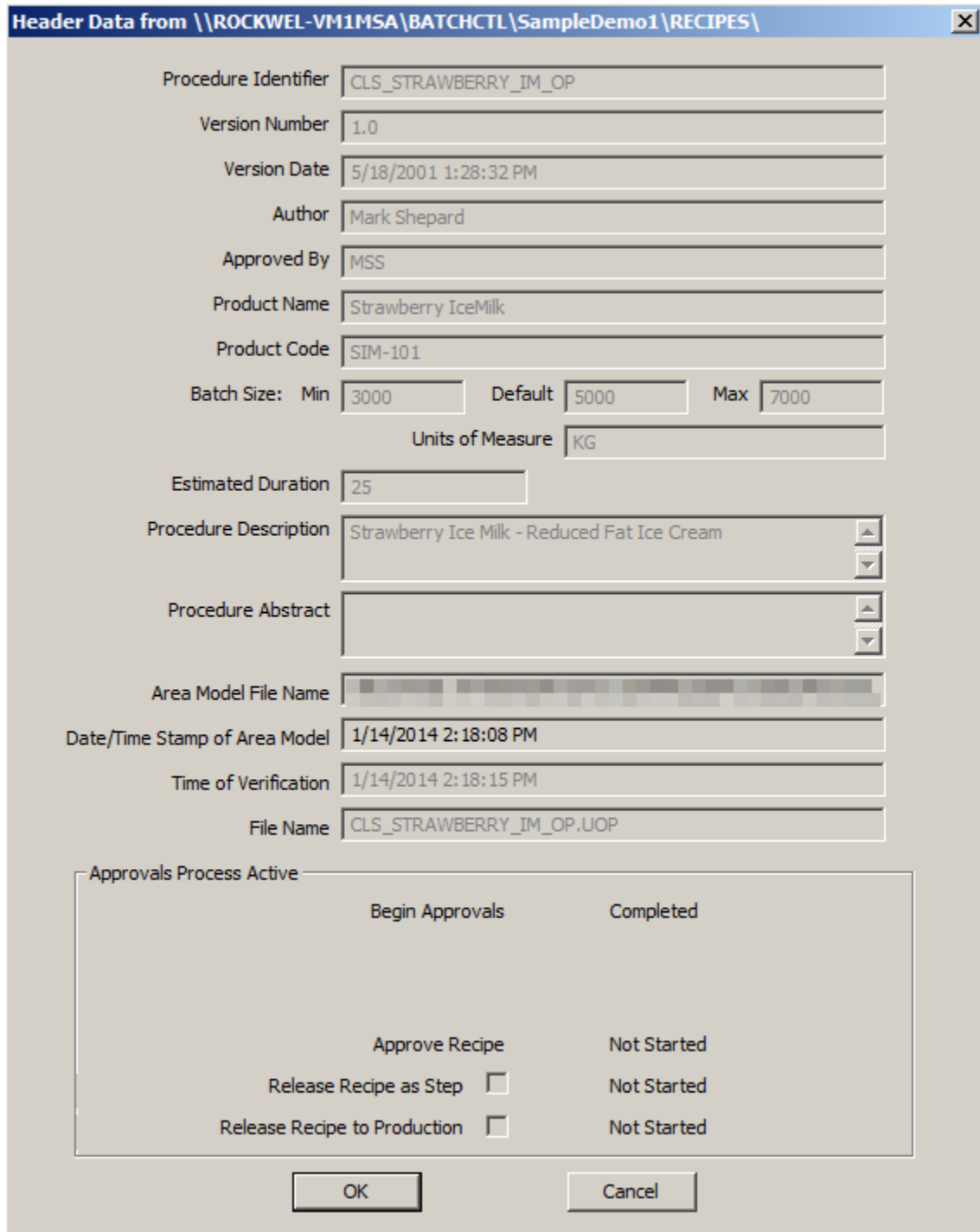
Item	Description
Procedure Identifier	The unique name that displays when working with a recipe in the FactoryTalk Batch View. It is also used as the file name when the recipe is saved. Follow methodologies for assigning product identification codes. The Procedure Identifier can contain letters (A-Z), numbers (0-9), and underscores (_). For Procedures, the Procedure Identifier can begin with either a letter or a number. For Unit Procedures and Operations, it must begin with a letter.
Version Number	Legacy, user-set version number (not related to the Recipe Versioning feature introduced in FactoryTalk Batch version 12). Manually update this version number as desired.
Version Date	Defaults to today's date, cannot be changed.
Author	The name of the individual that created the recipe.
Approved By	The name of the individual who approved the recipe.
Product Name	The product name the recipe produces.
Product Code	A short label that uniquely identifies the product.

Item	Description
Batch Size	<ul style="list-style-type: none"> • Min - a numeric value indicating the minimum size of the completed batch. • Default - a numeric value indicating the default size of the completed batch. • Max - a numeric value indicating the maximum size of the completed batch. • Units of Measure - the type of measurement used to designate the minimum, default, and maximum batch sizes, such as kilograms, pounds, liters, and other units.
Estimated	Approximate length of time it takes to run and complete a batch using the recipe (minutes).
Procedure Description	Product or process description at the procedure level. Indicate whether this is a class-based recipe, or any other distinguishing information required.
Procedure Abstract	A summary or outline of the steps of the procedure.
Area Model File Name	Indicates the location and filename of the area model file against which the recipe was last verified. If the entire path and filename is not visible, place the cursor in the box and use the right arrow key to scroll to the right.
Date/Time Stamp of Area Model	Date and time the area model was last saved.
Time of Verification	Date and time the recipe was verified. If the recipe was never verified, this information is also indicated.
File Name	Recipe filename.
Approval Process Area (recipe approvals <i>enabled</i>)	<p>The title of this area changes to reflect the active Approval Process for the recipe:</p> <ul style="list-style-type: none"> • Approval Process Not Started • Primary Approval Active • Expedited Approval Active • Pending Recipe Selection <p>Recipe Approval Process steps, and their current state, are listed in this area. Hover the cursor over an approval process step to view a description of the approval step.</p> <ul style="list-style-type: none"> • Checkboxes are read-only and controlled by the Approval Process. • When checked, Release Recipe as Step signifies the recipe (or unit procedure or unit operation) is approved for inclusion as a step in another recipe. • When checked, Release Recipe to Production signifies the recipe is approved for inclusion on the Recipe List in FactoryTalk Batch View, in eProcedure, or in custom applications that use Control Recipe List HMI Controls. <p>The state of each approval step:</p> <ul style="list-style-type: none"> • Not Started - approval sign off not started. • In Progress - approval started but one or more signoffs still required. • Reverting - revert started but one or more signoffs still required. • \$System Signoff Pending - translated (migrated) recipes, or recipes reentering an approval process, are set to begin with the Expedited approval process. Before verification, the Expedited approval steps for these recipes are set to a pending state. • Completed - all signoffs made and approval is complete.
(Recipe approvals <i>disabled</i>)	<p>This area contains checkboxes indicating the states of the Release Recipe as Step and Release Recipe to Production recipe properties.</p> <p>The checkboxes are active and can be selected to toggle between true and false.</p> <ul style="list-style-type: none"> • Release Recipe as Step, when set to true, signifies the recipe (or unit procedure or unit operation) can be included as a step in another recipe, but batches cannot be created from it alone. • Release Recipe to Production, when set to true, signifies the recipe can be added to the Recipe List in FactoryTalk Batch View, in eProcedure, or in custom applications that use Control Recipe List HMI Controls.

Add recipe header data

Use this procedure to add recipe header information to a recipe.

1. With the appropriate recipe level open in the FactoryTalk Batch Recipe Editor, select **Header Data** . The **Header Data** dialog box opens. Type the required data (**Procedure Identifier**, **Version Number**, and **Author**) in the **Header Data** dialog box.



Header Data from \\ROCKWEL-VM1MSA\BATCHCTL\SampleDemo1\RECIPES\

Procedure Identifier: CLS_STRAWBERRY_IM_OP

Version Number: 1.0

Version Date: 5/18/2001 1:28:32 PM

Author: Mark Shepard

Approved By: MSS

Product Name: Strawberry IceMilk

Product Code: SIM-101

Batch Size: Min 3000 Default 5000 Max 7000

Units of Measure: KG

Estimated Duration: 25

Procedure Description: Strawberry Ice Milk - Reduced Fat Ice Cream

Procedure Abstract:

Area Model File Name:

Date/Time Stamp of Area Model: 1/14/2014 2:18:08 PM

Time of Verification: 1/14/2014 2:18:15 PM

File Name: CLS_STRAWBERRY_IM_OP.UOP

Approvals Process Active

Begin Approvals	Completed
Approve Recipe	Not Started
Release Recipe as Step <input type="checkbox"/>	Not Started
Release Recipe to Production <input type="checkbox"/>	Not Started

OK Cancel

2. (optional) Enter information in the remaining boxes.



Tip: If an Approvals Process is enabled for the area model, then the approval process, along with configured approval steps, is displayed in the bottom area of the dialog box. The two checkboxes are read-only (disabled) but the status displayed reflects their current state.

3. If the Approvals Process is disabled, then the two checkboxes are enabled. Select one or both of the boxes as required.
 - Select **Release Recipe As Step** to use the recipe or operation as a component in another recipe. With just this box checked, the recipe does not appear in the Recipe List.
 - Select **Release Recipe To Production** for the recipe to appear in the FactoryTalk Batch View, eProcedure, or HMI Controls Recipe List, to create production batches.
4. Select **OK**.

Recipe storage

After building the recipe, save the recipe by selecting **Save**. The recipe is saved in the location and format defined in the FactoryTalk Batch Equipment Editor **Server Options** dialog box.

If recipes are stored using binary files, the recipes save separate files according to the recipe level:

- Procedures are stored as **.bpc** files
- Unit Procedures are stored as **.upc** files
- Operations are stored as **.uop** files

If recipes are stored using XML, the recipes save separate files according to the recipe level:

- Procedures are stored as **.pxml** files
- Unit Procedures are stored as **.uxml** files
- Operations are stored as **.oxml** files

If recipes are stored using the RDB format, recipes are stored in a single SQL Server database (MasterRecipes is the default database name).

Copy recipe options

Copy recipes by using the **Save As** function or using the backup directory. When selecting a procedure level or unit procedure level recipe and the **Save As** function is used, only the selected recipe level (procedure or unit procedure) saves with a new name. Recipe Approval steps and their states do not carry over to the new copy.

IMPORTANT Any lower level recipes (unit procedure, operation, or phase) are not copied with the selected recipe level. Any changes made to those lower levels modify the original version. To modify the original versions, perform a **Save As** on each of the lower level recipes.

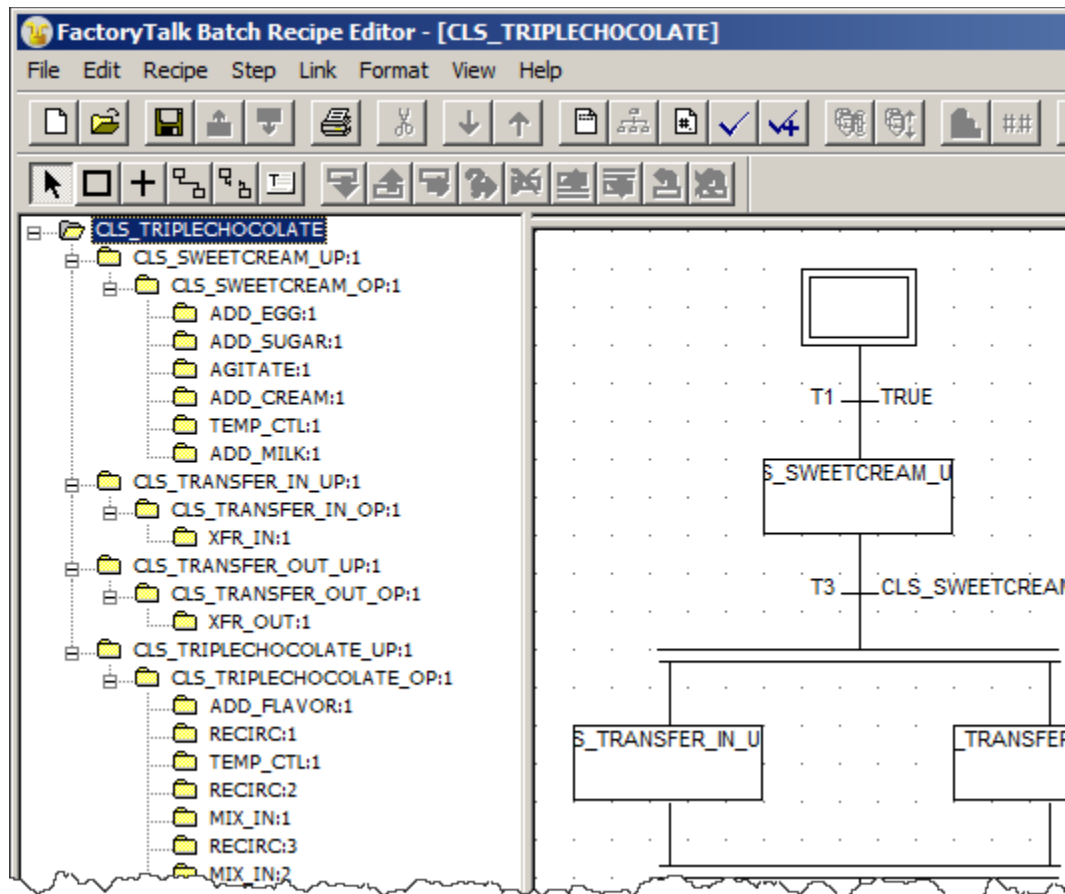
Many recipes use deferred parameters in order to minimize the number of different recipes maintained. Because deferred parameter values assign at a higher level recipe, using the **Save As** command to copy the recipes with deferred parameters does not affect the original recipe.

Operations can be copied using **Save As** because they are the lowest level and do not point to upper levels. Changes made to the copy do not affect the original operation.



Tip: With Recipe Approvals enabled, using **Save As** clears all signature-approved approval steps in the new copy. The approval steps in the original recipe are not affected.

In this example, performing a **Save As** on the procedure level saves a new copy of that procedure only. The new procedure still points to the original unit procedures, operations, and phases. If changing a parameter value on a phase in the copied recipe, that parameter value changes in the original recipe. The same applies to the operations and unit procedures; all changes affect the original recipe.



Copy recipe with backup directory

Use this procedure to copy a recipe using the backup directory.

1. Open File Explorer to create a backup directory for recipes.
2. Copy all recipes to duplicate to the backup directory.
3. In the FactoryTalk Batch Equipment Editor, open the **Server Options** dialog box and change the **Recipe Directory** to point to the backup directory.
4. Open the FactoryTalk Batch Recipe Editor. If it was already open, exit the FactoryTalk Batch Recipe Editor and restart it.

5. Open the recipe to copy. The phases, operations, and unit procedures display in the **Procedure View** pane hierarchical list.
6. Select a phase in the first operation of the hierarchical list. From the **Recipe** menu, select **Header Data** to display the operation information.
7. Change the **Procedure Identifier** to a new name, and then select **OK**. Change the other information if needed.
8. A message indicates recipes affected by the change. Select **Proceed**, and then select **OK**.
9. Select the operation that just renamed. Open the **Header Data** and change the **Procedure Identifier** to a new unit procedure name. Select **OK**, and a message lists recipes affected by the change. Select **Proceed**, and then select **OK**.
10. Repeat steps 6 through 9 for each operation and unit procedure.
11. Select any unit procedure. Open the **Header Data** and change the **Procedure Identifier** to a new name for the procedure. Select **OK** and a message displays the recipes affected by the change. Select **Proceed**.
12. Save the procedure.
13. Export the renamed recipe back to the original recipe directory.
14. In the FactoryTalk Batch Equipment Editor, open the **Server Options** dialog box and change the **Recipe Directory** to point back to the original recipe directory.
15. Open the FactoryTalk Batch Recipe Editor to display the newly created recipe copies.

Recipe unit requirements

When initially building a recipe, define two things:

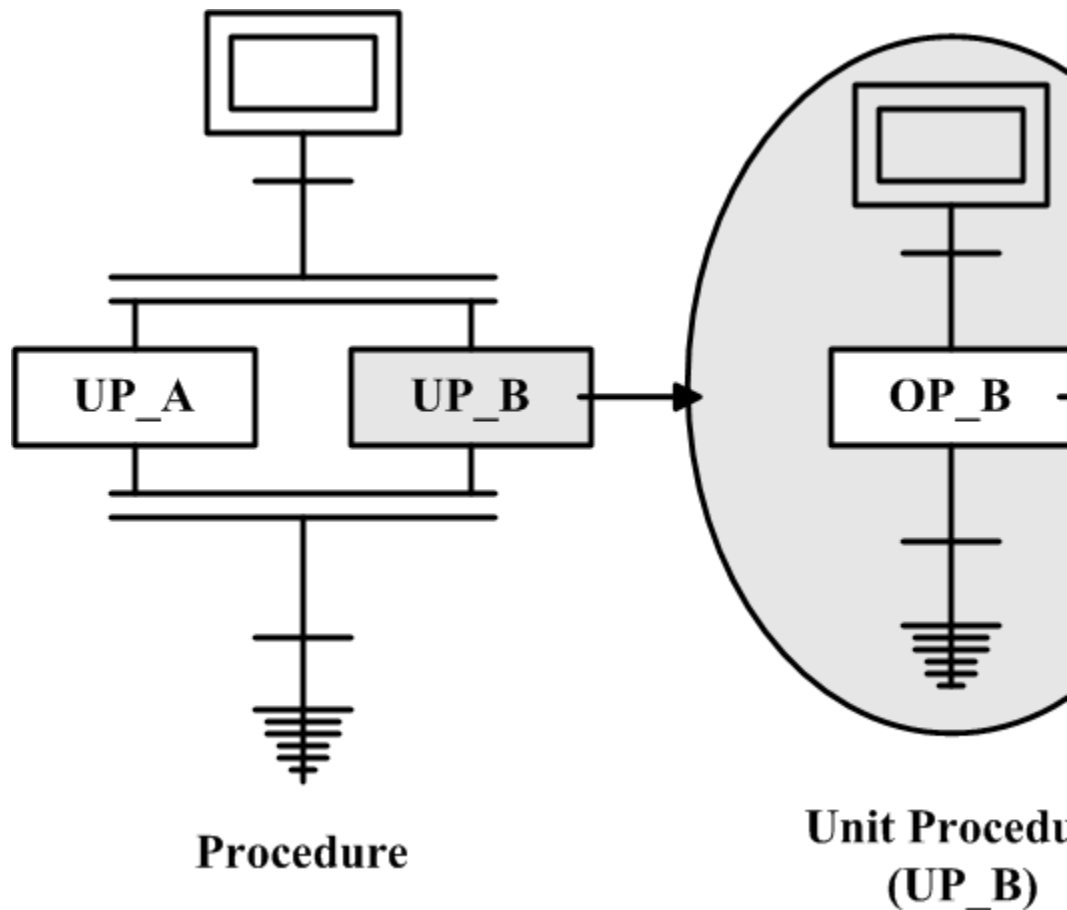
- The procedure level: procedure, unit procedure, or operation.
- The recipe unit requirements.

To properly define the recipe requirements, have a good understanding of the area model, the equipment classes, and the actual equipment required for recipe execution. If phases are material-enabled, know what materials and containers are needed for the recipe.

Prior to creating a recipe, determine if the recipe is unit-based, class-based, or a combination of the two. If Dynamic Unit Allocation is enabled and the recipe is a procedure containing some or all class-based unit procedure and operation level recipes, choose a unit allocation method.

Recipe procedure levels

The first step in planning a new recipe is determining the level of the procedure. A batch recipe may consist of up to four procedure levels. These levels follow the S88.01 Procedural Model:



- Procedure
- Unit Procedure
- Operation
- Phase

Procedure-level recipes contain steps called unit procedures. Unit procedure-level recipes contain steps called operations. Operation-level recipes contain steps called phases. Because of this hierarchical structure of the S88.01 Procedural Model, plan recipes from the top down, but build recipes from the bottom up. First, create the operation, then create the unit procedure, and lastly create the procedure.

If a recipe is run as a single unit (no unit-to-unit transfers are involved), then building a recipe as an operation may be adequate. If the recipe intends to operate across multiple units, create a procedure containing a unit procedure for each unit involved. Each unit procedure may consist of one or more operations.

To begin, have a good definition of what the recipe needs to do, then decide what process levels are needed. Start by building the operations and configuring the phases. Add the operations to the unit procedures, then add the unit procedures to the top-level procedure.



Tip: Every unit procedure must have at least one operation, and every procedure must have at least one unit procedure.

Recipe formulations

Recipe formulations build a common recipe that can make a variety of products by changing recipe input parameters. Users build and save a recipe formulation, and then select from the list of saved formulations when adding a batch to the batch list.

Consider the following when using recipe formulations:

- Formulations only apply to recipes. They cannot be written for manually running operation sequences or phases.
- Formulations only apply at the highest level of the control recipe. Recipes within a control recipe cannot use its formulations.
- Formulations are optional. Master recipes without a formulation run with the formula values in the Master recipe.
- Formulations not specified when creating a batch run with the default formula values from the Master recipe.

Formulation parameters

Recipe formulation parameters change the input of a recipe formula to quickly change the production line from one recipe variant to another, such as vanilla to chocolate ice cream.

Review these guidelines when using recipe formulation parameters:

- The formulation value must be within the **Min** and **Max** ranges defined in the Master recipe parameter.

If the formulation value is out of range, recipe verification fails. Open the **Formulation Parameters** dialog box to review any out of range values. A recipe with an out of range **Min** and **Max** value can still be saved.

If the **Min** and **Max** values of a phase change to values that cause the recipe parameter values to be out of range, the phase step parameters update to the new value(s), the origin is updated to the new value(s), which breaks the deferral, and the value(s) for the formulation parameters remain the same.

- The formulation value must correspond to the data type defined in the Master recipe parameter:
 - **Integers:** Numeric values only.
 - **Real:** Numeric values, with a decimal space as needed.

- **String:** Any character, with a maximum of 255 characters.
- **Enumerations:** Select from a list associated with the defined enumeration set for the Master recipe parameter.

If a data type mismatch exists, recipe verification fails with an error that the recipe formulation data type does not match the data type for the recipe parameter.

If the data types of a phase change in the area model, when verifying the recipe, the phase step parameters update to the new value and data type, the origin updates to the new value and data type, which breaks the deferral, and the value and data type for the formulation parameters remain the same.

- When defining new input parameters on the Master recipe, the FactoryTalk Batch Recipe Editor automatically updates existing formulations with the new parameter.
- The value of the new formulation parameter defaults to the default value of the Master recipe parameter.
- When deleting an existing Master recipe parameter, the FactoryTalk Batch Recipe Editor automatically deletes the corresponding formulation parameter from existing formulations.
- Parameter modifications other than creation and deletion do not apply to recipe formulations and may cause formulation verification errors.

Add a recipe formulation

Use these instructions to add a common recipe that can make a variety of products by changing recipe input parameters.




Tip: When viewing recipe formulations for read-only recipes (SAI protected, view-only user, versioned recipes, or recipes with approval signatures), the grid displaying the formulations, and the **Add Formulation**, **Delete Formulation**, and **OK** buttons are disabled.

1. Open the master recipe in the Recipe Editor.
2. Select **Recipe > Formulations**.
3. In the **Recipe Formulations** dialog box, select **Add Formulation**. A new row with empty **Name** and **Description** boxes displays.
4. Enter a name for the recipe formulation.



Tip: There is a maximum limit of 128 characters, and valid characters are A-Z, 0-9, and underscore.

5. Select browse  beside the **Parameters** box to display the **Parameters for Formulation** dialog box.



Tip: Define parameters for this dialog box by selecting **Recipe > Recipe Parameters /Reports**.

6. Modify any values in the **Formulation Value** boxes and select **OK**.

Review [Guidelines when using formulation parameters](#) on [page 73](#) for more information.

7. Enter a description for the recipe formulation.



Tip: There is a maximum limit of 255 characters, and all characters are valid.

8. (optional) To add another recipe formulation, repeat steps 3-7.
9. Select **OK**. The dialog box closes and the formulation is added to the recipe.

Delete a recipe formulation

Use these instructions to delete a recipe formulation.

1. Select **Recipe > Formulations**.
2. In the **Recipe Formulations** dialog box, select the number beside the row from the list of formulations.
3. Select **Delete Formulation** to remove the row from the list.
4. Select **OK**. The dialog box closes.

Build a Sequential Function Chart

The creation of an Sequential Function Chart (SFC) involves the addition of steps, transitions, and links. During the recipe creation process, an initial SFC displays. This SFC consists of the initial step, initial transition, final transition, and final step. Add steps to the chart and then define the steps appropriately. If needed, reconfigure transitions that require special expressions and include any necessary comments.




Tip: In addition to using the mouse for navigation, keyboard controls are also available. When focus is in the **SFC** view, use the arrow keys to navigate between recipe elements. Use the arrow keys while pressing the **Ctrl** key to move between selected recipe elements. When deleting a selected step with the delete key, the SFC is not automatically rearranged. The active step outlines in dark blue and **SFC** displays on the status bar at the bottom of the screen when the SFC view has focus.


Add a branch structure to an SFC

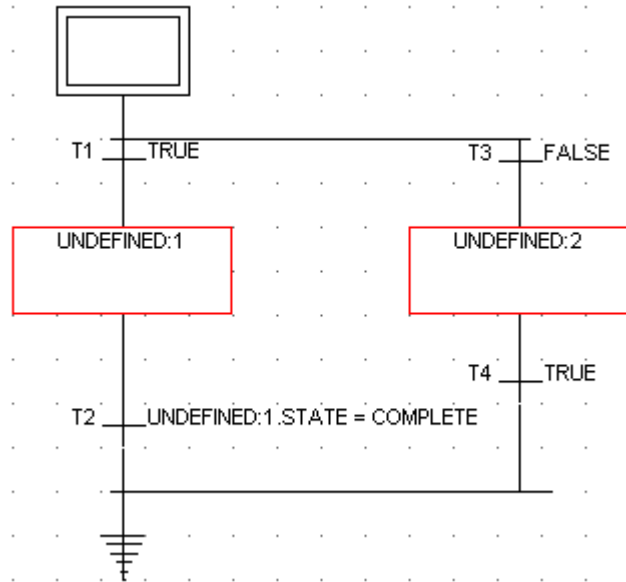
Add a branch structure to an SFC using the **Add Branch** tool, which creates a new step, a new transition preceding the step, and a new transition following the step. The new preceding transition contains the transition condition **FALSE** and the new following transition contains the transition condition **STEP.STATE=COMPLETE**.



Tip: Due to the variation and complexity of SFC branch structures, it is necessary to redefine one or both transitions.

1. Use the **Selection Tool**  to select the step on which the new branch structure is to border.

2. Select **Add Branch** . A new step creates on the SFC and outlines in blue (indicated by the dashed lines). A new transition precedes the new step, and a new transition follows the new step. Additionally, the SFC structure automatically rearranges to make room for the new elements. One of these dialog boxes opens, based on the recipe level: **Select Phase**, **Operation Select**, or **Unit Procedure Select**. Make the appropriate selection and select **OK**.



3. Define the new preceding and following transition expressions, if necessary.

Add transitions to an SFC

Use the **Transition Tool** to add transitions to an SFC. The **Transition Tool** creates a single transition requiring a manual link to the appropriate steps and defines the transition expression.

1. Select **Transition Tool** .
2. Place the cursor in the desired location in the SFC View and click. The transition, with its name (T followed by a number), is placed on the SFC.



Tip: The transition displays in blue until another transition is added or the Recipe Construction tool is selected. The transition then displays in red until configured.

3. Link the transition to the appropriate steps.
4. Assign a transition expression.



Tip: In FactoryTalk eProcedure recipes, all loops and transitions must branch to the right side of the SFC steps.


Define a step

Newly added steps are UNDEFINED. Only valid lower level recipes are used in the step definition. When a process for a step is selected, that process is assigned to that particular step in the recipe, and the step is marked with the process name. Redefine a step at any time.

1. Double-click a step marked UNDEFINED. One of these dialog boxes opens, based on the recipe level: **Select Phase**, **Operation Select**, or **Unit Procedure Select**.
2. To define a phase or operation either select from the option list and then select **OK**. Or, select **New Phase** or **New Operation** to create a new phase or operation. See **Create operations and unit procedures**.
3. To define a unit procedure:
 - a. From the **Alias** list, select the alias to associate with this unit procedure.
 - b. The **Alias** list contains a list of all units defined in the procedures unit requirements.
- a. If this unit runs in parallel with other units, and if each of the units at this level must acquire in order to initiate this level of the recipe, then select **Acquire unit prior to starting unit procedure**.
 - b. If the units running in parallel can run independently of each other, clear this check box for all of the units at this level. The recipe then runs the individual units acquired, regardless of whether the other units acquire.
- c. Select the appropriate unit procedure to associate with this step. Or, select **New Unit Procedure** to create a new unit procedure.
- d. Select **OK**.

Link SFC components

Use this procedure to link SFC components.

1. Select **Link Tool** .
2. Place the cursor at the links starting step. Drag the link to the connecting step, and release the mouse button to complete the link. The elements are now connected.




Tip: In FactoryTalk Procedure recipes, all loops and transitions must branch to the right side of the SFC steps.

Remove a link from an SFC

Follow these instructions to remove a link from an SFC.

To remove a link from an SFC:

1. Select **Remove Link Tool** . The cursor changes to a + sign.
2. Place the cursor at the links starting step. Drag the cursor to the connecting step.
3. Release the mouse button to remove the link.

See also

[Recipe construction toolbox](#) on [page 19](#)

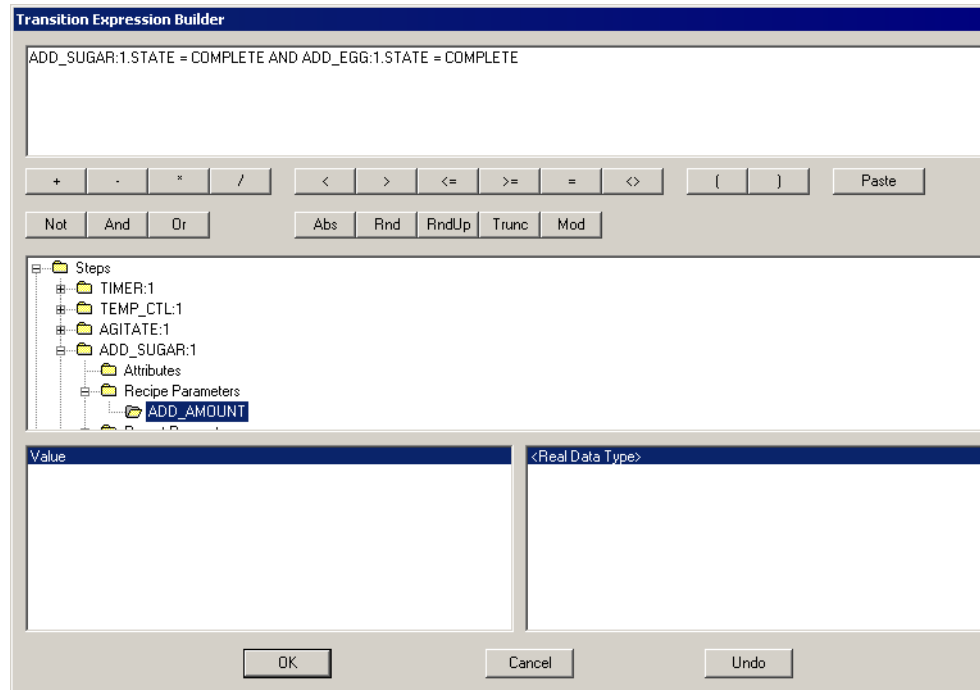
[Link SFC components](#) on [page 79](#)

Assign a transition expression

After adding a transition to the SFC, assign a transition expression.

To assign a transition expression:

1. Double-click a transition in the SFC view. The **Transition Expression Builder** dialog box appears.



2. Enter the expression for the transition condition or build an expression by selecting options from the three different sections. Press **Enter** to continue the expression on the next line.
3. Select **OK** to return to the **SFC** view.

See also

[Transition expressions](#) on [page 24](#)

Remove a step from an SFC

Remove a step from an SFC using the **Selection** tool.

1. Using the **Selection Tool** , select the step to remove.
2. Select **Remove Step** .

The selected step is removed and all affected transitions are removed or modified to reflect the new SFC structure. Additionally, the SFC structure rearranges automatically to recover the space created by any removed elements. If there are no recipe elements bordering the

removed step (if the step was added to the **SFC** view without being linked), a blank space is left in the SFC structure. Any blank spaces left within the SFC structure must be linked manually using the **Link Tool**






Tip: Transitions following parallel structures can be extremely complex so they are not automatically reconfigured. If the transition following the removed parallel step requires reconfiguration, it must be configured manually.

Create material loops

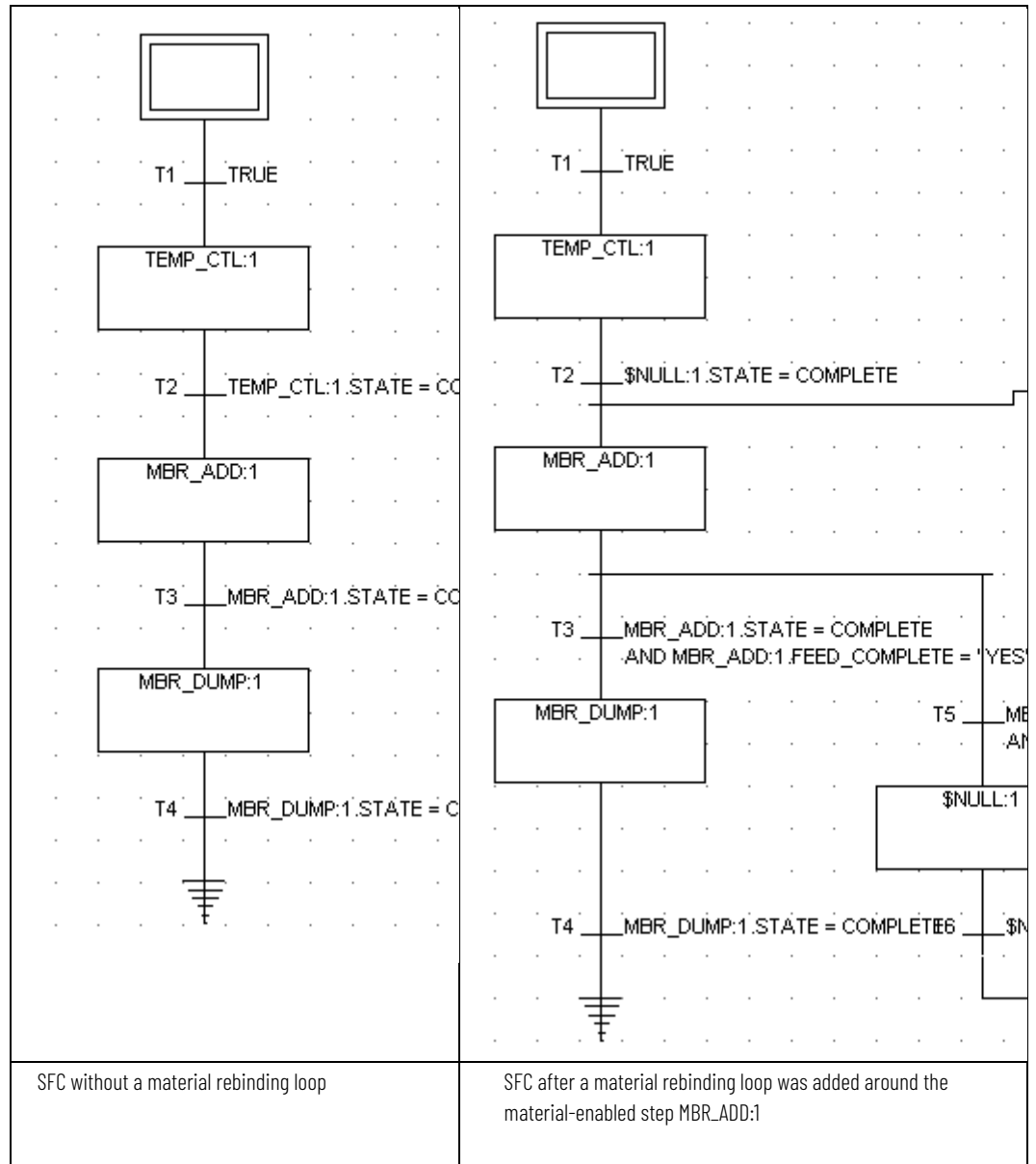
If the SFC contains a material-enabled step, create a material loop that allows the step to automatically rebind when a feed is not completed. Because material loops disable the ability to create complex recipe structures, add them after the SFC is set up with AND and OR **convergences** and **divergences**. For example, to create three material equipment phases in parallel, create the three parallel recipe phases first, and then add the material loop.

To create material loops:

1. Using the **Selection Tool** , select the material enabled step to add to the loop.
2. Select **Create Material Loop** . The material loop is added, including the two required null procedures.

To remove a material loop immediately after adding it, select **Undo Material Loop** .

This example shows how a material loop appears when the material-enabled step is preceded and followed by a transition. The material loop may look different and contain more than one null procedure if it is preceded or followed by a convergence or a divergence.



SFC without a material rebinding loop

SFC after a material rebinding loop was added around the material-enabled step MBR_ADD:1

Recipe comments


Use recipe comments to give written instructions to operators, share notes with other engineers, or access a reference. Recipe commenting associates data with a step, transition, or the entire recipe. The comment is viewable at design and at run time.

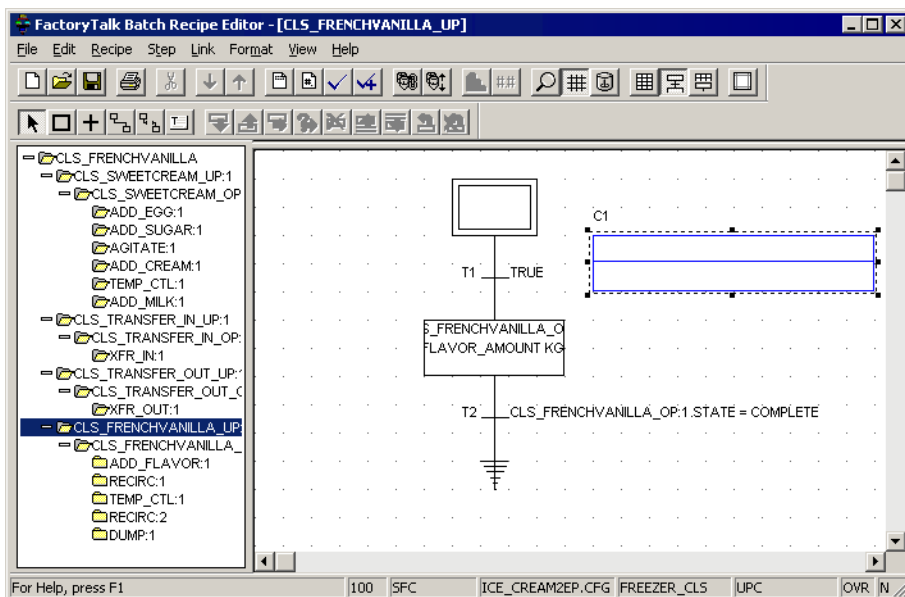
The comment, and its associated information, is a text box. Above each text box, the assigned reference name displays as a C followed by a number. View added SFC text boxes in the SFC views of FactoryTalk Batch View and FactoryTalk Batch View HMI Controls.

Add comments to an SFC

Use these instructions to add comments to an SFC.

To add comments to an SFC:

1. Select **Text Box Tool** .
2. Move the cursor (text box) to the desired location, click, and the text box is displayed. When placing the text box, pick an area in the SFC that does not interfere with viewing existing steps and transitions as these elements do not relocate when the text box is generated. If necessary, text boxes, steps, and transitions can be moved by selecting and dragging them to new locations.



3. Use the selection tool to add the required information inside the text box. When finished typing the text, select outside the box. The text box resizes to accommodate the comment. If needed, use the Selection Tool to edit existing text.



Tip: The maximum number of characters contained in a text box is 1024. If more capacity (space) is required for a recipe comment, use additional text boxes and associated them with the step or transition.

See also

[Recipe construction toolbox](#) on [page 19](#)


[Associate recipe comments](#) on [page 83](#)

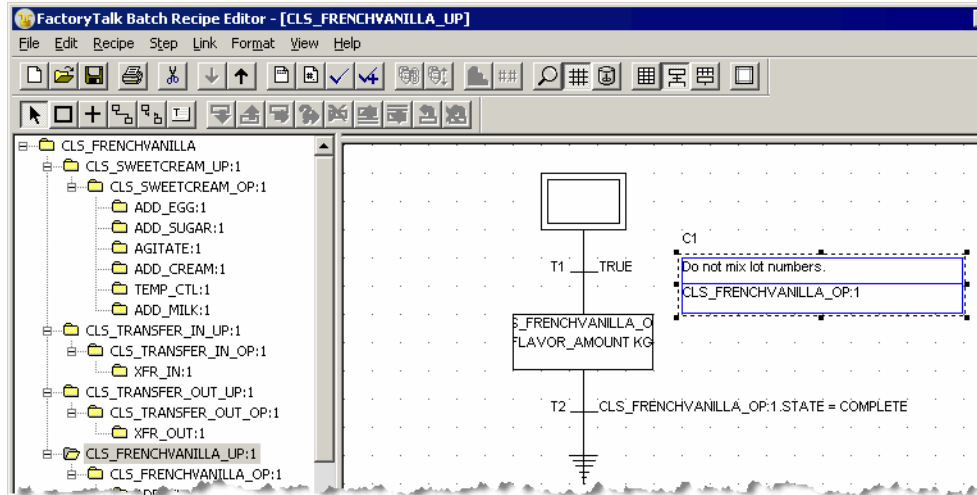
[Disassociate recipe comments](#) on [page 84](#)

Associate recipe comments

When a text box is created, the default is a general recipe procedure comment. The associated information section is blank.

To associate a recipe comment with a step or transition:

1. Select **Link Tool** , select inside the text box, and drag to the desired step or transition. Completing this step in reverse order is also an option.



2. Release the button, and the bottom portion of the text box displays the associated step or transition name.



Tip: A text box can only be associated with one step or transition. Linking a text box to a step or transition that is already associated to another element results in assigning the most recently associated step or transition name.

See also


[Recipe construction toolbox](#) on [page 19](#)

[Disassociate recipe comments](#) on [page 84](#)

Disassociate recipe comments

Use these instructions to disassociate a comment.

To disassociate a recipe comment from a step or transition

1. Select **Remove Link Tool** , select inside the text box, and drag to the associated step or transition. This step can be performed in reverse order with the same result.
2. The associated step name is removed from the text box and now associated with the entire recipe.



Tip: If a step or transition associated with a text box is deleted, the text box becomes associated with the entire recipe procedure and the association label is blank.

Delete recipe comments

Follow these instructions to delete recipe comments.

To delete recipe comments



1. Select the text box to be deleted.
2. From the **Edit** menu, select **Delete Selection** or press **Delete**.

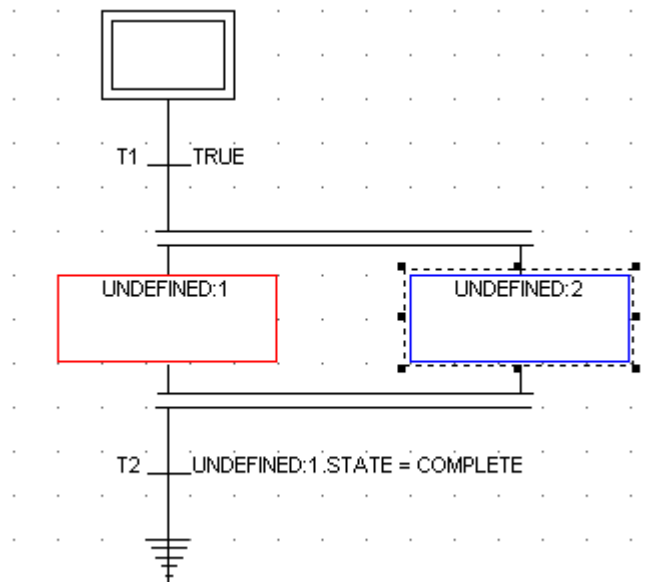
The comment and text box are removed from the SFC.

Add a parallel step to an SFC

Use the **Add Parallel** tool to add a step to an SFC in parallel with another. The **Add Parallel** tool creates a parallel step, the appropriate links, and redefines the existing transition to reflect the new parallel structure.

To add a parallel step:

1. Use the **Selection Tool**  to select the step with which the new step is to run in parallel.
2. Select **Add Parallel** . A new UNDEFINED step creates in parallel with the selected step on the SFC and outlines in blue. Additionally, the SFC structure automatically rearranges to make room for the new elements. One of these dialog boxes opens, based on the recipe level: **Select Phase, Operation Select, or Unit Procedure Select**. Make the appropriate selection and select **OK**. The existing transition modifies to reflect the new step and the parallel structure.



See also

[Recipe construction toolbox](#) on [page 19](#)

[Define a step](#) on [page 78](#)

Parallel steps use same resource

If parallel steps require the same dedicated resource, the FactoryTalk Batch Server automatically determines how the resources allocate between steps when the batch runs. This is subarbitration. If parallel steps requiring the same resource occur just after an AND divergence or just before an AND convergence, add null procedures to the recipe in order for batches to run successfully.

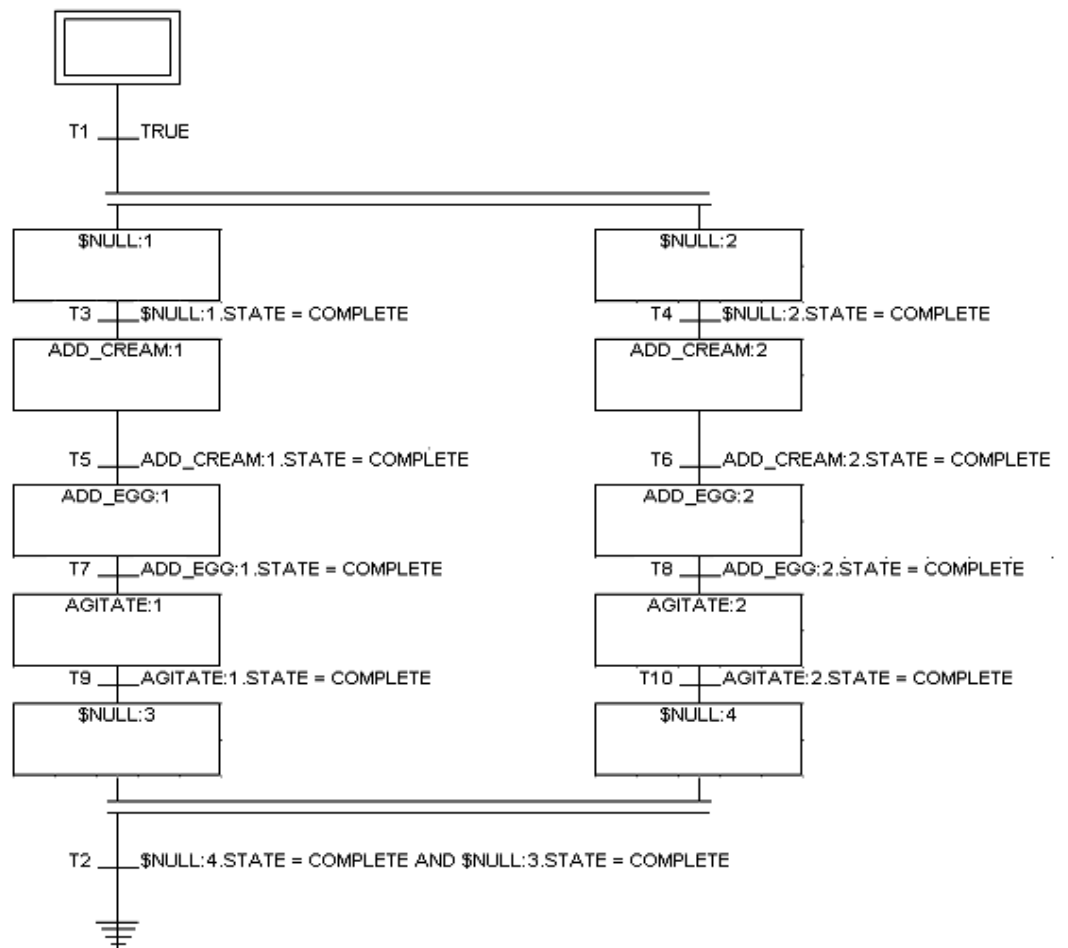
This example shows the correct placement of null procedures in a sequential function chart (SFC) that contains parallel steps requiring the same phase.

In this SFC, each set of parallel steps is requesting the same phase.

Add null procedures after an AND Divergence of two parallel steps requesting the same dedicated resource.

Batch Server subarbitrates resources for parallel steps requesting the same dedicated resource.

Add null procedures before an AND Convergence of two parallel steps requesting the same dedicated resource.



- The first set of steps after the AND Divergence, ADD_CREAM:1 and ADD_CREAM:2, require a set of null procedures before them so subarbitration can work. Without the null procedures, the batch goes into a HELD state before it reaches these steps.
- The FactoryTalk Batch Server automatically subarbitrates for the second set, ADD_EGG:1 and ADD_EGG:2.
- The third set, AGITATE:1 and AGITATE:2, requires a set of null procedures before the AND Convergence so subarbitration can work.

Without the null phases, the batch appears as though it is still running, but it never is able to transition past these steps to the Convergence.

Insert steps into an SFC




When constructing a recipe, insert steps into an SFC using the **Add Step** tool or the **Insert Step** tool.

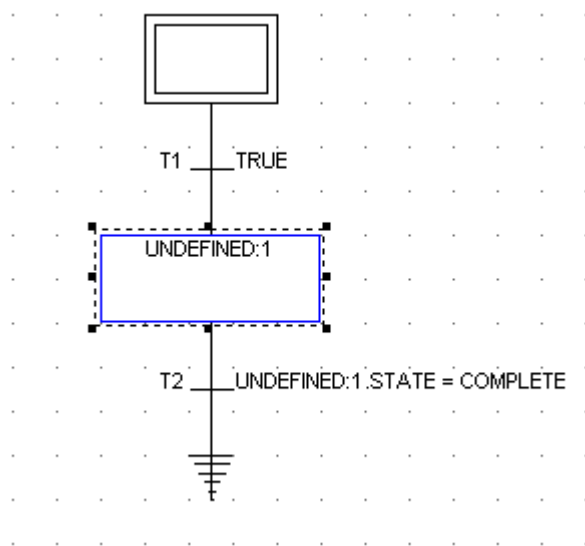
- The **Add Step** tool creates a new step after the selected recipe element.
- The **Insert Step** tool creates a new step before the selected element.

If required, a new transition automatically creates for the new step and a default common transition expression automatically configures. If a new transition is not required, the existing transition modifies to reflect the new step.



Tip: Add material-enabled phases to operations just like other phases. When adding a material-enabled phase to the recipe, define the material parameters.

1. Using the **Selection Tool** , select a step or transition that either precedes or follows the new step.
2. Select **Insert Step**  to insert a new step before the selected element, or select **Add Step**  to insert a new step after the selected element. A new UNDEFINED step displays on the SFC with a blue box outlined in a black dashed line. Additionally, the SFC structure automatically rearranges to make room for the new elements. One of these dialog boxes opens, based on the recipe level: **Select Phase**, **Operation Select**, or **Unit Procedure Select**.
3. Make the appropriate selection and select **OK**.






4. Repeat Steps 1 through 3 until all of the required steps are on the SFC.



Tip: The word UNDEFINED displays inside the step until configured.

Insert timer steps

Timer steps are available at all levels of a recipe: operations, unit procedures, and procedure.

1. Using the **Selection Tool** , select the transition that either precedes or follows the new step.
2. Select **Insert Step**  to insert a new step before the selected element, or select **Add Step**  to insert a new step after the selected element. One of these dialog boxes opens, based on the recipe level: **Select Phase**, **Operation Select**, or **Unit Procedure Select**. These instructions are for using the **Select Phase** dialog box. For information on inserting a step into a Unit Procedure or Operation, see **Define a step**.
3. Select **TIMER** from the dialog box and select **OK** to display the **Select Timer - TIMER** dialog box.

Enter these values:

- a. Type a new name for the Timer step.
- b. Select the Timer type (COUNT_DOWN or COUNT_UP).
- c. Select the units of measure (seconds, minutes, hours, or days).
- d. Select **OK**. The new Timer step displays in the SFC.




Tip: If a COUNT_UP timer in parallel with existing steps displays, the transition expression does not change. If it is the only step, then the transition defaults to TRUE.

4. Set parameter values for the Timer step. For more information, see [Assign formula values to Timer steps](#) on [page 109](#).

Manually add steps to an SFC

Manually add a step to a sequential function chart using the **Step Tool** on the Recipe Construction toolbox. When using the **Step Tool**, no new transitions are added, no existing transitions are modified, no links are created for the step, and the SFC structure is not automatically rearranged to make room for the new step.

1. Select **Step Tool** .
2. In the SFC view, place and click the cursor in the desired location.



Tip: The step is placed on the sequential function chart and is outlined in blue (indicated by the dashed lines) until another step is placed or another tool is selected. The step displays in red until configured. UNDEFINED is displayed inside the step until it is defined.

3. Repeat Step 2 until all of the required steps for the process display on the sequential function chart.

Build a table

The creation of a table involves the addition and configuration of steps to the table structure. Add steps to the table and then define the steps appropriately.



Tip: While in the **Table** view, the active step is highlighted in dark blue and **Table** is displayed on the status bar at the bottom of the screen when the **Table** view has focus. Delete a selected step in the table by pressing the **Delete** key.



Insert steps into a table

When constructing a recipe, steps can be inserted into a table using the **Add Step** tool or the **Insert Step** tool. The **Add Step** tool creates a new step after the selected step, while the **Insert Step** tool creates a new step before the selected step.



Tip: Material-enabled phase steps are added to operation recipes similar to all other phase steps.

To insert steps into a table:

1. Select the table row that either precedes or follows the new step.
2. Select **Insert Step**  to insert a new step before the selected row, or select **Add Step**  to insert a new step after the selected row. A new UNDEFINED step is created in the table and highlighted in dark blue. One of these dialog boxes opens, based on the recipe level: **Select Phase**, **Operation Select** or **Unit Procedure Select**.
3. Make the appropriate selection and select **OK**.
4. Repeat steps 1 through 3 until all of the required process steps are placed in the table.


STEPS		
1	↓	UNDEFINED:1





Tip: The word UNDEFINED displays inside the step until it is configured.

Add a parallel step to a table

To add a step to a table in parallel with another step use the **Add Parallel** tool.

1. Select the step that is to run in parallel with the new step.
2. Select **Add Parallel** . A new UNDEFINED step is created in parallel with the selected step and highlighted in blue. To the left of the step, a parallel icon signifies that the step is a parallel step. One of these dialog boxes opens, based on the recipe level: **Select Phase, Operation Select** or **Unit Procedure Select**.
3. Make the appropriate selection and select **OK**.


STEPS		
1		UNDEFINED:1
2		UNDEFINED:2







Tip: For parallel steps that request the same resource, the FactoryTalk Batch server determines how the resources are allocated through the subarbitration process. If parallel steps require the same resource to occur just after an AND divergence or just before an AND convergence, add null procedures to the recipe in order for batches to run successfully.

Add step before and after parallel

Use the **Insert Step Before Parallel** and **Add Step After Parallel** tools to add steps before and after parallel steps.

1. Select the parallel step that is to run in parallel to the new step.
2. Select **Insert Step Before Parallel** . A new UNDEFINED step is created before the selected parallel step and is highlighted in blue. One of these dialog boxes opens, based on the recipe level: **Select Phase, Operation Select** or **Unit Procedure Select**.
3. Make the appropriate selection and select **OK**.

STEPS		
1		UNDEFINED:3
2		UNDEFINED:1
3		UNDEFINED:2

4. Select the parallel step that runs after the new step.
5. Select **Add Step After Parallel** . A new UNDEFINED step is created after the selected parallel step and highlighted in blue. One of these dialog boxes opens, based on the recipe level: **Select Phase, Operation Select** or **Unit Procedure Select**.

6. Make the appropriate selection and select **OK**.

STEPS		
1	↓	UNDEFINED:3
2	↑↑	UNDEFINED:1
3	↑↓	UNDEFINED:2
4	↓	UNDEFINED:4

Define a step


Newly added steps are UNDEFINED. Only valid lower level recipes are used in the step definition. When a process for a step is selected, that process is assigned to that particular step in the recipe, and the step is marked with the process name. Redefine a step at any time.

1. Double-click a step marked UNDEFINED. One of these dialog boxes opens, based on the recipe level: **Select Phase**, **Operation Select**, or **Unit Procedure Select**.
2. To define a phase or operation either select from the option list and then select **OK**. Or, select **New Phase** or **New Operation** to create a new phase or operation. See **Create operations and unit procedures**.
3. To define a unit procedure:
 - a. From the **Alias** list, select the alias to associate with this unit procedure.
 - b. The **Alias** list contains a list of all units defined in the procedures unit requirements.
- a. If this unit runs in parallel with other units, and if each of the units at this level must acquire in order to initiate this level of the recipe, then select **Acquire unit prior to starting unit procedure**.
 - b. If the units running in parallel can run independently of each other, clear this check box for all of the units at this level. The recipe then runs the individual units acquired, regardless of whether the other units acquire.
- c. Select the appropriate unit procedure to associate with this step. Or, select **New Unit Procedure** to create a new unit procedure.
- d. Select **OK**.

Remove a step from a table

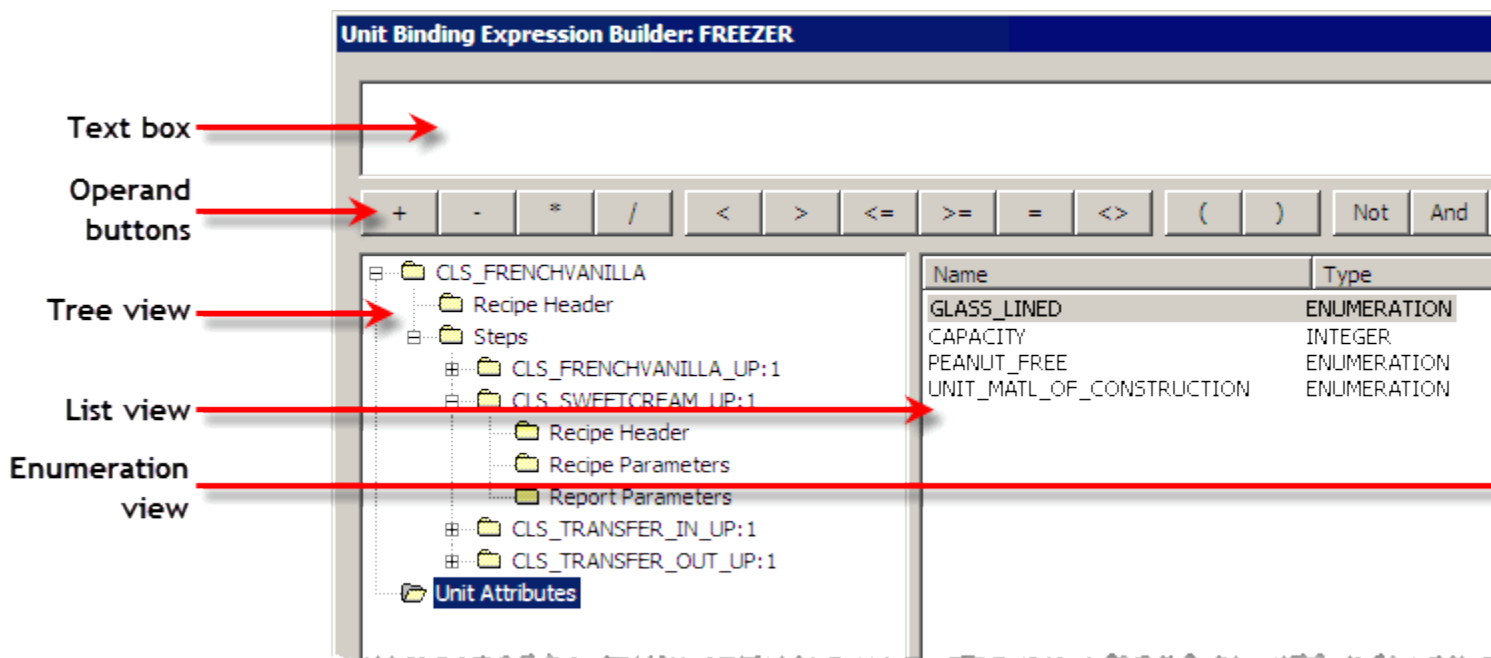
Use these instructions to remove a step from a table.

To remove a step from a table:

1. Select the step to remove.
2. From the **Recipe Construction** toolbox, select **Remove Step**  or press **DELETE**.

Bind Expressions

Use the Unit Binding Expression Builder to build bind requirement and bind preference expressions. If the bind requirement or preference type is Expression, selecting browse next to the **Expression** box displays the **Unit Binding Expression Builder** dialog box. The **Unit Binding Expression Builder** dialog box contains a tree view in the left pane. The tree view shows Recipe Header and Unit Attribute data used to build an expression.



Select **Recipe Header**, a list of recipe header parameters displays in the list view (in the middle). The Recipe Header names map to the same boxes defined within a Recipe Header.

Create or edit an expression by typing directly in the Expression Builder text box, select items in the list view and enumerations view, or select various operators. The operator buttons of the Expression Builder are always enabled. The selected operator is inserted into the expression at the current cursor position and displayed in the Expression Builder text box.

When the expression is complete, selecting **OK** validates and saves the expression and returns to the **Bind Requirements** or **Bind Preferences** dialog box.

There are three types of Bind Expressions: Boolean, Minimize, Maximize. Boolean expressions must evaluate to true or false. Minimize and maximize expressions evaluate to real or integer values.

Expression operators expanded

Parameter expressions support these operators. The precedence of the execution depicts from highest to lowest. An operator with a higher precedence executes before an operator of lower precedence.

Operator	Description
()	Expressions within parentheses are evaluated before expressions outside of parentheses
NOT, -	Logical NOT, negation (of a single argument)
*, /, AND	Multiplication, division, logical AND
+, -, OR	Addition, subtraction, logical OR
<, <=, >, >=	Less than, less than or equal to, greater than, greater than or equal to
=, <>	Equal to, not equal to



Tip: If you want the result of the expression to be an Integer, the values used to build the expression must be Integers – Real is not compatible with an Integer. However, using division in an expression always results in the value being presented as a Real number.

Expression data types

The data types supported are integer, real, string, and enumeration.



Tip: Unsigned data types are not supported.

These are data type examples:

Integer: 423

Real: 423.123456789012

String: The string constant must be in quotes: "READY".

Enumeration: As a string, the enumeration constant must be in quotes: "BUTTER_PECAN". As an integer, the ordinal for the enumeration may be: 4.



Tip: If you want the result of the expression to be an Integer, the values used to build the expression must be Integers – Real is not compatible with an Integer. However, using division in an expression always results in the value being presented as a Real number.

Operands

Operands used within a binding expression can reference Recipe Header data, Unit Attributes, and constants. This table illustrates examples of allowable expression entities and the required syntax.

Expression Entity	Expression Syntax	Example
Recipe Header data	[Recipe Header item]	[VERSION_NUMBER]
Global Unit Attribute	Unit Attribute Name	CAPACITY
Constant		100

Expression validation

Although an invalid expression cannot be created, a previously valid expression may become invalid through subsequent changes elsewhere in the area model. When the **Edit Global Unit Binding Requirements** dialog box is invoked all binding expressions are re-validated. If any current expressions have become invalid due to subsequent changes, an error message opens and the dialog displays an asterisk beside each invalid expression.

The expression validates for consistency and syntactical correctness. If the defined expression validates successfully, the dialog box closes. If the expression is invalid, then the location of where the first error encountered highlights, a description of the error displays in a message box, and the dialog box does not close.

Validation rules:

- An expression must resolve to either TRUE or FALSE and meet the criteria:
 - An expression must contain at least one operator (+ (addition), - (subtraction), * (multiplication), / (division), = (equals), > (greater than), < (less than), <= (less than, or equal to), >= (greater than, or equal to), <> (not equal to), Not, And, and Or.
Tip: OR transitions fire if at least one side of the expression is TRUE.
 - Separate each entity (recipe header, global unit attribute, or constant) with an operator.
 - The operated entities must evaluate to acceptable data types (string with a string; integer with integer or real; real with integer or real).
- Maximum expression length must not exceed 1023 characters.
- Duplicate expressions are allowed.

Parameter types in a recipe

Within a recipe, there are three kinds of parameters:

- Phase parameters
- Operation sequence parameters
- Recipe parameters

Recipe properties

Use the **Procedure Properties** dialog box to configure parameters and reports for recipe procedures and steps. There are two tabs on the **Procedure Properties** dialog box – **Parameters** and **Reports**.

Recipe parameters

Recipe formula parameters provide flexibility when creating recipes. Use recipe parameters in transition conditions or to substitute for lower-level phase or recipe parameters. Create a recipe parameter for Operation, Unit Procedure, or Procedure recipe levels.

Recipe parameter values assign as follows:

- If the operation is the highest level recipe for the batch, operation recipe parameter values assign when the operation step is added to a unit procedure-level recipe or at batch creation.
- If the unit procedure is the highest level recipe for the batch, unit procedure recipe parameter values assign when the unit procedure step is added to a procedure-level recipe or at batch creation.
- Procedure-level recipe parameter values always assign at batch creation because this is the highest possible recipe level for the batch.

A **recipe formula parameter** is a parameter specific to a recipe. A parameter can be used to pass values from one level of a recipe to the next lower level. A recipe formula parameter is configured in the FactoryTalk Batch Recipe Editor on a procedure or operation step. Recipe parameters can be deferred to recipe formula parameters.


See also

[Assign recipe formula parameters](#) on [page 99](#)

Assign recipe formula parameters

Use these instructions to assign recipe formula parameters on the **Procedure Properties** dialog box, **Parameters** tab.

To assign recipe formula parameters

1. Select **Recipe Formula Parameters** .
2. On the **Parameters** tab, select **Add Parameter** to create a new recipe formula parameter. A new parameter row displays and is populated with default values.
3. Type the appropriate information in the boxes.
4. Repeat steps 2 and 3 until all recipe formula parameters are defined.
5. Select **Apply** to save changes and keep the **Properties** dialog box open; select **OK** to save any changes and return to the FactoryTalk Batch Recipe Editor.
6. To delete a recipe formula parameter, select the row to delete and select **Delete**. Select **OK** to save any changes and return to the FactoryTalk Batch Recipe Editor.



Report parameters

Use report parameters in transition conditions or to substitute for lower-level phase or recipe parameters. Create a report parameter for Operation, Unit Procedure, or Procedure recipe levels. Reports on operation, unit procedure, and procedure recipes accumulate data from their child steps to aid report generation activities. Report values are calculated and recorded in the batch event journal when the state of the procedure is Complete, Aborted, or Stopped.

Assign report parameters

Use these instructions to assign report parameters on the **Properties** dialog box **Reports** tab.

Assign report parameters

1. Select **Recipe Formula Parameters** .
2. On the **Reports** tab, select **Add Report** to create a new report parameter. A new row displays.
3. Enter a name for the report.
4. Select the data type from the list in the **Type** column, and enter the appropriate engineering units in the **Enum/EU** column.
If the Type is Enumeration, select an enumeration from the **Enum/EU** list.
5. Select browse  next to the **Report Expression** box to display the **Report Expression Builder** dialog box.
6. Create the desired expression to use as the report value, and then select **OK** to save the expression into the **Report Expression** box.
Repeat steps 2 through 6 until all reports are defined.

7. Select **Apply** to save changes and keep the **Properties** dialog box open; select **OK** to save any changes and return to the FactoryTalk Batch Recipe Editor.
8. To delete a report, select the row to delete and select **Delete**. Select **OK** to save any changes and return to the FactoryTalk Batch Recipe Editor.

Aggregate report values

The report parameters of phases cannot have expressions because a control program running outside the FactoryTalk Batch Server supplies these values. There is one exception—the ability to accumulate the values uploaded by phase logic. The phase logic does not sum integer and real values for reporting purposes.

Phase parameters

When adding recipe phases to an operation-level recipe, specify the formula values for its associated phase parameters. Recall that a recipe phase links to an equipment phase configured in the area model. The recipe phase parameters originate from the parameters defined in the FactoryTalk Batch Equipment Editor for the associated equipment phase. Therefore, the recipe phase formula value parameters are the phase parameters defined for the associated equipment phase in the area model. These parameters pass data directly to the process connected device (PCD).

Configure the phase parameter values in one of four ways:

- Specify them in the FactoryTalk Batch Recipe Editor for the recipe phase at recipe creation time.
- Specify them prior to the execution of the recipe phase in a running recipe after batch creation (done most often with an operator prompt).
- Specify them as a parameter expression.
- Defer them to a recipe parameter, which allows defining at batch creation time.

Operation sequence parameters

When adding an operation sequence to a unit procedure recipe, specify the formula values for its associated operation sequence parameters.

Configure to have values in one of five ways:

- Assign values to operation sequence parameters using the FactoryTalk Batch Recipe Editor.
- Specify them before the phase or operation sequence step runs, by assigning a value from an FactoryTalk Batch user control.
- Specify them when an unacknowledged prompt requests the value for a phase parameter. The **Origin** value for the parameter is **Operator**.
- Specify the value is calculated from a parameter expression. The **Origin** value for the parameter is **Expression**.

- Defer them to a recipe parameter, which allows defining at batch creation time. The **Origin** value for the parameter is **Defer**.

Recipe parameters

Recipe parameters are parameters associated with the recipe as a whole, not a particular phase. Recipe parameters:

- Are created and defined in the FactoryTalk Batch Recipe Editor.
- Are specific to the recipe.
- Define any recipe level: operation, unit procedure, or procedure.

The values of the highest level recipe parameters within a batch are defined at batch creation time.

Recipe parameters are:

- Used in transition conditions.
- Used to assign parameter values to lower level steps in a process called deferring.
 - Phase formula value parameter values may defer to the value of Operation-level recipe parameters.
 - Operation-level recipe parameters may defer to the value of Unit Procedure-level recipe parameters.
 - Unit Procedure-level recipe parameters may be deferred to the value of Procedure-level recipe parameters.



Tip: Recipe parameters of unit procedures and operation steps can also use expressions to reference parent and peer step data (for example, to calculate a value that other parameter expressions reference).

- Procedure-level recipe parameters define at batch creation.

Recipe Formula Parameters	Formula Values
Procedure	Batch creation
Unit Procedure	Procedure
Operation	Unit Procedure
Phase in FactoryTalk Batch Equipment Editor	Operation



Tip: The grayed cells indicate the function cannot perform within the FactoryTalk Batch Recipe Editor.

Defer parameters

Define the values of the highest level recipe parameters within a batch at batch creation time. For example, assume:

- A phase parameters value defers to an operation-level recipe parameter defined for its containing operation-level recipe.

- The operation-level recipe parameters value defers to a unit-procedure-level recipe parameter defined for its containing unit procedure-level recipe.

When creating a unit procedure-level recipe instance, the operator defines the value for the unit-procedures recipe parameter at batch creation time because this is the highest level recipe of the batch. The value the operator defines for the unit-procedure's recipe parameter passes down to the operation-level recipe parameters value, which then passes down to the phase parameters value at batch creation time. When FactoryTalk Batch Server downloads parameters to the process-connected-device before starting the recipe phase, the phase parameters value is sent to the process-connected device.

Deferring phase parameter values to recipe parameters provides a way to define the recipe parameter value at batch creation time rather than waiting for the server to initiate the phase within the recipe after batch creation time. This frees the operator from closely monitoring the running recipe for phase parameter value prompts that would not occur until the Server initiates the phase.

Deferring phase parameters also provides a way for third-party scheduling packages or other applications working through the Server API (such as FactoryTalk Batch View HMI Controls) to create batches and define the values of the corresponding recipe parameters for those batches at batch creation time.

See also

[Recipe parameters](#) on [page 102](#)

Formula values and report limits

Assign formula values by specifying the values and choosing the step parameter displayed within an operation. Report parameters pass phase information to the FactoryTalk Batch server for reports. Typically, this information is a process value. The parameters and report limits that display in the **Formula Value Entry/Report Expression** dialog box are the configured phase parameters and report limits defined in the area model. Configure the step formula values and modify the report limits in the **Formula Value Entry/Report Expression** dialog box.

If control str configured for the phase class, the first item in the **Parameters** values area is named CONTROL_STRATEGY and the **Value** column for this item contains a list of configured control strategies. Only the parameters and reports associated with the selected control strategy display. Select the appropriate control strategy before configuring parameters and report limits. When configuring parameters, the parameter values may be assigned as literal values or deferred to another parameter at a higher level in the recipe.

If the step selected is a material-enabled phase class, the **Formula Value Entry/Report Expression** dialog box also contains the recipe parameters (MATERIAL and AMOUNT) and the report parameters (ACTUAL_AMOUNT and FEED_COMPLETE) associated with material phases. If optional parameters are enabled, it will also contain the optional material recipe parameters CONTAINER, LOT, LABEL, and MATERIAL_CLASS.



Tip: To enable the optional parameters in the FactoryTalk Batch Equipment Editor, double-click the phase class to open the **Edit Phase Class** dialog box. On the **Parameters** tab, make sure the **Add Optional Material Parameters** box is selected.

Material recipe parameters

The values of the MATERIAL, LOT and LABEL parameters are combined to create the material specification. The material specification is used to determine the container from which a material is drawn, or into which a material is distributed, when a batch is run. The material specification always contains a material name. Optionally, the specification can include a lot, a label, or both.

Material recipe parameters:

Parameter Name	Purpose
AMOUNT	Indicates the quantity of the material to be used, or created, in the recipe. A positive number or zero (0), indicates a material addition. A negative number or zero (0), indicates a material distribution. If needed, defer the value assignment for the AMOUNT parameter to a higher recipe level or to the operator.
CONTAINER	Cannot be modified. This parameter is used only at run time. Each time a step is bound to a container, the FactoryTalk Batch Server stores the container's Controller ID (the CONTAINER's enumeration set ordinal value) and string into this parameter. The container associations are defined for each phase in the area model.
LABEL	Indicates a specific subplot of material to be used in the recipe, either as a material addition or a material distribution. If a specific subplot is not needed, leave the value of the parameter blank and the FactoryTalk Batch Server selects a subplot based on the material specified, the unit in which the phase is running, and the phases container associations. When distributing, the value of the LABEL parameter is assigned to the Distributed Sublot.
LOT	Indicates the lot to be used in the recipe either as an addition or a distribution. If the recipe does not require a specific lot, leave the parameter value blank and FactoryTalk Batch Server selects a lot based on the material specified, the unit in which the phase is running, and the phase's container associations. In distributions, the LOT parameter value is assigned to the Distributed Sublot.
MATERIAL	Indicates a specific material to be used in this particular phase. A value must be assigned; If the operator selects from a class of materials at the time the batch is run, select NULL_MATERIAL to enable the MATERIAL CLASS parameter.

MATERIAL CLASS	Indicates a class of materials to use in this particular recipe. Assign a value; If a material class is selected, the operator is prompted to select a specific material from the material class at the time the batch is added to the batch list in FactoryTalk Batch View. If only one specific material is used in this phase, select NULL_CLASS to enable the MATERIAL parameter.
----------------	---



Tip: Materials and material classes are configured in the FactoryTalk Batch Material Editor. Use the same recipe to change back and forth between a material-specific recipe and a material class-based recipe by changing the MATERIAL and MATERIAL_CLASS parameters.

Material report parameters


The FEED_COMPLETE parameter reports that a material addition or distribution is complete.

The ACTUAL_AMOUNT report parameter records the actual quantity of the material produced or consumed.

Formula Value Entry Report

The **Formula Value Entry/Report Limit Entry** dialog box is used to define the phase, operation sequence, or recipe parameters and reports associated with a recipe step. The **Formula Value Entry/Report Limit Entry** dialog box contains:

Item	Description
Parameters value entry	
Name	The formula parameter name associated with this step. (view-only)
Type	The parameter data type as configured in the area model (for phase parameters) or the FactoryTalk Batch Recipe Editor (for recipe parameters). Valid types include Real, Integer, Enumeration, and String. (view-only)

Item	Description
Parameters value entry	
Origin	<p>A single word description of where the value for this parameter is assigned. Valid entries for the Origin box are:</p> <p>Value: This assigns a default fixed value for the parameter (not valid for Timer steps).</p> <p>Operator: Only available for timer, phase, or operation sequence step formula values. Postpones the value assignment to the operator. A prompt is generated in FactoryTalk Batch View when the recipe is run. The operator is prompted for a value when the phase or operation sequence is executing. The operator must enter a value for the parameter for that phase or operation sequence to continue running.</p> <p>Defer: Defers the value assignment to the next higher recipe level. When Defer is selected, a list of appropriate recipe parameters to which this phase or recipe parameter may be deferred is presented in the Value column. This list only includes existing recipe formula parameters that have the same data type as the lower level phase or recipe parameter, and a range less than or equal to the lower level phase or recipe parameter.</p> <p>These material parameters cannot be deferred: MATERIAL, LOT, LABEL, or MATERIAL CLASS.</p> <p>Expression: Allows assignment of an expression result as the value for the parameter. Select browse () to open the Parameter Expression Builder.</p>
Recipe formula parameters must be created if the step formula parameter is deferred.	
Min	The minimum value for this parameter. This is determined when the phase parameter is defined in the area model or the recipe parameter is defined in FactoryTalk Batch Recipe Editor. (view-only)
Value	<p>The value that is to be assigned to this parameter. This is dependent on the selected Origin.</p> <p>For material-enabled phases:</p> <p>MATERIAL – contains a list of materials retrieved from the material database.</p> <p>AMOUNT – dependent on the selected Origin. May be set to a material amount if the Origin box is Value. The sign of this value specifies whether the phase is a material distribution (amount is negative) or a material addition (amount is positive or zero). The default value is zero.</p>
Max	The maximum value for this parameter. This is determined when the phase parameter is defined in the area model or the recipe parameter is defined in FactoryTalk Batch Recipe Editor. (view-only)
Enum/EU	Enumeration/Engineering Units. The engineering units associated with the parameter value. This is assigned when the phase parameter is defined in the area model or the recipe parameter is defined in FactoryTalk Batch Recipe Editor. (view-only)
Display	Indicates display of this parameter value in the step in the SFC. Only one parameter may be displayed per step.
Verification Method	<p>The algorithm that specifies which policy and set of limits verify the parameter value validity. There are four possible verification methods:</p> <ul style="list-style-type: none"> • High-High-High/Low-Low-Low • High-High/Low-Low • High/Low • No Limits <p>The Verification Method is set in FactoryTalk Batch Equipment Editor. (view-only)</p>
LLL	The Low-Low-Low limit value for the parameter (Real, Integer). This is only enabled if the Verification Method is High-High-High/Low-Low-Low.


Item	Description
Parameters value entry	
LL	The Low-Low limit value for the parameter (Real, Integer). This is only enabled if the Verification Method is High-High-High/Low-Low-Low or High-High/Low-Low .
L	The Low limit value for the parameter (Real, Integer). This is disabled if the Verification Method is No Limits .
H	The High limit value for the parameter (Real, Integer). This is disabled if the Verification Method is No Limits .
HH	The High-High limit value for the parameter (Real, Integer). This is only enabled if the Verification Method is High-High-High/Low-Low-Low or High-High/Low-Low .
HHH	The High-High-High limit value for the parameter (Real, Integer). This is only enabled if the Verification Method is High-High-High/Low-Low-Low .
Container Binding area (material-enabled phase only)	Specifies the method used to bind the material-enabled phase to a container. Automatic: The FactoryTalk Batch Server selects the appropriate container based upon the container associations configured in the area model and the material specification. Prompt: The operator is prompted to choose the container and phase to which the phase is to be bound.
Feed Type area (material-enabled phase only)	The selection available is dependent upon the phase parameter values defined in the area model. If the phase is an addition (material is added to a recipe), Addition is selected and cannot be edited. If the phase is a distribution (material is added to inventory), Distribution is selected and cannot be edited. If the phase is configured as both, specify the material feed type associated with this step. The Max and Min values adjust accordingly.
Many aspects of recipe configuration are dependent upon accessing data from the material database using the FactoryTalk Batch Material Server. If the Material Server is not available, the MATERIALS, CONTAINERS, and MATERIAL CLASS enumeration sets are recreated with the NULL_MATERIAL, NULL_CONTAINER, and NULL_CLASS, respectively. reconfigure the materials after the server becomes available.	
Report Limits area	
Name	The parameter name. (view only)
ID	The number, unique to the phase that identifies this parameter. (view only)
Type	The parameter data type. Valid types include: Real, Integer, Enumeration, and String. (view only)
Accumulate	Displays the type of accumulation for the report as defined in the phase class in the area model. (view only)
Verification Method	The type of verification method used for the parameter. This method sets the limits and policies for verifying report values (view-only).
Limit Calculation	The equation type used to calculate the limit. There are three ways to calculate this limit: Absolute: Absolute value for the parameter. Percentage: Target Parameter + (Target Parameter * Percentage) Relative: Target Parameter + Value (view-only)
Target Parameter	The recipe parameter used in calculating the limit when the Limit Calculation is Percent or Relative . (view-only)
LLL	The Low-Low-Low limit for the parameter (Real, Integer). This is only enabled if the Verification Method is High-High-High/Low-Low-Low .

Item	Description
Parameters value entry	
LL	The Low-Low limit for the parameter (Real, Integer). This is only enabled if the Verification Method is High-High-High/Low-Low-Low or High-High/Low-Low .
L	The Low limit for the parameter (Real, Integer). This is disabled if the Verification Method is No Limits .
H	The High limit for the parameter (Real, Integer). This is disabled if the Verification Method is No Limits .
HH	The High-High limit for the parameter (Real, Integer). This is only enabled if the Verification Method is High-High-High/Low-Low-Low or High-High/Low-Low .
HHH	The High-High-High limit for the parameter (Real, Integer). This is only enabled if the Verification Method is High-High-High/Low-Low-Low .
EU	The engineering units associated with the parameter, such as Degrees F or Kilograms . (view only)

Assign values and phase limits

Use these instructions to assign parameter values and phase report limits.

To assign parameter values and phase report limits

1. Select a step in either the **SFC** view or **Table** view.
2. Select **Value Entry**  to display the **Formula Value Entry/Report Expression** dialog box.



Tip: If the step is a phase with configured control strategies, the first item in the **Formula Value Entry/Report Expression** dialog box is named **CONTROL_STRATEGY**, and the **Value** column for this item defaults to the control strategy assigned as the area model default.

3. If the step is a phase for which control strategies are defined, select the appropriate control strategy from the **Value** list in the row named **CONTROL_STRATEGY**. Only parameters and reports associated with the selected control strategy display.
4. For each formula parameter, select the appropriate origin from the **Origin** list. (If the step is a phase or operation sequence, the list includes **Operator**.)
5. In the **Value** box, type a value for each parameter.
6. If the data **Type** is **Enumeration**, select an enumeration from the **Value** list based on the selected origin.



Tip: The list of enumerations displays in alphabetical order and configure in the area model.

7. Select **Display** to see the step formula value for this parameter in the SFC view step (step formula values always display in the **Table** view).
8. To change the default parameter deviation limit values defined for the phase in the area model, edit the limit values for this recipe in the **LLL**,

- LL, L, H, HH, and HHH** boxes. The values entered determine deviation events at run time.
9. If this is a material-enabled phase, select the type of binding in the **Container Binding** area.
 10. If this is a material-enabled phase, and the parameter is configured for addition and distribution, specify the material feed type associated with this step in the **Feed Type** area.
 11. In the **Report Limits** area, enter the report deviation limit values for this recipe in the **LLL, LL, L, H, HH, and HHH** boxes. The values entered determine deviations in a report value at run time.
 12. Select **OK** to return to the FactoryTalk Batch Recipe Editor window.

Assign formula values to Timer

There are three parameters available for a COUNT_DOWN timer and two parameters available for a COUNT_UP timer.

To assign formula values to Timer steps

1. Select a timer step in either the **SFC** view or **Table** view.
2. Select **Value Entry** . The **Parameter Value Entry/Report Limit Entry** dialog box appears.
3. For the HOLD_BEHAVIOR formula parameter, select the appropriate origin from the **Origin** list.
 - The origin of the SETPOINT and HOLD_BEHAVIOR parameters of a COUNT_DOWN Timer can be set to OPERATOR, DEFER, or to the result of an EXPRESSION. When a COUNT_DOWN Timer parameter is OPERATOR, the FactoryTalk Batch Server prompts the operator on start of the Timer step to supply a runtime value for the parameter. When a COUNT_DOWN Timer parameter is DEFER, the FactoryTalk Batch Server looks to the parent procedure (or upper level recipe) for the value as defined on the **Recipe Formula Parameter** list.
 - Select one of three values for HOLD_BEHAVIOR: CONTINUE, RETENTIVE, and RESET. The default value is CONTINUE.
 - The origin of the HOLD_BEHAVIOR parameter of a COUNT_UP Timer can be OPERATOR, DEFER, or EXPRESSION. When a COUNT_UP Timer parameter is OPERATOR, the FactoryTalk Batch Server prompts the operator on start of the COUNT_UP Timer step to supply a runtime value for the parameter. When a COUNT_UP Timer parameter is set to DEFER, the FactoryTalk Batch Server looks to the recipe parameter to supply the value for the step parameter.

- When the **Origin** is VALUE, at runtime, the value for the parameter comes from the value specified in the **Value** column.
4. The second parameter displayed is SETPOINT. The SETPOINT determines how long the timer step runs. The default value for the SETPOINT parameter is the default value configured for the SETPOINT parameter of the TIMER system phase. The SETPOINT parameter is of the type **Real**.
 5. The third parameter displayed is TIMER_TYPE. It is set to the enumeration value of COUNT_UP or COUNT_DOWN. If all the boxes of this parameter disable; default values cannot change.
 6. Select the **Display** check box to display the SETPOINT, TIMER_TYPE, or HOLD_BEHAVIOR parameter values on a timer step.
 7. Select **OK** to return to the FactoryTalk Batch Recipe Editor window.

Parameter expressions and reports

A parameter expression is an arithmetic expression assigned to a recipe parameter or report parameter of a recipe or a step. Assign a parameter expression to all levels of the procedural control hierarchy (phase, operation, unit procedure, batch procedure) where it is appropriate.

Parameters provide a means to perform calculations in the structure of the recipe. The use of parameter expressions avoids complex and awkward downloading and uploading of parameter data so the control can perform needed mathematical operations. A key application is the ability to aggregate report (output) parameter data into key process indicators (KPIs) at the appropriate level of the recipe. A secondary application is to enable flexibility in calculating recipe (input) parameters.

The terms of the expression may reference several different sources of data. Recipe and report parameters behave differently because their expected roles are different. A recipe parameter supplies data to the process. A report parameter reports a result of the process. The two kinds of parameters evaluate and the result is stored as the value of the parameter at different times.

A parameter expression contains one or more terms with arithmetic operations connecting them. A term may be a constant value, a data reference, or another expression. For recipe parameters, the data references are subscribed. As the data sources change their values, the expression is notified. The expression evaluates the value and stores the new value for the recipe parameter.

For report parameters, evaluation of the expression does not update as references change, but as the expression triggers. Conceptually, report parameters intend to record data resulting from the execution of the process the step represents. This data typically comes from below the recipes level of the recipe hierarchy (child data). Or the data references the current level (peer data or data on the same step) and may include recipe header data, equipment data, and runtime data.

Configure report parameter expressions on operation, unit procedure, and batch procedure recipes. Expressions on phase report parameters are not supported.

Evaluate parameter expressions

At runtime, parameter expressions constantly evaluate. Subscriptions create to each data reference. As these values update, the expression evaluates and updates the value of its parameter. To avoid overwhelming the event journal with frivolous events, not every change in value of an input expression records in the event journal.

It is possible for an expression to evaluate to a value outside the range limits of the recipe parameter. When this happens, the parameter assigns the value `BAD_VALUE`. There are several ramifications for a parameter with the `BAD_VALUE` assignment:

- A download request to a controller will cause the requesting equipment phase to be HELD.
- A transition expression referencing the `BAD_VALUE` parameter will itself evaluate to `BAD_VALUE`, which will prevent the transition from firing.
- A binding expression referencing the `BAD_VALUE` parameter is unable to find binding candidates. Binding is pending and the recipes execution does not advance.
- An input parameter expression referencing a parameter with a `BAD_VALUE` will evaluate its expression to `BAD_VALUE`.

Validate parameter expressions

When a parameter expression evaluates, the result compares to the verification policy (Hi/Low, Hi-Hi/Low-Low, or Hi-Hi-Hi/Low-Low-Low). If the value falls outside the expected range, the appropriate verification policy action triggers. A value of `BAD VALUE` does not trigger a verification policy.



Tip: Parameter validation limits, which can result in the invocation of an electronic signature, only support phases.

Parameter expression override

The value of a parameter having an expression cannot change unlike parameters having an origin of `VALUE`. FactoryTalk Batch View and FactoryTalk Batch View HMI Controls provide the ability to override the expression and assign a value when responding to extraordinary circumstances. The new displayed value is marked as using the Override function.

Report parameter expressions

At runtime, report parameter expressions differ from recipe parameter expressions when evaluated. Recipe parameter expressions update continuously. Report parameter expressions are configured to update on triggers. The report values are uploaded to the FactoryTalk Batch Server based on the configured report parameter to upload on terminal state, or if an upload is requested by the phase logic.

There are two scenarios for phases:

- One, the phase step transitions to a terminal state and the values automatically upload.

- Two, an upload is requested by the phase logic (2XXX and 12XXX upload commands). So the upload is made on demand. Thus, report parameter expressions update when their step transitions to a terminal state (COMPLETE, STOPPED, and ABORTED).

For operation sequences, report parameter expressions update when their step transitions to a terminal state (COMPLETE, STOPPED, and ABORTED).

Override a report expression

The operator under any circumstances may not change calculated report parameters. This corrupts the record of the execution of a procedure. There are two exceptions where the value of a report expression may be changed:

1. A transition expression references the report parameter and its value is preventing the transition from firing. In this case, use the Force Transition function to force the transition to fire.
2. A recipe parameter is dependent on a report parameter to provide its value. If the report parameter is incorrect or undesirable, there is cause to change it. In this case, use the Override function to change the recipe parameter's value.

Expression value configurations

A parameter expression is an arithmetic expression that may be assigned to an input parameter or output parameter of a phase, operation, or unit procedure step that can reference other parameters and recipe header data within a recipe. The expression is evaluated and the result stored as the value of the parameter.

A recipe parameter expression may also reference recipe/formula parameters of the parent operation recipe, data in the operations recipe header, the phases own recipe parameters, and the phases own report parameters. FactoryTalk Batch Server evaluates the expression and then stores the result in the phase steps parameter.

For example, in a control system, the quantity of sugar added is a function of the quantity of water added by another phase. Configure a recipe parameter referencing the ADD_WATER report parameter of another phase and use an expression to calculate the amount of sugar added as the ADD_SUGAR formula parameter on this phase.

Configure expression values

Use these instructions to configure expression values.

Make sure:


- The data types of the referenced parameters in the expression are compatible.
- The data types real, integer, enumeration, and string are supported (no string manipulation is allowed).



Tip: Unsigned data types are not supported.

- The expression references phase parameters and report parameters that are within the open active step.

To configure expression values in recipe and report parameters:

1. Select a step in either the **SFC** view or **Table** view.
2. Select **Value Entry** to display the **Parameter Value Entry/Report Limit Entry** dialog box.
3. Select a recipe parameter or report.
4. Select **Expression** from the **Origin** list, and then select browse  in the **Value** column. (If the step is a phase, the list will include **Operator**.) The **Parameter Expression Builder** opens.
5. From the tree view in the left pane, select the step containing the recipe or report parameter.
6. From the right pane, double-click the recipe or report parameter used for the expression to copy it to the text box (or select it and select **Paste**).
7. Select an operator or function button to enter the desired operation or math function into the expression. Type directly into the **Expression** text box.

For this example, a value of 2 times the amount of milk added is used to determine the amount of sugar to add:



Tip: The maximum length of a parameter expression is 1023 characters.

8. Select **OK** to save and validate the expression and to return to the **Parameter Value Entry/Report Limit Entry** dialog box. The report parameter expression displays.
9. Select **OK** to close the **Parameter Value Entry/Report Limit Entry** dialog box and validate the expression.

In a running batch, the parameter expression continuously evaluates. In the example used in these instructions, the calculation and resulting value is the amount for the ADD_SUGAR phase. This amount would be 2 times the water added (50 KG) for a total amount of 100 KG of sugar.

Parameter expression operators

Parameter expressions support these operators. The precedence of the execution depicts from highest to lowest. An operator with a higher precedence executes before an operator of lower precedence.

Operator	Description
()	Expressions within parentheses are evaluated before expressions outside of parentheses
*, /	Multiplication, division
+, -	Addition, subtraction



Tip: If you want the result of the expression to be an Integer, the values used to build the expression must be Integers – Real is not compatible with an Integer. However, using division in an expression always results in the value being presented as a Real number.

Parameter expression functions

Functions determine how the expression parser handles Real and Integer data types used in a parameter expression. This table lists available functions and their behavior on positive and negative values:

Function	Description	Behavior: Value	Behavior: Result
RND()	Round – Numeric values round to the nearest integer.	6.7 6.5 6.3 -6.7 -6.5 -6.3	7 7 6 -7 -6 -6
RDUP()	Round up – Numeric values round to the next larger integer.	6.7 6.5 6.3 -6.7 -6.5 -6.3	7 7 7 -6 -6 -6
TRNC()	Truncate – Retains only the integer portion of the numeric value.	6.7 6.5 6.3 -6.7 -6.5 -6.3	6 6 6 -6 -6 -6
ABS()	Absolute – Numeric values are positive values. For example, if the Real or Integer is 6.7, there is no effect. If the Real or Integer is -6.7, it multiplied by -1 and is 6.7.	6 6.7 -6 -6.7	6.7 6 6.7
MOD()	Modulo – Returns the modulo, or remainder, of a division. (Integers only)	7 MOD 6	1

Phase link groups

Phase link groups identify phases that may communicate and work together for these reasons:

- **Synchronization** – For example, one phase running logic will not proceed until another phase is at a certain point within its own phase logic.
- **Permissives** – For example, one phase must pass a certain point in its phase logic before its partners can begin their phase logic.
- **Data Transfer** – For example, moving data from one phase to another.

Phases that need to communicate with each other are **message partners**. Message partners organize into link groups in the FactoryTalk Batch Recipe Editor. When a recipe containing message partner phases is on the Batch List, the link group definition informs the FactoryTalk Batch Server which phases need to communicate with each other.

There are three programs involved in implementing message partners:

- **FactoryTalk Batch Equipment Editor** – defines the number of message partners for a phase class within the Area Model.
- **FactoryTalk Batch Recipe Editor** – defines the grouping of message partner phases into link groups.
- **Phase Logic** – where the actual communication between the phases occurs.



Tip: Before defining phase link groups, define the corresponding message partners in the area model.

A phase link group must contain the same number of phases as there are message partners plus one (the phase itself), configured in the area model for each of the phases that are included in the group. For example, if each phase in a phase link group is configured with three message partners, then there must be a total of four phases in the phase link group (each phase has three message partners plus itself). Message partners are configured in the FactoryTalk Batch Equipment Editor.

Message partner phases

When a recipe containing message partner phases is on the Batch List, the link group definition informs the FactoryTalk Batch Server which phases need to communicate with each other.

There are three programs involved in implementing message partners:

- **FactoryTalk Batch Equipment Editor** – defines the number of message partners for a phase class within the Area Model.
- **FactoryTalk Batch Recipe Editor** – defines the grouping of message partner phases into link groups.
- **Phase Logic** – where the actual communication between the phases occurs.



Tip: Before defining phase link groups, define the corresponding message partners in the area model.

A phase link group must contain the same number of phases as there are message partners plus one (the phase itself), configured in the area model for each of the phases that are included in the group. For example, if each phase in a phase link group is configured with three message partners, then there must be a total of four phases in the phase link group (each phase has three message partners plus itself). Configure message partners in the FactoryTalk Batch Equipment Editor.

Message partner application

A common message partner application is with a Transfer-Out phase on one unit and a Transfer-In phase on another unit running in parallel within a Procedure-level recipe.

When the Transfer-Out phase on a unit is ready to send its material to the next unit, the Transfer-Out phase's running logic sets its `_RQ` value equal to 5501 (where 01 is the message number). The Transfer-Out phase's running logic, if coded to do so, halts and waits for the Transfer-In phase to set its `_RQ` value. When the Transfer-In phase is ready to receive the material from the previous unit, the Transfer-In phase's running logic sets its `_RQ` value equal to 5201 (where 01 is the message number) and, if coded to do so, halts and waits.

The FactoryTalk Batch Server then coordinates the messaging when it sees that both message partners have either a 55xx or a 52xx value in their respective `_RQ` memory registers and their xx value (the message number) is the same. The FactoryTalk Batch Server then commands the PLI to reset both message partners' `_RQ` values to 0. Once both message partners see their `_RQ` registers reset to 0, their running logic (which had halted while they were waiting for their respective message partner) then continues to execute.

The sequence of events is as follows:

- **T1:** Transfer-Out phase is ready; sets `_RQ = 5501` and halts its running logic.
- **T2:** Transfer-In phase is ready; sets `_RQ = 5201` and halts its running logic.
- **T3:** FactoryTalk Batch Server commands the PLI to reset both `_RQ` registers to 0.
- **T4:** PLI resets both `_RQ` registers to 0.
- **T5:** All running logic resumes execution.

Link group rules

The rules involving link groups are as follows:

- Up to 200 link groups per recipe.
- Up to 20 phases per link group.
- Define all phases in a link group with the same number of message partners in the area model's phase class definition.
- The total number of phases within a link group must be equal to the number of configured message partners plus one.
- Assign each phase instance to only one link group per recipe.
- Phases in phase link groups must have unique names. If the phase names are not unique, it will result in missing message partners in the linked groups.

Message partners must run in parallel within a recipe and the phase logic must run in parallel so the phases can synchronize.



Tip: When using phase link groups in conjunction with control strategies, the number of phase link groups cannot vary between control strategies configured for a single phase.

For example, if a phase configures with two control strategies, CONTROL_STRATEGY_1 and CONTROL_STRATEGY_2, and the phase has two configured phase link groups, then both control strategies must also have two phase link groups. CONTROL_STRATEGY_1 cannot have a different number of phase link groups than CONTROL_STRATEGY_2.

Phase communication

The actual communication between the phases is programmed in the phase logic using the `_RQ = 5Xxx` request category. Commonly implemented between two individual phases using `55xx` and `52xx` where:

- **55xx** means Request to Wait for a message from another phase.
- **52xx** means Send Message and Wait for One Receiver.
- **xx** means the message number from 00 to 99. All message partners must utilize the exact same message number.

One phase is responsible for sending a message to its partners at the appropriate time. The synchronization occurs when one phase in the group sends the message and the appropriate number of phase partners wait for the message.

Create a phase link group

Use these instructions to create a phase link group.

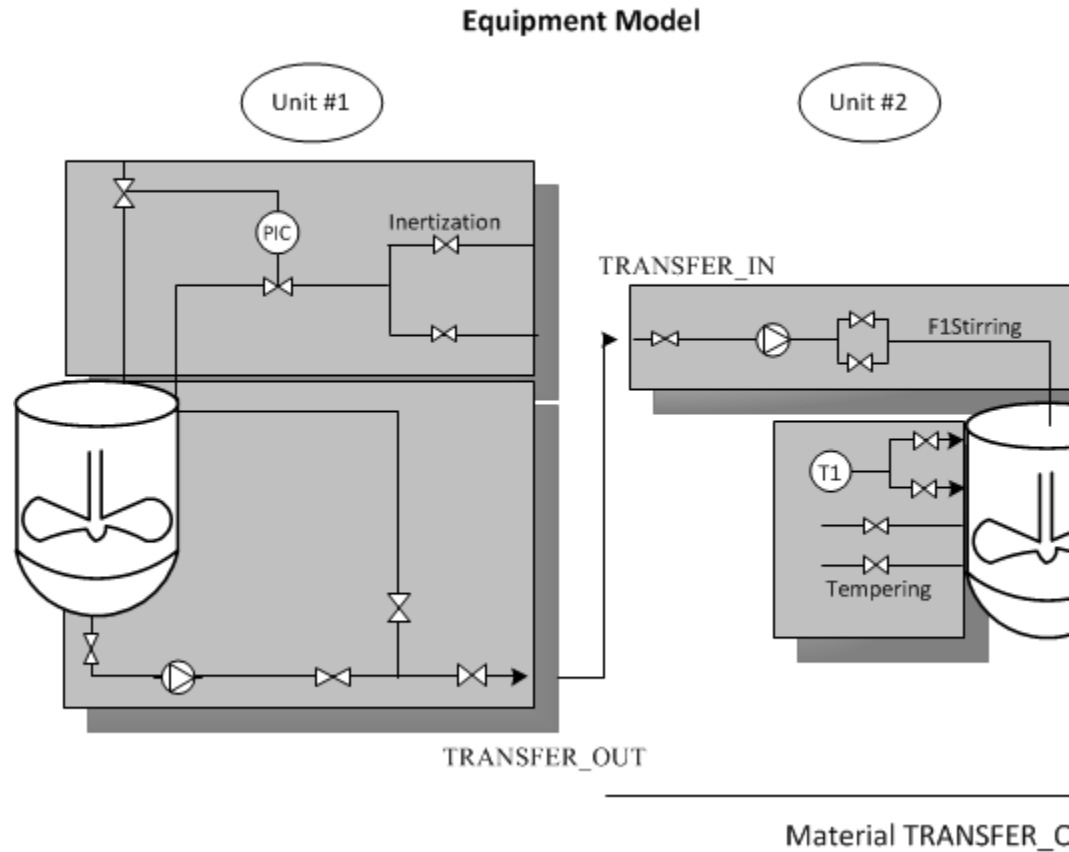


Tip: A phase may only be contained in one phase link group.

To create a phase link group:

1. From the **Link** menu, select **Link Groups**.
2. Select the appropriate group column heading. The cells in the column highlight.
3. Select the appropriate phase in the **Procedure View** pane, or select a phase step in the **Recipe Construction** pane.

4. Select **Add**. The selected phase is added to the link group.
5. Repeat steps 3 and 4 until all phases have been added to the group.
6. Select **OK** to save changes.



Delete a phase link group

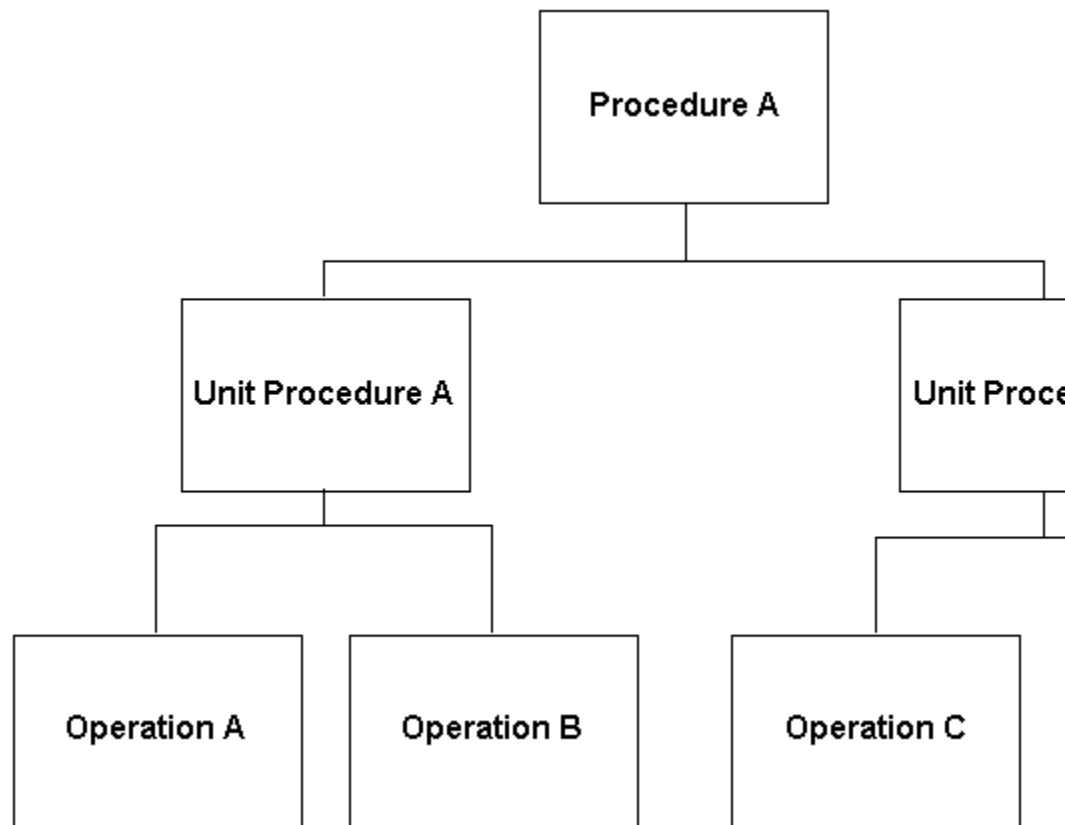
Follow these instructions to remove a phase link group.

To remove a phase link group:

1. From the **Link** menu, select **Link Groups** to display the **Phase Link Group** dialog box.
2. Select the column header of the group to delete. The cells in the column are highlighted.
3. Select **Delete**. The selected group is removed.



Tip: Phase link groups can exist at any level of a recipe, but only the root level link groups can be seen or edited.



All of these recipe elements can contain phase link groups. To edit the phase link group contained in Operation A, open Operation A by itself.

Recipe approval process overview

Use the recipe approval process to validate the development and maintenance of batch recipes. Signature certification allows the recipe approval process to safeguard the design workflow in a formalized manner. This ensures validation of each recipe by authorized personnel before released to production, or released as a component within a larger recipe.

Configure and enable Recipe Approval in the FactoryTalk Batch Equipment Editor, and executed in the FactoryTalk Batch Recipe Editor.

In addition to the Primary Approval process with up to six approval steps (three optional), a two-step Expedited Approval process is available.

Use the Expedited Approval process instead of the Primary Approval process, for example in the early stages of recipe design. A recipe can go through initial review using the expedited approval process. After the recipe is deemed ready, the Expedited approval process can be reverted to its starting point, and then the Primary approval process can be used to validate it for release as a step (procedure or operation) used in a larger recipe, or to release it to production.

Revert option

Both primary and expedited processes provide a **Revert** option to allow forward and reverse progress through the recipe approval process.

- Revert into the initial Not Started state. The Not Started state allows restart of the recipe approval process, obtain area model recipe approval process changes, or switch between the primary and expedited approval processes.
- Revert returns to the beginning of any incomplete approval step and/or to any previous approval step without restarting the entire approval process.
- The revert process operates the same for both the primary approval process and the expedited approval process.

Release Recipe to Production

With Recipe Approval enabled, **Release Recipe to Production** is a recipe approval step, and must be approved to allow the recipe to generate production batches. Similarly, with Recipe Approval enabled, **Release Recipe**

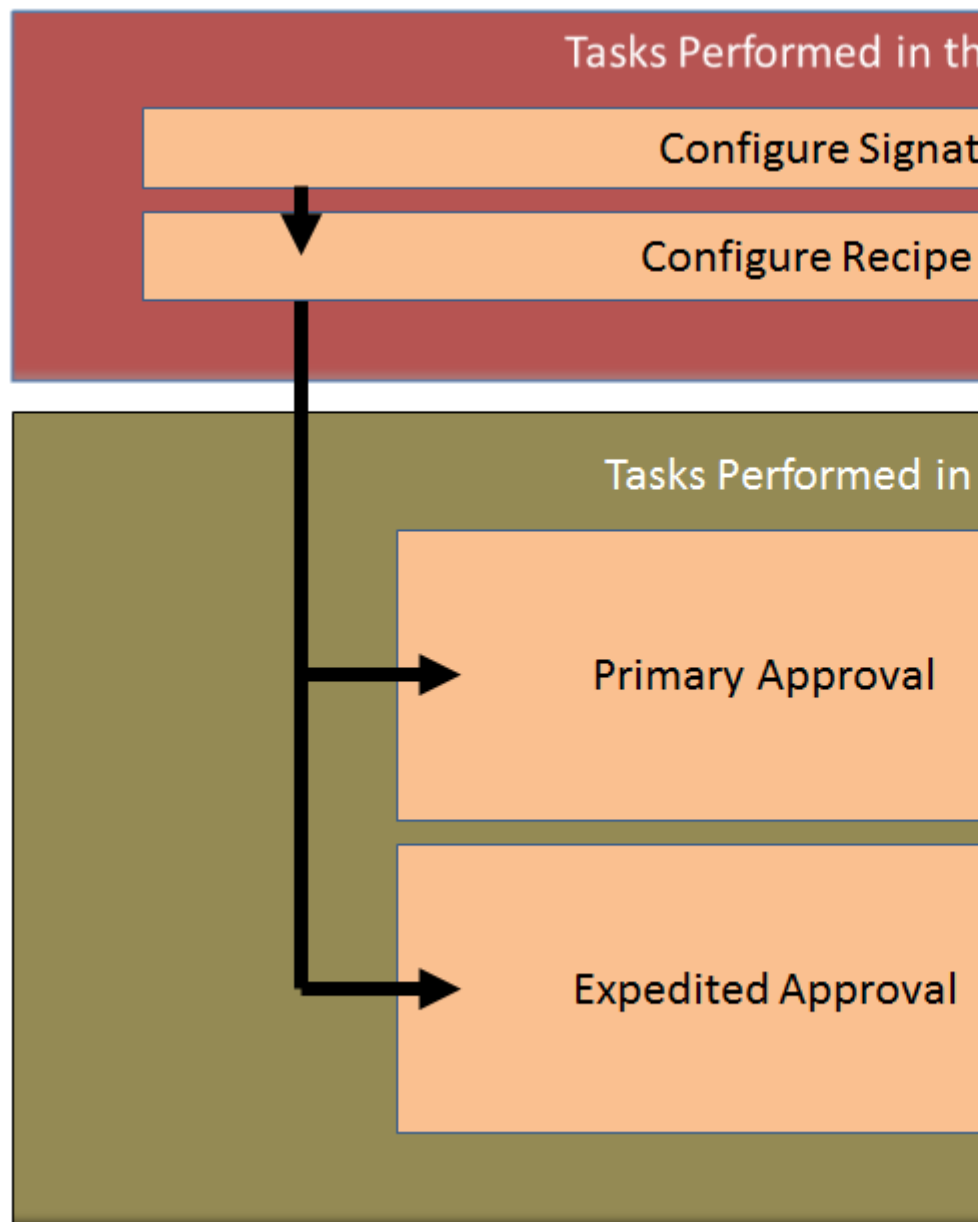
as Step is a recipe approval step, and must be approved to allow the recipe to be incorporated into other recipes.

With Recipe Approval **disabled**, **Release Recipe to Production** is a recipe property and must be checked (set to true) to allow the recipe to generate production batches. The same is true for **Release Recipe as Step**--when this recipe property is set to true, the recipe can be incorporated into other recipes.

When Recipe Approval changes from disabled to enabled in the area model, a recipe property set to true becomes an approved step, with Recipe Editor using the **System** approver.

Configure recipe approval

This chart shows the sequence of tasks in the setup, configuration, and execution of recipe approval.



Configure the signature templates used in the Recipe Approval process within the FactoryTalk Batch Equipment Editor. Approval steps and their signoffs are also configured in the FactoryTalk Batch Equipment Editor. The process configuration becomes part of the specific Area Model used by the recipe elements. Manage the Approval process, primary or expedited, within the FactoryTalk Batch Recipe Editor.


Approve a recipe



Use these instructions to approve a recipe.

Before you begin:

- In the FactoryTalk Batch Equipment Editor:
 1. Enable and configure the Recipe Approval process.
 2. Create the required signature templates in the appropriate area model.
 3. Save the area model.

To approve a recipe:

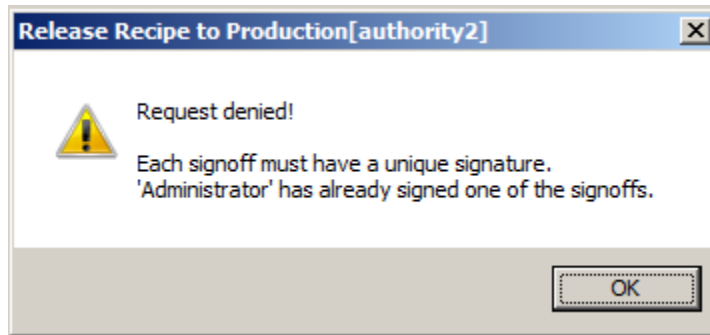
1. Open the FactoryTalk Batch Recipe Editor. If prompted to translate the recipes, select **Yes**.
 - a. (optional) Select **Verify All Recipes**.
 - b. Select **Accept** when the verification has finished.
 - c. If **Auditing** is enabled, add any desired audit comments. Select **OK** to close the **Audit Comments** dialog box.
 - d. Select **Close** to close the **Verification Process** dialog box.
2. From the **File** menu, select **Open Top Level**.
3. From the **Recipe Name** list, double-click the recipe to open, or select the appropriate recipe and select **Open**. The selected recipe opens in the **Procedure View** pane.
4. From the **Recipe** menu, select either **Primary Approval** or **Expedited Approval** depending on the current approval state of the recipe. If an approval process has not started, either process is available.
5. Depending on which process chosen, the **Primary Approval** or **Expedited Approval** dialog box opens to the **Approve** tab, showing the next step available for approval (marked with a pencil  icon). If all step approvals are completed, the dialog box opens to the **Summary** tab. Signature credentials are entered in the area below the signoff step list.
 - In the **User** box, type the name of a user with the required permissions for the signoff (configured in the FactoryTalk Batch Equipment Editor under the **Edit** menu).
 - Enter the correct password in the **Password** box.

- Enter a **Comment** (mandatory or optional, as configured in the area model).
- If a signoff is required to be last in a series of signoffs for an approval step, it is marked with a checkered flag  icon.
- If a FactoryTalk Security Permission is invalid, it is preceded by a warning icon  in the **Security Permission(s)** column. Hover over an invalid permission to show any valid permissions for that signoff.

6. Select **Approve**.

Continue with the approval process for all steps to approve. If a different user is required to sign off on an approval step, save and close the recipe after any approval step. The recipe can then be opened by the next authorized signer.

When a signature requires multiple signoffs, each signoff must be made using a unique UserID and password. Reuse of the same UserID and password results in this warning:



Be sure to provide a unique UserID and password (different from the UserID quoted in the warning message) and continue with the signoff.



Tips:

- With the completion of the first signoff on an approval step, the recipe becomes read-only. To make the recipe editable, revert all approvals back to the *not started* state.
- Any time a recipe is copied using **Save As**, the **Release Recipe as Step** and **Release Recipe to Production** states are cleared in the copy.
- Approval steps and their status are included when a recipe is exported, and retained during import.

Revert a recipe approval

Any completed approval step can be reverted. When a step is reverted, all completed approval steps that follow it in the approval process are also reverted to their Not Started state. Reverting the first formal step has the effect of reverting an entire recipe approval back to its Not Started state. To revert an approved step:

To revert a recipe approval:

1. Open the desired recipe in the FactoryTalk Batch Recipe Editor.
2. From the **Recipe** menu, select either Primary Approval or Expedited Approval, determined by which process was used to approve the recipe.

of the desired approval process, and continue the approval process. Reverting back to the first step provides the option of switching between approval processes (primary to expedited, or vice versa).



Tip: Changing the recipe approval process from enabled to disabled in the area model has the effect of removing all signoffs from the current recipe set.

When Recipe Approval is disabled in the area model, and recipes in the current working set have existing signoffs, the **Inconsistency between Area Model and Recipes** dialog box opens.

- **Remove** opens the recipe set with all approval step signoffs removed.
- **Exit** closes the FactoryTalk Batch Recipe Editor and retains existing signoffs. Return to the FactoryTalk Batch Equipment Editor, re-enable recipe approvals, and continue with the recipe approval process in the FactoryTalk Batch Recipe Editor.

Automatic system signoff

If Recipe Approvals are enabled, FactoryTalk Batch Recipe Editor automatically creates two Expedited Approvals process steps after FactoryTalk Batch is upgraded: **Release Recipe to Production** and **Release Recipe as Step**.

The FactoryTalk Batch Recipe Editor creates the **Release Recipe to Production** step when:

- The recipe was migrated from a previous FactoryTalk Batch version that did not support recipe approvals.
- Recipe Approvals are enabled in the area model.

If the **Release to Production** recipe property in the recipe was previously set to true, then the **\$System** signoff is applied to approve the step.

Similarly, **Release Recipe as Step** is added (always with approval signoff **\$System**) when:

- The recipe was migrated from a previous FactoryTalk Batch version that did not support recipe approvals.
- Recipe Approvals are enabled in the area model.

Check if the \$System signoff has been applied to a recipe in the FactoryTalk Batch Recipe Editor. Open the **Recipe > Expedited Process** dialog box, then select the **Summary** tab. Look under the **Approver** column:

The screenshot shows a dialog box titled "Expedited Approval - CLS_FRENCHVANILLA" with three tabs: "Approve", "Revert", and "Summary". The "Summary" tab is active, displaying a table with the following data:

#	Approval Step	Approve State	Signoff Meaning	Approver
1	Release Recipe as Step	Completed		\$System
				\$System
1	Release Recipe to Production	Completed		\$System
				\$System

Release Recipe as Step and **Release Recipe to Production** are managed differently depending on whether Recipe Approvals are enabled or not:

- When Recipe Approvals are **enabled**, **Release Recipe as Step** is a **recipe approval step** and is approved as part of a formal or expedited approval process. When approved, the recipe can be used within other recipes.
- With Recipe Approvals **disabled**, **Release Recipe as Step** is a **recipe property**, and set to true or false by the user in the **Recipe > Header Data** dialog box. When true, the recipe can be used within other recipes.
- When Recipe Approvals are **enabled**, **Release Recipe to Production** is a **recipe approval step** and is approved as part of a formal or expedited approval process. When approved, the recipe is placed on the Recipe List and the recipe can be used to generate production batches.
- With Recipe Approvals **disabled**, **Release Recipe to Production** is a **recipe property**, and set to true or false by the user in the **Recipe > Header Data** dialog box. When true, the recipe is placed on the Recipe List and the recipe can be used to generate production batches.

Recipe versioning overview

A versioned recipe is a saved, read-only snapshot of the recipe taken at a particular point in time. Recipe versioning is useful when an author, or a team of authors, need to store and protect unique versions of the recipe at chosen development milestones.

Recipe versioning starts with a new or existing recipe before it has had a version of it created. When the recipe has reached a development state the recipe author wants to protect, the **Check In** command creates the first instance of a **versioned** recipe. At that point, the recipe version is saved, set to read-only, and can no longer be edited.

Using the **Check Out** command, an editable work-in-progress (WIP) copy of a versioned recipe can be created. Typically, a WIP recipe undergoes further development before being checked in as the next version of the recipe. Subsequently, a new WIP copy can be made and used in the next iteration of recipe development.

If a versioned recipe fails verification, it is marked by the system as **obsoleted**. Verification can fail due to modifications to the recipe's underlying area model, or if the recipe references other missing or obsoleted recipes. An obsoleted recipe cannot be revised in order to pass verification. A WIP copy of an obsoleted version can be created and then modified to pass verification.

Below the **Store Recipes Using** area is the **Enable Recipe Versioning** check box. Check this box to enable Recipe Versioning, a system-enforced naming convention that stores and protects recipe revisions. By default the box is unchecked and recipe versioning is disabled.

IMPORTANT FactoryTalk Full Edit access to FactoryTalk Batch Equipment Editor is required to enable and disable recipe versioning.

Recipe versioning

How recipe versions are named

Recipe versions are maintained through a naming convention enforced by the FactoryTalk Batch Recipe Editor.

Recipe version naming has three components

- **Basename** is the unique name of a recipe. The recipe author gives the recipe this name. The basename identifies all related versions and work-in-progress (WIP) copies.
- **Version name** is the name assigned by the FactoryTalk Batch Recipe Editor to a versioned recipe created from a new, existing, or WIP

recipe. The versioned recipe's name includes the basename and an appended version number (for example **~V1**).

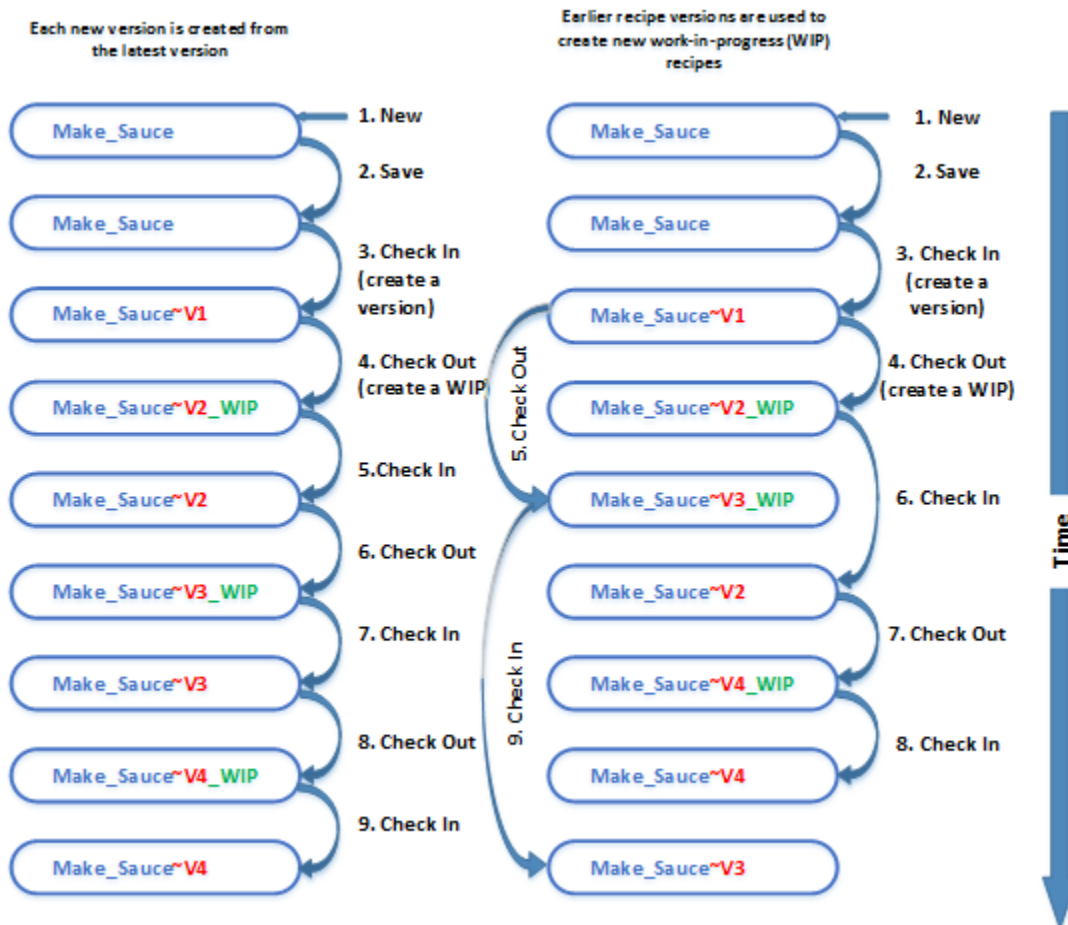
- **WIP name** is the name assigned by the FactoryTalk Batch Recipe Editor to a WIP copy of a recipe. The name includes the basename, the version number of the copied versioned recipe, and an appended **_WIP**, indicating it is a work-in-progress recipe. The FactoryTalk Batch Recipe Editor increments the version number for a WIP copy to the next unassigned version number.

This illustration includes two examples of how recipe version names are derived. In both examples, the recipe basename is **Make_Sauce**. The first version saved is **Make_Sauce~V1**, and the first work-in-progress copy of that version is **Make_Sauce~V2_WIP**.



Tip: When a WIP copy is made using the **Check Out** command, FactoryTalk Batch Recipe Editor looks ahead to when the WIP is saved as a version. For example, the WIP copy of **Make_Sauce~V1** is saved as the second version **Make_Sauce~V2**.

The example on the left shows a straightforward, linear progression from V1 to V4 of the Make_Sauce recipe. The right shows the WIP numbering method, where earlier versions are used as the basis for creating the next version. FactoryTalk Batch Recipe Editor manages version naming to avoid duplicates of version numbers and WIP copy names.



Restrictions for recipe versions

When using recipe versioning, observe these restrictions:

- A versioned recipe can only reference other versioned recipes or steps.
- Once a recipe version, or work-in-progress recipe (WIP), is created, its basename cannot be changed afterwards, even if recipe version control is disabled.
- A new recipe's basename must be unique; it cannot be the same as an existing recipe basename or existing recipe name (in the current working set).
- A recipe's basename cannot contain any of the naming conventions used by the system, namely ~V or _WIP. These strings are reserved for use by the FactoryTalk Batch Recipe Editor.
- Prior to creating a versioned recipe, the recipe is automatically verified by the FactoryTalk Batch Recipe Editor; it must pass this verification without any errors.
- If a recipe makes reference to missing or obsoleted recipes, it will not pass verification, and is marked by the system as obsoleted.
- Import of versioned recipes can result in conflicts, both with version numbering and version naming.
- A recipe formulation is versioned when the master recipe is versioned, and follows the same versioning behavior as the master recipe:
 - The formulation cannot be created, modified, or deleted from a versioned recipe. Modifications include changing the formulation name, formulation description, or formulation parameter values.
 - The formulation can be created, modified, or deleted from a checked out WIP recipe.
 - If the formulation in a WIP recipe is invalid, the formulation cannot be checked in until the issue is resolved. However, the WIP can be saved.



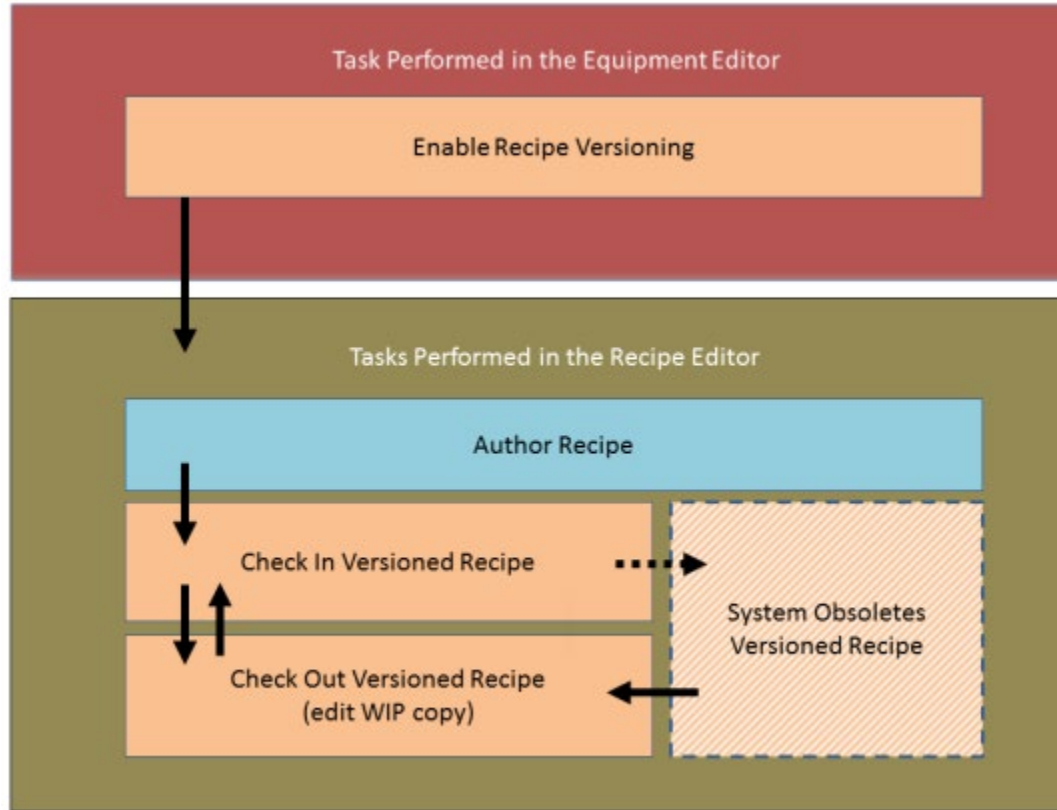
Tip: Recipe versioning and the recipe approval process are independent of each other. Recipe steps can be approved or reverted without being considered as edits to a versioned recipe.

Recipe version control works with all recipe storage formats (BINARY, XML, and RDB).

Enable recipe versioning option

Configure Recipe Versioning within the FactoryTalk Batch Equipment Editor. Area Model authors must have FactoryTalk Full Edit access to FactoryTalk Batch Equipment Editor to enable and disable versioning.

The FactoryTalk Batch Recipe Editor handles enforcement of versioning naming. This flow chart shows the sequence of tasks to enable and use recipe versioning.



Disable recipe versioning option

When recipe versioning is disabled, previously versioned recipes remain read-only and have all header boxes disabled. It is still prohibited to name a new recipe in conflict with any existing basename, in order to avoid name collisions if recipe versioning is re-enabled.

Rename a previously versioned recipe by using the **Save As** command to create a new recipe. Edit and save a previous work-in-progress (WIP) recipe--use **Save As** and provide a new recipe name to disassociate the recipe from its former basename.

Create a recipe version

Prior to starting this procedure, ensure that recipe versioning is enabled in the **Server Options** dialog box in the FactoryTalk Batch Equipment Editor.

To create a recipe version:


1. Create a new recipe using any of these options:
 - Use the **File** menu **New** command.
 - Use the **Save As** command.
 - Create a new recipe as part of adding a step to an open recipe.



- Be sure to specify a unique basename when creating the new recipe. Observe naming restrictions.
2. Edit the recipe, and then **Save** the recipe.
 3. Use the **Check In** command to create the first version.
 - To edit this version of the recipe, use the **Check Out** command to create an editable copy and make the necessary changes.
 - When finished editing, use the **Check In** command again to create the next version of the recipe.
 4. Use the **Check Out** command to create an editable copy of this version.

This process can be repeated as many times as required (the only restriction is that the length of the recipe name cannot exceed 50 valid characters).

Recipe version control

With recipe version control **enabled**, these menu items (and their corresponding toolbar icons) are available:

Check In (under the **File** menu)  - Creates a read-only recipe version from a new or Checked Out work-in-progress (WIP) recipe. Enabled if a new or Checked Out recipe is open in the FactoryTalk Batch Recipe Editor. A Checked In recipe cannot be edited, but recipe step approvals can be signed off.

- **Check Out** (under the **File** menu)  - Creates an editable WIP recipe from a recipe version. Enabled if a versioned or obsolete recipe is open in the FactoryTalk Batch Recipe Editor.
- **Version History** (under the **Recipe** menu)  - Opens a dialog box displaying information about the Previous recipe version and the current recipe. Enabled when any recipe is open in the FactoryTalk Batch Recipe Editor.


With recipe version control enabled, the **File** menu commands in the FactoryTalk Batch Recipe Editor operate as follows, depending on the type of recipe (unversioned, WIP, or versioned) that is open:

Command	Unversioned recipe or new recipe	Checked Out (WIP) recipe	Checked In (versioned) recipe
New	<ul style="list-style-type: none"> • Creates a new recipe. • Prompts to provide a unique name for the recipe. Observe the naming restrictions. 		Not allowed.
Save	Saves changes in the open file to the working set.	Saves the changes in the WIP file to the working set.	<ul style="list-style-type: none"> • Versioned recipes cannot be edited--they cannot be saved. • If Recipe Approvals are enabled, the recipe with its approval state is saved automatically with every approval or revert signoff.

Command	Unversioned recipe or new recipe	Checked Out (WIP) recipe	Checked In (versioned) recipe
Save As	<ul style="list-style-type: none"> Saves the current recipe as a new recipe with a new name. The original recipe is closed and unchanged. Any approval steps are removed. 	<ul style="list-style-type: none"> Saves the current recipe as a new recipe with a unique, user-specified basename. Observe the naming restrictions. The original recipe is closed and unchanged. Version history information is removed. Any approval steps are removed. 	
Modify	<ul style="list-style-type: none"> If Recipe Approvals are enabled and an approval process is not active: <ul style="list-style-type: none"> Any user with editing rights is allowed to edit all aspects of the recipe. See Rename command below. 		<ul style="list-style-type: none"> Versioned recipes cannot be edited. If Recipe Approvals are enabled, approval steps can be signed and reverted.
Export	<ul style="list-style-type: none"> Save an exact copy of a specified recipe to the specified working set of recipes. User is prompted before overwriting an existing versioned file. Existing recipe approval steps are maintained. 		
Import	<ul style="list-style-type: none"> Import an exact copy of a recipe from another working set of recipes into the current working set of recipes. Existing recipe approval steps are maintained. 	<ul style="list-style-type: none"> If a version conflict exists, choose which version to import. If a recipe basename conflict exists with one or more recipes in the working directory, import the recipe that causes the conflict. Existing recipe approval steps are maintained. 	<ul style="list-style-type: none"> Confirm to overwrite an existing versioned file. If no name conflict exists, this imports an exact copy of a recipe from another working set of recipes into the current working set of recipes. If a version conflict exists, prompts to resolve the conflict. If a recipe basename conflict exists with one or more recipes in the working directory, the recipe that causes the conflict cannot be imported. Existing recipe approval steps are maintained.
Remove Recipe	<ul style="list-style-type: none"> Deletes the recipe from the working set (any referencing recipe will then have a missing element). 		
Rebuild Recipe Directory	<ul style="list-style-type: none"> If a recipe is added or deleted from the recipe directory without using the FactoryTalk Batch Recipe Editor, this command rebuilds the recipe directory and provides an option to verify all recipes within it. 	<ul style="list-style-type: none"> If a recipe basename or version numbering conflict exists with two recipes in the recipe directory, choose which recipe to retain in the directory. 	
Rename (Change Header data & Save)	<ul style="list-style-type: none"> Changes the name of the file and automatically propagates to all referencing files. Updates all the transition expressions and parameters with the new name. Name cannot change if an approval process is active. The recipe remains a New Recipe. 	Not allowed.	
Print	Print the recipe.		

Command	Unversioned recipe or new recipe	Checked Out (WIP) recipe	Checked In (versioned) recipe
Verify	Standard verification process.		Additional verification rules apply. If the recipe refers to a missing or obsolete unit procedure or operation, the recipe is marked as obsolete.
Check In	<ul style="list-style-type: none"> • If the recipe verifies, create a new file (basename~V1) • Replace all references to the basename with basename~V1. • Recipe basename is deleted. • Existing recipe approval steps are maintained. 	<ul style="list-style-type: none"> • If the recipe verifies, create a new file (basename~Vn) • Replace all references to the WIP recipe with basename~Vn. • WIP recipe is deleted. • Existing recipe approval steps are maintained 	Not Applicable.
Check Out	Not Applicable.		<ul style="list-style-type: none"> • Creates the next WIP recipe for the current recipe (basename~Vn_WIP). • Existing recipe approval steps are removed.

Recipe version history

Select **Version History**  to view recipe version information.

The version history consists of:

- Recipe name
- Version description
- Version creation date
- Version verification date
- Area model date (when last saved)
- Area model name against which the version was verified

All boxes are read-only with the exception of the **Version Description** box under **Current Recipe**. This box can be edited if:

- The recipe itself is editable (editable recipes are both unversioned and work-in-progress (WIP) recipes that have not started a recipe approval process).
- The FactoryTalk user has full access rights.

Obsolete recipe versions

During verification, the area model configuration stored in the recipe is compared against the current area model. If the recipe configuration and area model do not match, the recipe fails verification and is subsequently marked by FactoryTalk Batch as **Obsolete**.

Obsolete recipes:

- Are still versions of the basename recipe.
- Can be imported and exported, and removed from the working recipe set (using the **Remove Recipe** command).

- Cannot be opened, modified, renamed, checked in, printed, checked in, or saved.
- Do not allow a containing recipe to verify--an obsolete recipe is treated as a missing recipe element. See **Verification issues with a versioned recipe** in the section on Troubleshooting for more information.
- Any step in a visible recipe that references an obsoleted recipe is outlined in red.

To work with an obsoleted recipe:

- Use the **Save As** command to create a new recipe. The new recipe is not marked as obsoleted.
- Use the **Check Out** command to create a new, editable, work-in-progress (WIP) recipe. The FactoryTalk Batch Recipe Editor will make all required area model configuration changes to the WIP as it is created, indicating the modifications in the verification progress dialog box. The original obsoleted recipe remains unchanged.

Once the WIP recipe is checked back in:

- It is given the latest version number.
- The original obsoleted recipe remains in the working directory.

Alternatively, to reinstate an obsoleted recipe (clear its obsoleted status), modify the area model in the FactoryTalk Batch Equipment Editor to match the recipe configuration. When the recipe next undergoes verification by the FactoryTalk Batch Recipe Editor, it is reinstated as a current versioned recipe.



Tip: If a Recipe Approval Process is associated with a recipe when it is marked as obsoleted, its approval properties remain in effect and can be approved or reverted as needed.

Enable recipe versioning or approval

Use these examples to determine whether to enable recipe versioning or recipe approvals:

Recipe versioning can be used in conjunction with the Recipe Approval Process--playing a complementary role in recipe development.



Tip: When using recipe versioning and approvals together, consider applying expedited approvals to a checked-out (WIP) recipe, and the formal approval process to a checked-in (Versioned) recipe.

This section details the four scenarios when enabling or disabling Recipe Version Control and the Recipe Approvals Process:

	Recipe Versioning Disabled	Recipe Versioning Enabled
Recipe Approvals Process Disabled	Scenario 1	Scenario 2
Recipe Approvals Process Enabled	Scenario 3	Scenario 4

- **Scenario 1:** Recipe Versioning and Recipe Approvals are **disabled**. Commands in the FactoryTalk Batch Recipe Editor operate in their normal fashion. No enhanced version control or formal approval

process is involved in authoring and readying a recipe for production use. The **Release Recipe as Step** and **Release Recipe to Production** properties function in this way:

- To use a recipe or operation in another recipe, select the **Release Recipe as Step** checkbox in the **Recipe Header Data** dialog box.
- To add a recipe to the Recipe List, select the **Release Recipe to Production** checkbox in the **Recipe Header Data** dialog box.
- **Scenario 2:** Recipe Versioning is **disabled** and Recipe Approvals are **enabled**. The approval process governs if the **Release Recipe as Step** and **Release Recipe to Production** steps are approved or not.
- **Scenario 3:** Recipe Versioning is **enabled** and Recipe Approvals are **disabled**: These additional commands are available in the FactoryTalk Batch Recipe Editor (enabled or disabled depending on what versioning state the current open recipe is in): **Check In**, **Check Out**, and **Parent information**.
 - When a version of a recipe is created using **Check In**, the **Release Recipe as Step** and **Release Recipe to Production** properties are carried over unchanged.
 - When a work-in-progress (WIP) recipe copy is created using **Check Out**, the **Release Recipe as Step** and **Release Recipe to Production** properties are always cleared (set to false).
 - When a new recipe is created using **Save As**, the **Release Recipe as Step** and **Release Recipe to Production** properties are always cleared (set to false). The original recipe is unchanged.
 - To use a versioned or WIP recipe in another recipe, select the **Release Recipe as Step** checkbox in the **Recipe Header Data** dialog box.
 - To add a versioned or WIP recipe to the Recipe List in FactoryTalk Batch View, FactoryTalk eProcedure, or the FactoryTalk Batch HMI Controls, select the **Release Recipe to Production** checkbox in the **Recipe Header Data** dialog box. All sub-recipes and procedures within a recipe must also have their **Release Recipe to Production** checkboxes selected.
- **Scenario 4:** Recipe Versioning and Recipe Approvals are **enabled**: Versioning commands in the FactoryTalk Batch Recipe Editor are available (some may be disabled depending on the state of the open recipe), as are menu commands for **Approvals Process** and **Expedited Approval**.



Tip: Changing the **Release Recipe as Step** and the **Release Recipe to Production** approval states is not considered an edit or change to a recipe itself.

- A new recipe (created using **Save As** or **New**) is set to **Approval process not started** and can go through either the formal approval process or expedited process.
- A newly created version or WIP recipe keeps any completed or in-progress approval steps from the original recipe.

- When a version of a recipe is created, the **Release Recipe as Step** and **Release Recipe to Production** properties (Recipe Approvals disabled) or approval steps (Recipe Approvals enabled) are carried over unchanged.
- Approve and revert steps in a versioned recipe--there is no need to create a WIP recipe to do this.

Security authority overview

FactoryTalk Batch Security Authority, when enabled, protects intellectual property (as contained in recipes) and ensures it is only used within its intended scope. Security Authority helps secure recipes from unauthorized copying, editing, import, export, and use. Binary is the only recipe format that is allowed for the Security Authority setting.

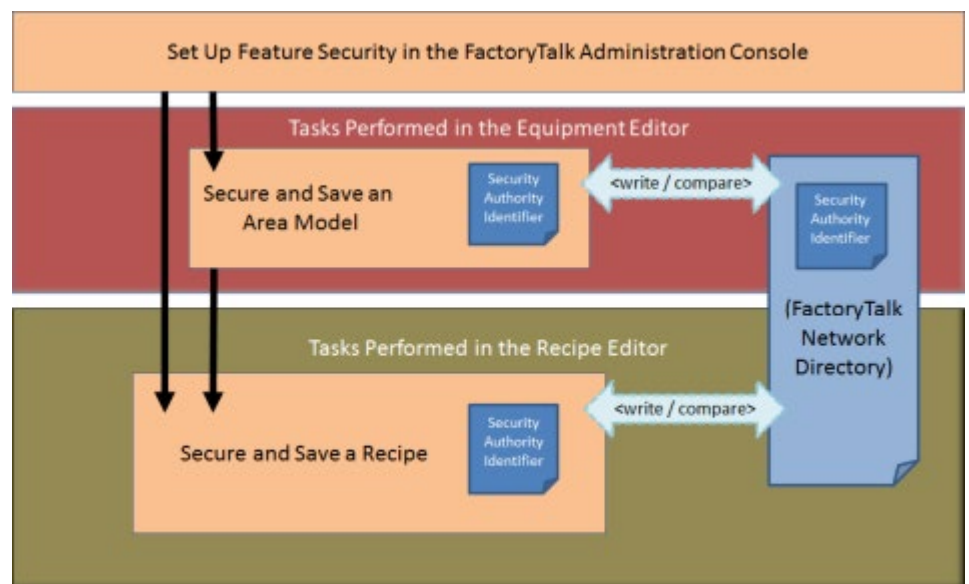
The FactoryTalk Batch Recipe Editor, when directed by an authenticated user to secure a recipe, associates the Security Authority Identifier (SAI) stored in the FactoryTalk Network Directory with the recipe. When the FactoryTalk Batch Recipe Editor is later directed to open the secured recipe, the SAI in the recipe is compared to the **current** SAI in the **current** FactoryTalk Network Directory. If they match, the FactoryTalk Batch Recipe Editor opens the recipe. If they do not match, for example when running under another instance of FactoryTalk or if the SAI has been regenerated, the editing session does not have the authority to open the recipe and stops the process.



Tip: If the SAI in a FactoryTalk Network Directory is changed or lost, access to any recipe that is bound to it may be lost. Rockwell Automation recommends that, before securing a recipe, back up the FactoryTalk Network Directory, and store unsecured versions of the recipe files in binary (.OUP, .UPC or .BPC), XML (.oxml, .uxml, or .bxml), or RDB formats in a secure location.

Security authority configuration

This flow chart shows the sequence of tasks in the configuration and use of security authority.



Authority to secure area models and recipes is assigned in the FactoryTalk Administration Console. Security Authority is disabled by default.

Area models are secured within FactoryTalk Batch Equipment Editor using the **Security Authority** command. The FactoryTalk Network Directory Security Authority Identifier (SAI) is written into the area model schema. To subsequently open and edit the area model, the SAI in the area model must match that of the current FactoryTalk Network Directory. No match prevents the opening and editing of an area model and its associated recipes in FactoryTalk Batch Recipe Editor.

Recipes are secured within FactoryTalk Batch Recipe Editor using the **Security Authority** command. The FactoryTalk Network Directory SAI is written into the recipe header data. To subsequently open and edit the recipe in FactoryTalk Batch Recipe Editor, the SAI in the recipe must match the one in the current FactoryTalk Network Directory.

Import and export operations with secured recipes are restricted.

See also

[Import and export restrictions for secured recipes](#) on [page 167](#)

[Security authority overview](#) on [page 141](#)

Secure recipe

Recipes are secured using the **Security Authority** command. Use these instructions to secure the recipe.

1. Open the recipe to secure in the FactoryTalk Batch Recipe Editor.



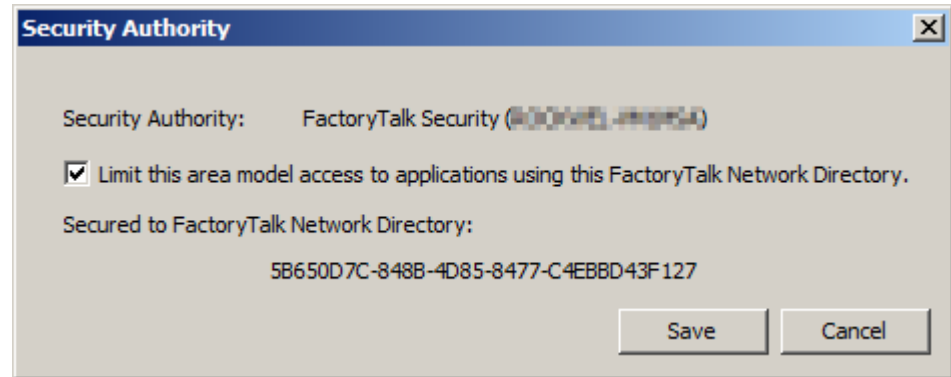
Tip: Before securing a recipe to a specific FactoryTalk Network Directory, Rockwell Automation recommends backing up the FactoryTalk Network Directory and store unsecured versions of the recipe files in binary (.OUP, .UPC or .BPC), XML (.oxml, .uxml, or .bxml), or RDB formats in a secure location.

For backup details, see **FactoryTalk Help**: Select **Start**, point to **All Programs > Rockwell Software > FactoryTalk Tools**, and then select **FactoryTalk Help**.

Once a recipe is secured to a particular FactoryTalk Network Directory, it cannot be opened if the security authority identifier associated with that directory no longer exists.

2. Select **Recipe > Security Authority**. The **Security Authority** dialog box opens.

3. Select the check box to secure the recipe. If the check box is not enabled, the user account is not authorized to use this feature.



4. Select **Save**.

Complete and maintain recipe

Once the recipe is saved, verify the recipe and release to production for batch operator access.

See also

[Verify recipe](#) on [page 145](#)

[SFC validation](#) on [page 146](#)

[Release Recipe as Step option](#) on [page 154](#)

[Release a recipe to production option](#) on [page 155](#)

[Rebuild the recipe directory](#) on [page 155](#)

Verify recipe

Verify recipe is the process that checks the recipe for completion and accuracy. A recipe must be verified to ensure that all connections and references are properly made. The verification process examines the selected recipe and all lower-level recipes. If there are errors in the recipes being verified, messages explaining the nature of the error are displayed.



Tip: To verify all recipes, from the File menu select Verify All Recipes or Verify All Recipes and Validate All SFCs.

1. Open the recipe to verify.
2. Select Verify or select Verify Recipes from the Recipe menu. The FactoryTalk Batch Recipe Editor reads the current recipe and displays a list of the recipe's outstanding problems, if any, and saves the recipe in the process.

IMPORTANT FactoryTalk users with ViewOnly permissions to the FactoryTalk Batch Recipe Editor cannot open the FactoryTalk Batch Recipe Editor to verify recipes. The FactoryTalk Batch Recipe Editor will close.

3. If there are errors, double-click an error message to open the recipe to the recipe level where the error was detected.



Tip: Copy the error text and paste it to a text document or spreadsheet if desired. Hold down Ctrl and select on the errors to copy. Use the Ctrl+C command to copy to the Clipboard.

4. Fix the problem and run the verification again. Continue this process until the recipe verifies successfully.
5. When the recipes are successfully verified, the SFC structure can be validated. From the File menu select Verify All Recipes and Validate All SFCs.

What gets verified?

When a recipe is opened in FactoryTalk Batch Recipe Editor, FactoryTalk Batch Recipe Editor checks that all the phases, operation sequences, parameters, and reports in the recipe also exist in the area model. If any parameters or reports have changed in the area model since the recipe was originally built, FactoryTalk Batch Recipe Editor automatically adds or removes parameters or reports from the recipe accordingly. Material references are also checked when the recipe is opened. If the material is not present in the material database, the material parameter is reset to NULL_MATERIAL.

When a recipe is verified, these items are also checked:

- **Basic Structure** is verified to ensure there is an initial step, a terminal step, other steps, and that they are all linked together.
- **Unit Requirements** are verified against the area model. For example, does the Unit Class or the Unit Instance exist in the area model? Is there a flow path in the area model that satisfies the Procedure Unit Requirement specified flow path?
- **Deferred Parameters** are within the min/max and limits of the parameters that are deferred to them.
- **Enumeration** set names and elements found in the area model.
- **Phase links** referential integrity (have partners), along with the message partner information found in the area model.
- **Phase class names** found in the area model.
- **Header Identifier, version, timestamp and author** are not empty.
- **Recipe Approval Steps** that are incomplete use valid signoffs (signature templates are assigned to valid FactoryTalk users and/or groups).
- **Release Recipe as Step property** is enabled on recipes.
- **Recipe Versions** are compatible with other parent or offspring recipes with the same basename. The verification results include information on **Recipes to be Obsoleted** and **Obsoleted Recipes to be Repaired**.
- **All step recipe paths are valid** (only if **Verify All Recipes and Validate All SFCs** is selected). For Procedures, this means that each referenced Unit Procedure recipe exists. For Unit Procedures, it means that each referenced Operation exists. For Operations, it means the Phase is in the area model.
- **Referenced recipes** existence (through the step name).

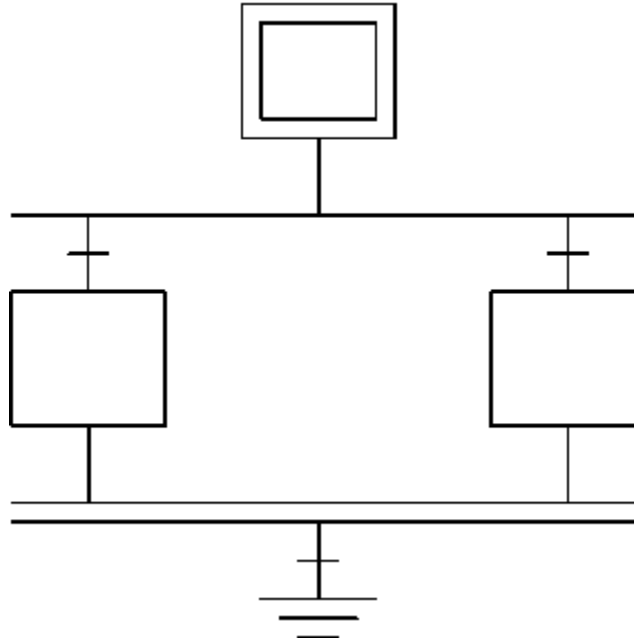
IMPORTANT If a working recipe is changed and fails verification, the **Release Recipe as Step** and the **Release Recipe to Production** properties in the recipe header are preserved. Due to verification errors, the recipe may be unavailable for use in production. Fix all verification errors to make the recipe available for use.

SFC validation

The **verification** feature ensures that basic SFC legality rules, such as step-transition-step sequencing, are enforced, but the recipe verification check does not detect logic errors within the SFC structure. Beginning with FactoryTalk Batch v. 10, the FactoryTalk Batch Recipe Editor provides **SFC**

validation which is an error check that looks for **logic** errors in the SFC structure defined within a recipe.

For example, this SFC is structurally invalid:



While the above SFC structure would pass the current verification checks done by the FactoryTalk Batch Recipe Editor, this structure would never run to completion for this reason: The final transition condition following the AND Convergence cannot execute unless all prior steps above it are concurrently active. The OR Divergence ensures that only one of the steps prior to the AND Convergence can be active at one time. While the error in the SFC shown above is relatively easy to identify, this type of SFC flaw and others can be much more difficult to identify in more complex recipe structures.

The SFC Validation tool can detect a variety of programming errors that result in invalid or illegal SFC programs. SFC structures identified as invalid by the tool results in recipe verification warnings. Note that the **Release Recipe to Production** approval or check box is **not** cleared by the verification checks.

Run SFC validation

While SFC validation is optional, it is good practice to get in the habit of validating the SFC structures.



Tip: To validate all recipes, from the **File** menu, select **Verify all Recipes and Validate all SFCs**.

To run SFC validation:

1. Open the recipe containing the SFC to validate.
2. From the **Recipe** menu, select **Verify all Recipes and Validate all SFCs**, or select **Verify and Validate**.

The FactoryTalk Batch Recipe Editor reads the current recipe structure, determines its validity, displays a list of the recipe's outstanding problems, and saves the recipe.

Errors display in the **Recipe Verification** dialog box.

3. If there are errors, double-click an error message to open the recipe containing the error.
4. Fix the problem and run the validation again. Continue this process until the recipe validates successfully.

Set allowable SFC permutations

The SFC Validation tool deals with **OR Divergences** by examining each possible leg of the divergence as if it were a separate SFC. This leads to a combinatorial number of possible SFCs when encountering multiple divergences within an SFC. For example, if an SFC contained three OR Divergences, one with three legs, one with two legs, and the third with four legs, the number of SFC permutations that would need to be examined would be 24 ($3 \times 2 \times 4 = 24$).

If configured a recipe with an extremely large number of OR Divergences, validation may take several minutes to complete. For this reason, define how many OR divergences the validation checks.



Tip: If the recipe contains more OR divergences than the upper limit allows, the recipe is considered too complex and validation stops. Configure the upper limit in the **FactoryTalk Batch Recipe Editor Options** dialog.

1. To set the maximum number of SFC permutations, select **Options** from the **Recipe** menu.
2. Change the value in the **SFC Validation: Max Number of OR Permutations** box to the desired number.

The default setting is **65535**. The allowable range for the configuration parameter is from **1024** to **2,097,152**, inclusive.

Select **OK** to save the changes.

SFC validation error types

Five basic errors can occur within an SFC structure. These examples show how to create and detect these errors.

The SFC Validation function identifies the linear segment of an error. Note that this may not be the location of the actual structural issue. The error message generated by SFC Validation when it detects an SFC structure error may include a linear segment specifier that provides the name of the first step in the linear segment of the error. If the first step in the linear segment is an Initial Step or Terminal Step of the SFC, then the step name, as currently used by the FactoryTalk Batch Server, is INITIALSTEP:1 or TERMINALSTEP:1.

Example:

WARNING: INITIALSTEP:1 >> Parallel Activation of Linear Segment

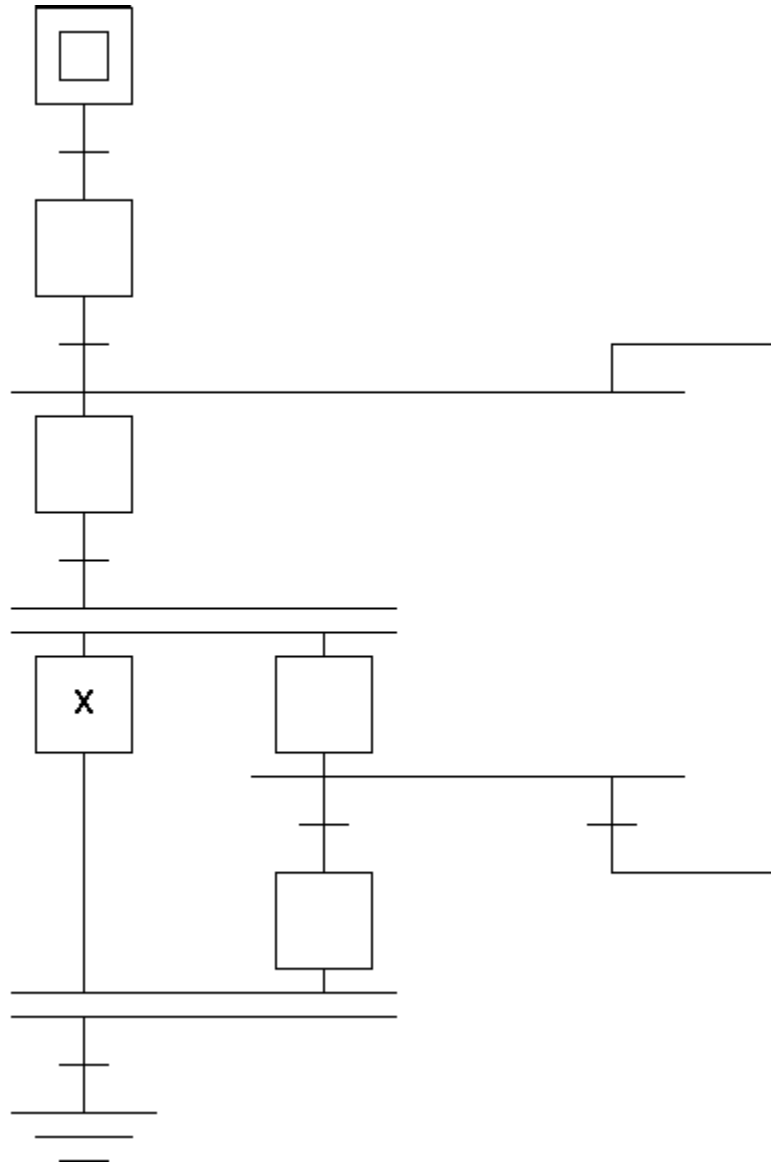
If there are no steps present in the linear segment, the string STEP NOT PRESENT IN LINEAR SEGMENT is used.

Example:

WARNING: STEP NOT PRESENT IN LINEAR SEGMENT >> Parallel Activation of Linear Segment

Parallel activation of segment

The loopback from the right leg of the OR Divergence back into the Linear Segment prior to the AND Convergence is the source of the error. Two execution tokens exists in the same linear segment, which is the segment containing the step marked with an **X**:

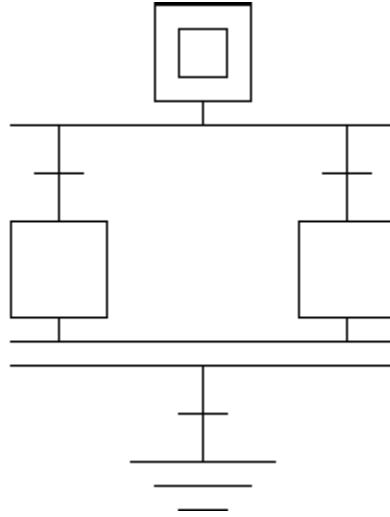


The actual error message text is similar to: **WARNING: SFC Validation Tool detects dual activation of Linear Segment beginning with step:%1 (where %1 is the actual segment identifier)**. The reported error location may not be close to the SFC structural issue. Refer to the product documentation for examples of SFC structures that can report this error. Other errors may or may not be present.

Unreachable terminal step

SFC Validation can detect when an SFC structure makes it impossible for a recipe execution to reach the SFC terminal step.

In this SFC, the OR Divergence activates only one of the two steps below it. This means that the transition below the AND Convergence never activates, since it is not permitted to fire unless **all** prior steps are active. This error makes it impossible for an execution token to reach the terminal step of the SFC.

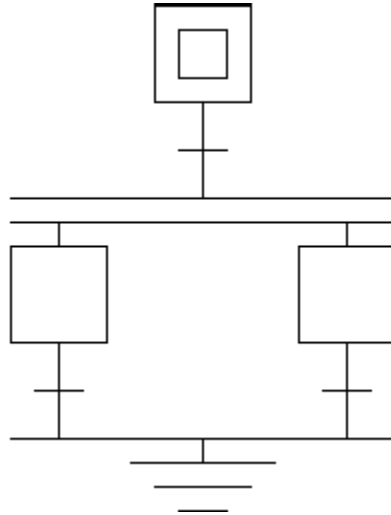


The actual error message text is similar to: **WARNING: SFC Validation Tool detects unreachable Terminal Step**. Refer to the product documentation for examples of SFC structures that report this error. Other errors may or may not be present.

Parallelism with terminal step

SFC Validation can detect when an SFC structure makes it possible to reach the terminal step of the SFC while other linear segments within the structure still have active execution tokens.

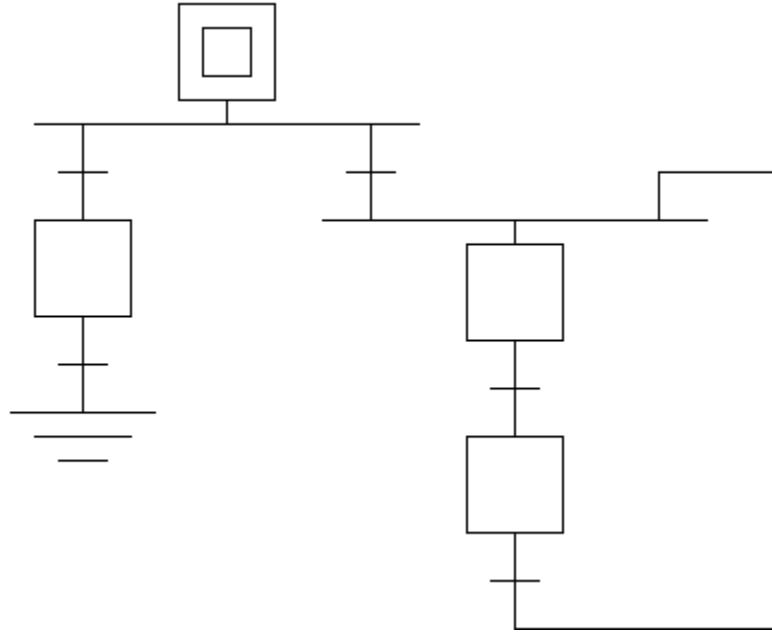
For example, in the SFC below, the AND Divergence activates both steps below it. Then, whichever leg of the parallelism completes first has its execution token reach the terminal step. This results in an execution token reaching the terminal step of the SFC while another execution token is active within the structure. This is defined as an illegal behavior — when an execution token reaches the terminal step of the SFC, it should be the **only** execution token present in the SFC structure.



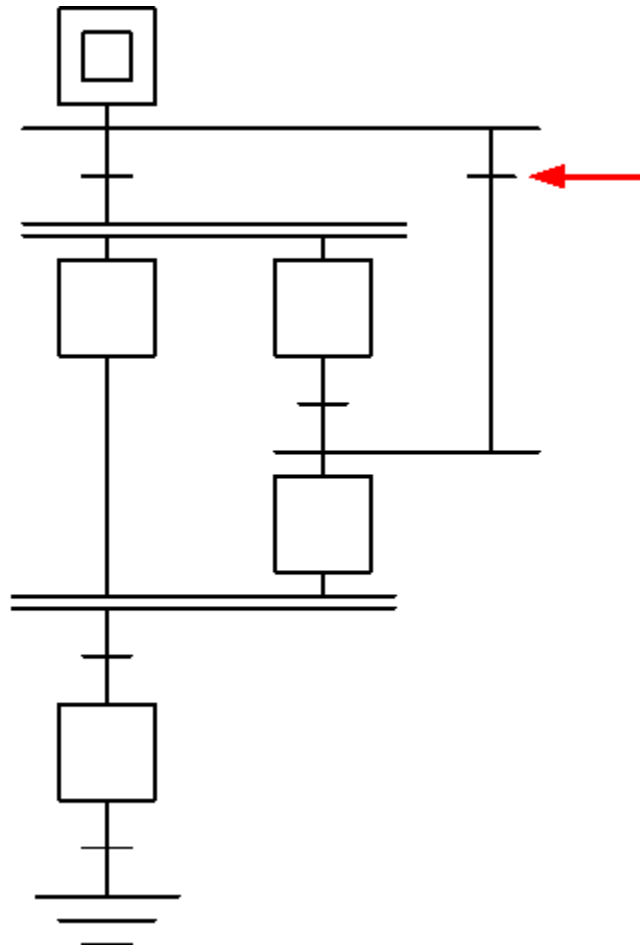
The error message text is similar to: **WARNING: SFC Validation Tool detects parallelism with Terminal Step of Linear Segment beginning with step:%1 (where %1 is the segment identifier)**. The reported error location may not be close to the SFC structural issue. Refer to the product documentation for examples of SFC structures that can report this error. Other errors may or may not be present.

Token cannot reach terminal step

SFC Validation Tool detects when an SFC structure makes it impossible for an active execution token contained within a linear segment to reach the SFC terminal step. For example, in the SFC example below, if the active token goes down the right branch of the first OR Divergence, then the execution token enters an infinite loop through which it can never reach the terminal step of the SFC.



A second type of SFC structure can also generate this error. In the SFC structure below, if the recipe execution goes through the right-most transition, indicated by the arrow, then the execution token reaches a dead end in the second step of the right-most leg of the AND Divergence and recipe execution is unable to reach the terminal step.

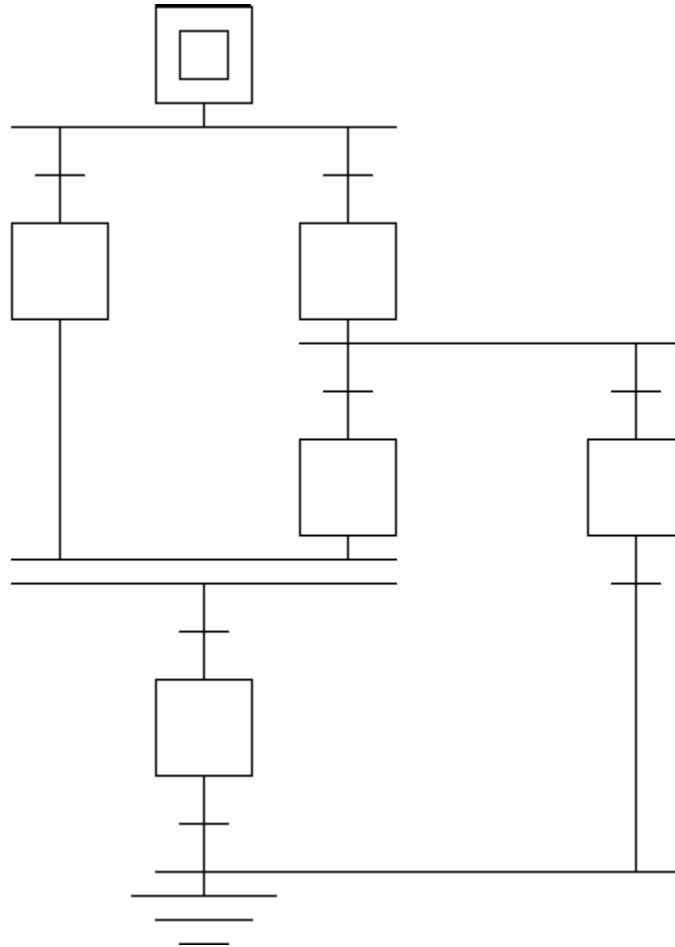


In this SFC example, the linear segment reported by the Validation Tool as the source of the error actually contains no steps.

The actual error message is similar to: **WARNING: SFC Validation Tool detects Linear Segment beginning with step:%1 (where %1 is the segment identifier) has no path to Terminal Step.** Refer to the product documentation for examples of SFC structures that can report this error. Other errors may or may not be present.

Unreachable linear segment

SFC Validation also detects when an SFC structure makes it impossible for an active execution token to reach one or more linear segments within the SFC. For example, in this SFC, the linear segment under the AND Convergence is unreachable because it is impossible for both prior steps to have active execution tokens simultaneously due to the initial OR divergence. This means that if recipe execution takes the left branch of the first OR Divergence, or the left branch of the second OR Divergence, then recipe execution becomes hung and unable to proceed.



The error message is similar to: **WARNING: SFC Validation Tool detects unreachable Linear Segment beginning with step:%1 (where %1 is the segment identifier)**. Refer to the product documentation for examples of SFC structures that report this error. Other errors may or may not be present.

Release Recipe as Step option

To use a recipe within another recipe, set the **Release Recipe as Step** property to true. When no Recipe Approval process is in use, select the **Release Recipe as Step** check box in the **Recipe Header** dialog box. When enabled Recipe Approval process, set this property by completing the signoff for Release Recipe as Step during the recipe approval process.



Tip: A recipe is valid for release to production only when all steps and procedures within it are approved as **Release Recipe as Step**.

Release recipe to production

Only recipes created in FactoryTalk Batch Recipe Editor have access to FactoryTalk Batch View, FactoryTalk Batch HMI Controls, or other client applications. If a recipe is to appear as a choice in a client application **Recipe List**, release the recipe to production. When no Recipe Approval process is in use, a recipe is released to production by checking the **Release Recipe to Production** checkbox in the **Recipe Header** dialog box. When a Recipe Approval process is enabled, the release to production property is set by completing all signoffs in the recipe approval process.

A recipe is valid for release to production only when all procedures within it are respectively approved as **Release Recipe as Step**.

Rebuild the recipe directory

The working set of recipes is stored in the location specified on the **Project Settings** tab of the FactoryTalk Batch Equipment Editor **Server Options** dialog box. If a recipe has been added to or deleted from this recipe storage location without using the FactoryTalk Batch Recipe Editor, rebuild the recipe directory.

1. There are two options:
 - If the FactoryTalk Batch Recipe Editor is closed, open it to rebuild the recipe directory. Any recipe file additions or deletions are shown when using the **Open, Remove, Import, Export, or Generate Reports** menus in the FactoryTalk Batch Recipe Editor.
 - If the FactoryTalk Batch Recipe Editor is already open, select **File > Rebuild Recipe Directory**. The FactoryTalk Batch Recipe Editor reads all of the currently stored recipes and updates the recipe directory file.
2. If a recipe version conflict occurs when rebuilding the directory, the **Resolve Version Conflict for Rebuild Recipe Directory Request** dialog box opens. A prompt opens to retain one of the two recipes that conflict.



Tip: If there are unverified recipes in the recipe directory, the option to verify all recipes or verify and validate all recipes displays.

Recipe maintenance

Recipe maintenance includes find recipe references, page setup, generating reports, remove recipe, and translate recipes.

See also

[Find Recipe References overview](#) on [page 156](#)

[Page setup](#) on [page 157](#)

[Generate reports option](#) on [page 157](#)

[Remove a recipe on page 160](#)

[Translation on page 160](#)

Find Recipe References overview

Use **Find Recipe References** to locate recipes that contain a specified operation or unit procedure. The **reference recipe** is the operation or unit procedure referenced in container recipes searched. A **container recipe** contains one or more steps assigned to a specified operation-level recipe or unit procedure-level recipe retrieved. A **step reference** is a step within a container recipe configured to reference an operation or unit procedure.

For example, an ice cream factory wants to use a new SWEETMILK_ORGANIC operation recipe in place of their SWEETMILK operation recipe. The recipe author uses **Find Recipe References** to search for all recipes containing steps with SWEETMILK (reference recipe) and manually updates them with SWEETMILK_ORGANIC.

The **reference list** of recipes that contain steps configured with the specified operation or unit procedure displays in a table. Every entry in the resulting list is the full recipe path and corresponding recipe levels to that step. When a container recipe opens, the highest level recipe opens. In this example, CLS_SWEETMILK_UP / CLS_SWEETMILK_OP:1 is selected. The recipe that opens is CLS_SWEETMILK_UP. For an explanation of recipe levels, see the **FactoryTalk Batch Recipe Editor introduction**.

The screenshot shows a dialog box titled "Recipe References" with the instruction "Select a recipe that references CLS_SWEETMILK_OP". Below the instruction is a table with four columns: "Row", "Recipe Reference Path (Procedure)", "Recipe Reference Path (Unit Procedure)", and "Recipe Refer". The table contains four rows of data.

Row	Recipe Reference Path (Procedure)	Recipe Reference Path (Unit Procedure)	Recipe Refer
1		CLS_SWEETMILK_UP	CLS_SWEETM
2	CLS_CHERRY_ICEMILK	CLS_SWEETMILK_UP:1	CLS_SWEETM
3	CLS_MANGO_ICEMILK	CLS_SWEETMILK_UP:1	CLS_SWEETM
4	CLS_STRAWBERRY_ICEMILK	CLS_SWEETMILK_UP:1	CLS_SWEETM

Find recipe references

Search recipes for a reference to an operation or unit procedure. Recipes can be stored as a binary file, XML file, or SQL Server database.

Prerequisites

- Obtain at least view-only security privileges to view the results.
- Close all recipes.
- Verify all recipes (full edit privileges required).

To search for recipe references

1. Select **File > Find Recipe References**. The **Select Reference Recipe** dialog box opens.
2. (optional) Set the **Recipe Filter** to narrow the list of displayed recipes.
3. Select one operation or unit procedure to find. The selected recipe is the **reference recipe**.
4. Select **Find**. The **Recipe References** dialog box opens. A reference list of recipes having steps matching the reference recipe displays. The recipes are **container recipes** and the steps are **step references**. **Total** indicates the number of step references listed.



Tip: The search includes checked-in recipes, checked-out recipes, recipes that are not versioned (versioning is disabled), obsolete recipes, and recipes having invalid Security Authority Identifiers.

5. (optional) Select **Copy All** to copy the **entire** list to the clipboard and paste in a document for future reference.
6. Select one step reference and then select **Open** to view and modify a container recipe in the list. Edit privileges and recipe status determines if the recipe can be modified. This table describes possible outcomes (assuming edit privileges):

Container Recipe Status	Open Result
Invalid Security Authority	Container recipe cannot be opened.
Checked-in	Container recipe cannot be edited.
Checked-out	Container recipe can be edited.
Obsolete	Prompts to update the recipe to be consistent with area model changes.
Versioning disabled	Container recipe can be edited.
Previously versioned recipes with versioning disabled	Container recipe cannot be edited.

Page setup

Page Setup is accessed through the **File** menu or **Ctrl+P**, and allows adjustment of:

- Margins (must be greater than 0.25 inches)
- Orientation
- Paper size and source
- Preview (view only)
- Printer and associated properties



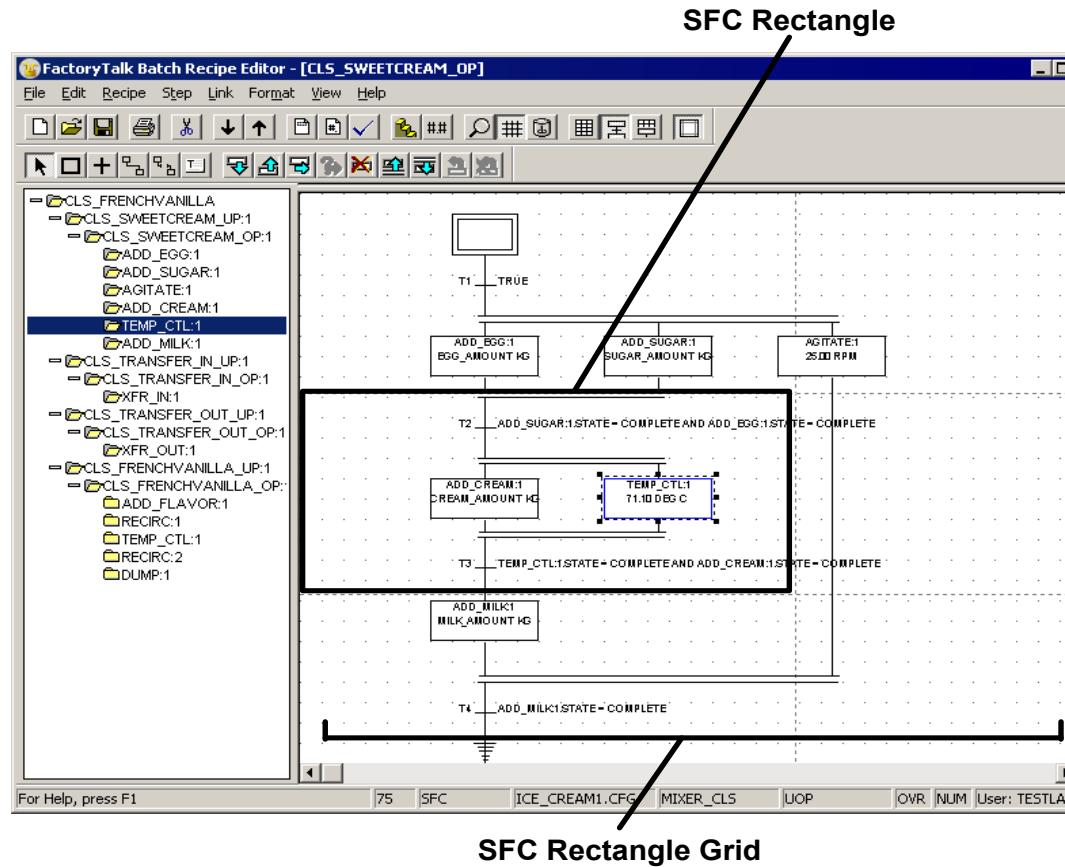
Tip: The print area cannot be smaller than 4 inches (10 cm) wide or high.

Generate reports option

Filter and print a complete working set of recipes, or if desired, a single recipe using the **Generate Reports** option in the FactoryTalk Batch Recipe Editor. The report format consists of these sections: the Overview Page, the SFC, Recipe Descriptive Information, and Recipe Formulations.

When generating reports, the entire recipe procedure appears in the SFC rectangle grid. An SFC rectangle represents a portion of the SFC that prints on

one page at 100% scale as defined in the **Page Setup**. Activating the **Show/Hide Page Boundaries** button displays the SFC rectangles in the SFC rectangle grid.



If a procedure description is in the **Header Data** dialog box for a recipe, it displays under the report header information.


If enabled, the approval process data displays in the recipe descriptive information. If approval process data is defined for the recipe, but disabled, nothing related to the approval process displays.

Print a working set of recipes

Use the **Generate Reports** option to select a set of recipes to print.

The Print function in the FactoryTalk Batch Recipe Editor uses the configured default printer. If there is no default printer specified, a prompt to establish one opens. Set or change the default printer without having to close the FactoryTalk Batch Recipe Editor application.

To print a working set of recipes

1. Close all open recipes by selecting **File > Close All**.
2. Select **Generate Reports** . The **Generate Reports** dialog box opens. The **Recipe Procedures** list displays the current working set of recipes.

3. Use **Select All** to include all displayed recipes in the report.
4. To customize the recipe list for printing, use the filters in the **Recipe Filter** tab to refine which recipes display. To select nonadjacent files, select the name of any file, then hold down the **Ctrl** key and select the name of each additional file.



Tip: Improperly secured recipes in the current working set cannot be printed--a notification window opens and lists the improperly secured recipes. Recipe names listed can be selected and copied to the clipboard.

5. In the **Report Selection** area, choose the report elements to print.
 - **Overview Page:** Displays the entire SFC grid scaled on a single report page with its page or SFC rectangle boundaries. A cross reference grid provides numeric vertical and alphabetic horizontal coordinates. The SFC rectangles are numbered from left to right, top to bottom. Numbers only appear in the upper right corner of rectangles containing a portion of the SFC.
 - **SFC:** Displays the SFC at 100% scale and includes a cross reference grid with numeric vertical and alphabetic horizontal coordinates. Off-page references indicate the SFCs continued direction.
 - **Descriptive Information:** Displays the recipe header information (with security authority and version history information), unit requirements, phase link groups, recipe formula parameters, steps, step parameters, reports, binding preferences, requirements, report expressions, transitions, and text boxes. Steps and transitions print in the SFC report order, left to right, top to bottom. If a group, requirement, parameter value, report limit, or text box does not exist, the report notes that status also.
 - **Recipe Formulations:** Displays a Recipe Formulations section in the report that includes the formulation names and descriptions, and, for each formulation, the name, formulation value, and Enum/Eng Unit of each parameter.
 - **Report Description:** Enter a name or descriptive phrase to display in the reports footer.
6. Select **Page Setup** to finalize any page printing adjustments.
7. After setting the report options, select **Print**.




Tip: **Print Preview** can only view a single procedure and is not available when printing an entire recipe set.

Print a single recipe

Open a recipe and use **Generate Reports** to print a single recipe instead of a set of recipes.


The Print function in the FactoryTalk Batch Recipe Editor uses the configured default printer. If there is no default printer specified, a prompt to establish one opens. Set or change the default printer without having to close the FactoryTalk Batch Recipe Editor application.

1. Open the recipe to print.

2. Select **Generate Reports** . The **Generate Reports** dialog box opens. The **Recipe Procedures** list displays the selected (root) recipe and its sub-recipes if any.
3. In the **Report Selection** area, choose the report elements to print.
 - **Overview Page:** Displays the entire SFC scaled on a single report page with its page boundaries. A cross reference grid provides numeric vertical and alphabetic horizontal coordinates. The SFC rectangles are numbered from left to right, top to bottom. Numbers only display in the upper right corner of rectangles containing a portion of the SFC.
 - **SFC:** Displays the SFC at 100% scale and includes a cross reference grid with numeric vertical and alphabetic horizontal coordinates. Off-page references indicate the SFC's continued direction.
 - **Descriptive Information:** Displays in text form the recipe header information (with security authority and version history information), unit requirements, phase link groups, recipe formula values, steps, step parameters/reports, binding preferences/requirements, report expressions, transitions, and text boxes. Steps and transitions print in the SFC report order, left to right, top to bottom. If a group, requirement, parameter value, report limit, or text box does not exist, the report notes that status also.
 - **Recipe Formulations:** Displays a Recipe Formulations section in the report that includes the formulation names and descriptions, and, for each formulation, the name, formulation value, and Enum/Eng Unit of each parameter.
 - **Report Description:** Enter a name or descriptive phrase to display in the reports footer.
4. Select **Page Setup** to finalize any page printing adjustments.
5. To view the finished report, select **Print Preview**.
6. After setting the report options, select **Print**.

Remove a recipe

When a recipe file is no longer needed, remove it from the current working set of recipes.

1. Select **File > Remove Recipe**. The **Remove Recipe** dialog box opens.
2. (optional) Set filters with the **Recipe filter** to narrow the number of recipe names displayed.
 -  **Tip:** The filter options are the same options as the **Open Recipe** dialog box.
3. From the **Recipe Name** list, select the recipe or recipes to remove, and then select **Remove**.

Translation

Recipes created in the previous version of FactoryTalk Batch automatically translate as long as the area model translated correctly, and the recipes are

not stored in XML or SQL.

If you need to translate recipes created in FactoryTalk Batch versions previous to FactoryTalk Batch version 11.x, please contact your Rockwell Automation Customer Support Representative.



Tip: If Recipe Approvals are enabled, when recipes from the previous version are translated in FactoryTalk Batch Recipe Editor, they are assigned a default Expedited Approvals process with two steps:

Release Recipe as Step and **Release Recipe to Production**

These steps initially have the **\$\$System** signoff following verification. Before verification, these steps are in a **\$\$System signoff pending** state.

Import and export recipes

The recipe **Import** and **Export** commands move exact copies of recipes in and out of the current working set. Recipes can be imported and exported in binary, XML, or RDB format.

When Recipe Version Control is enabled, the FactoryTalk Batch Recipe Editor prompts if version conflicts are encountered during import and export.

- Recipe Approval states are exported as part of a recipe, and are brought in unchanged when a recipe is imported.
- Security Authority supports only the binary file format.

Import recipes

Import recipes from any format into the current working set of recipes (binary only for secured recipes).

IMPORTANT Importing or exporting FactoryTalk Batch recipes to or from a computer containing more than one instance of Microsoft SQL Server is not supported. The SQL Server instance name cannot be defined in the **Import/Export Recipe** dialog box.

Prerequisites

- Make sure the recipe file format (binary, RDB, or XML) for the working set of recipes is set to the correct format.
- Create the recipe database on the computer running SQL Server. FactoryTalk Batch uses a database called **MasterRecipes**.



Tip: If the recipe file format in the FactoryTalk Batch Equipment Editor was changed, restart the FactoryTalk Batch Server if it is running, and re-open the FactoryTalk Batch Recipe Editor for the change to take effect.

To import recipe

1. Select **File > Import Recipe Into Working Set** option. The **Import Recipe** dialog box opens. The title bar of the dialog box indicates the file storage type selected for the current working recipe set in the FactoryTalk Batch Equipment Editor. Imported recipes convert to this format.
2. (optional) In the **Recipe directory from which recipes will be imported** area, change the storage format for importing recipes.
3. Select the recipe directory or database that contains the files to import.
 - **Binary Files** or **XML Files**

- **Microsoft SQL Server Database**

4. (optional) Set the **Recipe Filter** to narrow the list of displayed recipes.



Tip: The filter options are the same options as the **Open Recipe** dialog box.

5. From the **Recipe Name** list, select the recipes to be imported.
 - To select nonadjacent files in the **Open** dialog box, select the name of any file. Hold down the **Ctrl** key and select the name of each additional file.
 - To select adjacent files in the **Open** dialog box, select the name of the first file in the sequence. Hold down the **Shift** key and select the name of the last file.
 - To clear a file selection, hold down the **Ctrl** key and select the file name again.
6. Select **Import**. The recipes selected in the **Recipe Name** list are imported into the current working set.



Tip: If one of the imported recipes has the same name as a recipe in the working set, a notification dialog box opens.


If a versioned recipe would conflict, when imported, with a version in the working set, a notification dialog box opens.

Improperly secured recipes in the current working set cannot be imported—a notification window opens and lists the improperly secured recipes. Recipe names listed can be selected and copied to the clipboard.

Select a directory

If Binary Files or XML Files are imported, select the directory that contains the binary or XML files to import.

To select a directory:

1. Select browse  next to the **Recipe Directory** box of the selected storage format. The **Select Directory** dialog box opens. The default directory, BATCHCTL, displays unless another directory was previously selected.





Tip: If selected the same directory used by the working set of recipes, an error message displays. Choose another directory."

2. Select the directory, and then select **Open**. The **Recipe Name** list in the **Import Recipe** dialog box updates with the recipe list from the specified directory.

Select a SQL Server database

If RDB files or export files to RDB format are imported, select the Microsoft SQL Server database that contains the files to import or the database to export recipes.

To select a SQL Server database:

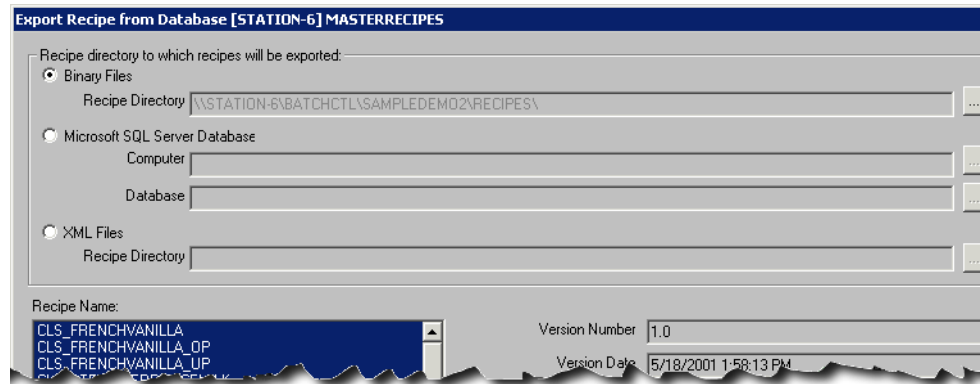
1. Select browse  next to the **Computer** box to select the computer on which the SQL Server database resides. The **Select Computer** dialog box opens.
2. In the **Enter the object name to select** box, type the name of the computer where the SQL Server database resides, and select **OK**.
3. Select browse  next to the **Database** box to select the database to import recipes or export recipes.
4. Select the database, and then select **OK**. The Recipe Name list on the **Import Recipe** dialog box updates with the list of recipes from the specified database.

Export recipes

Export recipes from the current working set of recipes into any supported storage format, including RDB, XML, or binary (secured recipes only maintain security when exported to binary).

To export recipes:



1. Open the FactoryTalk Batch Recipe Editor.
2. From the **File** menu, select **Export Recipe from Working Set**. The **Export Recipe** dialog box opens. The title bar of the dialog box indicates the file storage type selected for the current working set of recipes (set in the FactoryTalk Batch Equipment Editor). The recipes export from this format.



3. In the **Recipe directory to which recipes will be exported** area, change the storage format for exporting files.
 - **Binary Files** or **XML Files**
 - **Microsoft SQL Server Database**

IMPORTANT

Importing or exporting FactoryTalk Batch recipes to or from a computer containing more than one instance of Microsoft SQL Server is not supported. The **Import/Export Recipe** dialog box only allows you to enter the name of the computer hosting the Microsoft SQL Server and an existing database name. The SQL Server instance name cannot be defined in the **Import/Export Recipe** dialog box.

4. (optional) Set the **Recipe Filter** to narrow the list of recipes displayed for export.
 -  Tip: The filter options are the same options as the **Open Recipe** dialog box.
5. From the **Recipe Name** list, select the recipes to export.
 - To select nonadjacent files in the **Open** dialog box, select the name of one file. Hold down the **Ctrl** key and select the name of each additional file.
 - To select adjacent files in the **Open** dialog box, select the name of the first file in the sequence. Hold down the **Shift** key and select the name of the last file. To clear a selected file, hold down the **Ctrl** key and select the file name again.
 -  Tip: Improperly secured recipes in the current working set cannot be printed--a notification window opens and lists the improperly secured recipes. Recipe names listed can be selected and copied to the clipboard.
6. Select **Export** to initiate the export process. If recipes with the same name exist in the target format, indicate whether or not to overwrite them with the new recipe file.

Import conflicts

Recipe imports can create naming and versioning conflict. This can take several forms:

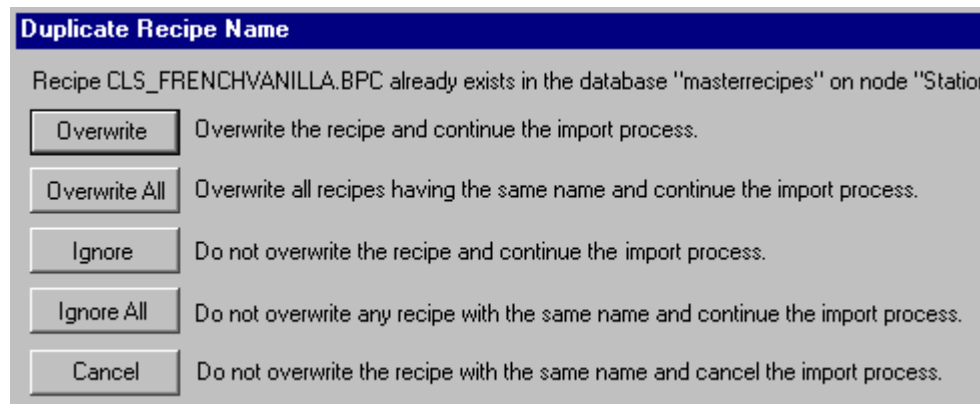
- Recipe name conflict
- Versioned recipe basename conflict
- Recipe version number conflict
 - During import
 - When rebuilding the current working recipe set

Resolve duplicate name conflict

If one of the imported recipes has the same name as a recipe in the working set, the **Duplicate Recipe Name** dialog box opens.

To resolve duplicate name conflict:

1. Select one of the listed options.





Tips:

- If the versioned recipe basename exists in the working recipe set, importing versioned recipes can result in a Recipe basename conflict.
- If Recipe Approvals are enabled, the approval states of the imported recipe overwrite those of the existing recipe.

Resolve recipe basename conflict

If a versioned recipe with the same basename as one or more versioned recipes is imported a warning message opens.

To resolve recipe basename conflict:

1. Select **OK** to close the warning.
2. Either rename the recipe to import, or remove the existing versioned recipes from the current working set that conflict with the recipe to import.

Resolve recipe version conflict

Use these instructions to resolve recipe version conflict.

To resolve recipe version conflict:

1. If a recipe version conflict occurs when importing a versioned recipe into the current working set, the **Resolve Version Conflict for Import Request** dialog box opens.
2. Select one of these options:
 - Select **Import with Delete**, which continues the import operation. This deletes the conflicting recipe in the working directory and replaces it with the imported recipe.
 - Select **Cancel Import**. This does not overwrite the recipe in the working set.

Resolve conflicts rebuilding directory

When rebuilding the recipe directory (**File > Rebuild Recipe Directory**), and a recipe version conflict occurs, the **Resolve Version Conflict for Rebuild Recipe Directory Request** dialog box opens:

To resolve recipe conflict when rebuilding the recipe directory

1. Choose which recipe to retain in the recipe directory.
2. Select the corresponding **Retain** button.

Import and export recipe restrictions

When recipes are secured with a security authority identifier (SAI), secure import and export can only occur when:

- The recipe to import or export is BINARY.
- The SAI in the recipe matches the SAI in the current FactoryTalk Network Directory.

These tables summarize recipe import and export operation restrictions:

Import or Export of binary recipes

	SAI match	SAI mismatch	No SAI
To binary	Allowed	Not allowed	Allowed
To RDB	Not allowed	Not allowed	Allowed
To XML	Not allowed	Not allowed	Allowed

Import or Export of XML or RDB recipes

	SAI import	SAI export	No SAI
To binary	Allowed, SAI ignored	Not applicable	Allowed
To RDB	Allowed, SAI ignored	Not applicable	Allowed
To XML	Allowed, SAI ignored	Not applicable	Allowed

Operations marked as Not allowed open a warning message.

Recipe formats

FactoryTalk Batch recipes can be stored in the proprietary FactoryTalk Batch binary format, XML, or a relational database (RDB) format. The three formats are mutually exclusive. Converting recipes from one format to the other is allowed. Select one of these formats in the FactoryTalk Batch Equipment Editor, to store all recipes created or imported into the working set of recipes in that format.

The binary format is the default for FactoryTalk Batch recipes.

IMPORTANT Only the binary format is secure from outside modification—it can be viewed and edited only in the FactoryTalk Batch Recipe Editor. Recipes stored in RDB or XML can be edited in external or third-party application software. To protect RDB and XML recipe files that have completed the Recipe Approvals process, save them in a secured location.

RDB format

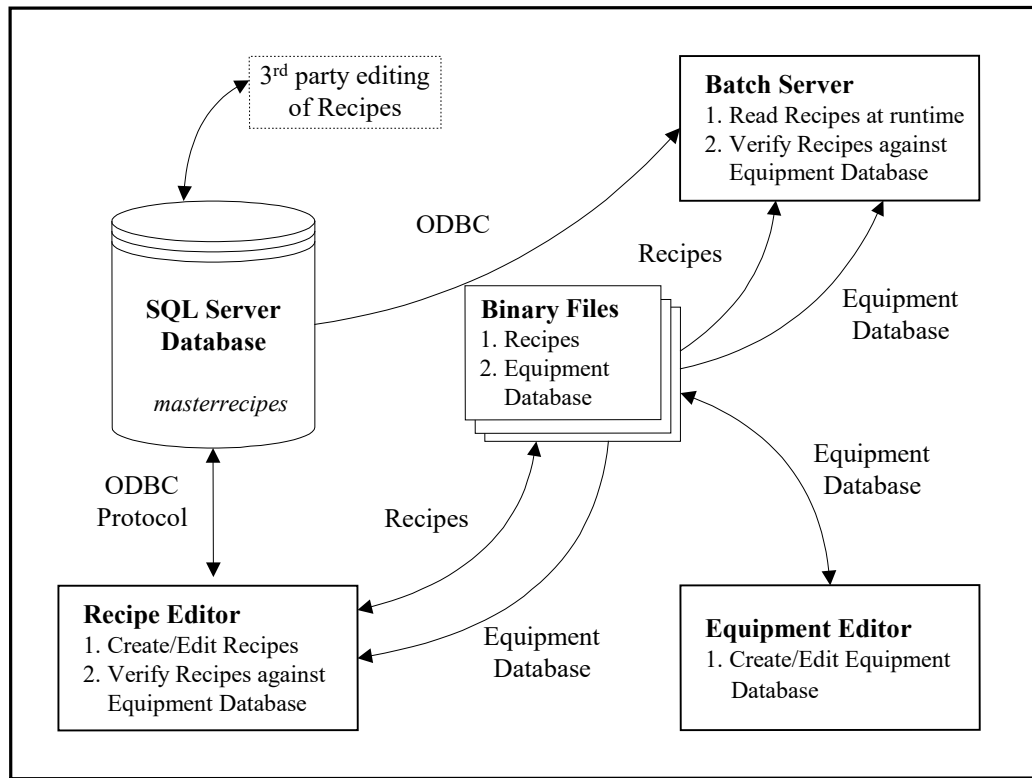
The RDB recipe format gives greater flexibility when generating reports. A default Microsoft SQL Server database, MasterRecipes, is created during FactoryTalk Batch installation and selected **Yes, I want to store recipes in SQL Server**.



Tip: The MS SQL Server must be installed before installing FactoryTalk Batch. Check software compatibility in the *FactoryTalk Batch Components Upgrade and Installation Guide*.

Open Database Connectivity (ODBC) is used as the database connectivity protocol between the FactoryTalk Batch Recipe Editor and the Microsoft SQL Server database.

This diagram describes the flow paths and functional relationships of FactoryTalk Batch using a relational database:



Add a Windows user to the MasterRecipeAuthor and MasterRecipeViewer local groups

Add a Windows user to the **MasterRecipeAuthor** and **MasterRecipeViewer** according to the function needed. Add recipe authors to the **MasterRecipeAuthor** local group and recipe viewers to the **MasterRecipeViewer** local group. The MasterRecipeAuthor user is allowed to edit recipes, and the MasterRecipeViewer user is only allowed to view recipes. FactoryTalk Batch uses a database called **MasterRecipes**.

1. Add the FactoryTalk Batch Server user account, or the domain account used by the FactoryTalk Batch Server, to the **MasterRecipeViewer** user group. This allows the FactoryTalk Batch Server to read the MasterRecipes database.
2. Add the Windows user to the **MasterRecipeAuthor** or **MasterRecipeViewer** local group.

Create the MasterRecipes database

Before importing recipes into the RDB format, first create the recipe database on the computer running SQL Server. FactoryTalk Batch uses a database called **MasterRecipes**.



Tip: Follow these instructions if the **Yes, I want to store recipes in SQL Server** option was not selected during FactoryTalk Batch installation, and MS SQL Server is installed.

Prerequisite

- Install and run SQL Server.

To create the MasterRecipes database:

1. If they do not exist, create the **MasterRecipeAuthor** and **MasterRecipeViewer** local user groups on the computer to install.
2. If workgroup security is used, create a local user account for the FactoryTalk Batch Server.
3. Add the FactoryTalk Batch Server user account, or the domain account used by the FactoryTalk Batch Server, to the **MasterRecipeViewer** user group. This allows the FactoryTalk Batch Server to read the MasterRecipes database.
4. Open the **Scripts** folder (in the Batch directory).
5. Double-click **createmasterrecipedb.bat**. The batch file runs and creates the MasterRecipes database.
6. After the batch file runs, configure FactoryTalk Batch to use the database. This is done in the FactoryTalk Batch Equipment Editor.

XML format

XML schemas have become an industry standard for providing implementation-neutral information. An XML schema defines what a textual file should look like to represent structured data, for instance a master recipe. Recipes are an instance document of an XML schema and can easily exchange between different companies.

The XML recipe format gives greater flexibility when generating recipes that must be usable across multiple operating systems and platforms. The FactoryTalk Batch Recipe Editor is capable of editing and creating XML recipes, and the FactoryTalk Batch Server can execute these recipes.

XML recipe schema

The FactoryTalk Batch Server and FactoryTalk Batch Recipe Editor can only read a valid XML recipe. An invalid recipe, one that does not conform to the MasterRecipe schema (**MasterRecipe.xsd**), is rejected, thus protecting the FactoryTalk Batch Recipe Editor and Server from faulty input. A valid recipe does not imply that the recipe has been verified by the FactoryTalk Batch Recipe Editor.

In addition to the information that is verified by FactoryTalk Batch Recipe Editor and FactoryTalk Batch Server, these items in XML recipes are also validated by the FactoryTalk Batch Recipe Editor and the MasterRecipe schema:

- **Referential Integrity:** An element referenced by another element exists.
- **Uniqueness:** Element names within a collection are unique.
- **Name Length:** Element names are restricted to a certain length. The recipe name cannot exceed 50 valid characters. The maximum length of an expression is 1023 characters. The maximum number of characters contained in a text box is 1024.

Set recipe file format

- **Name Pattern:** Names conform to a certain syntax. Sequential function charts must follow a step-transition-step pattern.



Store the working set of recipes in one file format. All recipes that imported into the working set must be in the format designated for the working set of recipes.

IMPORTANT Shut down and restart the FactoryTalk Batch Recipe Editor when changing between recipe storage types.

To set or change the recipe file format

1. Select **Start > Rockwell Software > FactoryTalk Batch Suite > FactoryTalk Batch**, and then select **Equipment Editor**. The FactoryTalk Batch Equipment Editor application opens.
2. From the **Options** menu, select **Server Options**. The **Server Options** dialog box opens with the **Project Settings** tab displayed.
3. In the Store Recipes Using area, select the recipe file format to store recipes.


If the Microsoft SQL Server Database was selected:

- Locate the SQL Server by selecting browse  next to **Node**. The **Select Computer** dialog box displays.
- Select the SQL Server computer and select **OK** to return to the **Server Options** dialog box.
- Locate the appropriate database by selecting browse  next to **Database**. A list of databases display.
- Select the SQL Server database and select **OK** to return to the **Server Options** dialog box. The selected database name is inserted into the **Database** box.



Tip: If the SQL Server is not installed on the same computer as the FactoryTalk Batch Client, to view a list of available SQL Server databases install the SQL Server Client Tools Connectivity option. Do a custom installation of SQL Server to install the Client Tools Connectivity option.

If Binary Files or XML Files was selected:

- Locate the recipe directory by selecting browse  next to **Recipe Directory**. The **Select Directory** dialog box displays.
 - Select the appropriate directory and select **OK** to return to the **Server Options** dialog box. The selected directory path name is inserted into the **Recipe Directory** box.
4. Select **OK** and then exit the FactoryTalk Batch Equipment Editor.

Import/Export error issues

Errors may occur during recipe import or export due to security issues or verification issues.

See also

[Improperly secured recipe ineligible for selection](#) on [page 173](#)

[Verification results for a versioned recipe](#) on [page 173](#)

[Invalid recipe folder or directory path warning](#) on [page 175](#)

Improperly secured recipes

Improperly secured recipes in the current working set cannot be selected when using the **Import**, **Export**, or **Generate Reports** command in the FactoryTalk Batch Recipe Editor. An improperly secured recipe is one in which the security authority identifier (SAI) in the recipe does not match that in the current FactoryTalk Network Directory.

This notification window opens to list any improperly secured recipes.

The recipe name listed in the notification window can be selected and copied to the clipboard for reference.

Verification results


When checking in a recipe or procedure as part of the versioning process, the recipe or procedure must pass verification. During the verification process, the FactoryTalk Batch Recipe Editor opens the **Verification Results** dialog box, listing any warnings or errors found. Warnings allow creation of a versioned recipe. Errors prevent successful verification and the creation of a versioned recipe.

Remove area model conflict

A unit procedure, CLS_FRENCHVANILLA_UP, is verified when opened in the FactoryTalk Batch Recipe Editor. An operation within the unit procedure, CLS_FRENCHVANILLA_OP~V1, is marked as **Obsoleted** during the verification, because of a conflict detected with the area model. The conflict is because parameter INGREDIENT_A was added to a phase after the recipe CLS_FRENCHVANILLA_OP~V1 was versioned.

The formula engineer determines that the parameter INGREDIENT_A is needed in the area model and for the operation which is currently marked **Obsoleted**.

To use Check Out and Redefine Step to remove an area model conflict

1. Select **File > Open Top Level**. In the open recipe dialog box, select the **Recipe Filter** tab, then select **Obsoleted** from the **Versioning Data** group.
CLS_FRENCHVANILLA_OP~V1 appears in the Recipe List.
2. Double-click CLS_FRENCHVANILLA_OP~V1 to open and verify it. The **Verification Results** dialog box shows that CLS_FRENCHVANILLA_OP~V1 has a **Conflict** and is **Not Updated**.
3. In the **Verification Results** dialog box, select **Check Out**.
4. In the **Check Out** dialog box, select **OK**. Verification is performed and the **Verification Results** dialog box shows that CLS_FRENCHVANILLA_OP~V2_WIP is created, and updated to include the parameter INGREDIENT_A.
5. Select **Close** to dismiss the **Verification Results** dialog box.
6. Select **File > Check In**. The **Verification Results** dialog box opens.
7. Select **Create Version**. The **Check In** dialog box opens.
8. Select **OK**.
9. In the **Search for recipes has completed** dialog box, select **Proceed**. CLS_FRENCHVANILLA_OP~V2 appears in the left pane view.
10. Double-click CLS_FRENCHVANILLA_UP to open it.
Verification is performed, and the **Verification Results** dialog box shows that CLS_FRENCHVANILLA_OP~V1 is **Obsoleted**.
11. In the **Verification Results** dialog box, select **Close**.
12. In the left pane tree view of the FactoryTalk Batch Recipe Editor, select CLS_FRENCHVANILLA_OP~V1:1.
13. Select the **Step > Redefine Step** menu command (toolbar icon ).
The **Redefine Step** command allows reference to an alternative copy or version of the operation or unit procedure.
14. From the **Operation Select** dialog box, select CLS_FRENCHVANILLA_OP~V2.
15. Select **OK**. Unit procedure CLS_FRENCHVANILLA_UP can be checked in as a versioned recipe.

Reinstate an obsolete procedure

A Unit Procedure, CLS_FRENCHVANILLA_UP, is verified when opened in the FactoryTalk Batch Recipe Editor. An operation within the unit procedure, CLS_FRENCHVANILLA_OP~V1, is marked as **Obsoleted** during verification, due to a conflict detected with the area model. The conflict is because parameter INGREDIENT_A was added to a phase after the recipe CLS_FRENCHVANILLA_OP~V1 was versioned.

The formula engineer determines that the parameter, INGREDIENT_A, is **not** needed in the area model.

1. Close the FactoryTalk Batch Recipe Editor.

2. Open the FactoryTalk Batch Equipment Editor, delete the unnecessary parameter INGREDIENT_A, then save the area model.
3. Open the FactoryTalk Batch Recipe Editor.
4. Open the unit procedure CLS_FRENCHVANILLA_UP. When it opens, verification is performed and the operation within the unit procedure, CLS_FRENCHVANILLA_OP~V1, is marked as **Reinstated**.

No further action is required.

Verification warnings and errors

When a lower-level operation or unit procedure prevents verification of its parent recipe or procedure, these are the likely causes:

Warning or Error	Solution
A lower-level operation or unit procedure does not have its Release Recipe as Step property enabled.	In the Verification Results dialog box, select Recipe > Header Data command to check each operation or procedure identified with this warning, and enable its Release Recipe as Step property. If enabled Primary or Expedited Approvals, sign off on its Release Recipe as Step approval step.
All lower-level operations or unit procedures must be versioned before the parent can be versioned.	In the Verification Results dialog box, use the Check in command to apply versioning to each operation or procedure identified with this error.
A lower-level operation or unit procedure has been marked as obsolete, because of conflicts with the area model.	As a best practice, apply either of these methods starting with the lowest level of the procedure's hierarchy and work up to the top level. <ul style="list-style-type: none"> • Method 1: Modify the area model to reinstate an obsolete procedure • Method 2: Use Check Out and Redefine Step to remove an area model conflict

Invalid recipe folder or path

Invalid recipe folder or directory path warning may occur when performing recipe file operations such as **Export** in the FactoryTalk Batch Recipe Editor:

This error can result if:

- The selected folder is not valid.
- The path to the folder is not valid.
- The selection is not a folder.

Confirm the folder is valid in File Explorer, and then retry the operation.

Index

.
.bpc files 67
.oxml files 67
.pxml files 67
.uop files 67
.upc files 67
.uxml files 67

A

adding
 branch structures to an SFC 77
 operations to a recipe 52
 parallel steps to a table 91
 parallel steps to an SFC 85
 phase link groups 119
 procedure 55
 recipe header data 64
 steps before parallel steps 92
 steps to an SFC 88
 transitions to an SFC 78
 unit procedures 52

Amount parameter 104
AND convergence 33, 86
AND divergence 33, 86
AND path, example 27
AND structure 23

arbitration
 null phase 31

assigning
 recipe formula parameters 99, 100
 report limits 108
 step formula values 108
 Timer step formula values 109
 transition expressions 80

associating comments 83
associating recipe comments 83
associating text boxes 83

B

begin parallel step 40
binary format
 exporting recipes to 165

binary recipe format 169
binding expressions
 data types 96
 operators 114

binding methods 105
binding preferences
 configuring 62
 printing 158, 159

binding requirements
 configuring 60
 printing 158, 159

branch structures
 adding to an SFC 77

building a recipe 20

C

changing
 recipe storage format 172
 unit requirements 55, 59

command buttons 19

comments
 associating 83
 disassociating 83
 editing 82

configuring
 binding preferences 62
 binding requirements 60
 Recipe Editor options 15

container binding 105
Container parameter 104
control strategies
 formula values 103, 108
 phase link groups 117

copying a recipe 67
creating
 operations 52
 phase link groups 119
 procedure 55
 unit procedures 52

D

data types 25
 binding expressions 96
 enumeration 96
 examples 96
 integer 96
 real 96
 string 96

decimal places
 setting 15

decision logic 27
default printer 158
defer
 parameters 105

defining
 steps 78

deleting
 equipment requirements 59

- phase link groups 120
- recipes 160
- descriptive information 159**
- descriptive information report 158**
- disassociating recipe comments 83**
- disassociating text boxes 83**
- display 105**
- dynamic unit allocation 15, 47**

E

- Edit menu 17**
- editing comments 82**
- editing text boxes 82**
- end parallel step 40**
- enum/EU 105**
- enumeration**
 - data type 25
- equipment requirements**
 - deleting 59
- EU 105**
- examples**
 - AND path 27
 - complex parallel structure 41
 - data types 25
 - decision logic 27
 - loop logic 27
 - OR path 27
 - parallel logic 27
 - REPEAT path 27
 - SFC execution 29
 - simple parallel structure 42
 - transitions 24
- export**
 - RDB files 164
 - recipes 165
- exporting**
 - recipes to binary 165
 - recipes to SQL 165
 - with same name 166
 - XML recipes 165
- expression builder**
 - unit binding 95
- expressions**
 - operands-Recipe Editor 96
 - operators-Recipe Editor 114
- expressions, transition 24**

F

- FactoryTalk Batch**
 - upgrading older versions 160
- FactoryTalk Batch Equipment Editor**
 - log on to the Recipe Editor 14
- FactoryTalk Batch Recipe Editor**
 - menu bar 17
 - recipe construction toolbox 19

- toolbar 17
- feed type 105**
- File menu 17**
- file types 67**
- final step 23**
- footer**
 - report description 159
- Format menu 17**
- formula values**
 - assigning 108, 109
 - control strategies 103, 108
 - material-enabled phase parameters 103

G

- generate reports 157**

H

- H 105**
- header data 64**
- Help menu 17**
- HH 105**
- HHH 105**
- hiding**
 - recipe construction toolbox 19

I

- ID 105**
- import**
 - RDB files 164
- importing**
 - binary recipes 163
 - RDB recipes 163
 - recipes 163
 - with same name 166
 - XML recipes 163
- initial step 23**
- inserting**
 - steps into a table 91
 - steps into an SFC 87
- integer**
 - data type 25

L

- L 105**
- Label parameter 104**
- limit calculation 105**
- limitations**
 - SFC 28
- Link menu 17**
- linking**
 - SFC components 79
 - SFC elements 27
- LL 105**

LLL 105
loop
 material 81
 rebinding 81
loop logic 27
Lot parameter 104

M

maintaining recipes 145
Material Class parameter 104
material class-based recipes 46
material loop
 creating 81
 examples 81
Material parameter 104
material report parameters 105
material-based recipes 46
material-enabled phase
 feed type 105
 parameters 103
max 105
menu bar 17
min 105
modifying
 unit requirements 55, 59
moving text boxes 82

N

name 105
naming a recipe 64
null
 procedure 30
null phase
 arbitration 31
NULL_CLASS 105
NULL_CONTAINER 105
NULL_MATERIAL 105

O

Open Database Connectivity 169
opening
 a recipe 50
 Equipment Editor 14
operands
 binding expression-Recipe Editor 96
operation 70
 changing unit requirements 55
 creating 52
 file types 67
 S88.01 definition 12
operators, expressions-Recipe Editor 114
operators, transitions 26
Options dialog box 15
OR convergence 81

OR divergence 81
OR path, example 27
OR structure 23
origin 105
overview page report 158, 159

P

Page Boundaries 157
parallel logic 27
parallel steps-recipe structure 40
 adding step before 92
 adding to a table 91
 adding to an SFC 85
parallel structure
 complex 41
 simple 42
 table view 41
parameter values
 assigning 108, 109
parameters
 defer 105
 recipe formula 99, 100
 setting decimal places 15
phase link groups
 control strategies 117
 creating 119
 deleting 120
phase report limits
 assigning 108
phase, Procedure levels 70
 S88.01 definition 12
printing
 binding preferences 158, 159
 binding requirements 158, 159
 recipes 157
 working set 158
printing report expressions 158, 159
procedure 70
 changing unit requirements 59
 creating 55
 file types 67
 S88.01 definition 12
procedure filtering 158
procedure level
 operation 70
 phase 70
 procedure 70
 unit procedure 70
procedure levels 70
process levels 70
process-connected device 15

R

RDB files
 export 164

- import 164
 - RDB format 169**
 - exporting recipes to 165
 - process flow diagram 169
 - RDB recipe format 169**
 - real**
 - data type 25
 - rebinding loop**
 - creating 81
 - rebuilding**
 - recipe directory 155
 - recipe**
 - assigning recipe formula parameters 99, 100
 - building overview 20
 - copying 67
 - exporting to binary 165
 - exporting to SQL 165
 - exporting XML 165
 - file types 67
 - header data 64
 - importing binary 163
 - importing RDB 163
 - importing XML 163
 - material class-based 46
 - material-based 46
 - opening 50
 - printing 157
 - multiple recipes 158
 - procedure levels 70
 - process levels 70
 - rebuilding directory 155
 - releasing to production 155
 - removing 160
 - saving 67
 - SFC view 23
 - translating 160
 - unit-based 45
 - upgrading
 - FactoryTalk Batch 160
 - verifying 145
 - Recipe menu 17**
 - recipes**
 - printing a working set 158
 - rectangle**
 - SFC 157
 - rectangle grid 157**
 - releasing**
 - recipe to production 155
 - removing**
 - recipes 160
 - SFC link 79
 - steps
 - from a table 93
 - from an SFC 80
 - REPEAT path, example 27**
 - report**
 - descriptive information 158, 159
 - header 157
 - report description 158, 159**
 - report expressions**
 - printing 158, 159
 - report footer**
 - report description 159
 - report information 157**
 - report limits**
 - assigning 108
 - report parameters**
 - material 105
 - report selection 158, 159**
 - reports**
 - generate 157
 - print 157
 - procedure filtering 158
 - resource**
 - arbitration 31
- S**
- S88.01 Procedural Model 70**
 - saving a recipe 67**
 - scalable 99**
 - setting**
 - binary recipe format 172
 - RDB recipe format 172
 - Recipe Editor options 15
 - XML format 172
 - SFC 23, 40**
 - adding
 - branch structures 77
 - parallel steps 85
 - steps 88
 - transitions 78
 - assigning transition expressions 80
 - branch structures 77
 - components
 - AND structure 23
 - final step 23
 - initial step 23
 - OR structure 23
 - step 23
 - transition 23
 - defining steps 78
 - element linking 27
 - examples 81
 - AND path 27
 - execution 29
 - OR path 27
 - REPEAT path 27
 - inserting steps 87
 - limitations 28
 - linking components 79

- null procedures 30
 - removing
 - links 79
 - steps 80
 - SFC rectangle 157**
 - SFC rectangle grid 157**
 - SFC report 158, 159**
 - SFC validation 146**
 - SFC validation errors**
 - Linear Segment Cannot Reach Terminal Step 151
 - Parallel Activation of Linear Segment 149
 - Parallelism with Terminal Step 150
 - Unreachable Linear Segment 153
 - Unreachable Terminal Step 150
 - SFC view, overview of SFC 23**
 - Show/Hide Page Boundaries 157**
 - showing**
 - recipe construction toolbox 19
 - simple recipes 40**
 - simple step 40**
 - smart binding 59**
 - starting**
 - Equipment Editor 14
 - step formula values**
 - assigning 108
 - assigning to Timers 109
 - Step menu 17**
 - steps 23**
 - adding
 - before parallel steps 92
 - parallel steps to a table 91
 - parallel steps to an SFC 85
 - to an SFC 88
 - defining 78
 - inserting
 - into a table 91
 - into an SFC 87
 - removing
 - from SFC 80
 - removing from table 93
 - string**
 - data type 25
 - symbols 40**
 - SFC recipe structure 40
- T**
- table**
 - adding parallel steps 91
 - adding step before parallel steps 92
 - defining steps 78
 - inserting steps 91
 - removing steps 93
 - table, viewing table-based recipes**
 - parallel structures 41
 - recipe structure 40
 - transition expressions 40
 - target parameter 105**
 - text boxes**
 - associating 83
 - disassociating 83
 - editing 82
 - moving 82
 - timer steps 37**
 - inserting 88
 - toolbar-Recipe Editor 17**
 - toolbox 19**
 - transition name 78**
 - transitions-SFC components 23**
 - adding to an SFC 78
 - assigning conditions 80
 - data types-Recipe Editor 25, 96
 - examples 24
 - expressions 24, 40
 - expressions in table view 40
 - identifiers 24
 - operators 26
 - translating**
 - recipes 160
 - type-Parameter Value/Report Limit definition 105**
- U**
- unit binding expression builder-Recipe Editor 95**
 - unit procedure 70**
 - changing unit requirements 55
 - creating 52
 - file types 67
 - S88.01 definition 12
 - unit requirements 69**
 - changing 55, 59
 - unit-based recipe 45**
 - upgrading**
 - older versions of FactoryTalk Batch 160
 - recipes 160
- V**
- validating SFCs 146**
 - value 105**
 - verification method 105**
 - verifying recipes 145**
 - View menu 17**
 - viewing**
 - SFC recipes 23
- X**
- XML recipe format 171**
 - XML recipes**

exporting 165
importing 163

Rockwell Automation support

Use these resources to access support information.

Technical Support Center	Find help with how-to videos, FAQs, chat, user forums, and product notification updates.	rok.auto/support
Knowledgebase	Access Knowledgebase articles.	rok.auto/knowledgebase
Local Technical Support Phone Numbers	Locate the telephone number for your country.	rok.auto/phonesupport
Literature Library	Find installation instructions, manuals, brochures, and technical data publications.	rok.auto/literature
Product Compatibility and Download Center (PCDC)	Get help determining how products interact, check features and capabilities, and find associated firmware.	rok.auto/pcdc

Documentation feedback

Your comments help us serve your documentation needs better. If you have any suggestions on how to improve our content, complete the form at rok.auto/docfeedback.

Waste Electrical and Electronic Equipment (WEEE)



At the end of life, this equipment should be collected separately from any unsorted municipal waste.





Rockwell Automation maintains current product environmental information on its website at rok.auto/pec.

Allen-Bradley, expanding human possibility, Logix, Rockwell Automation, and Rockwell Software are trademarks of Rockwell Automation, Inc.

EtherNet/IP is a trademark of ODVA, Inc.

Trademarks not belonging to Rockwell Automation are property of their respective companies.

Rockwell Otomasyon Ticaret A.Ş. Kar Plaza İş Merkezi E Blok Kat:6 34752, İçerenköy, İstanbul, Tel: +90 (216) 5698400 EEE Yönetmeliğine Uygundur

Connect with us.    

rockwellautomation.com — expanding **human possibility**[™]

AMERICAS: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204-2496 USA, Tel: (1) 414.382.2000, Fax: (1) 414.382.4444

EUROPE/MIDDLE EAST/AFRICA: Rockwell Automation NV, Pegasus Park, De Kleetlaan 12a, 1831 Diegem, Belgium, Tel: (32) 2 663 0600, Fax: (32) 2 663 0640

ASIA PACIFIC: Rockwell Automation, Level 14, Core F, Cyberport 3, 100 Cyberport Road, Hong Kong, Tel: (852) 2887 4788, Fax: (852) 2508 1846