

# ArchestrA Sequencer Object User's Guide

Invensys Systems, Inc.

Revision A  
Last Revision: 6/21/06



## Copyright

© 2006 Invensys Systems, Inc. All Rights Reserved.

All rights reserved. No part of this documentation shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of Invensys Systems, Inc. No copyright or patent liability is assumed with respect to the use of the information contained herein. Although every precaution has been taken in the preparation of this documentation, the publisher and the author assume no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained herein.

The information in this documentation is subject to change without notice and does not represent a commitment on the part of Invensys Systems, Inc. The software described in this documentation is furnished under a license or nondisclosure agreement. This software may be used or copied only in accordance with the terms of these agreements.

Invensys Systems, Inc.  
26561 Rancho Parkway South  
Lake Forest, CA 92630 U.S.A.  
(949) 727-3200

<http://www.wonderware.com>

## Trademarks

All terms mentioned in this documentation that are known to be trademarks or service marks have been appropriately capitalized. Invensys Systems, Inc. cannot attest to the accuracy of this information. Use of a term in this documentation should not be regarded as affecting the validity of any trademark or service mark.

Alarm Logger, ActiveFactory, ArchestrA, Avantis, DBDump, DBLoad, DT Analyst, FactoryFocus, FactoryOffice, FactorySuite, FactorySuite A<sup>2</sup>, InBatch, InControl, IndustrialRAD, IndustrialSQL Server, InTouch, Manufacturing Execution Module, MaintenanceSuite, MuniSuite, QI Analyst, SCADAAlarm, SCADASuite, SuiteLink, SuiteVoyager, WindowMaker, WindowViewer, Wonderware, and Wonderware Logger are trademarks of Invensys plc, its subsidiaries and affiliates. All other brands may be trademarks of their respective owners.

---

# Contents

<b>Welcome.....</b>	<b>9</b>
Documentation Conventions .....	9
Technical Support.....	9
<b>Chapter 1 About the Sequencer Object.....</b>	<b>11</b>
What is the Sequencer Object?.....	11
Sequencer Object Structure .....	11
Step Program.....	12
Steps.....	12
Outputs.....	13
Aliases .....	14
Settings.....	16
Step Structure .....	16
Step Entry, Step Body and Step Exit.....	17
Minimal Steps.....	17
Step Condition and Jump Condition .....	18
Triggers.....	19
Timers .....	20
Combined Triggers and Timers .....	20
Always True, Always False.....	20
Write on Exit .....	21
Jump To.....	21
Steps and Aliases Naming Conventions.....	21
Limitations.....	21

<b>Chapter 2</b>	<b>Defining Steps.....</b>	<b>23</b>
	Creating Steps .....	23
	Configuring Steps .....	24
	Renaming Steps.....	28
	Appending Steps .....	28
	Inserting Steps.....	29
	Changing the Position of Steps .....	29
	Deleting Steps .....	30
	Defining Minimal Steps .....	31
	Defining Initial Step and Final Step.....	31
	Naming the Step Program .....	32
	Validating the Step Program .....	33
	Adding Comments to Sequencer Objects .....	34
	Locking the Step Program.....	34
<b>Chapter 3</b>	<b>Defining Step and Jump Conditions .....</b>	<b>37</b>
	Defining Steps with Trigger-Based Conditions .....	37
	Defining Steps with Timers .....	38
	Defining Steps with a Cyclic Timer.....	39
	Defining Steps with Monthly Timers .....	39
	Defining Steps with Weekly Timers .....	40
	Defining Steps with Daily Timers .....	41
	Defining Steps with Hourly Timers.....	42
	Defining Steps with Minute-Based Timers .....	42
	Defining Steps with Always True or Always False Conditions ...	43
	Defining Combinations of Triggers and Timers .....	44
	Using Operators.....	45
	Using Retentive and Non-Retentive Timers.....	45
	Configuring Steps with Combined Conditions.....	45
	Overview of Trigger and Timer Combinations .....	46
<b>Chapter 4</b>	<b>Defining Outputs for a Step .....</b>	<b>53</b>
	Creating an Alias.....	53
	Assigning an Output Value to a Step .....	54
	Assigning a Literal Output Value .....	55
	Assigning a Literal Output Value in the Output Matrix.....	55
	Assigning a Literal Output Value in the Edit Outputs Dialog Box.....	56
	Assigning the Value of another Alias as Output Value.....	57
	Assigning the Value of another Alias as Output Value in the Output Matrix .....	57

Assigning the Value of another Alias as Output Value in the Edit Outputs Dialog Box .....	58
Removing Output Values.....	60
Removing Output Values from the Output Matrix.....	60
Removing Output Values from the Edit Outputs Dialog Box .....	60
Renaming Aliases.....	60
Deleting Aliases .....	61

## Chapter 5 Using Alias Definitions..... 63

Creating an Alias .....	63
Assigning an Attribute Reference to an Alias .....	64
Renaming an Alias .....	64
Changing the Appearance Order of Aliases .....	65
Deleting Aliases .....	65
Deleting an Alias.....	65
Clearing Aliases .....	66
Locking the Alias Definition List.....	66
Viewing Alias References .....	67
Viewing Alias Crossreferences.....	68

## Chapter 6 Defining Sequencer Program Settings..... 69

Configuring the Initial Command .....	69
Handling Errors and Reporting Alarms.....	71
Configuring Initialization Timeout .....	71
Halting Program Execution on Error .....	72
Using Alarms to Report Errors .....	73
Configuring Security .....	74
Modifying Access Rights to the Program Execution Command .....	74
Modifying Access Rights to Step Settings.....	75
Modifying Access Rights to the Active Step Command Settings .....	75
Modifying Access Rights to the Initial Step Command Setting .....	76
Modifying Access Rights to the Final Step Command Setting .....	76
Modifying Access Rights to the Sequence Configuration Command .....	77
Modifying Access Rights to the Output Value Arrays of a Selected Step .....	77
Modifying Access Rights to the Timer Presets of a Selected Step .....	78

Modifying Access Rights to the Jump Destination Setting of a Selected Step .....	79
Clearing Settings .....	80
<b>Chapter 7 Importing and Exporting the Sequencer Program Configuration .....</b>	<b>81</b>
Exporting the Step Program .....	81
Exporting the Alias Definitions .....	82
Exporting the Program Settings .....	83
Importing the Step Program .....	83
Importing the Alias Definitions .....	84
Importing the Program Settings .....	85
<b>Chapter 8 Running the Sequencer .....</b>	<b>87</b>
Deploying the Sequencer Object .....	87
Handling Restart/Failover Conditions .....	88
What are Restart/Failover Conditions?.....	88
How does the Running Sequence Program React to a Restart/Failover Event? .....	88
Detecting Restart/Failover Events .....	89
Configuring Alarms to Detect Program Execution Halting ..	89
Using Attributes to Detect Program Execution Halting .....	89
Resuming Operation after a Restart/Failover Event .....	90
Automatically Resuming Execution after a Restart/Failover Event .....	90
What happens to the Execution State after it Resumes?.....	91
<b>Chapter 9 Using Sequencer Program Commands and States .....</b>	<b>93</b>
Viewing and Monitoring the Currently Active Step .....	93
Viewing the Configuration of the Currently Active Step .....	93
Monitoring the Execution of the Currently Active Step.....	94
Controlling Program Flow at Run Time .....	95
Starting or Stopping Program Execution .....	96
Starting Program Execution .....	96
Stopping Program Execution .....	97
Resetting Program Execution .....	99
Resetting Program Execution.....	99
Holding or Resuming Program Execution.....	100
Holding Program Execution.....	100
Resuming Program Execution.....	102
Advancing a Step.....	103

Advancing a Step .....	103
Running in Single Step Mode .....	105
Initiating Single Step Mode .....	105
Confirming a Transition When Running in Single Step Mode .....	106
Jumping to a Specific Step .....	107
Jumping to a Specific Step .....	107
Setting Initial Command at Run Time .....	109

## Chapter 10 Modifying the Sequencer Program at Run Time ..... 111

Viewing and Changing the Configuration of a Selected Step .....	111
Viewing Configuration of a Selected Step .....	111
Changing Configuration of a Selected Step .....	113
Viewing the Execution Order of Aliases .....	114
Changing Step Program and Alias Configuration at Run Time ..	115
Loading and Saving .....	116
Saving Sequencer Object Configuration at Run Time .....	116
Loading Sequencer Object Configuration at Run Time .....	117
Uploading Run-Time Changes .....	119
Setting Initial Step and Final Step .....	119
Detecting Errors at Run Time .....	120
Detecting Program Execution Halting .....	120
Detecting Condition Trigger Failure .....	121
Detecting On Entry Output Failure .....	122
Detecting On Exit Output Failure .....	122
Detecting Errors during Sequencer Configuration Change ....	123
Using the Sequencer Object with Redundancy .....	123

## Appendix A Sequencer Object Help 125

Configuration Object Attributes .....	125
Step Program .....	126
Aliases .....	127
Settings .....	127
Settings .....	128
Alarms .....	129
Locking and Security .....	130
Run-Time Object Attributes .....	132
Execution Attributes .....	133
Current Attributes .....	140
Program Attributes .....	143
Selected Attributes .....	148

Alarm Attributes .....	152
History Attributes.....	154

## Appendix B Sequencer Program XML Schema 155

General Information.....	155
XML Roots and Sub-Roots.....	155
About Steps .....	155
Step/Jump Condition Syntax .....	156
Condition Type.....	157
OnExit Output Flag .....	157
Timer Preset .....	157
Trigger Expression .....	158
Examples of Conditions .....	158
Alias Configuration.....	158
Settings.....	158
Example XML .....	159

## Appendix C Sequencer State Transition Tables 161

Terminology .....	161
Current Execution State: Running .....	162
Current Execution State: RunningSingleStep .....	163
Current Execution State: SingleStepTransitionReady .....	164
Current Execution State: RunningHeld.....	165
Current Execution State: Stopped .....	166
Current Execution State: StoppedComplete .....	167
Current Execution State: StoppedError.....	168

## Appendix D Error Codes for Run-Time Updating 169

Glossary .....	171
----------------	-----

Index .....	181
-------------	-----



---

# Welcome

This guide assumes you know how to use Microsoft Windows, including navigating menus, moving from application to application, and moving objects on the screen. If you need help with these tasks, see the Microsoft online help.

You can view this document online or you can print it, in part or whole, by using the print feature in Adobe Acrobat Reader.

## Documentation Conventions

This documentation uses the following conventions:

Convention	Used for
Initial Capitals	Paths and filenames.
<b>Bold</b>	Menus, commands, dialog box names, and dialog box options.
Monospace	Code samples and display text.

## Technical Support

Wonderware Technical Support offers a variety of support options to answer any questions on Wonderware products and their implementation.

Before you contact technical support, refer to the relevant section(s) in this guide for a possible solution to any problem you have with Sequencer. If you need to contact technical support for help, have the following information ready:

- The type and version of the operating system you are using. For example, Microsoft Windows XP, SP2.

- Details of how to recreate the problem.
- The exact wording of the error messages you saw.
- Any relevant output listing from the Log Viewer or any other diagnostic applications.
- Details of what you did to solve the problem(s) and your results.
- If known, the Wonderware Technical Support case number assigned to your problem, if this is an ongoing problem.

# Chapter 1

## About the Sequencer Object

This section helps you understand the basic features of the Sequencer object, what they do, and how they interact.

### What is the Sequencer Object?

The Sequencer Object is an ArcestrA application object. It lets you configure, execute, and manipulate a sequence of operations associated with ArcestrA attributes within an Industrial Application Server application.

You can use it to automate:

- repetitive manufacturing procedures with a finite number of steps
- supervisory processes with a finite number of steps

### Sequencer Object Structure

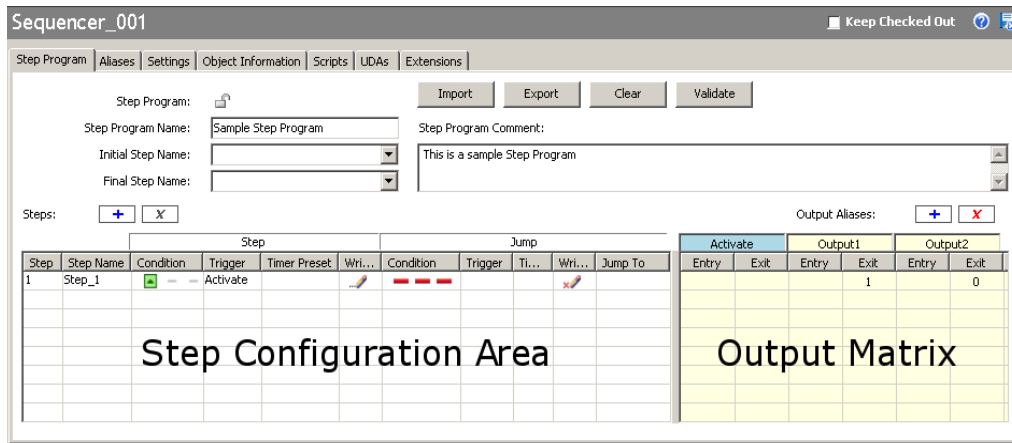
A Sequencer Object consists of three main parts:

- **Step Program** - a collection of steps that define the reading and writing operations associated with ArcestrA attributes via Aliases. When in operation, the Sequencer object advances cyclically through the steps or jumps based on various events, conditions, and timers. Inputs are evaluated and values are written to defined outputs.
- **Aliases** - a list of names that are mapped to ArcestrA attributes. The Sequencer Object uses these names internally to associate inputs and outputs with ArcestrA attributes.

- **Settings** - other settings that control the behavior of the Sequencer Object.

## Step Program

The Sequencer Object hosts a Step Program that has a finite number of steps. You can configure the Step Program in the following panel:



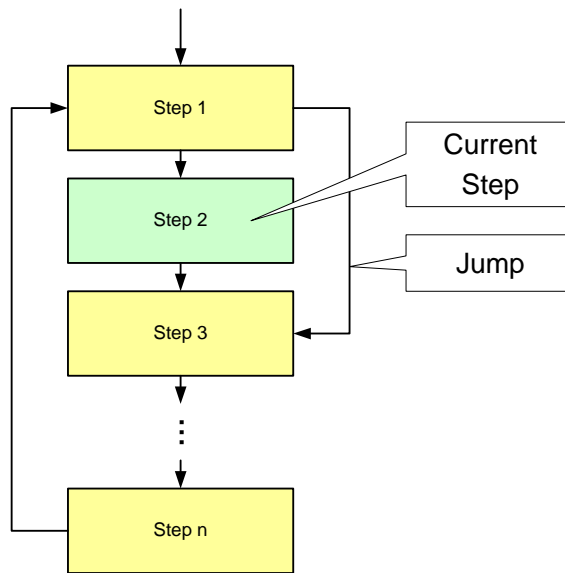
## Steps

Steps are processed sequentially unless certain conditions are met that instruct the processing to continue at a specified step (Jump To).

The step that is being processed at any given time, is called the **Current Step**. There is only one current or active step at any given time.

The conditions for processing the next step or any specified step are controlled by events (triggers, timers, or a combination of triggers and timers).

The following diagram shows this:



## Outputs

You can write values to ArcestrA attributes whenever a certain step becomes active or before Sequencer jumps to the next or another step.

---

**Note** To write to an ArcestrA attribute you must first map it to an Alias. For specifics, see Aliases on page 14.

---

Every step of the Step Program allows you to associate the step entry and/or the step exit with values that are written to the **Aliases**. These values form an **output matrix**.

An output value can be one of two types:

- **literal value** (number, string, date time, elapsed time or boolean)
- **value of another Alias** (referencing another ArcestrA attribute)

Empty cells in the matrix mean that no write to the the associated Alias takes place.

### Literal Values

You can configure a step to write literal values to attributes either when a step becomes active or after the step is triggered, either by the step condition or the jump condition. Some examples of literal values are 3.141, 17, “Start”, and “Pause”.

---

**Note** Literal values that are either string, date time, or elapsed time data type must be enclosed by double quotes to be recognized as such. If they are not enclosed by double quotes they are considered to be Aliases. You can embed double quotes in literal strings without any escape character. For example: "Joe said "Hello"". Most special characters are accepted within double quotation marks, except for tabs and carriage returns.

---

### Aliases

You can configure a step to write the value of another attribute to the output. In this case the output causes the step program to pass the value of one attribute to another attribute. This applies to any Alias you have defined in your Step Program.

## Aliases

At run time, the Sequencer Object interacts with attributes of other ArcestrA objects. These attributes can be:

- **Inputs** that cause the step condition or jump condition to be fulfilled. Typical inputs are sensory devices that trigger the step execution when a certain condition in a factory device is fulfilled. For example, a tank level sensor.
- **Outputs** are written either before or after the step or jump condition is fulfilled. Typical outputs are to device controls, such as valve controls.
- **Inputs and Outputs** at the same time. A typical example is a condition reset.

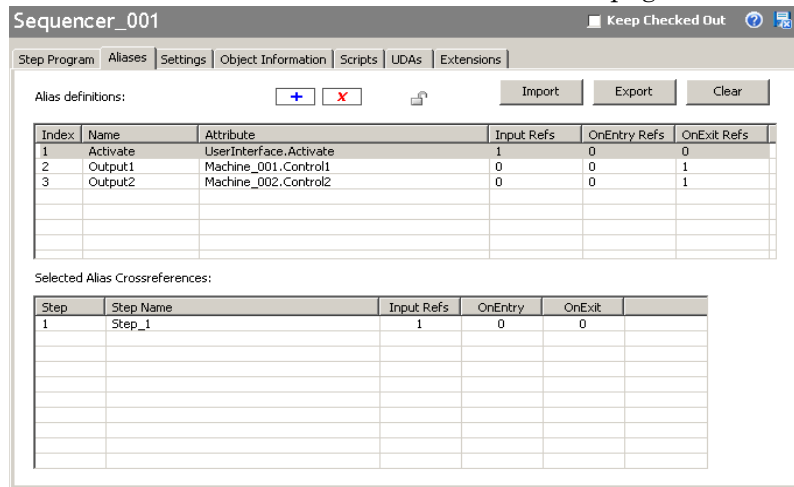
The Step Program can reference the same ArcestrA attribute multiple times in the Step Program. Instead of referencing these attributes directly, the Sequencer Object uses a list of **Aliases** to reference the attributes. Using Aliases simplifies the appearance of the step program and saves time when the referenced attribute needs to be updated.

---

**Note** Writes are not guaranteed. Only invalid references are reported. The Sequencer Object does not request or evaluate write confirmations.

---

In the Sequencer Object, you can configure the mapping of Aliases to ArchestrA attributes on the Aliases page.



The screenshot shows the 'Aliases' page of a Sequencer Object named 'Sequencer\_001'. The page has a tabbed interface with 'Aliases' selected. Below the tabs are buttons for '+', 'X', and a refresh icon, along with 'Import', 'Export', and 'Clear' buttons. The main area contains two tables.

**Alias definitions:**

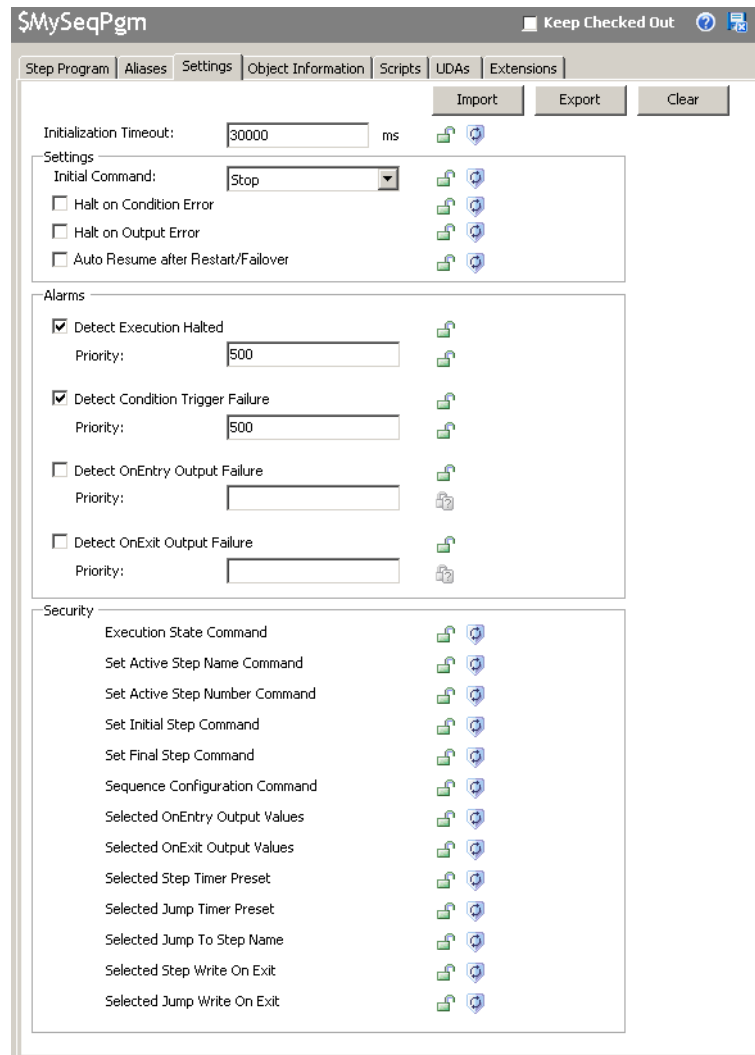
Index	Name	Attribute	Input Refs	OnEntry Refs	OnExit Refs
1	Activate	UserInterface.Activate	1	0	0
2	Output1	Machine_001.Control1	0	0	1
3	Output2	Machine_002.Control2	0	0	1

**Selected Alias Crossreferences:**

Step	Step Name	Input Refs	OnEntry	OnExit
1	Step_1	1	0	0

## Settings

The Settings panel looks like the following:



You can set generic parameters of the Sequencer Object in the Settings panel, such as:

- **Settings** - you can set the initial command and the Sequencer behavior after errors and failover
- **Alarms** - you can instruct the Sequencer Object to generate an alarm if an error or a warning occurs
- **Security** - you can lock access to certain Sequencer Object features at design time and run time

## Step Structure

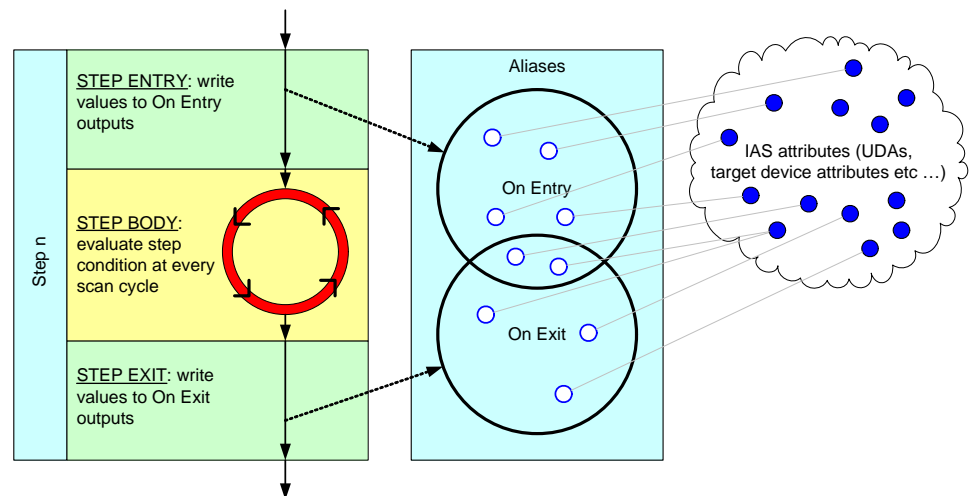
This section describes the functionality embedded in every step.



## Step Entry, Step Body and Step Exit

Every step includes a:

- **Step Entry** section where values are written to ArchestrA attributes after the step is activated. The On Entry outputs are processed on the first scan cycle when the step becomes active.
- **Step Body** section where the step condition and jump condition are continuously evaluated. Step and jump conditions are not executed until the second scan of the step.
- **Step Exit** section where values are written to ArchestrA attributes after the step condition or jump condition is fulfilled. The On Exit outputs are processed in the same scan cycle when the step condition or jump condition is fulfilled.




---

**Note** Once a step is active, it is evaluated once per scan cycle until it is no longer the active step.

---

### Minimal Steps

Minimal Steps are empty steps with

- an Always True step condition
- no jump condition
- no values being written to outputs

The minimal step can be considered a No Operation (NOP) Step. The Sequencer stays in an empty step exactly two engine execution cycles before advancing to the next step. No Operation (NOP) steps are useful for inserting dummy steps for future enhancements.

## Step Condition and Jump Condition

Every step body contains a:

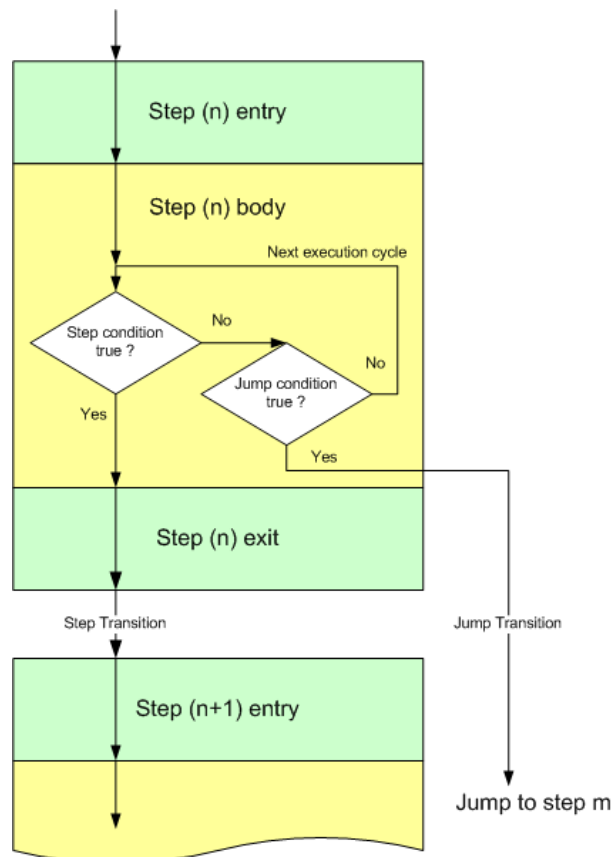
- **Step Condition** - if the step condition is fulfilled, Sequencer continues processing at the next step
- **Jump Condition** - if the jump condition is fulfilled, Sequencer continues processing at a specified step (which can also be the current step).

---

**Note** If both the step condition and the jump condition are true, advancing to the next step takes precedence over the jump.

---

The following diagram shows this:



By default:

- **Jump Write OnExit** is **False**, which means On Exit output values are not written for a jump transition.
- **Step Write OnExit** is **True**, which means On Exit output values are written for a step transition.

You can change this setting for any given step in your program.

A single step with a jump condition is processed as follows:

- 1 The step becomes active. On Entry values are written to Outputs.
- 2 Sequencer cycles continuously at the step body, evaluating the step condition and the jump condition at every scan cycle.
- 3 When the step condition is met, the On Exit output values are written if the **Step Write OnExit** is True, and the Sequencer advances to the next step.
- 4 When the jump condition is met, the On Exit output values are written if **Jump Write OnExit** is True, and the Sequencer moves the active step to the specified step.

---

**Tip** Jumps can be forward, backward, or to the current step. If the current step is the target of the jump, the on entry outputs are written once just like at any other step destination.

---

You can use Jump transitions for:

- Conditional Program Flow
- Error Handling
- Monitoring

## Triggers

In the step program, triggers cause the step or jump condition of the active step to be met and the step program processing to continue. Triggers are linked to Aliases that reference ArchestrA attributes.

---

**Note** Triggers are usually linked to Boolean attributes, but can also use any other numeric data type (integer, float, double). The trigger value is coerced to True or False.

---

In the simplest While True case, when a Trigger (Boolean attribute) is true, the condition the Condition evaluates to true, the On Exit Output are written and the step is ready to transition the next scan. The **Aliases** panel maps attributes to Aliases. The Aliases are mapped to steps via values to be written at that step.

Step and Jump Conditions are triggered by referenced Boolean attributes in different ways, such as when the Boolean attribute:

- is true: **While True**
- becomes true while the step is active: **On True**
- is false: **While False**
- becomes false while the step is active: **On False**

- changes from true to false or vice-versa: **Data Change**

A condition can also be

- never fulfilled: **Always False**
- always fulfilled: **Always True**

without using any Aliases.

## Timers

Timers can either be used on their own or in combination with triggers to fulfill a certain processing requirement.

### Simple Timer

A Simple timer is started when the step becomes active. When the timer elapses, the condition is set to true.

### Cyclic Timers

**Cyclic Timers** are pulse timers that evaluate to true for one scan at a certain time each month, each week, each day, each hour or each minute.

Cyclic timers do not retain the fired state. If a cyclic timer event did not lead to a firing condition due to combinations with triggers, the cyclic timer enters the non-firing state until the next cyclic timer event occurs.

## Combined Triggers and Timers

You can combined a trigger and timer to form a **combined condition**. This is done by using a logical operators (AND, OR) or a special operator called DELAY.

### Delay Timer

The Step or Jump Condition is fulfilled if the delay timer has elapsed. Delay Timers can either be:

- Delay Simple timer - is started when the trigger condition is met. Once started, it runs down unconditionally. When the timer elapses, the condition is set to true.
- Retentive - allows summing up the times a trigger condition is true until it reaches the specified interval.
- Non-Retentive - tests for a minimum time a trigger condition needs to be continuously true before triggering.

## Always True, Always False

The **Always True** condition instructs Sequencer to always execute the step, or in the case of a jump condition, to enforce a jump to a specified step.

The **Always False** condition instructs Sequencer to ignore the step execution, or to never jump, in the case of a jump condition.

## Write on Exit

For the Step Condition and the Jump Condition there is a **Write on Exit** option. You can use the **Write on Exit** option to instruct Sequencer to write the outputs to the Aliases either when the Step or Jump Condition is fulfilled.

By default the **Write on Exit** option is set for the Step Condition, and not set for the Jump Condition.

## Jump To

The **Jump To** parameter of any step specifies the Step Name where Sequencer jumps to if the Jump Condition is fulfilled and the Step Condition is not fulfilled.

# Steps and Aliases Naming Conventions

Steps and Aliases follow the same naming conventions:

- They can only consist of alphanumeric characters, underscore (“\_”) and period (“.”).
- They can only start with an alphanumeric character and an underscore (“\_”).
- They must include at least one alphabet character.
- They can be up to 32 characters in length.
- They are case-insensitive (for example it is not possible to create the Aliases “Alias1” and “aLiAs1” in the same Alias definition list).
- Capitalization is allowed and maintained (for example it is possible to rename “Step1” to “sTeP1” and the change persists)
- Alias names cannot be True or False.

## Limitations

A Sequencer Object has the following limitations:

- maximum of 1000 steps
- maximum of 1000 Aliases
- maximum of 250 outputs (OnEntry and OnExit) per step

- maximum of 10,000 outputs (OnEntry and OnExit) across all steps

---

# Chapter 2

## Defining Steps

This section shows you how to configure the steps of your Sequencer Object step program.

Using the Step Program panel, you can

- add steps to the step program at any position
- configure or modify steps in the step program, such as the conditions or Aliases
- change the order of steps in the step program
- validate steps in the step program for correct syntax and structure
- delete steps from the step program.

## Creating Steps

You can either add steps by appending them to the end of the step program or by inserting them at any given location in the step program.

---

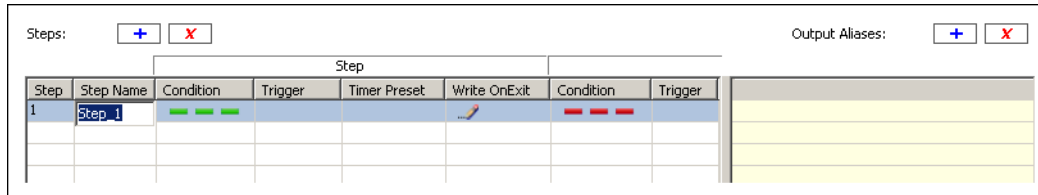
**Note** You can create up to 1000 steps in a single Sequencer Object.

---

**To add steps to a Step Program**

- 1 Click the Step Program tab. The Step Program panel appears.

- Click the + button next to **Steps**. A template line is added to the step program.



## Configuring Steps

Configuring a step tells the step program what to do when this step becomes active. You can configure the following:

Step Name	Arbitrary name you can use for easier identification.
Step Section	
Condition	The combination of a trigger and/or timer required for the step to be executed. The condition is evaluated at each scan cycle until the condition is fulfilled, or any other action is taken.
Trigger	The trigger that is associated with the condition. The trigger is represented by an Alias which is defined in the <b>Aliases</b> panel.
Timer Preset	The time setting that is associated with the step condition. This can either be an actual time or an offset.
Write On Exit	Determines whether the step program writes the On Exit values to the outputs when the step condition is met.
Jump Section	
Condition	The combination of a trigger and/or timer required for Sequencer to jump to a different step. The jump condition is evaluated at each scan cycle. It causes Sequencer to jump, if the jump condition is met.
Trigger	The trigger that is associated with the condition. The trigger is represented by an Alias which is defined in the <b>Aliases</b> panel.
Timer Preset	The time setting that is associated with the jump condition. This can either be an actual time or an offset.



Write On Exit	Determines whether the step program writes the On Exit values to the outputs when the jump condition is met.
Jump To	The name of the step Sequencer jumps to, if the jump condition is met.

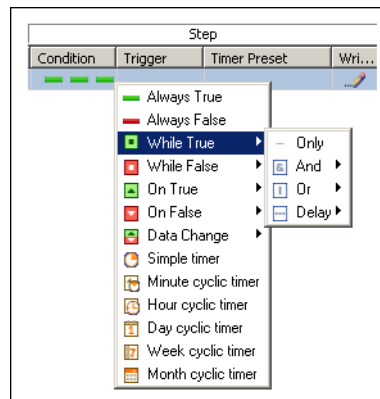
#### To configure or modify a step name

- 1 Click the **Step Program** tab. The **Step Program** panel appears.
- 2 Locate the step to configure or change.
- 3 Click the step name cell of that step. Enter a new step name and press Enter.

Step	Step Name	Condition	T
1	Initialize	— — —	

#### To configure or modify the step or jump condition

- 1 Click the **Step Program** tab. The **Step Program** panel appears.
- 2 Locate the step to configure or modify the condition.
- 3 Click the **Condition** cell of that step. Select the trigger and/or timer.

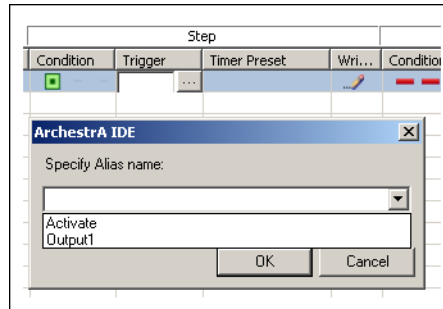


For more information about the types of triggers and timers, see *Defining Step and Jump Conditions* on page 37.

#### To assign a step or jump trigger to a condition

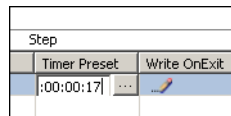
- 1 Click the **Step Program** tab. The **Step Program** panel appears.
- 2 Locate the step to configure or modify the condition.

- 3 In the Trigger cell of either the Step section or the Jump section, enter the name of the Alias or select it from the list.



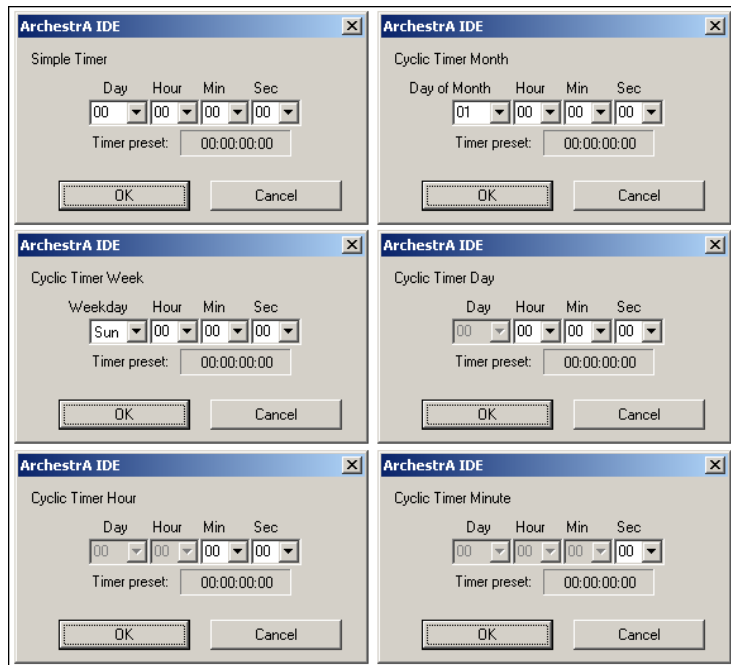
#### To configure or modify the Step Timer Preset or Jump Timer Preset

- 1 Click the **Step Program** tab. The **Step Program** panel appears.
- 2 Locate the step to configure or modify the timer preset.
- 3 Click on the **Step Timer Preset** or **Jump Timer Preset** cell of that step.



- 4 Click the **Browse** button. Depending on the type of timer you selected in the timer condition, one of the following ArchestrA IDE dialog boxes appears:
  - Simple Timer
  - Cyclic Timer Month
  - Cyclic Timer Week
  - Cyclic Timer Day
  - Cyclic Timer Hour

- Cyclic Timer Minute



Select the time to set as a timer preset in the **Simple Timer** or **Cyclic Timer** dialog box. You can also manually enter the time. The format is: dd.hh.mm.ss (dd=days, hh=hours, mm=minutes, ss=seconds).

Click **OK**.

- 5 When you are done, press **Enter**.

---

**Note** In the case of **Cyclic Timer Month**, the time setting is a time offset from the beginning of the month. For example: A cyclic timer setting for the 12th day of each month at 2:35:07 PM would be 11:14:35:07.

---

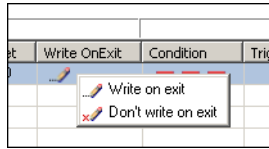
**Note** Sunday is considered the first day of the week.

---

To configure or modify whether **On Exit** values are written when the step is executed or a jump is initiated

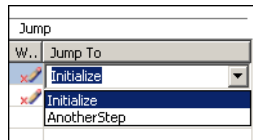
- 1 Click the **Step Program** tab. The **Step Program** panel appears.
- 2 Locate the step to configure or modify the **Write OnExit** property.

- 3 Click on the icon. From the list, select either **Write on Exit** or **Don't write on Exit**.



To configure the location where Sequencer jumps to

- 1 Click the **Step Program** tab. The **Step Program** panel appears.
- 2 Locate the step to configure or modify the jump location.
- 3 Click on the **Jump To** cell for that step. Select a step name from the list. You can also type in the name of the step Sequencer jumps to.



## Renaming Steps

You can rename your steps at any time. Jump references to the renamed step are automatically updated by the step program editor.

To rename a step

- 1 Click the **Step Program** tab. The **Step Program** panel appears.
- 2 Locate the step to change. Click the step name cell of that step.
- 3 Enter a new step name and press **Enter**.

## Appending Steps

You can add additional steps to your step program at the end of the step program. Any step can be moved to any position within the step program by dragging and dropping.

To add additional steps to a Step Program

- 1 Click the **Step Program** tab. The **Step Program** panel appears.

- 2 Click the **Add** button next to **Steps**. An additional template line is added to the step program.

Steps: + x

Step	Step Name	Step				Jump	
		Condition	Trigger	Timer Preset	Write OnExit	Condition	Trigger
1	Initialize	--		00:00:00:00			Alias_1
2	Step_3	---				---	
3	AnotherStep2	---				---	
4	Step_4	---				---	
5	Step_5	---				---	

## Inserting Steps

You can add additional steps to your step program at any position in the step program. You can drag any step to a new position.

To insert steps in a Step Program

- 1 Click the **Step Program** tab. The **Step Program** panel appears.
- 2 Right-click on the line in the step program above where you want to insert an additional step. A context menu appears.

Step	Step Name	Step		
		Condition	Trigger	Timer
1	Step_1	---		
2	Step_2	---		
3	Step_3	---		
4	Step_4	---		
5	Step_5	---		
6	Step_6	---		

Insert Step  
 Append Step  
 Delete Step  
 Edit Outputs

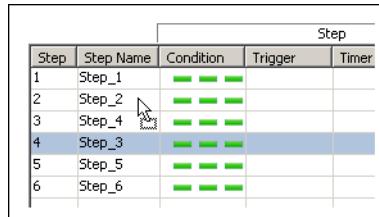
- 3 Click **Insert Step** on the context menu. A template line is added to the step program at the selected position and the other steps in the step program are re-arranged accordingly.

## Changing the Position of Steps

The position of the steps in the step program is crucial to the flow of the step program logic. After you create or update a step program, some steps may not be in the correct position. You can change the position of steps in the step program with the Sequencer object editor.

**To change the position of steps**

- 1 Click the **Step Program** tab. The **Step Program** panel appears.
- 2 Locate the step to move.
- 3 Click on the **Step Number** cell of that step. Drag it to its destination line.




Step	Step Name	Condition	Trigger	Timer
1	Step_1	■■■■	■■■■	
2	Step_2	■■■■	■■■■	
3	Step_4	■■■■	■■■■	
4	Step_3	■■■■	■■■■	
5	Step_5	■■■■	■■■■	
6	Step_6	■■■■	■■■■	

## Deleting Steps

You can delete single steps either by the delete icon or by the context menu. You can also delete the whole step program at once.

**To delete single steps from your step program**

- 1 Click the **Step Program** tab. The **Step Program** panel appears.
- 2  Select the step to delete by clicking on the **Step Number** cell. Click the **Delete** button.

You are asked to confirm this action as no undo is available.

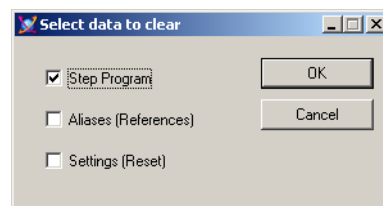
---

**Caution** Before you start, make sure you correctly select the steps you want to delete. After they are deleted, you cannot undelete them.

---

**To delete all steps from a step program**

- 1 Click the **Step Program** tab. The **Step Program** panel appears.
- 2 Click the **Clear** button at the top of the panel. The **Select data to clear** dialog box appears.



- 3 Select the **Step Program** check box and click **OK**.  
You are asked to confirm this action as no undo is available.

## Defining Minimal Steps

You can define minimal steps to act as placeholders for steps that are to be included in the future. In programming these steps are also known as No Operation steps (NOP). These minimal steps take exactly two scan cycles to execute.

### To define a minimal step

- ◆ Create a new step and make sure that
  - the condition is set to **Always True**
  - there are no values in the output matrix that belong to this step

---

**Note** You can configure Sequencer to skip No Operation steps by configuring an **Always True** jump condition on the step before the first No Operation step. The jump destination can be the step just after the last No Operation step.

---

## Defining Initial Step and Final Step

You can instruct Sequencer to begin the program at a specified step, the **Initial Step**. You can also instruct Sequencer to finish processing after a specified step, the **Final Step**.

When you deploy the Sequencer object or when the **ExecutionStateCmd** is set to **Reset**, Sequencer starts at the Initial step and runs through the remaining steps. If the Final Step is not specified, it runs through Step n and continues at Step 1.

---

**Note** You can change the initial step and final step at run time by writing to the attributes **PrgStepInitialCmd** and **PrgStepFinalCmd**. For more information, see **Setting Initial Step and Final Step** on page 119.

---

### To configure the Step Program to start at a specific step

- 1 Click the **Step Program** tab. The **Step Program** panel appears.

- From the **Initial Step Name** list, select the step name where you want to start the processing or enter the name manually.

To configure the Step Program to stop after a specific step

- Click the **Step Program** tab. The **Step Program** panel appears.
- From the **Final Step Name** list, select the step name you want the processing to stop after (or enter the name manually).

## Naming the Step Program

You can give your step program a meaningful name to help you identify it. The Step Program name does not have any functional significance.

To name the Step Program

- Click the **Step Program** tab. The **Step Program** panel appears.
- Enter a name for your Step Program in the **Step Program Name** text field.

---

**Note** For more information, see the section on Steps and Aliases Naming Conventions as it also applies to the Step Program Name.

---



## Validating the Step Program

Step Validation helps you track down errors by checking the syntax of your step program and reporting any missing settings. The results of the validation appears in the **Sequencer Program Validation Output** area at the bottom of the Sequencer Editor panel.

Common errors are:

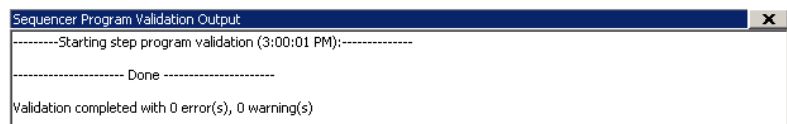
- If a Jump Trigger or Jump Timer Mode is assigned to a step with out a Jump Destination Step being assigned
- If transitions that are configured as Retentive or Non-Retentive Delay do not have Triggers assigned
- If “On True”, “On False” and “Data Change” Trigger Types are used for Non-Retentive or Retentive Timer modes
- If Jump Trigger or Jump Timer Mode is selected for each step and Jump Step Name is blank

Common warnings are:

- Input and Outputs references are empty.
- Input and Outputs references configured as “---.---“

To validate the step program

- 1 Click the **Step Program** tab. The **Step Program** panel appears.
- 2 Click the **Validate** button.
- 3 Check for possible errors and warnings in the **Sequencer Program Validation Output** area at the bottom of the **Sequencer Editor** panel.



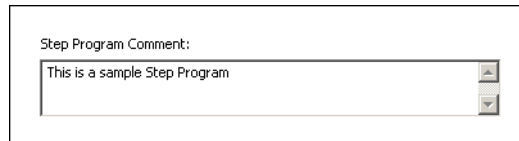
**Caution** If you update the attributes PrgStepProgram and PrgAliasConfig with GRAccess, invalid entries, such as duplicate or invalid step names are not detected by the Sequencer object. Make sure you only pass valid Sequencer configuration to these attributes when using GRAccess.

## Adding Comments to Sequencer Objects

You can give your step program a meaningful comment to help you describe it. The Step Program comment does not have any functional significance.

To add or modify a comment of a Step Program

- 1 Click the **Step Program** tab. The **Step Program** panel appears.
- 2 In the **Comment** box, enter a comment.

A screenshot of a software interface showing a text input field for a comment. The label "Step Program Comment:" is positioned above the field. The field contains the text "This is a sample Step Program". To the right of the text area are two small, vertically aligned arrow buttons (up and down) for scrolling.

## Locking the Step Program

You can lock your step program so that

- the step program cannot be edited in derived instances and templates.
- changes are propagated from templates to instances.
- users cannot change the step program configuration at run time.

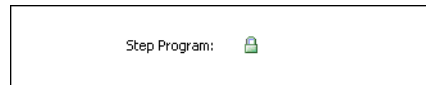
The following settings are locked if the step program is locked:

- Step Program Name
- Initial Step Name
- Final Step Name
- Step Program Comment
- Configuration of all steps
- Output values of all steps
- Importing Step Program
- Clearing Step Program
- Removing and renaming of Aliases that are used in the step program, even if the Alias configuration is not locked.

To lock the Step Program

- 1 Click the **Step Program** tab. The **Step Program** panel appears.

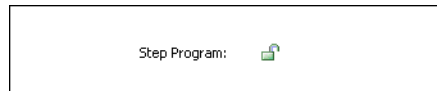
- 2 Click the Lock icon near the Step Program entry so that it is locked.



The step program for any derived templates or instances from the Sequencer Object template are locked. Any attributes related to the step program cannot be changed.

#### To unlock the Step Program

- 1 Click the **Step Program** tab. The **Step Program** panel appears.
- 2 Click the Lock icon near the Step Program entry so that it is unlocked.



---

**Note** When either the Step Program or the Alias configuration is locked, certain constraints apply. When the Step Program template is locked, an Alias referenced by the Step Program cannot be deleted or renamed and importing an Alias configuration that does not include all of the Aliases referenced by the Step Program will fail.

---

---

**Caution** When the Step Program is locked, at run time, attempts to change it by the Selected.\* attributes or the PrgSeqConfigCmd attribute are not allowed.

---



# Chapter 3

## Defining Step and Jump Conditions





This section shows you how to configure conditions for your Sequencer Object step program.

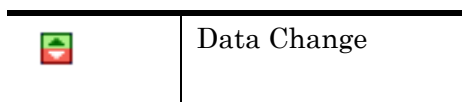
### Defining Steps with Trigger-Based Conditions

You can use triggers to control the processing of each step. Triggers are associated with Aliases that reference attributes.

To configure a step with a trigger

- 1 Click the **Step Program** tab. The Step Program panel appears.
- 2 Locate the step for which you want to change the step condition or jump condition.
- 3 Click the **Condition** cell or **Jump Condition** cell of that step. From the menu, select either **While True**, **While False**, **On True**, **On False** or **Data Change**.

	While True
	While False
	On True
	On False



- 4 Click the **Trigger** cell to the right of the step condition cell or jump condition cell.
- 5 From the list, select the Alias you want to use as a trigger or enter and define a new Alias.


## Defining Steps with Timers

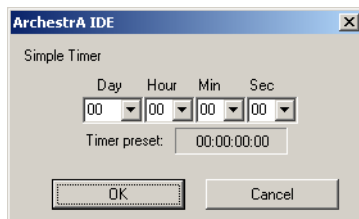
You can configure a step with a timer that only delays step execution. This timer is called a **Simple Timer**. You can use the Simple Timer to make Sequencer wait a certain time period before writing values to the On Exit outputs.

**Tip** In the Timer Preset cell, you can enter the time delay in dd:hh:mm:ss (dd=days, hh=hours, mm=minutes, ss=seconds).

**Note** The time format in the Sequencer editor is dd:hh:mm:ss does not correspond to the ElapsedTime format of the equivalent attribute (such as seen in Object Viewer), which is “d hh:mm:ss”.

### To configure steps with a simple timer

- 1 Click the **Step Program** tab. The **Step Program** panel appears.
- 2 Locate the step for which you want to change the step condition or jump condition.
-  3 Click the **Condition** cell in the Step section or Jump section for that step and select **Simple Timer**.
- 4 Click the **Timer Preset** cell to the right of the condition cell.
- 5 Click the **Browse** button. The **Simple Timer** dialog box appears.








- 6 Configure the timer and click **OK**.

## Defining Steps with a Cyclic Timer

You can configure a step condition or jump condition to be fulfilled on a periodic basis. For example, you can specify the 3rd day, at 10:12:40 each month or 42 seconds after every minute. These timers are called **cyclic timers**.

Timers can also be combined with Boolean attributes. For more information, see *Defining Combinations of Triggers and Timers* on page 44.

There are 5 different cyclic pulse timers.

 Minute-based Timer	On for one scan once a minute at the top of the minute plus the preset (xx:xx:xx:00)
 Hourly Timer	On for one scan once an hour at the top of the hour plus the preset (xx:xx:00:00)
 Daily Timer	On for one scan once a day at midnight plus the preset (xx:00:00:00)
 Weekly Timers	On for one scan once a week on Sunday at midnight plus the preset (00:00:00:00)
 Monthly Timers	On for one scan once a month on the first day of the month at midnight plus the preset (00:00:00:00)

---

**Note** The time format in the Sequencer editor is dd:hh:mm:ss does not correspond to the ElapsedTime format of the equivalent attribute (such as seen in Object Viewer), which is “d hh:mm:ss”.

---

**Note** If the Day, Week or Month Cyclic Timer is configured to have an offset of 1 hour (timer event occurs at 1am) and if the step is entered between 1 AM and 2 AM on the day of the fallback daylight saving time schedule, then when the time rolls back from 2 AM to 1 AM, the timer condition is not fired at 1 AM. To work around this, configure the Cyclic Timer to have an offset of 1 hour and 1 second.

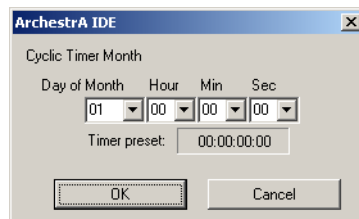
---

## Defining Steps with Monthly Timers

You can use monthly timers to fulfil a step condition or jump condition once a month. You can specify at what time precisely you want the step condition or jump condition to be fulfilled.

**To configure a step with a monthly timer**

- 1 Click the **Step Program** tab. The **Step Program** panel appears.
- 2 Locate the step for which you want to change the step condition or jump condition.
- 3 Click the **Condition** cell or **Jump Condition** cell of that step and select **Cyclic Timer Month**.
- 4 Click on the **Step Timer Preset** or **Jump Timer Preset** cell near the condition cell.
- 5 Click the **Browse** button. The **Cyclic Timer Month** dialog box appears.



- 6 Select the activation time from the pulldown menus and click **OK**.

---

**Note** In the **Timer Preset** field the value for **Day** is calculated as day of month - 1. For example: The 5th day of the month appears as 04:00:00:00 in the **Timer Preset** Field.

---

**Important** Specifying an offset of days greater than or equal to the number of days in the current month causes the event to fire on the last day of the month. However, if the step is entered on the last day of the month and exceeds the timer preset, the value does not fire in this month. For example: An activation time of 30:5:15:10 entered on February 28 at 6:00AM (non leap year) does not fire in February because it exceeds the preset time of 5:15:10. In this scenario, the event triggers on March 31 at 5:15:10.

---

## Defining Steps with Weekly Timers

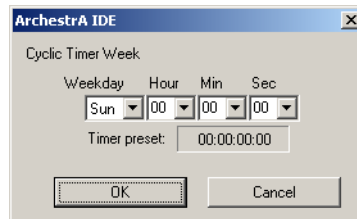
You can use weekly timers to fulfill a step condition or jump condition once a week. You can specify at what time precisely you want the step condition or jump condition to be fulfilled.

**To configure a step with a weekly timer**

- 1 Click the **Step Program** tab. The **Step Program** panel appears.
- 2 Locate the step for which you want to change the step condition or jump condition.
- 3 Click on the **Condition** cell or **Jump Condition** cell of that step and select **Cyclic Timer Week**.



- 4 Click on the **Step Timer Preset** or **Jump Timer Preset** cell near the condition cell.
- 5 Click on the ellipsis. The **Cyclic Timer Week** dialog box appears.



- 6 Select the activation time from the pulldown menus and click **OK**.

---

**Tip** In the Timer Preset cell, you can enter the activation time in DD:hh:mm:ss (DD=Day of the week [00=Sunday, 01=Monday, 02=Tuesday, 03=Wednesday, 04=Thursday, 05=Friday, 06=Saturday], hh=hours, mm=minutes, ss=seconds).

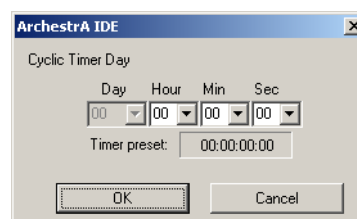
---

## Defining Steps with Daily Timers

You can use daily timers to fulfill a step condition or jump condition once a day. You can specify at what time precisely you want the step condition or jump condition to be fulfilled.

To configure a step with a daily timer

- 1 Click the **Step Program** tab. The **Step Program** panel appears.
- 2 Locate the step for which you want to change the step condition or jump condition.
- 3 Click on the **Condition** cell or **Jump Condition** cell of that step and select **Cyclic Timer Day**.
- 4 Click on the **Step Timer Preset** or **Jump Timer Preset** cell near the condition cell.
- 5 Click on the ellipsis. The **Cyclic Timer Day** dialog box appears.



- 6 Enter the activation time and click **OK**.

---

**Tip** In the Timer Preset cell, you can enter the activation time in xx:hh:mm:ss (xx=ignored, hh=hours, mm=minutes, ss=seconds).

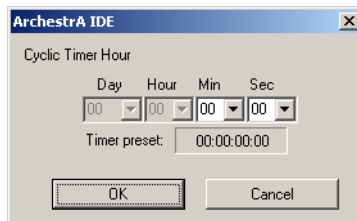
---

## Defining Steps with Hourly Timers

You can use hourly timers to fulfill a step condition or jump condition once an hour. You can specify at what time precisely you want the step condition or jump condition to be fulfilled.

To configure a step with a hourly timer

- 1 Click the **Step Program** tab. The **Step Program** panel appears.
- 2 Locate the step for which you want to change the step condition or jump condition.
- 3 Click on the **Condition** cell or **Jump Condition** cell of that step and select **Cyclic Timer Hour**.
- 4 Click on the **Step Timer Preset** or **Jump Timer Preset** cell near the condition cell.
- 5 Click on the ellipsis. The **Cyclic Timer Hour** dialog box appears.



- 6 Enter the activation time and click **OK**.

---

**Tip** In the Timer Preset cell, you can enter the activation time in xx:xx:mm:ss (xx=ignored, mm=minutes, ss=seconds).

---

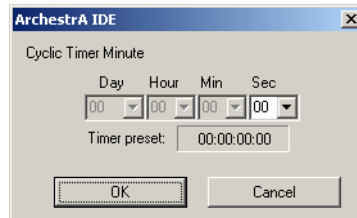
## Defining Steps with Minute-Based Timers

You can use minute-based timers to fulfill a step condition or jump condition once a minute. You can specify at what time precisely you want the step condition or jump condition to be fulfilled.

To configure a step with a minute-based timer

- 1 Click the **Step Program** tab. The **Step Program** panel appears.
- 2 Locate the step for which you want to change the step condition or jump condition.

- 3 Click on the **Condition** cell or **Jump Condition** cell of that step and select **Cyclic Timer Minute**.
- 4 Click on the **Step Timer Preset** or **Jump Timer Preset** cell near the condition cell.
- 5 Click on the ellipsis. The **Cyclic Timer Minute** dialog box appears.



- 6 Enter the activation time and click **OK**.

---

**Tip** In the Timer Preset cell, you can enter the activation time in xx:xx:xx:ss (xx=ignored, ss=seconds).

---

## Defining Steps with Always True or Always False Conditions

You can configure a step so that its step condition or jump condition is always or is never true.

This can be done with the following two conditions:

- **Always True**      The condition always evaluates to true.
- **Always False**      The condition always evaluates to false.

---

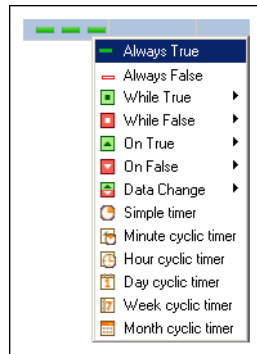
**Note** If both the Step condition and the Jump condition are configured as Always True, then the Step transition occurs since the Step takes precedence over the Jump. On the other hand, if the Step condition and the Jump condition are configured as Always False, then the Sequencer remains in the active step until an Advance ExecutionStateCmd or an StepNumCmd or StepNameCmd is issued.

---

To configure a step or jump to always execute

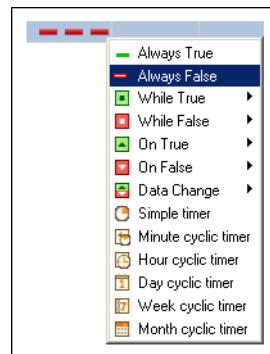
- 1 Click the **Step Program** tab. The **Step Program** panel appears.
- 2 Locate the step for which you want to change the step condition or jump condition.

- 3 Click the **Condition** cell or **Jump Condition** cell of that step and select **Always True**.



To configure a step or jump to never execute

- 1 Click the **Step Program** tab. The **Step Program** panel appears.
- 2 Locate the step for which you want to change the step condition or jump condition.
- 3 Click the **Condition** cell or **Jump Condition** cell of that step and select **Always False**.



## Defining Combinations of Triggers and Timers

Combining triggers and timers gives you more power over the conditional functionality of a step. Besides the triggers **While True**, **While False**, **On True** and **On False** and the simple timers and cyclic timers, you can also define

- logical operator to associate triggers with timers
- retentive and non-retentive timers

## Using Operators

You can use the following operators to logically connect event-based triggers and time-based triggers:



AND

Use this icon to logically connect an event-based trigger and a time-based trigger with the AND operator.



OR

Use this icon to logically connect an event-based trigger and a time-based trigger with the OR operator.



DELAY

Use this icon to logically connect an event-based trigger and a time-based trigger with the DELAY operator.

## Using Retentive and Non-Retentive Timers



Retentive timers count only the time the trigger condition is true. The timer pauses when the trigger condition changes to false and resumes when the trigger condition changes to true again.



Non-retentive timers count only the time the trigger condition is continuously true. If the trigger condition is no longer true, the timer is reset.

## Configuring Steps with Combined Conditions









You can define steps with combinations of triggers and timers. For more information on the numerous combinations of triggers and timers and what the combinations mean, see Overview of Trigger and Timer Combinations on page 46.









To configure steps with combined conditions









- 1 Click the Step Program tab. The Step Program panel appears.
- 2 Locate the step for which you want to change the step condition or jump condition.
- 3 Click the Condition cell or Jump Condition cell of that step and select the combination you want from menus.

## Overview of Trigger and Timer Combinations









The following table shows a complete list of combinations of timers and triggers.









Trigger Condition	The Trigger condition is true, when...
While True AND Simple Timer 	...the trigger is true and a specified time period has passed since step activation
While True AND Minute Cyclic Timer 	...the trigger is true and the specified offset (in seconds) is reached
While True AND Hour Cyclic Timer 	...the trigger is true and the specified offset (in minutes:seconds) is reached
While True AND Day Cyclic Timer 	...the trigger is true and the specified offset (in hours:minutes:seconds) is reached
While True AND Week Cyclic Timer 	...the trigger is true and the specified offset (in days:hours:minutes:seconds) is reached
While True AND Month Cyclic Timer 	...the trigger is true and the specified offset (in days:hours:minutes:seconds) is reached
While True OR Simple Timer 	...the trigger is true or a specified time period has passed since step activation
While True OR Minute Cyclic Timer 	...the trigger is true or the specified offset (in seconds) is reached









Trigger Condition	The Trigger condition is true, when...
While True OR Hour Cyclic Timer	...the trigger is true or the specified offset (in minutes:seconds) is reached
	
While True OR Day Cyclic Timer	...the trigger is true or the specified offset (in hours:minutes:seconds) is reached
	
While True OR Week Cyclic Timer	...the trigger is true or the specified offset (in days:hours:minutes:seconds) is reached
	
While True OR Month Cyclic Timer	...the trigger is true or the specified offset (in days:hours:minutes:seconds) is reached
	
While True DELAY Simple Timer	...the trigger is true and a specified time period has passed after the trigger became true, regardless if the trigger becomes false again
	
While True DELAY Retentive Timer	...the trigger is true and remains true for a specified time period. If the trigger becomes false, this timer is halted and continued when it becomes true.
	
While True DELAY NonRetentive Timer	...the trigger is true and remains true continuously for a specified time period. If the trigger becomes false before this time is reached, the timer is reset.
	
While False AND Simple Timer	...the trigger is false and a specified time period has passed since step activation
	







Trigger Condition	The Trigger condition is true, when...
While False AND Minute Cyclic Timer	...the trigger is false and the specified offset (in seconds) is reached
	
While False AND Hour Cyclic Timer	...the trigger is false and the specified offset (in minutes:seconds) is reached
	
While False AND Day Cyclic Timer	...when the trigger is false and the specified offset (in hours:minutes:seconds) is reached
	
While False AND Week Cyclic Timer	...the trigger is false and the specified offset (in days:hours:minutes:seconds) is reached
	
While False AND Month Cyclic Timer	...the trigger is false and the specified offset (in days:hours:minutes:seconds) is reached
	
While False OR Simple Timer	...the trigger is false or a specified time period has passed since step activation
	
While False OR Minute Cyclic Timer	...the trigger is false or the specified offset (in seconds) is reached
	
While False OR Hour Cyclic Timer	...the trigger is false or the specified offset (in minutes:seconds) is reached
	



Trigger Condition	The Trigger condition is true, when...
While False OR Day Cyclic Timer	...the trigger is false or the specified offset (in hours:minutes:seconds) is reached
	
While False OR Week Cyclic Timer	...the trigger is false or the specified offset (in days:hours:minutes:seconds) is reached
	
While False OR Month Cyclic Timer	...the trigger is false or the specified offset (in days:hours:minutes:seconds) is reached
	
While False DELAY Simple Timer	...the trigger is false and a specified time period has passed after the trigger became false, regardless if the trigger becomes true
	
While False DELAY Retentive Timer	...the trigger is false and remains true for a specified time period. If the trigger becomes true, this timer is halted and continued when it becomes false again.
	
While False DELAY NonRetentive Timer	...the trigger is false and remains false continuously for a specified time period. If the trigger becomes true before this time is reached, the timer is reset.
	
On True AND Simple Timer	...the trigger becomes true from false and a specified time period has passed since step activation
	
On True OR Simple Timer	...the trigger becomes true from false or a specified time period has passed since step activation
	

Trigger Condition	The Trigger condition is true, when...
On True OR Minute Cyclic Timer 	...the trigger becomes true from false or the specified offset (in seconds) is reached
On True OR Hour Cyclic Timer 	...the trigger becomes true from false or the specified offset (in minutes:seconds) is reached
On True OR Day Cyclic Timer 	...the trigger becomes true from false or the specified offset (in hours:minutes:seconds) is reached
On True OR Week Cyclic Timer 	...the trigger becomes true from false or the specified offset (in days:hours:minutes:seconds) is reached
On True OR Month Cyclic Timer 	...the trigger becomes true from false or the specified offset (in days:hours:minutes:seconds) is reached
On True DELAY Simple Timer 	...the trigger becomes true from false and a specified time period has passed since, regardless if the trigger becomes false in the meantime
On False AND Simple Timer 	...the trigger becomes false from true and a specified time period has passed since step activation
On False OR Simple Timer 	...the trigger becomes false from true or a specified time period has passed since step activation

Trigger Condition	The Trigger condition is true, when...
On False OR Minute Cyclic Timer 	...the trigger becomes false from true or the specified offset (in seconds) is reached
On False OR Hour Cyclic Timer 	...the trigger becomes false from true or the specified offset (in minutes:seconds) is reached
On False OR Day Cyclic Timer 	...the trigger becomes false from true or the specified offset (in hours:minutes:seconds) is reached
On False OR Week Cyclic Timer 	...the trigger becomes false from true or the specified offset (in days:hours:minutes:seconds) is reached
On False OR Month Cyclic Timer 	...the trigger becomes false from true or the specified offset (in days:hours:minutes:seconds) is reached
On False DELAY Simple Timer 	...the trigger becomes false from true and a specified time period has passed since, regardless if the trigger becomes true in the meantime
Data Change AND Simple Timer 	...the trigger changes from true to false (or from false to true) and a specified time period has passed since the step became active
Data Change OR Simple Timer 	...the trigger changes from true to false (or from false to true) or a specified time period has passed since step activation

Trigger Condition	The Trigger condition is true, when...
Data Change OR Minute Cyclic Timer	...the trigger changes from true to false (or from false to true) or the specified offset (in seconds) is reached
	
Data Change OR Hour Cyclic Timer	...the trigger changes from true to false (or from false to true) or the specified offset (in minutes:seconds) is reached
	
Data Change OR Day Cyclic Timer	...the trigger changes from true to false (or from false to true) or the specified offset (in hours:minutes:seconds) is reached
	
Data Change OR Week Cyclic Timer	...the trigger changes from true to false (or from false to true) or the specified offset (in days:hours:minutes:seconds) is reached
	
Data Change OR Month Cyclic Timer	...the trigger changes from true to false (or from false to true) or the specified offset (in days:hours:minutes:seconds) is reached
	
Data Change DELAY Simple Timer	...the trigger changes from true to false (or from false to true) and a specified time period passes since, regardless if the trigger changes again
	

---

# Chapter 4

## Defining Outputs for a Step

This section shows you how to define outputs for the steps of your step program.

### Creating an Alias

The steps in your step program use Aliases to reference ArcestrA attributes. Aliases are used to

- write outputs once a step is active (On Entry Write)
- trigger the step or jump condition
- write outputs after the step or jump condition is fulfilled (On Exit Write)
- specify the output value to be written when an Alias appears in the output matrix as a pass by reference value.

You must define an Alias before using it to read from or write to an ArcestrA attribute.

---

**Note** You cannot map an Alias to an ArcestrA attribute from the Step Program tab. Do this on the Aliases tab. For more information, see Assigning an Attribute Reference to an Alias on page 64.

---

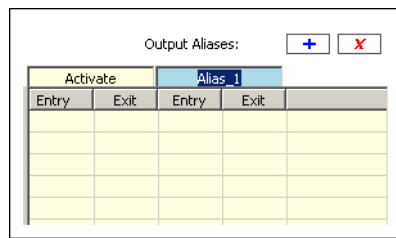
**Note** You can create up to 1000 Aliases in a single Sequencer Object.

---

To create an Alias from the Step Program panel

- 1 Click the Step Program tab. The Step Program panel appears.

- 2 Click on the **Add** button near the Outputs entry. An additional column is added in the **Outputs** area with a default name as column header.



- 3 Enter a name for the new Alias.

---

**Note** You cannot add an Alias if the Alias Configuration (PrgAliasConfig) is locked, even if the Step Program (PrgStepProgram) is unlocked.

---

## Assigning an Output Value to a Step

You can assign output values to each step in your step program. These values are written to the attributes that are mapped to the Aliases.

---

**Note** You can create up to 250 outputs per step and up to 10,000 outputs across all steps.

---

Use:

- **On Entry outputs** to write outputs as soon as the step becomes active.
- **On Exit outputs** to write outputs as soon as the step or jump condition is fulfilled.

Output values can either be literals (numerical values, True/False, date time values, elapsed time values and strings) or Aliases.

---

**Note** Hexadecimal values are not supported.

---

To assign output values to Aliases in steps, you can either use:

- Output matrix
- **Edit Outputs dialog box** - is especially useful if you have many outputs as it simplifies how they are shown.

---

**Note** Unlike the output matrix, you cannot use the **Edit Outputs** dialog box to add or rename Aliases.

---

## Assigning a Literal Output Value

You can either use the output matrix or the **Edit Outputs** dialog box to assign a literal output value in a step.

Literal values can be:

- numerical values, such as 3.141, 1000 or -141.22
- boolean values, such as TRUE or FALSE
- string values (enclosed by double quotes), such as “Start” or “Machine 1” or “John Smith”
- time values (enclosed by double quotes), such as “5/30/2006 8:37:43.504 PM”
- elapsed time values (enclosed by double quotes), such as “14:32:55.7328740”

### Assigning a Literal Output Value in the Output Matrix

You can write literal values to On Entry or On Exit outputs for any step in the step program. These values are specified in the output matrix.

To assign a literal value to a step as output value in the output matrix

- 1 Click the **Step Program** tab. The **Step Program** panel appears.
- 2 Locate the cell you need to place the output value at, which is a combination of:
  - a the step for which you want to assign the output value
  - b the Alias which references the ArchestrA attribute you want to write to
  - c On Entry or On Exit, depending if you want the output value to be written when the step becomes active or when the step or jump condition is fulfilled
- 3 Click the located cell, enter the output value and press **Enter**.

Entry	Exit	Entry
1	11	123.45
		"Hello World"
		"5/16/2006 10:40:34.109 AM"
		"00:10:30.000"
		Alias_3

**Important** String, date time or elapsed time data type values need to be enclosed with double quotes (“). Otherwise, they are interpreted as Aliases.

### Assigning a Literal Output Value in the Edit Outputs Dialog Box

You can write literal values to On Entry or On Exit outputs for any step in the step program. These can be entered in the Edit Outputs dialog box.

**Note** The Edit Outputs dialog box only lists Aliases that have output values assigned to them for the step the dialog is opened from.

To assign a literal value to a step as output value in the Edit Outputs dialog box

- 1 Click the Step Program tab. The Step Program panel appears.
- 2 Right-click the step you want to assign output values to and select **Edit Outputs**. The Edit Outputs dialog box appears.

No.	Alias name	Value
1	Output1	20
2	Output2	3.4

No.	Alias name	Value
1	Output1	5.5
2	Output2	"Run"

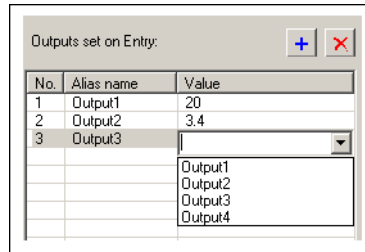


- 3 If the Alias you want to use does not appear in the list, click on either add button depending if you want to assign an OnEntry or an OnExit value. Then select the Alias from the pulldown list that appears in the list.

No.	Alias name	Value
1	Output1	20
2	Output2	3.4
3	Output3	



- 4 Select the Alias you want to assign a value to as output and then click the value cell. A pulldown menu appears.



- 5 Enter a value in the field and press Enter.
- 6 When you are done, click OK to close the **Edit Outputs** dialog box. All values are updated in the output matrix.

---

**Note** When you include an Alias in the **Edit Outputs** dialog box it may appear at the beginning, middle or the end of the list. The order of Aliases in the **Edit Outputs** dialog box depends on the order of Aliases on the **Aliases** panel.

---

## Assigning the Value of another Alias as Output Value

You can either use the output matrix or the **Edit Outputs** dialog box to assign the value of another Alias as output value in a step.

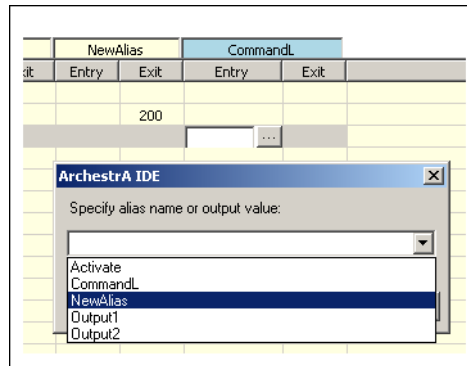
### Assigning the Value of another Alias as Output Value in the Output Matrix

You can configure a step so that the current value of one Alias is written to another Alias when that step becomes active or the step or jump condition is fulfilled.

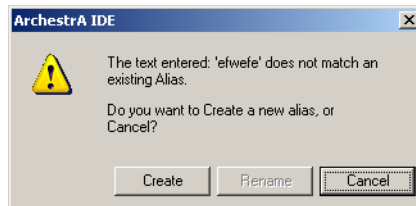
To assign the value of another Alias as output value

- 1 Click the **Step Program** tab. The **Step Program** panel appears.
- 2 Locate the cell you need to place the output value at, which is a combination of:
  - a the step for which you want to assign the output value
  - b the Alias which references the ArchestrA attribute you want to write to
  - c On Entry or On Exit, depending if you want the output value to be written when the step becomes active or when the step or jump condition is fulfilled
- 3 Click the located cell.

- 4 Select the Alias from the pulldown list or type the name in manually and press Enter.



- 5 If you enter an Alias name that does not exist, a confirmation dialog appears asking you to create the Alias.



- 6 Click Create to create the Alias.

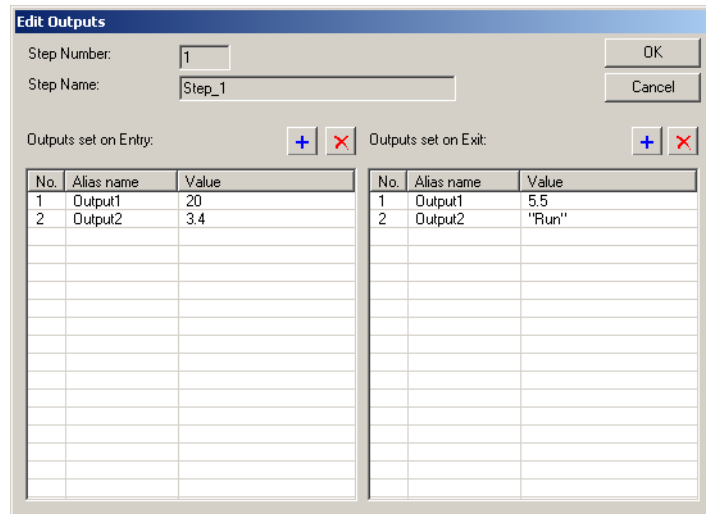
### Assigning the Value of another Alias as Output Value in the Edit Outputs Dialog Box

You can write literal values to On Entry or On Exit outputs for any step in the step program. These can be entered in the Edit Outputs dialog box.

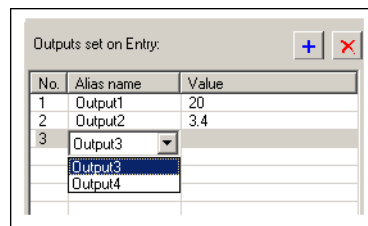
To assign a literal value to a step as output value in the Edit Outputs dialog box

- 1 Click the Step Program tab. The Step Program panel appears.

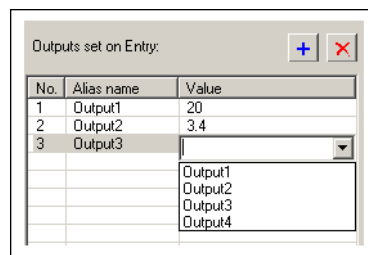
- Right-click the step you want to assign values of Aliases to and select **Edit Outputs**. The **Edit Outputs** dialog box appears.



- If the Alias you want to use does not appear in the list, click on either appear in the list, click on either add button depending if you want to assign an OnEntry or an OnExit Alias. Then select the Alias from the pulldown list that appears in the list.



- Select the Alias you want to assign a value to as output and then click the value cell. A pulldown menu appears.



- Select an Alias from the pulldown menu and press Enter.
- When you are done, click **OK** to close the **Edit Outputs** dialog box. All Aliases are updated in the output matrix.

## Removing Output Values

You can either use the output matrix or the **Edit Outputs** dialog box to remove output values from a step.

### Removing Output Values from the Output Matrix

If you do not want a step to write to an Alias you can blank out the corresponding cell in the output matrix.


To remove output values from the output matrix

- 1 Click the **Step Program** tab. The **Step Program** panel appears.
- 2 Locate the cell you need to want to remove the output value from. This cell is at the intersection of:
  - a the step for which you want to remove the output value
  - b the Alias which references the Arcestra attribute you want to remove the output value for
  - c On Entry or On Exit
- 3 Click the located cell, delete the contents and press Enter.

### Removing Output Values from the Edit Outputs Dialog Box

If you do not want a step to write to an Alias you can remove it from the **Edit Output** dialog box.

To remove output values from the Edit Outputs dialog box

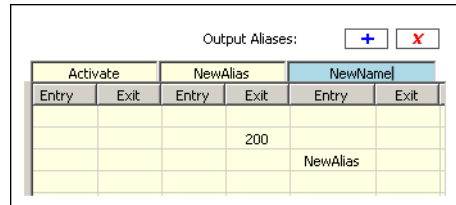
- 1 Click the **Step Program** tab. The **Step Program** panel appears.
- 2 Right-click the step you want to assign values of Aliases to and select **Edit Outputs**. The **Edit Outputs** dialog box appears.
- 3 Select the Alias assigned to the value that you want to remove.
- 4  Click the delete button of the corresponding list heading. The Alias and value are removed from the list.
- 5 Click **OK** to dismiss the **Edit Outputs** dialog box. The value is also removed from the output matrix.

## Renaming Aliases

You can rename Aliases at any time. When you rename an Alias, the Step Program Editor updates all references in the Sequencer Object.

To rename an Alias (from the Step Program panel)

- 1 Click the **Step Program** tab. The **Step Program** panel appears.
- 2 Locate the Alias from the **Output Matrix** and select its header.
- 3 Enter a new name for the Alias.



The name of the Alias is updated by the Step Program Editor in the **Output Matrix**, in the **Jump To** list and on the **Aliases** panel.

---

**Note** You cannot rename an Alias if the Alias Configuration (PrgAliasConfig) is locked, even if the Step Program (PrgStepProgram) is unlocked.

---

## Deleting Aliases

You can delete an Alias either from the Step Program panel or from the Alias panel. You cannot delete an Alias if it is used in a step condition or a jump condition.

When you delete the Alias, it is removed from


- the Step Program panel (as column in the output matrix)
- the Aliases panel Alias definition list
- in pulldown menus that use Aliases (such as triggers)

---

**Note** When you delete an Alias, you cannot undelete it. You must recreate it. If you delete an Alias that has output values associated with it, these values are deleted as well.

---

To delete an Alias from the Step Program panel

- 1 Click the **Step Program** panel. The **Step Program** panel appears.
- 2 Locate the Alias from the **Output Matrix** and select its header.
-  3 Click the **Delete** button at the top of the **Output Matrix**. The Alias is deleted from the **Output Matrix**, the **Jump To** lists and also from the **Aliases** panel.

---

**Note** You cannot delete an Alias if the Alias Configuration (PrgAliasConfig) is locked, even if the Step Program (PrgStepProgram) is unlocked.

---

---

# Chapter 5

## Using Alias Definitions

This section shows you how to work with Alias definitions. The list of Alias definitions shows you which Alias is mapped to which attribute. You can:

- create an Alias
- assign an attribute reference to an Alias
- rename an Alias
- change the order of appearance and execution of Aliases
- delete an Alias
- delete attribute references of Aliases
- lock the Alias definition list
- view Alias usage, such as references and cross references

### Creating an Alias

You can create a new Alias either from the Step Program panel when you are creating and configuring steps, or you can create it from the Aliases panel directly. When you create an Alias, it appears:

- on the **Step Program** panel as a new column in the output matrix
- on the **Aliases** panel as a new item in the Alias definition list
- in the menus where Aliases can be selected, such as Trigger columns.

To create an Alias from the Aliases panel

- 1 Click the **Aliases** tab. The **Aliases** panel appears.
- 2 Click the **Add** button next to **Alias Definitions**. A template line is added to the **Alias definition** list.

Index	Name	Attribute	Input Refs	OnEntry Refs	OnExit Refs
1	Alias_1	---	0	0	0

- 3 Enter a name for the created Alias and press **Enter**. The Alias is created and appears in the Alias definition list and also as a new column of the output matrix on the **Step Program** panel.

## Assigning an Attribute Reference to an Alias

You can assign an attribute reference for every Alias you create. Assigning an attribute to an Alias means that

- any value associated with the Alias is written to the assigned attribute
- when the step program uses the Alias as a trigger, the value of the assigned attribute acts as trigger value

To assign an attribute reference to an Alias

- 1 Click the **Aliases** tab. The **Aliases** panel appears.
- 2 Locate the Alias to which you want to assign the attribute reference.
- 3 Click on the attribute cell of the located row.
- 4 Either enter an attribute reference manually and press **Enter**, or click the ellipsis button and use the **Attribute** browser to select the attribute:

## Renaming an Alias

You can rename an Alias at any time. You can rename an Alias either from the **Step Program** panel or from the **Aliases** panel. When you rename the Alias, it is updated

- on the **Step Program** panel
- on the **Aliases** panel
- in menus that use Aliases, such as **Triggers**



To rename an Alias from the Aliases panel

- 1 Click the Aliases tab. The Aliases panel appears.
- 2 Locate the Alias you want to rename and click on the Name cell of that row.
- 3 Enter a new name and press Enter. The Alias is renamed.

## Changing the Appearance Order of Aliases

You can change the order in which Aliases appear on the Aliases panel according to your own preference. Changing the order of the Aliases on the Aliases panel also changes the order of the Aliases columns in the output matrix on the Step Program panel.

To change the order of Aliases

- 1 Click the Aliases tab. The Aliases panel appears.
- 2 Locate the Alias that you want to move to a different position and click on the Index cell of that row.
- 3 Drag and drop the Alias to a row in the list where you want it to appear. The Alias moves there, the other Aliases are arranged accordingly.

## Deleting Aliases

You can either delete a single Alias, all Aliases or clear their references.

### Deleting an Alias

You can delete an Alias either from the Step Program panel or from the Aliases panel. You cannot delete an Alias if it is used in a step condition or in a jump condition.

When you delete the Alias, it is removed from

- the Step Program panel (as a column in the output matrix)
- the Aliases panel Alias definition list
- pulldown menus that use Aliases (such as triggers and output matrix cells)
- step output values associated with the Alias are also deleted


If the Alias is referenced to an attribute, you cannot delete it.

---

**Note** When you delete an Alias, make sure you select the correct one. After you delete an Alias, you cannot undelete it. You must recreate it.

---

To delete an Alias from the Aliases panel

- 1 Click the **Aliases** tab. The **Aliases** panel appears.
- 2 Locate the Alias you want to delete and select it.
-  3 Click the **Delete** button at the top of the panel. The Alias is deleted from the list.

You are asked to confirm this action as no undo is available.

## Clearing Aliases

You can clear Alias from either the Step Program, Aliases or Settings panel. When you clear all Aliases

- all unused Aliases are deleted
- the attribute references for used Aliases are reset to “---.-.”

When unused Aliases are cleared, they are removed from

- the Step Program panel (as a column in the output matrix)
- the Aliases panel Alias definition list
- in pulldown menus that use Aliases (such as triggers and output matrix cells)

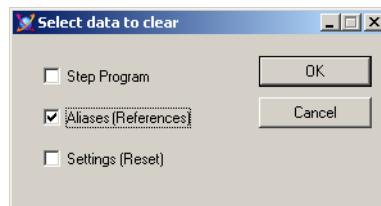
---

**Note** Clearing all Aliases cannot be undone.

---

To clear all Aliases.

- 1 From either the Step Program, Aliases or Settings panel, click **Clear**. The **Select data to clear** dialog box appears.



- 2 Make sure **Aliases (References)** is selected and click **OK**. You are asked to confirm this action as no undo is available.

## Locking the Alias Definition List

You can lock the Alias definition list so that:

- the Aliases in derived instances and templates cannot be edited
- users cannot change the Alias configuration at run time
- changes are propagated from templates to instances

The following actions are not allowed if the Alias definition list is locked:

- Adding, renaming or deleting Aliases
- Assigning attribute references to Aliases
- Importing Aliases

---

**Note** When either the Step Program or the Alias configuration is locked, certain constraints apply. When the Step Program template is locked, an Alias referenced by the Step Program cannot be deleted or renamed. Importing an Alias configuration that does not include all of the Aliases referenced by the Step Program fails.

---

## Viewing Alias References

You can view the following usage data for each Alias:

- **Input Refs:** Shows the number of times the Alias is used in the Step Program either as a trigger or for passing output values by reference.
- **OnEntry Refs:** Shows how many steps the Alias is assigned an On Entry value.
- **OnExit Refs:** Shows in how many steps the Alias is assigned an On Exit value.

To view Alias usage

- 1 Click the Aliases tab. The Aliases panel appears.
- 2 View the 3 columns on the right hand side of the Alias list.

Index	Name	Attribute	Input Refs	OnEntry Refs	OnExit Refs
1	Activate	MyContainer.Activate	1	1	0
2	InputValve	MyContainer.InputValve	0	3	1
3	Agitator	MyContainer.Agitator	0	3	1
4	OutputValve	MyContainer.OutputValve	0	3	1
5	Contaminated	MyContainer.Contaminated	4	0	0
6	DiscardValve	MyContainer.DiscardValve	0	2	1
7	TankEmpty	MyContainer.TankEmpty	2	0	0
8	TankFull	MyContainer.TankFull	1	0	0

## Viewing Alias Crossreferences

You can see Crossreference usage information on an Alias such as:

- **Step:** Shows the step numbers where the Alias is used.
- **Step Name:** Shows the step name of all steps where the Alias is used.
- **Input Refs:** Shows the number of times the selected Alias is used as a condition trigger (step and jump) and referenced in the output matrix for a particular step.
- **OnEntry:** Shows 1 if the selected Alias is used in an On Entry value assignment, otherwise 0, for a particular step.
- **OnExit:** Shows 1 if the selected Alias is used in an On Exit value assignment, otherwise 0, for a particular step.

To view Alias Crossreference usage

- 1 Click the **Aliases** tab. The **Aliases** panel appears.
- 2 Select an Alias for which you want to see Crossreference usage. The usage information appears at the bottom of the panel.

Index	Name	Attribute	Input Refs	OnEntry Refs	OnExit Refs
1	Activate	MyContainer.Activate	1	1	0
2	InputValve	MyContainer.InputValve	0	3	1
3	Agitator	MyContainer.Agitator	0	3	1
4	OutputValve	MyContainer.OutputValve	0	3	1
5	Contaminated	MyContainer.Contaminated	4	0	0
6	DiscardValve	MyContainer.DiscardValve	0	2	1
7	TankEmpty	MyContainer.TankEmpty	2	0	0
8	TankFull	MyContainer.TankFull	1	0	0

Selected Alias Crossreferences:

Step	Step Name	Input Refs	OnEntry	OnExit
2	TankFilling	1	0	0
3	Mixing	1	0	0
4	Settling	1	0	0
5	Emptying	1	0	0

---

# Chapter 6

## Defining Sequencer Program Settings

This section shows you how to define general Sequencer Program Settings, such as

- Initialization Timeout
- Initial Command
- Alarms on errors and warnings
- Security settings

### Configuring the Initial Command

You can control the way the step program is run after any of the following:

- the Sequencer object is deployed
- the attribute `ExecutionStateCmd` is set to **Reset**
- the attribute `PrgSeqConfigCmd` is set to **Load** and completes successfully
- the attribute `PrgSeqConfigCmd` is set to **Upload** and completes successfully

After deployment, resetting, loading or uploading, Sequencer can either

- run the program: **Start**
- not run the program: **Stop**

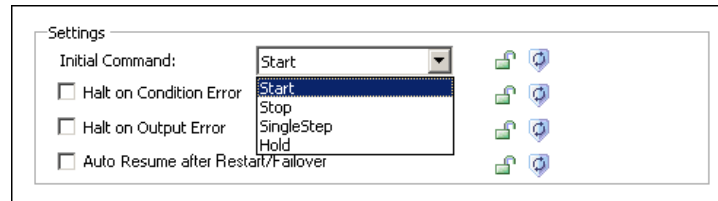
- run the program in RunningSingleStep mode: **Single Step**
- start the program but initially hold processing: **Hold**

---

**Note** At run time, the Program flow can be stopped and controlled in other ways. For more information, see Setting Initial Step and Final Step on page 119.

---

The Initial Command can be set on the Settings panel.



To configure the Step Program to start immediately after deployment, PrgSeqConfigCmd is set to Load or Unload and it is completed successfully or ExecutionStateCmd = Reset

- 1 Click the Settings tab. The Settings panel appears.
- 2 From the Initial Command list, select Start.

To configure the Step Program to not start immediately after deployment, PrgSeqConfigCmd is set to Load or Unload and it is completed successfully or ExecutionStateCmd = Reset

- 1 Click the Settings tab. The Settings panel appears.
- 2 From the Initial Command list, select Stop.

To configure the Step Program to run one step at a time after deployment, PrgSeqConfigCmd is set to Load or Unload and it is completed successfully or ExecutionStateCmd = Reset

- 1 Click the Settings tab. The Settings panel appears.
- 2 From the Initial Command list, select SingleStep.

To configure the Step Program to hold step processing after deployment, PrgSeqConfigCmd is set to Load or Unload and it is completed successfully or ExecutionStateCmd = Reset

- 1 Click the Settings tab. The Settings panel appears.
- 2 From the Initial Command list, select Hold.

---

**Note** The Initial Command can be changed at run time by setting the PrgInitialCommand attribute.

---

## Handling Errors and Reporting Alarms

You can specify how the Sequencer object handles irregular configuration and values at run time. You can:

- **halt program execution** if a condition trigger or Alias in the output matrix has bad quality or cannot be read, or if an Output OnEntry/OnExit Alias cannot be written or if the Sequencer Object is hanging in Initializing state.
- **report alarms** if an output or condition has a syntax error. You can log separate alarms for On Entry output errors and On Exit output errors.

Two types of errors can occur when the Sequencer object is deployed and interacts with ArcestrA attributes:

- **Condition Errors** - a condition error occurs when
  - the object failed to resolve a trigger reference
  - the quality of the trigger reference is not good
  - the data type is mismatched and cannot be handled by Message Exchange
- **Output Errors** - an output error occurs when
  - an output reference cannot be resolved or has bad quality
  - the reference of a pass by value Alias cannot be resolved or has bad quality

---

**Note** This affects the attributes ConditionTriggerFailure, OnEntryOutputFailure and OnExitOutputFailure.

---

Examples:

- configuring a **non-existent** attribute as an output in a step program causes an Output Error
- configuring a **non-numeric** attribute as an input (trigger) in a step program causes a Condition Error
- the quality of the attribute belonging to an input (trigger) or an output is bad.

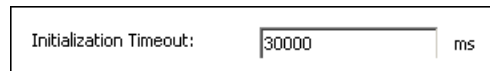
## Configuring Initialization Timeout

When the sequencer starts up, fails over, or loads or updates a new sequence program, its execution state is set to **Initializing** to allow the sequencer to connect to its input and output references.

In certain cases, for example when the Sequencer Object is put offscan during initialization, the Sequencer Object may be stuck in this state. You can configure the Initialization Timeout value to tell the object to go into a StoppedError state after a certain time period has elapsed.

#### To configure the Initialization Timeout

- 1 Click the **Settings** tab. The **Settings** panel appears.
- 2 In the **Initialization Timeout** text field, enter a value for the time period in milliseconds you want to pass before the Sequencer object goes into StoppedError state after initializing.



**Note** The default value for the initialization timeout is 30000 ms (30 seconds).

## Halting Program Execution on Error

The step program can be configured to report an alarm when either an Output Error or a Condition Error occurs. You can configure the Sequencer object to either continue executing or to halt program execution when such errors occur.

**Note** After an error occurs and the step program is configured to “Halt Program Execution on Error”, the program execution is in StoppedError state. You must reset Sequencer or set the **ExecutionStateCmd = Reset** to start processing again.

You can configure the Program Halting on the Settings panel.



#### To halt program execution on Condition Error

- 1 Click the **Settings** tab. The **Settings** panel appears.
- 2 In the **Settings** area, select the check box **Halt On Condition Error**.

#### To halt program execution on Output Error

- 1 Click the **Settings** tab. The **Settings** panel appears.



- 2 In the **Settings** area, select the check box **Halt on Output Error**.

To ignore Condition Errors and Output Errors

- 1 Click the **Settings** tab. The **Settings** panel appears.
- 2 In the **Settings** area, clear the check boxes **Halt On Condition Error** and **Halt on Output Error**.

## Using Alarms to Report Errors

You can configure the Sequencer Object to report an alarm if a Condition or Output error occurs, or if program execution halts for any reason.

This can be configured on the **Settings** panel.

Checkbox	Label	Priority	Lock Icon
<input checked="" type="checkbox"/>	Detect Execution Halted	500	✓
<input checked="" type="checkbox"/>	Detect Condition Trigger Failure	500	✓
<input type="checkbox"/>	Detect OnEntry Output Failure		✗
<input type="checkbox"/>	Detect OnExit Output Failure		✗

To report Program Halting as an alarm

- 1 Click the **Settings** tab. The **Settings** panel appears.
- 2 In the **Alarms** area, select the check box **Detect Execution Halted**.
- 3 Enter a valid priority in the corresponding text box. The range of valid priorities is from 0 to 999.

To report Condition Errors as alarms

- 1 Click the **Settings** tab. The **Settings** panel appears.
- 2 In the **Alarms** area, select the check box **Detect Condition Trigger Failure**.
- 3 Enter a valid priority in the corresponding text box. The range of valid priorities is from 0 to 999.

To report On Entry Output Errors as alarms

- 1 Click the **Settings** tab. The **Settings** panel appears.
- 2 In the **Alarms** area, select the check box **Detect OnEntry Output Failure**.
- 3 Enter a valid priority in the corresponding text box. The range of valid priorities is from 0 to 999.

To report On Exit Output Errors as alarms

- 1 Click the **Settings** tab. The **Settings** panel appears.
- 2 In the **Alarms** area, select the check box **Detect OnExit Output Failure**.
- 3 Enter a valid priority in the corresponding text box. The range of valid priorities is from 0 to 999.

## Configuring Security

You can restrict access to the following attributes of the Sequencer object on the Settings panel:

- **Program Execution** - prevent the user from changing the step program execution
- **Step Setting** - prevent the user from forcing a step jump or from setting the initial step and final step
- **Sequencer Configuration** - prevent the user from changing the Sequencer configuration.
- **Output Array Access** - prevent the user from changing the output values of a selected step
- **Timer Preset Access** - prevent the user from changing the timer preset values of a selected step
- **Jump Destination Access** - prevent the user from changing the jump destination of a selected step

---

**Note** You must enable security before security classification can be enforced.

---

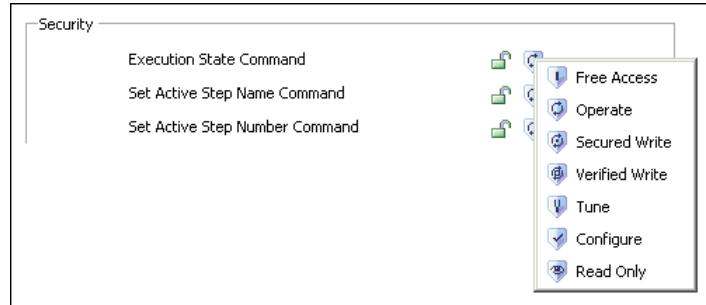
## Modifying Access Rights to the Program Execution Command

The Execution State setting lets you control the execution of the step program, such as stopping, starting, resetting, holding the step program, or running it in Single Step mode. You can prevent the user from changing this in derived templates, instances and at run time by restricting access to it.

To change access rights to the Execution State Command attribute

- 1 Click the **Settings** tab. The **Settings** panel appears.

- 2 Select the appropriate **Execution State Command** security settings using the security classification shield.



This changes your access rights to the **ExecutionStateCmd** attribute.

## Modifying Access Rights to Step Settings

You can prevent the user from instructing the step program to:

- jump to a specific step (at run time) by specifying a Step Number or a Step Name.
- change the Initial Step setting (at run time and in derived templates and instances)
- change the Final Step setting (at run time and in derived templates and instances)

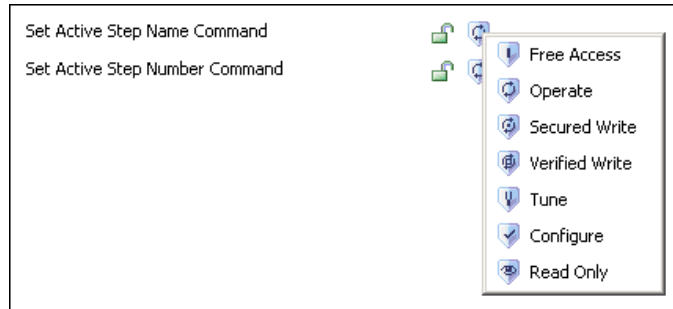
## Modifying Access Rights to the Active Step Command Settings

This section shows you how to change access to the Active Step Command settings.

**To change the access rights to the active step command settings**

- 1 Click the **Settings** tab. The **Settings** panel appears.

- 2 Select the appropriate security settings for either **Set Active Step Name Command** or **Set Active Step Name Command** using the security classification shield.



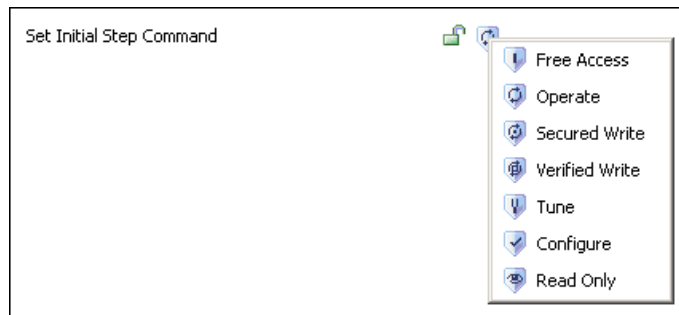
This changes your access rights to the **StepNameCmd** or **StepNumCmd** attributes.

### Modifying Access Rights to the Initial Step Command Setting

This section shows you how to change access rights to the Initial Step command setting.

To change the access rights to the initial step command setting

- 1 Click the **Settings** tab. The **Settings** panel appears.
- 2 Select the appropriate **Set Initial Step Command** security settings using the security classification shield.



This changes your access rights to the **PrgStepInitialCmd** attribute.

### Modifying Access Rights to the Final Step Command Setting

This section shows you how to change access rights to the Final Step command setting.

To change the access rights to the final step command setting

- 1 Click the **Settings** tab. The **Settings** panel appears.

- 2 Select the appropriate **Final Step Command** security settings using the security classification shield.



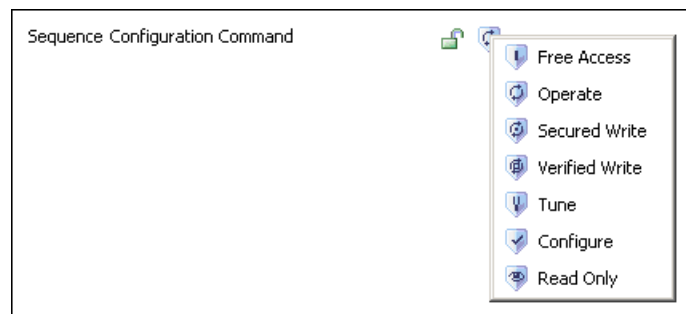
This changes your access rights to the **PrgStepFinalCmd** attribute.

## Modifying Access Rights to the Sequence Configuration Command

You can prevent the user from using the Sequence Configuration command at run time. This is done by restricting access to the attribute **PrgSeqConfigCmd**.

To change the access rights to the Sequence Configuration command

- 1 Click the **Settings** tab. The **Settings** panel appears.
- 2 Select the appropriate **Sequencer Configuration Command** security settings using the security classification shield.



## Modifying Access Rights to the Output Value Arrays of a Selected Step

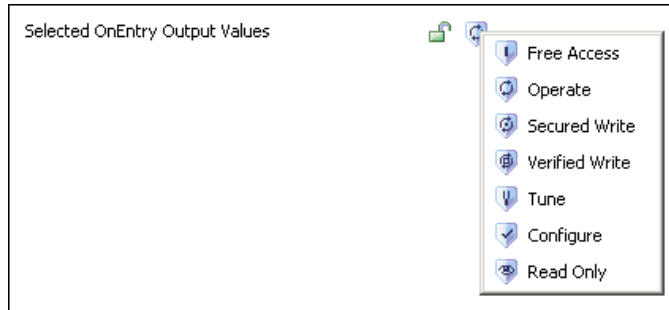
You can prevent the user from instructing the step program to change the on Entry and on Exit output values of the selected step. This is done by restricting access to the following two attributes:

- **Selected.OnEntryOutputValues**

- **Selected.OnExitOutputValues.**

To change the access rights to the On Entry output values of the selected step

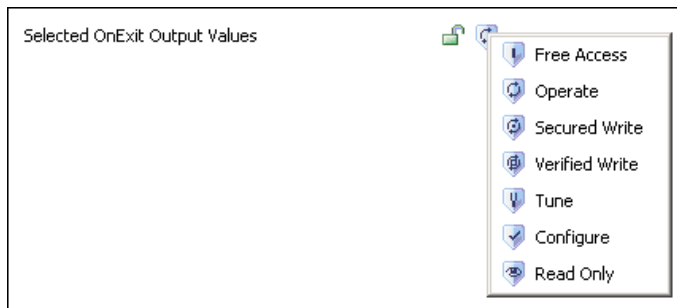
- 1 Click the **Settings** tab. The **Settings** panel appears.
- 2 Select the appropriate **OnEntryOutputValues** security classification by using the security classification shield.



This changes your access rights to the **Selected.OnEntryOutputValues** attribute.

To change the access rights to the On Exit output values of the selected step

- 1 Click the **Settings** tab. The **Settings** panel appears.
- 2 Select the appropriate **OnExitOutputValues** security classification by using the security classification shield.



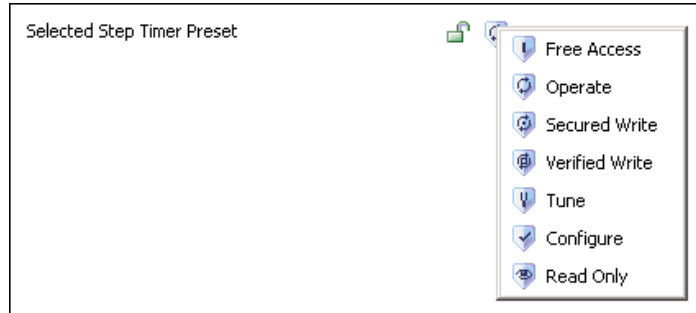
This changes your access rights to the **Selected.OnExitOutputValues** attribute.

## Modifying Access Rights to the Timer Presets of a Selected Step

You can prevent the user from instructing the step program to change the timer presets for the step condition and jump condition at run time. This is done by restricting access to the following two attributes; **Selected.StepTimerPreset** and **Selected.JumpTimerPreset**.

To change the access rights to the step timer preset of the selected step

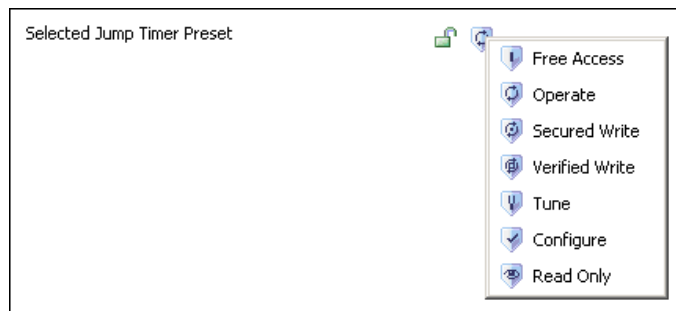
- 1 Click the **Settings** tab. The **Settings** panel appears.
- 2 Select the appropriate **StepTimerPreset** security classification by using the security classification shield.



This changes your access rights to the **Selected.StepTimerPreset** attribute.

To change the access rights to the jump timer preset of the selected step

- 1 Click the **Settings** tab. The **Settings** panel appears.
- 2 Select the appropriate **JumpTimerPreset** security classification by using the security classification shield.



This changes your access rights to the **Selected.JumpTimerPreset** attribute.

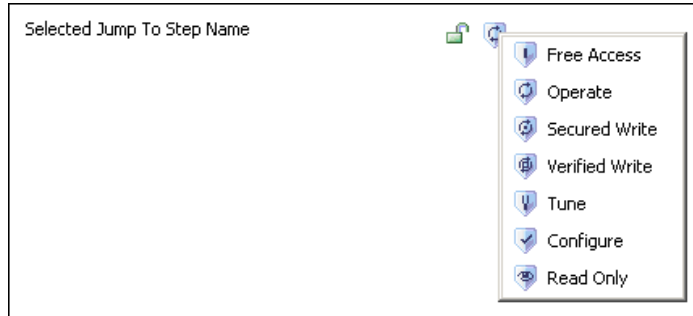
## Modifying Access Rights to the Jump Destination Setting of a Selected Step

You can prevent the user from changing the jump destination of any selected step in the step program. This is done by restricting access to the attribute **Selected.JumpToStepName**.

To change the access rights to the selected jump to Step Name of the selected step

- 1 Click the **Settings** tab. The **Settings** panel appears.

- 2 Select the appropriate **JumpToStepName** security classification by using the security classification shield.



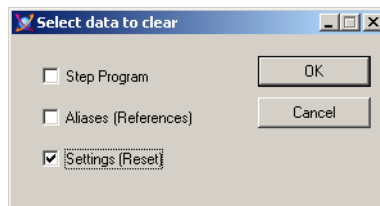
- 3 This changes your access rights to the **Selected.JumpToStepName** attribute.

## Clearing Settings

You can clear the settings on the settings panel with the **Clear** button. This resets the attributes in the **Settings** section of the **Settings** panel, except those that are locked.

### To clear all settings

- 1 Click the **Settings** tab. The **Settings** panel appears.
- 2 Click the **Clear** button. The **Select data to clear** dialog appears.



- 3 Make sure the **Settings (Reset)** option is selected and click **OK**. A confirmation dialog appears.
- 4 Click **Yes**. The settings are reset to their defaults if they are not locked.



---

# Chapter 7

## Importing and Exporting the Sequencer Program Configuration

This section shows you how to import and export the Sequencer program configuration, such as the step program, Alias definitions and general program settings at design time.

Sequencer program configurations can be imported and exported for use with Sequencer Objects or with third party applications that support XML.

### Exporting the Step Program

You can export the step program from the Sequencer object editor at any time. The exported data is written to an XML file that can also contain Alias configuration and settings for the Sequencer object.

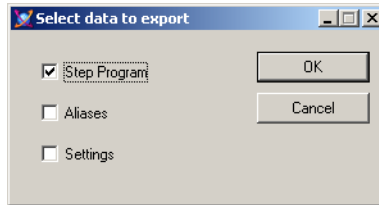
The following Step Program configuration data is exported.

- Step Program Name (attribute: PrgName)
- Step Program Comment (attribute: PrgComment)
- Initial Step (attribute: PrgStepInitial)
- Final Step Sequence (attribute: PrgStepFinal)
- Step Sequence (attribute: PrgStepProgram)

To export the Step Program

- 1 Click the Step Program tab. The Step Program panel appears.

- 2 Click **Export**. The **Select data to export** dialog box appears.



- 3 Select the **Step Program** check box. Click **OK**. The **Save As** dialog box appears. Browse to a location and enter a file name in the **File Name** field for the XML file.
- 4 Click **Save**. An XML file at the specified location is saved.

## Exporting the Alias Definitions

You can export the Alias definitions from the Sequencer object editor at any time. The exported data is written to an XML file that can also contain the step program and settings for the Sequencer object.

The following Alias configuration data is exported.

- Alias Name
- Attribute Name

To export all Alias definitions

- 1 Click the **Aliases** tab. The **Aliases** panel appears.
- 2 Click the **Export** button. The **Select data to export** dialog box appears.




---

**Note** You can export Aliases from either the Step Program panel or from the Settings panel. Depending from which panel you use, the **Select data to export** dialog box shows different default settings. Make sure the **Aliases** check box is selected to export the Aliases.

---

- 3 Select the **Aliases** check box. Click **OK**. Browse to a location and enter a file name in the **File Name** box for the XML file.
- 4 Click **Save**. An XML file is saved to the specified location.

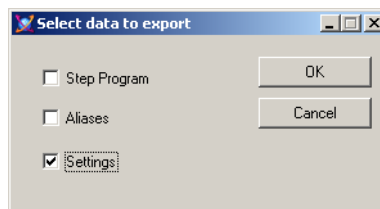
## Exporting the Program Settings

You can save the Sequencer program settings which contain the settings for following items:

- Initial Command
- Halt On Condition Error
- Halt On Output Error
- Auto Resume after Restart/Failover

To export the program settings

- 1 Click on the **Settings** tab. The **Settings** panel appears.
- 2 Click the **Export** button. The **Select data to export** dialog box appears.



- 3 Select the **Settings** check box and click **OK**. Browse to a location and enter a file name in the **File Name** box for the XML file.
- 4 Click **Save**. An XML file is saved to the specified location.

## Importing the Step Program

You can import the step program from an XML file at any time. The XML file may contain also Alias configuration and settings for the Sequencer object.

---

**Note** Before you import a step program, export any existing step program that you want to keep.

---

The following Step Program configuration data is imported.

- Step Program Name (attribute: PrgName)
- Step Program Comment (attribute: PrgComment)
- Initial Step (attribute: PrgStepInitial)
- Final Step Sequence (attribute: PrgStepFinal)
- Step Sequence (attribute: PrgStepProgram)

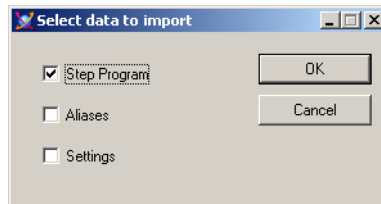
---

**Caution** You cannot import a step program if the Step Program (PrgStepProgram) is locked in the parent template.

---

### To import a Step Program

- 1 Click the **Step Program** tab. The **Step Program** panel appears.
- 2 Click the **Import** button. The **Open** dialog box appears.
- 3 Browse to the location of the XML file to load. Select it and click **Open**. The **Select data to import** dialog box appears.



- 4 Select the **Step Program** check box. Click **OK**. The existing Step Program is overwritten by the configuration from the XML file. Also the **Validation** window opens and validates the imported configuration.

---

**Note** If the **Step Program** option is disabled in the **Select data to import** dialog box, the corresponding attribute cannot be updated because it is locked in its parent template. When the **PrgAliasConfig** attribute is locked, importing a new step program with **Aliases** not defined in the **PrgAliasConfig** attribute terminates the import and the user is notified with an error message. If there are duplicate **StepNames** in the new Sequencer program, the import is terminated. If the new Sequencer program is missing information such as triggers, timers, jump to step names and/or attribute references, the import succeeds.

---

## Importing the Alias Definitions

You can import Alias definitions from an XML file at any time. The XML file may contain also the step program and settings for the Sequencer object.

---

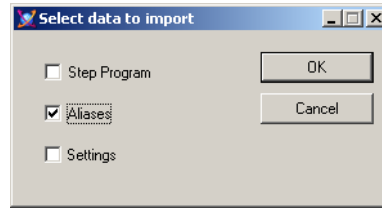
**Caution** You cannot import the Alias definitions if the Alias definitions (**PrgAliasConfig**) are locked in the parent.

---

### To import the Aliases definitions

- 1 Click the **Aliases** tab. The **Aliases** panel appears.
- 2 Click the **Import** button. The **Open** dialog box appears.
- 3 Browse to the location of the XML file to load and select it.

- 4 Click the **Open** button. The **Select data to import** dialog box appears.



- 5 Click **OK**. The existing Aliases are overwritten by the configuration from the XML file. Also the Validation window opens and validates the imported configuration.

---

**Note** If the **Aliases** option is disabled in the **Select data to import** dialog box, the corresponding attribute cannot be updated because it is locked in its parent template. When the **PrgAliasConfig** attribute is locked, importing a new step program with Aliases not defined in the **PrgAliasConfig** attribute terminates the import and the user is notified with an error message. If there are duplicate StepNames in the new Sequencer program, the import is terminated. If the new Sequencer program is missing information such as trigger, timer, jump to step name and/or attribute reference, the import succeeds.

---

## Importing the Program Settings

You can import program settings from an XML file at any time. The XML file may contain also the step program and Alias definitions for the Sequencer object.

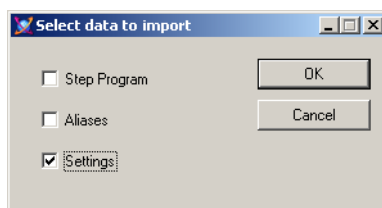
---

**Caution** You cannot update settings that are locked in the parent.

---

### To import the program settings

- 1 Click the **Settings** tab. The **Settings** panel appears.
- 2 Click the **Import** button. The **Open** dialog box appears.
- 3 Browse to the location of the XML file to load and select it.
- 4 Click the **Open** button. The **Select data to import** dialog box appears.



---

**Note** If the **Settings** option is disabled in the **Select data to import** dialog box, the corresponding attribute cannot be updated. This is due either to a security classification of read-only or to being locked in its parent template.

---

- 5 Click **OK**. The existing program settings are overwritten by the configuration from the XML file.

---

# Chapter 8

## Running the Sequencer

After you configure the Sequencer Object, you can deploy it. This section shows you important information

- on the regular deployment operation
- when restart/failover conditions occur that need your attention

It covers

- what the restart/failover condition is
- how the running sequence program is affected by them
- what you must do to make sure operations can continue smoothly and safely

## Deploying the Sequencer Object

When the Sequencer object is deployed, the Sequencer object starts up. It parses the sequence program that is stored as XML and invokes the code. The following attributes are affected:

- the `PrgSeqConfigStatus` attribute is set to the value **Updating** for a moment before it switches to **Ready**
- the `ExecutionState` attribute is set to **Initializing** while the sequencer acquires first update from all configured aliases. If the binding with the configured attributes takes longer time or the quality of the referenced attributes is initializing, the sequencer may remain in Initializing state until it times out.

---

**Note** If the Sequencer object is deployed offscan, its execution state is set to "Initializing". The initializing state does not time out.

---

## Handling Restart/Failover Conditions

This section shows you how the Sequencer Object handles restart/failover conditions.

### What are Restart/Failover Conditions?

The following events can be restart/failover events:

- the machine performs a failover
- the Sequencer object is switched offscan, then onscan again
- the hosting AppEngine is shutdown and restarted

### How does the Running Sequence Program React to a Restart/Failover Event?

The Sequencer object retains its configuration after a restart/failover event. By default, after a restart/failover event, the sequencer sets itself to a safe execution state where no further writes are automatically performed. If the execution state was **Running** or **RunningSingleStep** before the event, the sequencer sets itself to **RunningHeld** after the event.

---

**Note** During failover, it is not recommended to load, save or update the configuration file through the attributes `PrgSeqConfigCmd`, `PrgSeqConfig` and `PrgFileNameToLoadSave`. If a failover occurs during loading, saving or updating, the `PrgSeqConfigCmd` is aborted and the sequencer does not raise an alarm or indicate the failed action after the failover completes.

---

The following table shows you how the execution state changes after a restart/failover event:

---

Execution state <b>before</b> event	Execution state <b>after</b> event
Running	RunningHeld
RunningSingleStep	RunningHeld
SingleStepTransitionReady	SingleStepTransitionReady
RunningHeld	RunningHeld

---



Execution state <b>before</b> event	Execution state <b>after</b> event
Stopped	Stopped
StoppedComplete	StoppedComplete
StoppedError	StoppedError

## Detecting Restart/Failover Events

You can detect restart/failover events by configuring the Program Execution Halted alarm.

---

**Note** The alarm is raised only if prior to the restart/failover event the Execution State was Running or RunningSingleStep and AutoResume is false. Halting the program execution by setting the ExecutionStateCmd attribute to Hold does not raise this alarm.

---

### Configuring Alarms to Detect Program Execution Halting

You can configure the Sequencer to generate an alarm if the execution is halted after a restart/failover event. The alarm message includes the name and state of the Sequencer and the cause of the halting. The alarm returns to normal when an execution command is issued.

To configure Sequencer to generate an alarm when program execution is halted

- 1 Click the **Settings** tab. The **Settings** panel appears.
- 2 Select the check box **Detect Execution Halted**.
- 3 If necessary, enter an alarm priority in the corresponding text field. The range of valid alarm priorities is from 0 to 999. The default is 500.

## Using Attributes to Detect Program Execution Halting

You can detect a restart/failover event by looking at the values of the attributes **ExecutionHalted.Condition** and **ExecutionHaltedDesc**.

To detect that execution has halted

- 1 View the value of the **ExecutionHalted.Condition** attribute

- 2 View the value of the **ExecutionHaltedDesc** - when **ExecutionHalted.Condition** is true, this attribute provides the following description: *Sequencer Program:<Program Name> was halted due to <reason>*, where the reason can be either **offscan**, **shutdown** or **failover**.

---

**Note** For more information on the attributes **ExecutionHalted.Condition** and **ExecutionHaltedDesc**, see Sequencer Object Help on page 125.

---

## Resuming Operation after a Restart/Failover Event

After a restart/failover event occurs, the Sequencer object that was previously in **Running** or **RunningSingleStep** execution state is now in **RunningHeld** state. There are two possibilities of what may happen next, depending on the attribute **AutoResume** that you can configure on the **Settings** panel:

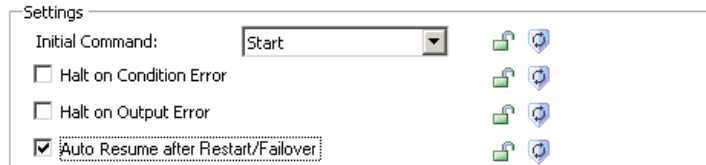
- if **AutoResume** is false, the operator needs to check the operation status and then to resume the sequence program by setting the attribute **ExecutionStateCmd** to **Resume**. This is the default setting.
- if **AutoResume** is true, the sequence program resumes automatically. No intervention of the operator is required.

### Automatically Resuming Execution after a Restart/Failover Event

You can instruct the Sequencer object to resume execution after a restart/failover event by setting the **Auto Resume after Restart/Failover** option in the **Settings** panel.

To set the **Auto Resume after Restart/Failover** option

- 1 Click the **Settings** tab. The **Settings** panel appears.
- 2 Select the check box **Auto Resume after Restart/Failover** in the **Settings** group.



## What happens to the Execution State after it Resumes?

When the sequence program resumes and the event is caused by:

- failover or by AppEngine shutting down and starting up, the execution state goes from **Initializing** to the execution state it was in before the event occurred.
- switching the Sequencer object offscan and then onscan, the execution state goes directly to the execution state it was in before the event occurred.

The following table shows how attributes and features behave after the sequence program is resumed:

Attribute or feature	Behavior
On Entry Outputs	are not written again when AutoResume is true
Timers	are frozen (excluding the time during failover, object offscan or object shutdown) and continues if AutoResume is set and the execution state before the failover event was Running or RunningSingleStep
Step Execution	is in the step body of the last active step  <b>Note</b> After switching offscan and then onscan, the active step is the next step if the restart/failover happened after the step condition was met (in case of a large scan period) .
Current.StepTimerRunning	persists the state of the timer so conditions can be evaluated accurately
Current.JumpTimerRunning	persists the state of the timer so conditions can be evaluated accurately

**Note** After a failover event, the object goes to the last (current) step and is put in RunningHeld state. If it happens that the failover event occurs after the condition is met and after the OnExit outputs are written then OnExit outputs are written again when the Resume or Advanced command is issued.



# Chapter 9

## Using Sequencer Program Commands and States

This section shows you how to work with the Sequencer Object after it is deployed. At run time, you can monitor and control the step program by viewing and changing the attributes of the Sequencer Object.

### Viewing and Monitoring the Currently Active Step

You can view the configuration or the execution of the step that is being currently processed (active step) by viewing the attributes that are prefixed with “Current”.

To view the number and name of the currently active step refer to the table below:

Tasks	Attributes
View step number	<code>.Current.StepNum</code>
View step name	<code>.Current.StepName</code>

For a detailed overview of the attributes and their properties, see Sequencer Object Help on page 125.

### Viewing the Configuration of the Currently Active Step

You can view the configuration of the step that is being currently processed.

The following table is an overview of the attributes for monitoring the active step configuration. You can view them in InTouch or Object Viewer:

Tasks	Attributes
View Step Configuration	<code>.Current.StepConditionCfg</code>
View Write On Exit Setting for the Step Transition	<code>.Current.StepWriteOnExit</code>
View Jump Configuration	<code>.Current.JumpConditionCfg</code>
View Write On Exit Setting for the Jump Transition	<code>.Current.JumpWriteOnExit</code>
View Jump destination step	<code>.Current.JumpToStepName</code>
View number of On Entry outputs	<code>.Current.OnEntryOutputCnt</code>
View names of Aliases associated with On Entry outputs	<code>.Current.OnEntryAliasNames[-1]</code>
View values that are written to On Entry outputs	<code>.Current.OnEntryOutputValues[-1]</code>
View number of On Exit outputs	<code>.Current.OnExitOutputCnt</code>
View names of Aliases associated with On Exit outputs	<code>.Current.OnExitAliasNames[-1]</code>
View values that are written to On Exit outputs	<code>.Current.OnExitOutputValues[-1]</code>

For a detailed overview of the attributes and their properties, see Sequencer Object Help on page 125.

## Monitoring the Execution of the Currently Active Step

You can view the processing state of the current step.

The following table is an overview of the attributes for monitoring the active step execution. You can view them in InTouch or Object Viewer:

Tasks	Attributes
View when the step became active	<code>.Current.StepStartTime</code>
View how long the step has been active	<code>.Current.StepDuration</code>
Identify if the step condition is fulfilled	<code>.Current.StepConditionState</code>
Identify if the step timer is running	<code>.Current.StepTimerRunning</code>
View how much time is remaining for the step timer	<code>.Current.StepTimeRemaining</code>
Identify if the jump condition is fulfilled	<code>.Current.JumpConditionState</code>
Identify if the jump timer is running	<code>.Current.JumpTimerRunning</code>
View how much time is remaining for the jump timer	<code>.Current.JumpTimeRemaining</code>

For a detailed overview of the attributes and their properties, see Sequencer Object Help on page 125.

## Controlling Program Flow at Run Time

You can control step program flow by telling Sequencer to

- start or stop program execution
- resetting program execution
- hold or resume program execution
- advance a step
- run in Single Step mode
- confirm a step transition when running in Single Step mode
- jump to a specific step

**Note** This is done by writing execution commands to the attribute `ExecutionStateCmd`. You can query the current state of execution by reading the value of the attribute `ExecutionState`.

---

The following terminology used in the tables of this section:

- **Current State** - the current execution of the step program. You can find out the current state by reading the attribute `ExecutionState`. Only certain combinations of current state and execution commands produce a meaningful result.
- **Target State** - the state of the step program execution after you have written the execution command to the attribute `ExecutionStateCmd`. This is your desired execution state.
- **Target State Timer** - after you write the execution command to the attribute `ExecutionStateCmd`, the timers act accordingly. They may restart, continue, or stop.
- **Target Step** - the step that becomes active after you write the execution command to the attribute `ExecutionStateCmd`.
- **Write OnExit** - informs you whether Sequencer writes values to the On Exit outputs, if configured, of the currently active step after you write the execution command.
- **Write OnEntry** - informs you whether Sequencer writes values to the On Entry outputs of the step that is active **after** the currently active step.

## Starting or Stopping Program Execution

At run time, you can start or stop program processing by writing to the Execution State Command (`ExecutionStateCmd`) attribute.

### Starting Program Execution

To start program execution write **Start** to the attribute `ExecutionStateCmd`.

You can start program execution if the current program execution state is one of the following;

- Stopped
- RunningHeld
- RunningSingleStep
- SingleStepTransitionReady



- **StoppedError**

If the command completes successfully the Execution State (attribute `ExecutionState`) changes to **Running**.

The following table shows you how the target state and target state timer change depending on the current execution state when the `ExecutionStateCmd` is set to **Start**. It also shows you whether the On Exit and On Entry values are written:

Current State	Target State Target State Timer Target Step	Execute OnExit Execute OnEntry
Stopped	<b>State:</b> Running <b>Timer:</b> Restart <b>Step:</b> Current	<b>OnExit:</b> No <b>OnEntry:</b> Yes
RunningHeld	<b>State:</b> Running <b>Timer:</b> Continue <b>Step:</b> Current	<b>OnExit:</b> No <b>OnEntry:</b> No
RunningSingleStep	<b>State:</b> Running <b>Timer:</b> Continue <b>Step:</b> Current	<b>OnExit:</b> No <b>OnEntry:</b> No
SingleStepTransitionReady	<b>State:</b> Running (StopComplete if in the Final step) <b>Timer:</b> Restart <b>Step:</b> Next Step (Current Step if it is the final step)	<b>OnExit:</b> No <b>OnEntry:</b> Yes (next step) No (if final step)
StoppedError	<b>State:</b> Running <b>Timer:</b> Restart <b>Step:</b> Current	<b>OnExit:</b> No <b>OnEntry:</b> Yes

For a detailed overview of the execution state attribute and its properties, see Sequencer Object Help on page 125.

For a detailed overview of the state transition and information on quality, timers and target steps that are effected by this execution command, see Sequencer State Transition Tables on page 161.

### Stopping Program Execution

To stop program execution write **Stop** to the attribute `ExecutionStateCmd`.

You can stop program execution if the current program execution state is one of the following;

- Running

- RunningHeld
- RunningSingleStep
- **SingleStepTransitionReady.**

If the command completes successfully the Execution State (attribute ExecutionState) changes to **Stopped**.

---

**Note** If you are using sufficiently large scan periods in the hosting AppEngine (such as 5 seconds) and you stop program execution after the current step completes (the conditions are satisfied and the OnExit outputs are written) but before the next execution cycle (in the middle between two execution cycles) then and only then the Current.xxx attributes correspond to the next step. If the step is still evaluating conditions and you stop program execution then Current.xxx attributes point to the current step.

---

The following table shows you how the target state and target state timer change depending on the current execution state when the **ExecutionStateCmd** is set to **Stop**. It also shows you whether the On Exit and On Entry values are written:

---

Current State	Target State Target State Timer Target Step	Execute OnExit Execute OnEntry
Running	<b>State:</b> Stopped <b>Timer:</b> Stopped <b>Step:</b> Current	<b>OnExit:</b> No <b>OnEntry:</b> No
RunningHeld	<b>State:</b> Stopped <b>Timer:</b> Stopped <b>Step:</b> Current	<b>OnExit:</b> No <b>OnEntry:</b> No
RunningSingleStep	<b>State:</b> Stopped <b>Timer:</b> Stopped <b>Step:</b> Current	<b>OnExit:</b> No <b>OnEntry:</b> No
SingleStepTransitionReady	<b>State:</b> Stopped <b>Timer:</b> Stopped <b>Step:</b> Current	<b>OnExit:</b> No <b>OnEntry:</b> No

---

For a detailed overview of the execution state attribute and its properties, see Sequencer Object Help on page 125.

For a detailed overview of the state transition and information on quality, timers and target steps that are effected by this execution command, see Sequencer State Transition Tables on page 161.

## Resetting Program Execution

At run time, you can reset program execution by writing to the Execution State Command (`ExecutionStateCmd`) attribute. Program processing is set to Initial Command and Initial Step. The first step is used if no Initial Step is specified.

### Resetting Program Execution

To reset program execution write **Reset** to the attribute **ExecutionStateCmd**.

You can reset program execution if the current program execution state is one of the following;

- Running
- RunningHeld
- RunningSingleStep
- SingleStepTransitionReady
- Stopped
- StoppedComplete
- **StoppedError**

If the command completes successfully the Execution State (attribute `ExecutionState`) changes to the initial command setting.

The following table shows you how the target state and target state timer change depending on the current execution state. It also shows you whether the On Exit and On Entry values are written:

Current State	Target State Target State Timer Target Step	Execute OnExit Execute OnEntry
Running	<b>State:</b> InitialCommand <b>Timer:</b> Restart <b>Step:</b> Initial step or first step	<b>OnExit:</b> No <b>OnEntry:</b> Yes (if InitialCommand <> Stop)
RunningHeld	<b>State:</b> InitialCommand <b>Timer:</b> Restart <b>Step:</b> Initial step or first step	<b>OnExit:</b> No <b>OnEntry:</b> Yes (if InitialCommand <> Stop)

Current State	Target State Target State Timer Target Step	Execute OnExit Execute OnEntry
RunningSingleStep	<b>State:</b> InitialCommand <b>Timer:</b> Restart <b>Step:</b> Initial step or first step	<b>OnExit:</b> No <b>OnEntry:</b> Yes (if InitialCommand <> Stop)
SingleStepTransitionReady	<b>State:</b> InitialCommand <b>Timer:</b> Restart <b>Step:</b> Initial step or first step	<b>OnExit:</b> No <b>OnEntry:</b> Yes (if InitialCommand <> Stop)
Stopped	<b>State:</b> InitialCommand <b>Timer:</b> Restart <b>Step:</b> Initial step or first step	<b>OnExit:</b> No <b>OnEntry:</b> Yes (if InitialCommand <> Stop)
StoppedComplete	<b>State:</b> InitialCommand <b>Timer:</b> Restart <b>Step:</b> Initial step or first step	<b>OnExit:</b> No <b>OnEntry:</b> Yes (if InitialCommand <> Stop)
StoppedError	<b>State:</b> InitialCommand <b>Timer:</b> Restart <b>Step:</b> Initial step or first step	<b>OnExit:</b> No <b>OnEntry:</b> Yes (if InitialCommand <> Stop)

For a detailed overview of the execution state attribute and its properties, see Sequencer Object Help on page 125.

For a detailed overview of the state transition and information on quality, timers, and target steps that are effected by this execution command, see Sequencer State Transition Tables on page 161.

## Holding or Resuming Program Execution

You can hold program execution to analyze outputs, debug your program or avert a critical condition. After you hold program execution you can tell it to continue by resuming processing. This is done by writing to the Execution State command (ExecutionStateCmd) attribute.

**Note** RunningHeld only occurs in the Step Body.

### Holding Program Execution

To hold program execution write **Hold** to the attribute **ExecutionStateCmd**.

You can hold program execution if the current program execution state is one of the following;

- Running
- RunningSingleStep
- SingleStepTransitionReady
- Stopped
- **StoppedError.**

If the command completes successfully the Execution State (attribute ExecutionState) changes to **RunningHeld**.

The following table shows you how the target state and target state timer change depending on the current execution state. It also shows you whether the On Exit and On Entry values are written:

Current State	Target State Target State Timer Target Step	Execute OnExit Execute OnEntry
Running	<b>State:</b> Running -> RunningHeld <b>Timer:</b> Freeze at current value <b>Step:</b> Current (or next step body)	<b>OnExit:</b> NO. (Yes, if target step is Next Step body) <b>OnEntry:</b> NO. (Yes, if target step is Next Step body)
RunningSingleStep	<b>State:</b> Running -> RunningHeld <b>Timer:</b> Freeze at current value <b>Step:</b> Current (or next step body)	<b>OnExit:</b> NO. (Yes, if target step is Next Step body) <b>OnEntry:</b> NO. (Yes, if target step is Next Step body)
SingleStepTransitionReady	<b>State:</b> Running -> RunningHeld <b>Timer:</b> Restart and freeze when reaching RunningHeld <b>Step:</b> Next step	<b>OnExit:</b> No <b>OnEntry:</b> Yes
Stopped	<b>State:</b> Running -> RunningHeld <b>Timer:</b> Restart and freeze when reaching RunningHeld <b>Step:</b> Current	<b>OnExit:</b> No <b>OnEntry:</b> Yes

Current State	Target State Target State Timer Target Step	Execute OnExit Execute OnEntry
StoppedError	<b>State:</b> Running -> RunningHeld <b>Timer:</b> Restart and freeze when reaching RunningHeld <b>Step:</b> Current	<b>OnExit:</b> No <b>OnEntry:</b> Yes

For a detailed overview of the execution state attribute and its properties, see Sequencer Object Help on page 125.

For a detailed overview of the state transition and information on quality, timers and target steps that are effected by this execution command, see Sequencer State Transition Tables on page 161.

### Resuming Program Execution

To resume program execution write **Resume** to the attribute **ExecutionStateCmd**.

You can resume program execution only if the current program execution state is **RunningHeld**. If the command completes successfully the Execution State (attribute ExecutionState) changes to the same state as before the hold command was written.

**Note** If the execution state was **Stopped** (or **StoppedError**) before holding program execution, resuming program execution puts the execution state back in **Stopped** (or **StoppedError**) state again, and not **Running** state. Resuming always puts the Sequencer Object into the state it was before its execution was held.

The following table shows you how the target state and target state timer change depending on the current execution state. It also shows you whether the On Exit and On Entry values are written:

Current State	Target State Target State Timer Target Step	Execute OnExit Execute OnEntry
RunningHeld	<b>State:</b> State before Hold command <b>Timer:</b> Continue in Timer state before Hold command <b>Step:</b> Current	<b>OnExit:</b> No <b>OnEntry:</b> No

For a detailed overview of the execution state attribute and its properties, see Sequencer Object Help on page 125.

For a detailed overview of the state transition and information on quality, timers and target steps that are effected by this execution command, see Sequencer State Transition Tables on page 161.

## Advancing a Step

You can tell the step program to advance to the next step without waiting for the step condition to become true. Sequencer then

- writes the On Exit values of the current step if the step Write OnExit is true
- transitions to the next step
- writes the On Entry values of the next step
- continuously evaluates the step and jump condition of the next step.

### Advancing a Step

To advance program execution to the next step write **Advance** to the attribute **ExecutionStateCmd**.

You can advance program execution if the current program execution state is one of the following;

- Running
- RunningHeld
- RunningSingleStep
- SingleStepTransitionReady
- Stopped
- **StoppedError.**

If the command completes successfully the Execution State (attribute ExecutionState) changes to the values as shown in the table below.

---

**Note** If you instruct the Sequencer Object to advance a step and the current execution state is either Running, RunningSingleStep or RunningHeld and the current step is configured to WriteOnExit it takes two execution cycles to finally advance to the next step.

---

The following table shows you how the target state and target state timer change depending on the current execution state. It also shows you whether the On Exit and On Entry values are written:

Current State	Target State Target State Timer Target Step	Execute OnExit Execute OnEntry
Running	<b>State:</b> Running (StoppedComplete if last step) <b>Timer:</b> Restart <b>Step:</b> Next step (or current if final step)	<b>OnExit:</b> Yes (if Step write on exit is true) <b>OnEntry:</b> Yes (No if last step)
RunningHeld	<b>State:</b> RunningHeld (StoppedComplete if last step) <b>Timer:</b> Restart <b>Step:</b> Next step (or current if final step)	<b>OnExit:</b> Yes, if Step write on exit is true No, if last step <b>OnEntry:</b> Yes (No if last step)
RunningSingleStep	<b>State:</b> RunningSingleStep (StoppedComplete if last step) <b>Timer:</b> Restart <b>Step:</b> Next step (or current if final step)	<b>OnExit:</b> Yes (if Step write on exit is true) <b>OnEntry:</b> Yes (No if last step)
SingleStepTransitionReady	<b>State:</b> RunningSingleStep (StoppedComplete if in the Final step) <b>Timer:</b> Restart <b>Step:</b> Next step (or current if final step)	<b>OnExit:</b> No (OnExit has already been executed in the previous state) <b>OnEntry:</b> Yes (No when this was the final step)
Stopped	<b>State:</b> Stopped <b>Timer:</b> Stopped <b>Step:</b> Next step (or current if final step)	<b>OnExit:</b> No <b>OnEntry:</b> No
StoppedError	<b>State:</b> Stopped <b>Timer:</b> Stopped <b>Step:</b> Next step (or current if final step)	<b>OnExit:</b> No <b>OnEntry:</b> No

For a detailed overview of the execution state attribute and its properties, see Sequencer Object Help on page 125.



For a detailed overview of the state transition and information on quality, timers and target steps that are effected by this execution command, see Sequencer State Transition Tables on page 161.

## Running in Single Step Mode

You can instruct Sequencer to execute the program in Single Step mode. In Single Step mode the program executes as follows:

- Step is active
- On Entry values are written to the outputs
- Step condition or jump condition is fulfilled
- On Exit values are written to the outputs
- Program waits for your confirmation
- After your confirmation, processing continues at the next step or a specified step to jump to

### Initiating Single Step Mode

To initiate the program in Single Step mode write **SingleStep** to the attribute **ExecutionStateCmd**.

You can run the program in Single Step mode if the current program execution state is one of the following;

- Running
- RunningHeld
- Stopped
- StoppedError

If the command completes successfully the Execution State (attribute ExecutionState) changes to **RunningSingleStep**.

The following table shows you how the target state and target state timer change depending on the current execution state. It also shows you whether the On Exit and On Entry values are written:

Current State	Target State Target State Timer Target Step	Execute OnExit Execute OnEntry
Running	<b>State:</b> RunningSingleStep <b>Timer:</b> Continue <b>Step:</b> Current	<b>OnExit:</b> No <b>OnEntry:</b> No

Current State	Target State Target State Timer Target Step	Execute OnExit Execute OnEntry
RunningHeld	<b>State:</b> RunningSingleStep <b>Timer:</b> Continue <b>Step:</b> Current	<b>OnExit:</b> No <b>OnEntry:</b> No
Stopped	<b>State:</b> RunningSingleStep <b>Timer:</b> Restart <b>Step:</b> Current	<b>OnExit:</b> No <b>OnEntry:</b> Yes
StoppedError	<b>State:</b> RunningSingleStep <b>Timer:</b> Restart <b>Step:</b> Current	<b>OnExit:</b> No <b>OnEntry:</b> Yes

For a detailed overview of the execution state attribute and its properties, see Sequencer Object Help on page 125.

For a detailed overview of the state transition and information on quality, timers and target steps that are effected by this execution command, see Sequencer State Transition Tables on page 161.

### Confirming a Transition When Running in Single Step Mode

To confirm a transition when the program is running in Single Step mode, write **Confirm** to the attribute **ExecutionStateCmd**.

You can confirm a transition only if the current program execution state is **SingleStepTransitionReady**. If the command completes successfully the Execution State (attribute ExecutionState) changes to **RunningSingleStep** or **StopComplete** (if the currently active step is the final step).

The following table shows you how the target state, target state timer change depending on the current execution state. It also shows you whether the On Exit and On Entry values are written:

Current State	Target State Target State Timer Target Step	Execute OnExit Execute OnEntry
SingleStepTransitionReady	<b>State:</b> RunningSingleStep (StopComplete if in the Final step) <b>Timer:</b> Restart <b>Step:</b> Next Step (Current step if it was the final step)	<b>OnExit:</b> No (OnExit has already been executed in the previous state) <b>OnEntry:</b> Yes (No when this was the final step)

For a detailed overview of the execution state attribute and its properties, see Sequencer Object Help on page 125.

For a detailed overview of the state transition and information on quality, timers and target steps that are effected by this execution command, see Sequencer State Transition Tables on page 161.

## Jumping to a Specific Step

You can force Sequencer to immediately jump to any step of the program. The On Exit values are written if the **Jump Write OnExit** setting is true.

### Jumping to a Specific Step

To jump to a specific step write the step number to the attribute **StepNumCmd** or the step name to the attribute **StepNameCmd**.

You can jump to a specific step if the current program execution state is one of the following;

- Running
- RunningHeld
- RunningSingleStep
- SingleStepTransitionReady
- Stopped
- StoppedComplete

- StoppedError

If the command completes successfully the Execution State (attribute ExecutionState) changes to the values as shown in the table below.

**Note** If you instruct the Sequencer Object to jump to a specific step and the current execution state is either Running, RunningSingleStep or RunningHeld and the current step is configured to WriteOnExit it takes two execution cycles to finally jump to the destination step.

The following table shows you how the target state and target state timer change depending on the current execution state. It also shows you whether the On Exit and On Entry values are written:

Current State	Target State Target State Timer Target Step	Execute OnExit Execute OnEntry
Running	<b>State:</b> Running <b>Timer:</b> Restart <b>Step:</b> StepNum	<b>OnExit:</b> Yes (if Jump write on exit is true) <b>OnEntry:</b> Yes
RunningHeld	<b>State:</b> RunningHeld <b>Timer:</b> Restart <b>Step:</b> StepNum	<b>OnExit:</b> Yes (if Jump write on exit is true) <b>OnEntry:</b> Yes
RunningSingleStep	<b>State:</b> RunningSingleStep <b>Timer:</b> Restart <b>Step:</b> StepNum	<b>OnExit:</b> Yes (if Jump write on exit is true) <b>OnEntry:</b> Yes
SingleStepTransitionReady	<b>State:</b> RunningSingleStep <b>Timer:</b> Restart <b>Step:</b> StepNum	<b>OnExit:</b> No (OnExit has already been executed in the previous state) <b>OnEntry:</b> Yes
Stopped	<b>State:</b> Stopped <b>Timer:</b> Stopped <b>Step:</b> StepNum	<b>OnExit:</b> No <b>OnEntry:</b> No
StoppedComplete	<b>State:</b> Stopped <b>Timer:</b> Stopped <b>Step:</b> StepNum	<b>OnExit:</b> No <b>OnEntry:</b> No
StoppedError	<b>State:</b> Stopped <b>Timer:</b> Stopped <b>Step:</b> StepNum	<b>OnExit:</b> No <b>OnEntry:</b> No

For a detailed overview of the StepNumCmd and StepNameCmd attributes and their properties, see Sequencer Object Help on page 125.

For a detailed overview of the state transition and information on quality, timers and target steps that are effected by this execution command, see Sequencer State Transition Tables on page 161.

## Setting Initial Command at Run Time

You can set the Initial Command at run time by writing the initial command to the attribute **PrgInitialCommand**. When the execution state is reset, Sequencer continues processing at the Initial Step, or first step if no Initial Step is defined, and changes its execution state to the value of the attribute PrgInitialCommand.

To set the Initial Command at run time

- ◆ Write the execution state that Sequencer should go to when it is reset (“Start”, “Stop”, “Hold”, “SingleStep”) to the attribute **PrgInitialCommand**.

For more information on these attributes, see Sequencer Object Help on page 125



# Chapter 10

## Modifying the Sequencer Program at Run Time

This section shows you how to modify some of the Sequencer Object parameters at run time, such as

- configuring a selected step
- setting initial step and final step
- configuring the step program or the Aliases by using XML
- loading and saving Sequencer object configuration data

### Viewing and Changing the Configuration of a Selected Step

You can view and change certain configuration parameters of any step by reading or writing to the attributes that are prefixed with the word “Selected”. You must select a step before using any of the attributes in the Selected group.

#### Viewing Configuration of a Selected Step

You can view the following configuration for any selected step in your step program:

- Step name
- Configuration of the Step Condition
- Step Timer Preset
- Write On Exit setting for the Step Transition

- Configuration of the Jump Condition
- Jump timer preset
- Write On Exit setting for the Jump Transition
- Jump destination
- Number of On Entry outputs
- Names of Aliases associated with On Entry outputs
- Values that are written to the On Entry outputs
- Number of On Exit outputs
- Names of Aliases associated with On Exit outputs
- Values that are written to the On Exit outputs

**To view the configuration of a specific step**

Write the step number of the step you want to select to the attribute *YourSequencerObject*.**Selected.StepNum** or **Selected.StepName**.

- 1 Refer to the following attribute table to retrieve the attribute reference:

Tasks	Attributes
View step name	<code>.Selected.StepName</code>
View Step Condition Configuration	<code>.Selected.StepConditionCfg</code>
View Step Timer Preset	<code>.Selected.StepTimerPreset</code>
View Write On Exit Setting for the Step Transition	<code>.Selected.StepWriteOnExit</code>
View Jump Condition Configuration	<code>.Selected.JumpConditionCfg</code>
View Jump Timer Preset	<code>.Selected.JumpTimerPreset</code>
View Write On Exit Setting for the Jump Transition	<code>.Selected.JumpWriteOnExit</code>
View Jump destination step	<code>.Selected.JumpToStepName</code>
View number of On Entry outputs	<code>.Selected.OnEntryOutputCnt</code>



Tasks	Attributes
View names of Aliases associated with On Entry outputs	<code>.Selected.OnEntryAliasNames[-1]</code>
View values that are written to On Entry outputs	<code>.Selected.OnEntryOutputValues[-1]</code>
View number of On Exit outputs	<code>.Selected.OnExitOutputCnt</code>
View names of Aliases associated with On Exit outputs	<code>.Selected.OnExitAliasNames[-1]</code>
View values that are written to On Exit outputs	<code>.Selected.OnExitOutputValues[-1]</code>

For a detailed overview of the attributes and their properties, see Sequencer Object Help on page 125.

## Changing Configuration of a Selected Step

You can change the following configuration for any selected step in your step program:

- Values that are written to On Entry outputs
- Values that are written to On Exit outputs
- Step Timer Preset
- Jump Timer Preset
- Write On Exit setting for the Step Transition
- Write On Exit setting for the Jump Transition
- Jump Destination of the Jump Transition

**Note** You can change the configuration of the current step by selecting it. When you use the Selected attributes to modify the Timer Preset and On Exit output values of the current step, these changes take effect immediately.

**Note** When the `PrgStepProgram` attribute is locked, run-time configuration changes via the Selected attributes are rejected with Operational Errors.

To change the configuration of a specific step

- 1 Write the step number of the step you want to select to the attribute *YourSequencerObject.Selected.StepNum*
- 2 Refer to the following attribute table to retrieve the attribute reference:

Tasks	Attributes
Change values that are written to On Entry outputs	<code>.Selected.OnEntryOutputValues[-1]</code>
Change Step Timer Preset	<code>.Selected.StepTimerPreset</code>
Change Jump Timer Preset	<code>.Selected.JumpTimerPreset</code>
Change values that are written to On Exit outputs	<code>.Selected.OnExitOutputValues[-1]</code>
Set or clear Write On Exit setting for the Step Transition	<code>.Selected.StepWriteOnExit</code>
Set or clear Write On Exit setting for the Jump Transition	<code>.Selected.JumpWriteOnExit</code>
Change the Jump Destination of the Jump Transition	<code>.Selected.JumpToStepName</code>

For a detailed overview of the attributes and their properties, see Sequencer Object Help on page 125.

**Note** When `PrgStepProgram` is locked, the set is rejected with Operational Error and the description is "Step Program Locked".

## Viewing the Execution Order of Aliases

The execution order of aliases determines in which order the output values are written to the OnEntry and OnExit aliases. You can view the writing execution order of aliases by reading the attribute **PrgStepProgram**.

To view the execution order of Aliases

- 1 Read the value of the attribute **PrgStepProgram**.
- 2 Locate the <ONENTRY> and <ONEXIT> section for any selected step. The order in which the aliases appear here is the order in which the outputs are written.

## Changing Step Program and Alias Configuration at Run Time

You can update the step program or the Alias configuration or both of them at the same time by using the **PrgSeqConfig** and **PrgSeqConfigCmd** attributes. The advantage of this is that you can add extra security to this attribute.

To update these attributes:

- Stop the step program execution so that it is either in Stopped, StoppedComplete or StoppedError state.
- Write the step program configuration and/or Alias configuration and/or settings to the PrgSeqConfig attribute. At this point there is no update to the run-time configuration.
  - Tell the Sequencer object to update its run-time configuration. The contents of the PrgSeqConfig attribute are written to the PrgStepProgram and PrgAliasConfig attributes and the run-time configuration is updated. Also the corresponding attributes from the Settings XML section are updated.

To update Sequencer object configuration at Run Time

- 1 Verify that the step program execution is stopped by reading the attribute **ExecutionState**. If it is not in one of the stopped states, write the value **Stop** to the enum attribute **ExecutionStateCmd**.
- 2 Write the step program configuration and/or Alias configuration as XML string to the PrgSeqConfig attribute.
- 3 Write the value **Update** to the enum attribute **PrgSeqConfigCmd**.
- 4 Check the attribute **PrgSeqConfigStatus** for details on the update:

Value	Description
Updating	The update is in progress.
Ready	The update has completed successfully.
Error	The update encountered a problem. This could be due to a syntax error in the XML data you tried to update the object with.

Additional feedback and results from the update are reflected in the following attributes:

- `PrgSeqConfigStatus.Error`
- `PrgSeqConfigStatus.Desc`
- `PrgSeqConfigStatus.Code`

For a detailed overview of these attributes and their properties, see Sequencer Object Help on page 125.

After the configuration data is successfully updated with the value from `PrgSeqConfig`, the step program starts running automatically. The attribute `PrgSeqConfig` is reset to an empty string after a successful update. When you upload configuration data that doesn't contain all parts of a Sequencer object, for example, if there are no Aliases, the configuration from the file is merged with the currently active run time configuration.

## Loading and Saving

At run time, you can load and save the Sequencer object configuration (step program, Aliases, and settings) from and to an XML file. Saving configuration at run time and loading it at design time is one way of uploading run-time changes.

---

**Important** When you perform a Load or Save at run time, your user credentials may not be the same as when you Import and Export at configuration time. Make sure that the shared locations of XML files are accessible by the Arcestra account under which the Sequencer object is running.

---

## Saving Sequencer Object Configuration at Run Time

At run time, you can save the Sequencer object configuration (step program, Aliases, and settings) by

- specifying a file name
- writing the save command

To save the run-time configuration to an XML file

- 1 Write the file name with path to the string attribute **`PrgFileNameToLoadSave`**.
- 2 Write the value **`Save`** to the enum attribute **`PrgSeqConfigCmd`**.

- 3 Check the attribute **PrgSeqConfigStatus** for details on the save:

Value	Description
Saving	The save is in progress.
Ready	The save has completed successfully and an XML file with the specified name is created at the specified location.
Error	The save encountered a problem. This could be due to insufficient privileges to create the XML file in the specified location.  A file error does not affect the error status of the Sequencer object.

Additional feedback and results from the save are reflected in the following attributes:

- PrgSeqConfigStatus.Error
- PrgSeqConfigStatus.Desc
- PrgSeqConfigStatus.Code

For a detailed overview of these attributes and their properties, see Sequencer Object Help on page 125.

## Loading Sequencer Object Configuration at Run Time

At run time, you can load Sequencer object configuration (step program, Aliases, and settings) by

- stopping the current step program execution
- specifying a file name
- writing the load command

To save the run-time configuration to an XML file

- 1 Verify that the step program execution is stopped by reading the attribute **ExecutionState**. If it is not stopped, write the value **Stop** to the enum attribute **ExecutionStateCmd**.

For more information on these attributes, see Sequencer Object Help on page 125.

- 2 Write the file name with path to the string attribute **PrgFileNameToLoadSave**.
- 3 Write the value **Load** to the enum attribute **PrgSeqConfigCmd**.

- 4 Check the attribute **PrgSeqConfigStatus** for details on the load:

Value	Description
Loading	The load is in progress.
Ready	The load has completed successfully and the run-time configuration now contains the configuration of the specified XML file.
	<p><b>Note</b> If a failover occurs during the load, the value of PrgSeqConfigStatus remains Ready.</p> <p><b>Caution</b> If a failover occurs while a Load, Save or Update operation is in progress, the operation is aborted. PrgSeqConfigStatus returns to Ready. The existing sequence program is retained. Error is not raised.</p>
Error	<p>The load encountered a problem. This could be due to invalid file location or invalid data contained within the file.</p> <p>A file error does not affect the error status of the Sequencer object.</p>

Additional feedback and results from the load are reflected in the following attributes:

- PrgSeqConfigStatus.Error
- PrgSeqConfigStatus.Desc
- PrgSeqConfigStatus.Code

For a detailed overview of these attributes and their properties, see Sequencer Object Help on page 125.

After configuration data is successfully loaded from the XML file

- the step program starts execution based on the value of the InitialCommand specified in the loaded file
- all attributes from the Settings node are updated per the configuration in the XML file.

**Caution** If any of these attributes such as InitialCommand are missing or are blank in the XML file then they are set to their default values. If the Settings node is not specified in the XML file, the corresponding object attributes are not updated.

After a successful load or update, the sequencer starts executing `InitialCommand` that it is explicitly specified in the sequence program that was just loaded or updated. If an `Initial Command` is not explicitly specified, the default `InitialCommand "Stop"` is used.

When you load configuration data from a file that does not contain all parts of a Sequencer object (for example, the `Aliases` section is missing), the configuration from the file is merged with the currently active run-time configuration.

## Uploading Run-Time Changes

In Sequencer you cannot upload run-time changes to the step program or Alias configuration by using the **Upload Runtime Changes** option on the context menu of the Sequencer instance. This is due to a current limitation of ArcestrA.

To upload run-time changes

- 1 At run time, save the run-time configuration as an XML file. For more information on how to do this, see [Saving Sequencer Object Configuration at Run Time](#) on page 116.
- 2 At design time import the XML file. The run-time data that you previously saved is imported. For more information on how to import data, see [Loading Sequencer Object Configuration at Run Time](#) on page 117.

## Setting Initial Step and Final Step

After deployment or a reset to `ExecutionStateCmd`, the Sequencer starts from the Initial Step. If an Initial Step is not specified, Sequencer starts from the first step. At run time, you can set the Initial Step and the Final Step. The changes take immediate effect.

If the Initial Step and Final Step are set so that:

- the current step is **before** the Initial Step, Sequencer continues until the final step is reached and continues at the Initial Step
- the current step is **after** the Final Step, Sequencer continues until the last step of the step program is reached, continues with the Initial Step (or first step) and then cycles between the InitialStep (or first step) and Final Step

- if the final step is specified and the transition conditions of the final step are met, the Sequencer writes OnExit outputs and moves to the StoppedComplete state
- if the final step is not specified and the transition conditions of the last step are met, the Sequencer writes OnExit outputs and returns to the first step.

#### To set the initial step at run time

- ◆ Write the step name to the attribute **PrgStepInitialCmd**. The update is reflected in the attribute **PrgStepInitial**.

---

**Tip** To clear the Final Step, set PrgStepFinalCmd to an empty string. When Final Step is blank, Sequencer does not stop at any step and returns back to the Initial or first step.

---

For more information on these attributes, see Sequencer Object Help on page 125.

#### To set the final step at run time

- ◆ Write the step name to the attribute **PrgStepFinalCmd**. The update is reflected in the attribute **PrgStepInitial**.

---

**Tip** To set the final step as the last step in the Step Program, pass an empty string to the attribute PrgStepFinalCmd.

---

## Detecting Errors at Run Time

At run time, you can use various attributes to detect errors and alarms caused by:

- program execution halting
- invalid condition triggers or bad quality values
- failure while writing values to On Entry outputs
- failure writing values to On Exit outputs

## Detecting Program Execution Halting

You can detect if the program execution is abnormally halted by the Sequencer by reading the values of the **ExecutionHalted.Condition** attribute and the **ExecutionHaltedDesc** attribute.

The program execution is halted and an alarm raised if, for example:

- you configure the **Halt on Condition Error** or **Halt on Output Error** options in the Sequencer editor



- a failover occurs and **AutoResume** is false (and the execution state is **Running** or **SingleStep** before the failover)
- the Sequencer object is taken Off Scan and then brought On Scan again and **AutoResume** is false (and the execution state is **Running** or **SingleStep** before taking it Off Scan)
- the hosting AppEngine object is restarted On Scan (and the execution state is **Running** or **SingleStep** before restarted the hosting AppEngine)
- the computer is rebooted (and the execution state is **Running** or **SingleStep** before restarted the hosting AppEngine).

To detect program execution halting

- ◆ Read the values from the following attributes:

Attribute	Description
ExecutionHalted.Condition	<b>TRUE</b> - The program execution was halted <b>FALSE</b> - The program execution is running as controlled by the user
ExecutionHaltedDesc	A description indicating possible causes

For more information on these attributes, see Sequencer Object Help on page 125.

## Detecting Condition Trigger Failure

You can detect if the Alias used as a condition trigger is invalid, inaccessible or has bad data quality by reading the values of the **ConditionTriggerFailure.Condition** attribute and the **ConditionTriggerFailureDesc** attribute.

To detect condition trigger failures

◆ Read the values from the following attributes:

Attribute	Description
ConditionTriggerFailure.Condition	<p><b>TRUE</b> - The Alias associated with a condition trigger is invalid or contains bad quality values.</p> <p><b>FALSE</b> - The Alias associated with a condition trigger is valid and contains good quality values.</p>
ConditionTriggerFailureDesc	A description indicating possible causes

For more information on these attributes, see Sequencer Object Help on page 125.

## Detecting On Entry Output Failure

You can detect if the write to one or more On Entry Outputs failed by reading the values of the **OnEntryOutputFailure.Condition** attribute or the **OnEntryOutputFailureDesc** attribute.

To detect OnEntry Output failures

Read the values from the following attributes:

Attribute	Description
OnEntryOutputFailure.Condition	<p><b>TRUE</b> - The write to an On Entry Output failed</p> <p><b>FALSE</b> - The write to an On Entry Output did not fail</p>
OnEntryOutputFailureDesc	A description indicating possible causes

For more information on these attributes, see Sequencer Object Help on page 125.

## Detecting On Exit Output Failure

You can detect if the write to one or more On Exit Outputs failed by reading the values of the **OnExitOutputFailure.Condition** attribute or the **OnExitOutputFailureDesc** attribute.

**To detect On Exit Output failures**

- ◆ Read the values from the following attributes:

Attribute	Description
OnExitOutputFailure.Condition	<b>TRUE</b> - The write to an On Exit Output failed <b>FALSE</b> - The write to an On Exit Output did not fail
OnExitOutputFailureDesc	A description indicating possible causes

For more information on these attributes, see Sequencer Object Help on page 125.

## Detecting Errors during Sequencer Configuration Change

You can detect if the Sequencer Configuration change failed during a load or update procedure. For this you can use the values of the **ProgramError** and **ProgramErrorDesc** attributes.

**To detect Program Errors**

- ◆ Read the values from the following attributes:

Attribute	Description
ProgramError	<b>TRUE</b> - A Program Error has occurred <b>FALSE</b> - No Program Error has occurred
ProgramErrorDesc	A description indicating possible causes

You can monitor the values of the following attributes for more information:

- PrgSeqConfigStatus.Error
- PrgSeqConfigStatus.Desc
- PrgSeqConfigStatus.Code

For more on these attributes, see Sequencer Object Help on page 125.

## Using the Sequencer Object with Redundancy

There are a few things to keep in mind when using the Sequencer object with redundancy:

- When the Sequencer object is hosted by a redundant application engine, avoid performing Load and Save operations using a path that refers to a local drive (i.e., C:\Sequence.xml). The Load and Save operations occur relative to the local engine.
- If a failover occurs during a Load, Save or Update operation, the operation is terminated and must be restarted manually. You are not notified of the failure of the operation.

# Appendix A

## Sequencer Object Help

This section includes the following:

- Configuration Object Attributes
- Run-Time Object Attributes
- Alarm Attributes
- Alarm Attributes

### Configuration Object Attributes

The following section lists the Sequencer object attributes associated with configuration panels.

- Step Program
- Aliases
- Settings

## Step Program

The following table lists the Sequencer object attributes associated with options on the **Step Program** panel:

Editor Option	Associated Attribute	Description	Run-Time Access
Step Program	.PrgStepProgram	<p>Stores the Step Program in an XML format. The step program consists of</p> <ul style="list-style-type: none"> <li>all the steps and corresponding output values</li> <li>StepProgramName, Step Program Comment, Initial Step Name and Final Step Name.</li> </ul> <p>The step program does not include information about the Alias References or settings.</p>	Read-Only
Step Program Name	.PrgStepProgram	Exposed at run time via the PrgName attribute. For more details, see attribute .PrgName in Program Attributes on page 143.	Read-Only
Initial Step Name	.PrgStepProgram	Exposed at run time via the .PrgStepInitial. Modifiable at run time via PrgStepInitialCmd. For more details, see attribute .PrgStepInitial in Program Attributes on page 143.	Read-Only
Final Step Name	.PrgStepProgram	Exposed at run time via the .PrgStepFinal. Modifiable at run time via PrgStepFinalCmd. For more details, see attribute .PrgStepFinal in Program Attributes on page 143.	Read-Only

Editor Option	Associated Attribute	Description	Run-Time Access
Step Program Comment	.PrgStepProgram	For run-time behavior, see attribute .PrgComment in Program Attributes on page 143.	Read-Only

## Aliases

The following table lists the Sequencer object attributes associated with options on the **Aliases** panel:

Editor Option	Associated Attribute	Description	Run-Time Access
Alias Definitions	.PrgAliasConfig	Stores the Sequencer Alias/attribute reference configuration in an XML format string.	Read-Only

## Settings

The following tables list the Sequencer object attributes associated with options on the **Settings** panel.

Editor Option	Associated Attribute	Description	Run-Time Access
Initialization Timeout	.InitializationTimeout	<p>Allows you to specify timeout period (ms) for the Initializing state. If the timeout period had elapsed and not all of the references returned valid updates, the following attributes are set:</p> <ul style="list-style-type: none"> <li>• ExecutionState = StoppedError</li> <li>• ExecutionHalted.Condition = True</li> <li>• ExecutionHaltedDescription = “Failed to acquire valid initial updates from input/output(s), program execution halted.”</li> </ul>	Supervisory, User

## Settings

The following table lists Sequencer object attributes associated with options on the **Settings** panel in the **Settings** group box. These four attributes can be imported/exported and load/saved as XML.

Editor Option	Associated Attribute	Description	Run-Time Access
Initial Command	.PrgInitialCommand	Sets the initial command the Sequencer starts after deployment or reset  "Start". Start the Sequencer (switch to running mode)  "Stop". Stop the Sequencer (switch to stopped mode)  "SingleStep". Switch to Single Step Mode  "Hold". Switch to hold	Supervisory, User
Halt on Condition Error	.PrgHaltOnConditionError	If TRUE (default) then when a Condition Trigger Failure is detected, the Sequencer is halted and ExecutionState is set to StoppedError.  If FALSE then when a Condition Trigger Failure is detected, the sequencer does not halt.	Supervisory, User
Halt on Output Error	.PrgHaltOnOutputError	If TRUE (default) then when an OnEntry or OnExit Output failure is detected, the Sequencer is halted and ExecutionState is set to StoppedError.  If FALSE then Sequencer continues.	Supervisory, User



Editor Option	Associated Attribute	Description	Run-Time Access
Auto Resume after Restart/Failover	.AutoResume	<p>If the ExecutionState is Running or SingleStep before restart/failover, the Sequencer returns to the RunningHeld state following restart/failover. Set this attribute to True to set the Sequencer to auto resume. Allows user to configure whether the object should auto resume on restart/failover.</p> <p>If false and ExecutionHalted alarm is enabled, an ExecutionHalted alarm is raised on restart/failover after the ExecutionState is set to RunningHeld.</p> <p>Restart refers to the following:</p> <ul style="list-style-type: none"> <li>going back to Onscan after toggling from OnScan to Offscan</li> <li>AppEngine Startup for checkpoint</li> </ul>	Supervisory, User

## Alarms

The following table lists Sequencer object attributes associated with options on the Settings panel in the Alarms group box:

Editor Option	Associated Attribute	Description
Detect Execution Halted	.ExecutionHalted.Alarmed	Enable or disable ExecutionHalted alarm. The ExecutionHalted alarm is raised when the sequencer is abnormally halted.
Priority	.ExecutionHalted.Priority	See Alarm Attributes.

Editor Option	Associated Attribute	Description
Detect Condition Trigger Failure	.ConditionTriggerFailure.Alarmed	Enable or disable ConditionTriggerFailure alarm. The ConditionTriggerFailure alarm is raised when the sequencer failed to evaluate condition trigger of the current step.
Priority	.ConditionTriggerFailure.Priority	See Alarm Attributes.
Detect OnEntry Output Failure	.OnEntryOutputFailure.Alarmed	Enable or disable OnEntryOutputFailure alarm. OnEntryOutputFailure alarm is raised if there is error writing to any of the OnEntry outputs.
Priority	.OnEntryOutputFailure.Priority	See Alarm Attributes.
Detect OnExit Output Failure	.OnExitOutputFailure.Alarmed	Enable or disable OnExitOutputFailure alarm. OnExitOutputFailure alarm is raised if there is error writing to any of the OnExit outputs.
Priority	.OnExitOutputFailure.Priority	See Alarm Attributes.

### Locking and Security

The following table lists Sequencer object attributes associated with options on the **Settings** panel in the **Security** group box:

Editor Option	Associated Attribute	Description
Execution State Command	.ExecutionStateCmd	For run-time behavior, see attribute .ExecutionStateCmd in Execution Attributes on page 133.
Set Active Step Name Command	.StepNameCmd	For run-time behavior, see attribute .StepNameCmd in Execution Attributes on page 133.

Editor Option	Associated Attribute	Description
Set Active Step Number Command	.StepNumCmd	For run-time behavior, see attribute .StepNumCmd in Execution Attributes on page 133.
Set Initial Step Command	.PrgStepInitialCmd	For run-time behavior, see attribute .PrgStepInitialCmd in Program Attributes on page 143.
Set Final Step Command	.PrgStepFinalCmd	For run-time behavior, see attribute .PrgStepFinalCmd in Program Attributes on page 143.
Sequence Configuration Command	.PrgSeqConfigCmd	For run-time behavior, see attribute .PrgSeqConfigCmd in Program Attributes on page 143.
Selected OnEntry Output Values	.Selected.OnEntryOutputValues	For run-time behavior, see attribute .Selected.OnEntryOutputValues in Selected Attributes on page 148.
Selected OnExit Output Values	.Selected.OnExitOutputValues	For run-time behavior, see attribute .Selected.OnExitOutputValues in Selected Attributes on page 148.
Selected Step Timer Preset	.Selected.StepTimerPreset	For run-time behavior, see attribute .Selected.StepTimerPreset in Selected Attributes on page 148.
Selected Jump Timer Preset	.Selected.JumpTimerPreset	For run-time behavior, see attribute .Selected.JumpTimerPreset in Selected Attributes on page 148.

---

Editor Option	Associated Attribute	Description
Selected Jump To Step Name	.Selected.JumpToStepName	For run-time behavior, see attribute .Selected.JumpToStepName in Selected Attributes on page 148.
Selected Step Write On Exit	.Selected.StepWriteOnExit	For run-time behavior, see attribute .Selected.StepWriteOnExit in Selected Attributes on page 148.
Selected Jump Write On Exit	.Selected.JumpWriteOnExit	For run-time behavior, see attribute .Selected.JumpWriteOnExit in Selected Attributes on page 148.

---

## Run-Time Object Attributes

The following section lists the Sequencer object attributes associated with the run-time environment.

- Execution Attributes
- Alarm Attributes
- Program Attributes
- Selected Attributes

## Execution Attributes

The following table lists the Sequencer object attributes associated with the Sequencer execution at run time.

Attribute	Description	Run-Time Access
.ExecutionState	<p>The execution state of the Sequencer can be:</p> <ul style="list-style-type: none"> <li>• "Running" (1)</li> <li>• "Stopped" (2)</li> <li>• "RunningHeld" (3)</li> <li>• "RunningSingleStep" (4)</li> <li>• "SingleStepTransitionReady" (5)</li> <li>• "StoppedComplete" (6)</li> <li>• "StoppedError" (7)</li> <li>• "Initializing" (8)</li> </ul>	Read-Only
	<hr/> <p><b>Note</b> Depending on the current ExecutionState, certain ExecutionStateCmds are allowed while others are not.</p> <hr/>	

---

Attribute	Description	Run-Time Access
.ExecutionStateCmd	<p>Set the mode at run time to control the operation mode of the Sequencer Object:</p> <p>"Start". Start the Sequencer (switch to running mode)</p> <p>"Stop". Stop the Sequencer (switch to stopped mode)</p> <p>"Reset". Start the Sequencer from the InitialStep in the state specified in by the InitialCommand. If InitialStep is blank, start from Step Number 1.</p> <p>"Advance". Advance unconditionally to the next step (manually forced step transition)</p> <p>Set at run time to force the value of StepNum to the next step without evaluating step conditions or timers or jump conditions or jump timers. If StepNum = StepCnt and StepNum &lt; &gt; FinalStep then StepNum advances to the first step (1). If the OnExit condition have not been set yet (step current step is in the step body) the OnExit outputs of the step transition is written if configured in the step condition.</p> <p>"SingleStep". Run Sequencer in Single Step Mode.</p>	Supervisory, User

---

---

Attribute	Description	Run-Time Access
.ExecutionStateCmd (continued)	<p>"Confirm". Confirm a pending single step transition. Set at run time to confirm the transition to the next step or jump destination in the state: "SingleStepTransitionReady"</p> <p>"Hold" Switch to RunningHeld</p> <p>"Resume" Resume to running from HOLD state</p>	Supervisory, User
	<hr/> <p><b>Note</b> When in "SingleStepTransitionReady" or "RunningHeld" state, the operator can still move to new steps using the "Advance", "Reset" commands or the StepNumCmd or StepNameCmd attributes.</p> <hr/>	

---

Attribute	Description	Run-Time Access
.ExecutionHalted.Condition	<p data-bbox="712 300 1089 363">Shows that the Sequencer is abnormally halted when:</p> <ul data-bbox="712 390 1089 1119" style="list-style-type: none"> <li data-bbox="712 390 1089 520">• a condition error occurs and HaltOnConditionError is true</li> <li data-bbox="712 548 1089 678">• an output error occurs and HaltOnConditionError is true</li> <li data-bbox="712 705 1089 800">• a failover occurs (without resume) and AutoResume is false</li> <li data-bbox="712 827 1089 957">• the object is switched OffScan and then OnScan and AutoResume is false</li> <li data-bbox="712 984 1089 1115">• the hosting AppEnging is shutdown and restarted OnScan and AutoResume is false</li> </ul> <p data-bbox="712 1142 1089 1245">TRUE: The program execution was abnormally halted by the Sequencer.</p> <p data-bbox="712 1272 1089 1360">FALSE: The program execution is running as controlled by the user.</p>	Read-Only
.ExecutionHaltedDesc	Warning description to indicate why the execution is halted.	Read-Only



Attribute	Description	Run-Time Access
.ConditionTriggerFailure.Condition	<p>Monitors the ConditionTrigger of the CurrentStep and is set to True when a problem is detected.</p> <p>TRUE: Failed to evaluate condition trigger of the current step because</p> <ul style="list-style-type: none"> <li>• either the trigger attribute quality is not Good</li> <li>• or the reference to the trigger cannot be resolved, coerced or made available.</li> </ul> <p>FALSE: No Condition Error detected in the current step.</p>	Read-Only
.ConditionTriggerFailureDesc	Warning description that indicates a possible reason for the ConditionTriggerFailure.	Read-Only
.OnEntryOutputFailure.Condition	<p>TRUE: Failed to evaluate condition trigger of the current step because</p> <ul style="list-style-type: none"> <li>• the output Alias reference cannot be resolved</li> <li>• or the output value cannot be coerced or the output Alias reference is unavailable.</li> </ul> <p>TRUE: In case of an On Entry Output error detected in the current step.</p> <p>FALSE: No OnEntry Output error detected in the current step.</p>	Read-Only
.OnEntryOutputFailureDesc	Warning description for the OnEntryOutputFailure attribute.	Read-Only

Attribute	Description	Run-Time Access
.OnExitOutputFailure.Condition	<p>TRUE: In case of an On Exit Output error detected in the current step because</p> <ul style="list-style-type: none"> <li>• the output Alias reference cannot be resolved</li> <li>• or the output value cannot be coerced or the output Alias reference is unavailable.</li> </ul> <p>FALSE: No OnExit Output error detected in the current step.</p>	Read-Only
.OnExitOutputFailureDesc	Warning description for the OnExitOutputFailure attribute.	Read-Only
.StepNameCmd	<p>Set to desired step name at run time to force the Sequencer Object to go unconditionally to this step without evaluating step conditions or timers or jump conditions or jump timers (manually forced jump transition). If the OnExit outputs of the current step have not been set yet, the OnExit values are written if JumpWriteOnExit is true before going to the new step location where the OnEntry output values are set. As soon as the new step position is reached all Current.xxx values are updated.</p> <p><b>Run time:</b> Forces the Sequencer Object to set the target step as the current step executing all OnExit and OnEntry output value settings.</p>	Supervisory, User

---

Attribute	Description	Run-Time Access
.StepNumCmd	<p>Set to desired step number at run time to force the Sequencer Object to go unconditionally to this step without evaluating step conditions or timers or jump conditions or jump timers (manually forced jump transition). If the OnExit outputs of the current step have not been set yet, the OnExit values are written if the JumpWriteOnExit is true before going to the new step location where the OnEntry output values are set. As soon as the new step position is reached all Current.xxx values are updated.</p> <p><b>Run time:</b> Forces Sequencer Object to set the target step as the current step executing all OnExit and OnEntry output value settings.</p>	Supervisory, User

---

## Current Attributes

The following table lists all Sequencer object attributes associated with the currently active step at run time:

Attribute	Description	Run-Time Access
.Current.JumpConditionConfig	<p>The string representation of the step condition configuration. For a full description, read the section XML Roots and Sub-Roots on page 155.</p> <pre>&lt;condition type&gt;   &lt;timer preset&gt;   &lt;trigger expression&gt;</pre> <p>Where:</p> <p>&lt;condition type&gt; consists of 3 characters indicating the condition, trigger, and timer type:</p> <p>&lt;Timer preset&gt; always has the format: &lt;dd&gt;:&lt;hh&gt;:&lt;mm&gt;:&lt;ss&gt;</p> <p>where the delimiter is always:</p> <p>&lt;trigger expression&gt; is the only variable length section of this syntax. It can have the form:</p> <p>&lt;Alias name&gt; for Boolean or data change evaluation (trigger T,F,t,f,c)</p> <p>Where Alias name is any symbolic Alias name used by the Sequencer.</p>	Read-Only
.Current.JumpConditionState	Shows the jump trigger condition evaluation result. TRUE if Jump condition is TRUE else FALSE.	Read-Only
.Current.JumpTimeRemaining	Remaining time of the jump timer in seconds.	Read-Only
.Current.JumpTimerRunning	Shows if the jump timer is running.	Read-Only
.Current.JumpToStepName	Represents the name of the destination step that the Sequencer Object jumps to on jump transition in the current step.	Read-Only

Attribute	Description	Run-Time Access
.Current.JumpWriteOnExit	Shows if WriteOnExit is set for the Jump Condition of the current step. TRUE: Outputs are written before jump transition. FALSE: Outputs are not written before jump transition.	Read-Only
.Current.OnEntryAliasNames	Shows the string array list of the current step OnEntry output names.	Read-Only
.Current.OnEntryOutputCnt	Shows the number of outputs in the OnEntry Alias and value arrays.	Read-Only
.Current.OnEntryOutputValues	Shows the string array list of the current step OnEntry output set values.	Read-Only
.Current.OnExitAliasNames	Shows the string array list of the current step OnExit output Alias names.	Read-Only
.Current.OnExitOutputCnt	Shows the number of outputs in the OnExit Alias and value arrays.	Read-Only
.Current.OnExitOutputValues	Shows the string array list of the current step OnExit output set values.	Read-Only

Attribute	Description	Run-Time Access
.Current.StepConditionConfig	<p>Shows the string representation of the step condition configuration. For a full description, read the section XML Roots and Sub-Roots on page 155.</p> <p>&lt;condition type&gt;   &lt;timer preset&gt;   &lt;trigger expression&gt;</p> <p>Where:</p> <p>&lt;condition type&gt; consists of 3 characters indicating the condition, trigger and timer type:</p> <p>&lt;Timer preset&gt; always has the format: &lt;dd&gt;:&lt;hh&gt;:&lt;mm&gt;:&lt;ss&gt;</p> <p>where the delimiter is always:</p> <p>&lt;trigger expression&gt; is the only variable length section of this syntax. It can have the form:</p> <p>&lt;Alias name&gt; for Boolean or data change evaluation (trigger T,F,t,f,c)</p> <p>Where Alias name is any symbolic Alias name used by the Sequencer.</p>	Read-Only
.Current.StepConditionState	Shows the step condition evaluation result. TRUE if the step condition is TRUE else FALSE.	Read-Only
.Current.StepDuration	Shows the elapsed time since the start of the current step.	Read-Only
.Current.StepName	Shows the name of the currently active step. Empty string if no step is active.	Read-Only
.Current.StepStartTime	Shows the time that the Sequencer Object entered the current step. Resets to the current time when step number changes or the current step is restarted.	Read-Only
.Current.StepTimeRemaining	Shows the remaining time of the step timer.	Read-Only
.Current.StepTimerRunning	Shows if the Step timer is running.	Read-Only

Attribute	Description	Run-Time Access
.Current.StepWriteOnExit	Shows if WriteOnExit is set for the Step Condition of the current step  TRUE: Outputs are written before step transition. FALSE: Outputs are not written before step transition.	Read-Only

## Program Attributes

The following table lists the Sequencer object attributes associated with the Sequencer program at run time:

Attribute	Description	Run-Time Access
.PrgAliasCnt	The total number of Aliases contained in the Sequencer program.	Read-Only
.PrgAliasNames	The string array list of the Sequencer Object Alias names.	Read-Only
.PrgComment	Stores an optional Sequencer program comment.	Read-Only
.PrgFileMatch	Set to False when the user has made a change to the program (step program, Alias configuration or settings) at run time but the program has not been saved to a file. This flag is also set to False when loading an XML file that does not contain all 3 sections (step program, Alias configuration, settings) so that existing sections are preserved or when loading an XML file into sections that have locked attributes.	Read-Only
.PrgFileName	The file name and path of the most-recently (successfully) loaded or saved file name.	Read-Only
.PrgFileNameToLoadSave	Stores the file name that the user specifies for loading and saving.	Supervisory, User
.PrgName	Displays the optional Sequencer program name.	Read-Only

---

Attribute	Description	Run-Time Access
.PrgReferenceNames	The string array list of the attribute references that are mapped to the Sequencer Object Aliases. The array is in the same sequence as the Alias names array.	Read-Only
.PrgSeqConfig	<ol style="list-style-type: none"><li>1 Allows user to update PrgStepProgram, PrgAliasConfig or Settings or any combination of them by specifying the corresponding section in the XML data set to this attribute.</li><li>2 The actual update takes place once PrgSeqConfigCmd is set to Update.</li><li>3 After Update is processed successfully, the value in the PrgSeqConfig is cleared.</li></ol>	Supervisory, User

---



Attribute	Description	Run-Time Access
.PrgSeqConfigCmd	<p>Set at run time to perform file IO or update the attribute PrgStepProgram, PrgAliasConfig or Settings or any combination of them. Feedback is provided by the PrgSeqConfigStatus attribute.</p> <p><b>"Save"</b> - save the Sequencer program XML configuration to the file which is specified in PrgFileNameToLoadSave (the filename from where it was loaded or where it was successfully saved to in a previous save as operation). This operation fails if PrgFileNameToLoadSave is empty or specifies an invalid or read only file.</p> <p><b>"Load"</b> - load the Sequencer XML data from the file specified in PrgFileNameToLoadSave. The Sequencer object loads whatever data is found in the XML file: Step Program data (steps and output values), Alias data (Alias names and attribute references) and Settings.</p> <p>If any of these sections are missing, the corresponding currently active data section remains unchanged. This allows replacing individual sections only (like loading a step program without modifying the existing IO configuration).</p> <p><b>"Update"</b> - similar to except that the data is coming from PrgSeqConfig attribute instead of a file.</p> <p><b>Run time:</b> Performs configuration operations and reflects the result in the attribute PrgSeqConfigStatus. Rejects with an Operational Error any attempt to issue a PrgSeqConfigCmd while PrgSeqConfigStatus is other than Ready or Error.</p>	Supervisory, User

Attribute	Description	Run-Time Access
.PrgSeqConfigStatus	Shows the result of the Save, Load or Update initiated by the PrgSeqConfigCmd. <ul style="list-style-type: none"> <li>• 1 - “Ready”</li> <li>• 2 - “Error”</li> <li>• 3 - “Loading”</li> <li>• 4 - “Saving”</li> <li>• 5 - “Updating”</li> </ul>	Read-Only
.PrgSeqConfigStatus.Code	Shows the result of the Save/Load/ Update initiated by the PrgSeqConfigCmd Possible values: <ul style="list-style-type: none"> <li>• 0 - The action completed successfully, no problem was detected</li> <li>• Severity 1000 (Warning) - reflects skipping update on some attributes due to lock during Load or Update</li> <li>• Severity 2000 (Critical) - means error and the whole Load, Update or Save is rejected</li> </ul>	Read-Only
.PrgSeqConfigStatus.Desc	Corresponds to PrgSeqConfigStatus.Code Includes Step Name and Step # when relevant. Includes Alias Name and # when relevant. It is empty when PrgSeqConfigStatus.Code = 0	Read-Only
.PrgSeqConfigStatus.Error	Shows the result of the Save/Load/ Update initiated by the PrgSeqConfigCmd It is set to True when PrgSeqConfigStatus.Code >= 2000	Read-Only
.PrgStepCnt	Shows the total number of steps contained in the Sequencer object.	Read-Only

Attribute	Description	Run-Time Access
.PrgStepInitial	Shows the starting step name of the Sequencer Object when deploying to run time or when reset.	Read-Only
.PrgStepInitialCmd	<p>Allows run-time modification of the initial step name.</p> <p><b>Run time:</b> Replaces initial step name in the Sequencer step program attribute and updates PrgStepInitial. The set is rejected with Operational error if PrgStepProgram attribute is locked.</p> <p>Validates the string value. A valid initial step string value is either an empty string or any existing step name in the step program. If this initial step name is empty the Sequencer starts on the first step in the sequence.</p>	Supervisory, User
.PrgStepFinal	Allows run-time modification of the final step name.	Read-Only
.PrgStepFinalCmd	<p>Allows run-time modification of the final step name.</p> <p><b>Run time:</b> Replaces final step name in the Sequencer step program attribute and updates PrgStepFinal. The set is rejected with Operational error if PrgStepProgram attribute is locked.</p> <p>Validates the string value. A valid final step string value is either an empty string or any existing step name in the step program.</p>	Supervisory, User
.PrgStepNames	Shows the string array list of the Sequencer Object step names.	Read-Only
.ProgramError	<p>TRUE: In case of an Program Error.</p> <p>FALSE: No Program Error detected.</p>	Read-Only
.ProgramErrorDesc	Description of the Program Error.	Read-Only

## Selected Attributes

The following table lists all Sequencer object attributes associated with the selected step at run time:

Attribute	Description	Run-Time Access
.Selected.JumpConditionConfig	<p>Represents the string representation of the jump condition configuration. For a full description, read the section XML Roots and Sub-Roots on page 155.</p> <p>&lt;condition type&gt;   &lt;timer preset&gt;   &lt;trigger expression&gt;</p> <p>Where:</p> <p>&lt;condition type&gt; consists of 3 characters indicating the condition, trigger, and timer type:</p> <p>&lt;Timer preset&gt; always has the format: &lt;dd&gt;:&lt;hh&gt;:&lt;mm&gt;:&lt;ss&gt;</p> <p>&lt;trigger expression&gt; is the only variable length section of this syntax. It can have the form:</p> <p>&lt;Alias&gt; for Boolean or data change evaluation (trigger T,F,t,f,c)</p> <p>Where Alias is any symbolic Alias used by the Sequencer.</p>	Read-Only
.Selected.JumpTimerPreset	<p>Allows run-time modification of the jump timer preset.</p> <p><b>Run time:</b> Replaces timer preset of the selected step with poked value if valid, updates Selected.JumpConditionCfg attribute. If Current = Selected, the new value is reflected in the calculation of the Current.JumpTimeRemaining attribute.</p>	Supervisory, User

Attribute	Description	Run-Time Access
.Selected.JumpToStepName	<p>Allows run-time modification of the destination step that the Sequencer Object jumps to on jump transition in the selected step.</p> <p>If Current = Selected the new value should be reflected in the Current.JumpToStepName attribute.</p> <p>The set is rejected with a Config Error if the new value does not match an existing step of the step program.</p>	Supervisory, User
.Selected.JumpWriteOnExit	<p>Shows if WriteOnExit is set for the Jump Condition of the selected step.</p> <p>TRUE: Outputs are processed before jump transition.</p> <p>FALSE: Outputs are not processed before jump transition.</p> <p><b>Run time:</b> Replaces jump WriteOnExit of the selected step with poked value, updates Selected.JumpConditionCfg attribute.</p> <p>If Current = Selected then Current.JumpWriteOnExit and Current.JumpConditionCfg are also updated.</p>	Supervisory, User
.Selected.OnEntryAliasNames	<p>OnEntry Output names corresponding to the Selected.OnExitOutputCnt and Selected.StepNum</p>	Read-Only
.Selected.OnEntryOutputCnt	<p>The number of outputs in the OnEntry Alias and value arrays of the selected step.</p>	Read-Only

Attribute	Description	Run-Time Access
.Selected.OnEntryOutputValues	Shows and allows modification of OnEntry Output values that correspond to the selected step.  <b>Run time:</b> Invalid values lead to an output error later when writing to the output.	Supervisory, User
.Selected.OnExitAliasNames	OnExit Output names corresponding to the selected step.	Read-Only
.Selected.OnExitOutputCnt	The number of outputs in the OnExit Alias and value arrays of the selected step.	Read-Only
.Selected.OnExitOutputValues	Shows and allows modification of OnExit Output values corresponding to the selected step.  If Current = Selected then the new value(s) are reflected in the Current.OnExitOutput values.  <b>Note</b> The size of the array cannot be changed, only the existing values.  <b>Run time:</b> Invalid values lead to an output error later when writing to the output.	Supervisory, User

Attribute	Description	Run-Time Access
.Selected.StepConditionConfig	<p>Represents the string representation of the step condition configuration. For a full description, read the section XML Roots and Sub-Roots on page 155.</p> <p>&lt;condition type&gt;   &lt;timer preset&gt;   &lt;trigger expression&gt;</p> <p>Where:</p> <p>&lt;condition type&gt; consists of 3 characters indicating the condition, trigger and timer type:</p> <p>&lt;Timer preset&gt; always has the format: &lt;dd&gt;:&lt;hh&gt;:&lt;mm&gt;:&lt;ss&gt;</p> <p>&lt;trigger expression&gt; is the only variable length section of this syntax. It can have the form:</p> <p>&lt;Alias&gt; for Boolean or data change evaluation (trigger T,F,t,f,c)</p> <p>Where Alias is any symbolic Alias used by the Sequencer.</p>	Read-Only
.Selected.StepName	<p>Represents the Step Name of the selected step. This attribute can be set at run time and behaves like .Selected.StepNum.</p>	Supervisory, User
.Selected.StepNum	<p>Run-time configurable Step Number value. Selected.StepNum is used for run-time (esp. HMI) editing of the selected step number parameters.</p> <p><b>Run time:</b> Value cannot be less than one and cannot exceed the StepCnt. RTErrror is generated when a set attempt is made on an out of range value.</p> <p>When a value is set to Selected.StepNum update all Selected.* attribute values.</p>	Supervisory, User

Attribute	Description	Run-Time Access
.Selected.StepTimerPreset	<p>Allows run-time modification of the step timer preset.</p> <p><b>Run time:</b> Replaces timer preset of the selected step with poked value if valid, updates Selected.StepConditionCfg attribute. If Current = Selected, the new value is reflected in the calculation of the Current.StepRemaining attribute.</p>	Supervisory, User
.Selected.StepWriteOnExit	<p>Shows if WriteOnExit is set for the Step Condition of the selected step.</p> <p>TRUE: Outputs are evaluated before step transition. FALSE: Outputs are not evaluated before step transition.</p> <p><b>Run time:</b> Replaces step WriteOnExit of the selected step with poked value, updates Selected.StepConditionCfg attribute. If Current = Selected then Current.StepWriteOnExit and Current.StepConditionCfg are also updated.</p>	Supervisory, User

## Alarm Attributes

If an alarm attribute is enabled, then the following additional alarm attributes become available. None of these attributes can be alarmed.

Attribute	Description
<Attribute>.Acked	Used to specify whether an alarm has been acknowledged. This attribute is updated when the AckMsg attribute is set. This attribute is always set to FALSE when a new alarm condition is detected (when the InAlarm attribute changes from FALSE to TRUE).



Attribute	Description
<Attribute>.AckMsg	The operator comment at the time the alarm is acknowledged. Any received text is stored, and the Acked attribute is set to TRUE. Also, the TimeAlarmAcked attribute is set to the current time. The maximum length is 256 characters.
<Attribute>.Category	The category of the alarm. The label of each alarm category is fixed.
<Attribute>.DescAttrName	The name of the attribute (within the same object) to be used as the alarm message. The attribute you specify must be of type String or InternationalizedString, with a maximum string length of 329 characters. If no attribute is specified, the short description of the object is used as the alarm message.
<Attribute>.InAlarm	The alarm state. This is exactly the same as the attribute in the host primitive that represents the alarm condition, except when the alarm state is disabled, in which case, InAlarm is set to Off, regardless of the actual condition state. The quality is set during execute to the quality of the attribute, except when the alarm is disabled, in which case the quality is always GOOD.
<Attribute>.Priority	The value for the urgency of the alarm. Valid values are 1 through 999, with 1 being the most urgent.
<Attribute>.TimeAlarmAcked	The timestamp indicating the last time this alarm was acknowledged. The date format reflects the current locale setting for the operating system.
<Attribute>.TimeAlarmOff	The timestamp indicating the last time this alarm (as represented by the InAlarm attribute) went off. The date format reflects the current locale setting for the operating system.
<Attribute>.TimeAlarmOn	The timestamp indicating the last time this alarm (as represented by the InAlarm attribute) went on. The date format reflects the current locale setting for the operating system.

## History Attributes

If a historization attribute is enabled, then the following additional history-related attributes become available. None of these attributes can be alarmed.

Attribute	Description
<Attribute>.ForceStoragePeriod	The time interval, in milliseconds, at which the value must be stored, even if the value has not changed. A value of 0 disables this feature. As an example, a setting of 3600000 indicates the value must be stored once per hour (measured from the time the object was last put onscan). If this value is less than the scan period of the host object, forced storage occurs every scan period (effectively equivalent to setting a value deadband of 0).
<Attribute>.TrendHi	The default top of the trend scale for clients. This value must be greater than or equal to the low value for the trend. If this value is changed at run time, the maximum engineering unit change is not reflected in the historian until you redeploy the object. This attribute only applies to numeric data types.
<Attribute>.TrendLo	The default bottom of the trend scale for clients. This value must be less than or equal to the high value for the trend. If this value is changed at run time, the minimum engineering unit change is not reflected in the historian until you redeploy the object. This attribute only applies to numeric data types.
<Attribute>.ValueDeadBand	The amount in engineering units by which a value referenced by the attribute must change to force a storage. A value of 0 means it is unused, and thus any value change forces a storage. Also, a quality change in the attribute always forces a storage regardless of this setting. This attribute only applies to numeric data types.

---

# Appendix B

## Sequencer Program XML Schema

This section describes the structure and syntax of the XML strings that are passed to attributes to control the Sequencer program at run time.

### General Information

The Sequencer XML file is case-sensitive for keywords and parameters. For example, passing the Initial Command value “hold” instead of “Hold” won’t work.

### XML Roots and Sub-Roots

All Sequencer XML documents have the common root <SEQ\_PRG> with the following sub-roots:

<STEPS/>	contains the step program
<ALIASES/>	contains the Alias configuration
<SETTINGS/>	contains global settings like “halt on condition errors”, initial step and final step

### About Steps

Steps are located in the <STEPS/> sub-root each with a <STEP> entry. Each step has following XML attributes:

NUMBER	specifies the step number (integer)
NAME	specifies the step name (string)

STEPCONDITION	specifies the step condition (string) - see below for details
JUMPCONDITION	specifies the jump condition (string) - see below for details
JUMPTOSTEPNAME	specifies the step to jump to (string)

Each step has two sub-sections <ONENTRY> and <ONEXIT> containing <OUT> keys to define output value assignments.

---

**Important** If you are passing literal string (or date, time or elapsed time) values as outputs you need to enclose these with the ampersand notation of double-quotes, which is: **&quot;**

---

**Examples:**

- `<OUT name="MachineInstruction" value="&quot;Start&quot;"></OUT>`  
which passes the literal value *Start* to the Alias *MachineInstruction* for a given step.
- `<OUT name="MachineInstruction" value="Start"></OUT>`  
which passes the value of the Alias named *Start* to the Alias *MachineInstruction* for a given step.
- `<OUT name="MachineInstruction" value="&quot;Joe says &quot;Start&quot;&quot;"></OUT>`  
which passes the literal value *Joe says "Start"* to the Alias *MachineInstruction* for a given step.

## Step/Jump Condition Syntax

Conditions are fully described by a string containing the condition type; timer preset, and trigger expression:

```
<condition type><onexit output flag><timer
preset> | <trigger expression>
```

## Condition Type

Condition type consists of 3 characters:

111	Always TRUE
000	Always FALSE
<T>--	Trigger only
--<I>	Timer only
<T>A<I>	Trigger AND Timer
<T>O<I>	Trigger OR Timer
<T>D<D>	Trigger Timer Delay

where the trigger character <T> can be any of the following:

T	Evaluate for TRUE (While TRUE)
F	Evaluate for FALSE (While FALSE)
t	Evaluate for transition to TRUE (On TRUE)
f	Evaluate for transition to FALSE (On FALSE)
c	Evaluate for data change

and the timer character <I> can be any of the following:

S	Simple timer
R	Retentive timer
N	Non-retentive timer
M	Cyclic timer month with offset (days, hours, minutes, seconds)
W	Cyclic timer week with offset (days, hours, minutes, seconds)
d	Cyclic timer day with offset (hours, minutes, seconds)
h	Cyclic timer hour with offset (minutes, seconds)
m	Cyclic timer minute with offset (seconds)

## OnExit Output Flag

OnExit Output Flag consists of one character:

!	all OnExit outputs are set on transition
	no OnExit outputs are set on transition

## Timer Preset

<timer preset> always has the format <dd>:<hh>:<mm>:<ss> where:

dd	two digits offset in days (with leading zeros)
hh	two digits offset in hours (with leading zeros)
mm	two digits offset in minutes (with leading zeros)

ss           two digits offset in seconds (with leading zeros)  
:            delimiter

### Trigger Expression

Trigger expression is the only variable length section of this syntax and is the symbolic Alias name.

### Examples of Conditions

- `stepcondition="T-!00:00:00:00 | TankFull"` - immediate transition to the next step when the attribute associated with the Alias named TankFull is true.
- `stepcondition="TDS!00:00:00:05 | TankEmpty"` - 5 seconds delayed transition to the next step after the attribute associated with the Alias named TankEmpty is true.
- `jumpcondition="T- | 00:00:00:00 | Contaminated"` - jump immediately to the step defined in the parameter `jumptostepname` (not shown here) as soon as the attribute associated with the Alias Contaminated is true.
- `stepcondition="111!00:00:00:00 |"` - transition to the next step regardless of any trigger or timer.

## Alias Configuration

Aliases (Inputs/Outputs) are located in the `<ALIASES>` sub-root each with a `<ALIAS>` entry. Each Output has the following XML attribute:

NAME   logical name (string)  
ATTR   attribute name (string)

## Settings

Settings are located in the `<SETTINGS>` sub-root each with a dedicated property:

`<InitialCommand>`       contains the initial command (Start, Stop, Hold, SingleStep)  
`<HaltOnConditionError>` specifies if program processing should halt on error condition (1) or not (0)

<HaltOnError>	specifies if program processing should halt on output error (1) or not (0)
<ResumeAfterFailover>	specifies if processing should resume after a failure (1) or not (0).

---

**Caution** The properties are case-sensitive. For example: If you specify <initialcommand> instead of <InitialCommand> the XML property is not recognized.

---

## Example XML

The following shows an example of a Sequencer XML file:

```
<SEQ_PRG>
  <STEPS name="Sample Sequencer Program" comment="Sample Sequencer Program
  demonstrating steps with different conditions" StepInitial="Step_NOP"
  StepFinal="TestAndExit">
    <STEP name="Step_NOP" stepcondition="111!00:00:00:00|"
    jumpcondition="000|00:00:00:00|" jumptostepname="">
      <ONENTRY>
        <OUT name="Output1" value="111" />
        <OUT name="Output2" value="112" />
      </ONENTRY>
      <ONEXIT>
        <OUT name="OutputString" value=""Hello World"" />
      </ONEXIT>
    </STEP>
    <STEP name="Trigger" stepcondition="T--!00:00:00:00|Input1"
    jumpcondition="000|00:00:00:00|" jumptostepname="">
      <ONENTRY>
        <OUT name="Output1" value="0" />
      </ONENTRY>
      <ONEXIT>
        <OUT name="Output2" value="222" />
        <OUT name="OutputString" value=""10/24/2006 10:30:00 AM"" />
      </ONEXIT>
    </STEP>
    <STEP name="Timer" stepcondition="--S!00:00:00:15|"
    jumpcondition="000|00:00:00:00|" jumptostepname="">
      <ONENTRY>
        <OUT name="Output1" value="10" />
      </ONENTRY>
```

```
<ONEXIT>
  <OUT name="OutputString" value="&quot;00:01:30.000&quot;" />
</ONEXIT>
</STEP>
<STEP name="TestAndExit" stepcondition="--S!00:00:00:20|"
jumpcondition="TDR|00:00:00:12|Input1" jumptostepname="Step_NOP">
  <ONENTRY>
    <OUT name="Output1" value="0" />
    <OUT name="Output2" value="15" />
  </ONENTRY>
</STEP>
</STEPS>
<ALIASES>
  <ALIAS name="Output1" attr="Motor.Speed" />
  <ALIAS name="Output2" attr="Filler.Test" />
  <ALIAS name="OutputString" attr="Object.Description" />
  <ALIAS name="Input1" attr="Valve.Open" />
</ALIASES>
<SETTINGS>
  <InitialCommand value="Stop" />
  <HaltOnConditionError value="0" />
  <HaltOnOutputError value="0" />
  <ResumeAfterFailover value="0" />
</SETTINGS>
</SEQ_PRG>
```



---

# Appendix C

## Sequencer State Transition Tables

This section shows the relationship between step program execution states and execution commands that you can write. The structure is from a current execution state point of view.

### Terminology

The following terminology is used in the State Transition Tables:

- **Current State** - the current execution state of the step program. You can find out the current state by reading the attribute `ExecutionState`.
- **Current Timer** - the timer state of the current step before you change the execution state.
- **Command** - the command you write to the execution state command attribute (`ExecutionStateCmd`).
- **Target State** - the state of the step program execution after you have written the execution command to the attribute `ExecutionStateCmd`. This is your desired execution state.
- **Target State Timer** - after you write the execution command to the attribute `ExecutionStateCmd`, the timers act accordingly. They may restart, continue, or stop.

- **Target Step** - the step that becomes active after you write the execution command to the attribute ExecutionStateCmd.
- **Execution OnExit** - informs you whether Sequencer writes values to the OnExit outputs (if configured) of the currently active step after you write the execution command.
- **Execution OnEntry** - informs you whether Sequencer writes values to the OnEntry outputs of the step that is active **after** the currently active step.

## Current Execution State: Running

The following table gives you an overview of how the state, timers, active step, and OnExit/OnEntry execution change depending on the execution state command you issue, when the current execution state is **Running**:

---

Current State: Running

Current Timer: Running

Calculated attributes quality set to BAD: none

---

Command	Target State Target State Timer	Target Step	Execute OnExit Execute OnEntry
Stop	<b>State:</b> Stopped <b>Timer:</b> Stop	current	<b>OnExit:</b> NO <b>OnEntry:</b> NO
Reset	<b>State:</b> InitialCommand <b>Timer:</b> Restart	Initial step or first step	<b>OnExit:</b> NO <b>OnEntry:</b> YES (if InitialCommand <> Stop)
Advance	<b>State:</b> Running (StoppedComplete if last step) <b>Timer:</b> Restart	Next step (or current if last step)	<b>OnExit:</b> YES (if Step write on exit is true) <b>OnEntry:</b> YES (NO if last step)
SingleStep	<b>State:</b> RunningSingleStep <b>Timer:</b> Continue	current	<b>OnExit:</b> NO <b>OnEntry:</b> NO
Hold	<b>State:</b> Running -> RunningHeld <b>Timer:</b> Freeze at current value	Current or next step body	<b>OnExit:</b> NO (YES if next step body) <b>OnEntry:</b> NO (YES if next step body)

---

---

Current State: Running

Current Timer: Running

Calculated attributes quality set to BAD: none

Command	Target State Target State Timer	Target Step	Execute OnExit Execute OnEntry
StepNumCmd/ StepNameCm d	<b>State:</b> Running <b>Timer:</b> Restart	StepNum/StepName	<b>OnExit:</b> YES (if Jump write on exit is true) <b>OnEntry:</b> YES

---

## Current Execution State: RunningSingleStep

The following table gives you an overview of how the state, timers, active step, and OnExit/OnEntry execution change depending on the execution state command you issue, when the current execution state is **RunningSingleStep**:

---

Current State: RunningSingleStep

Current Timer: Running

Calculated attributes quality set to BAD: none

Command	Target State Target State Timer	Target Step	Execute OnExit Execute OnEntry
Start	<b>State:</b> Running <b>Timer:</b> Continue	Current	<b>OnExit:</b> NO <b>OnEntry:</b> NO
Stop	<b>State:</b> Stopped <b>Timer:</b> Stopped	Current	<b>OnExit:</b> NO <b>OnEntry:</b> NO
Reset	<b>State:</b> InitialCommand <b>Timer:</b> Restart	Initial step or first step	<b>OnExit:</b> NO <b>OnEntry:</b> YES (if InitialCommand <> Stop)
Advance	<b>State:</b> RunningSingleStep (StoppedComplete if last step) <b>Timer:</b> Restart	Next step (or current if last step)	<b>OnExit:</b> YES (if Step write on exit is true) <b>OnEntry:</b> YES (NO if last step)
Hold	<b>State:</b> Running -> RunningHeld <b>Timer:</b> Freeze at current value	Current or next step body	<b>OnExit:</b> NO (YES, if next step body) <b>OnEntry:</b> NO (YES, if next step body)

---

---

Current State: RunningSingleStep

Current Timer: Running

Calculated attributes quality set to BAD: none

Command	Target State Target State Timer	Target Step	Execute OnExit Execute OnEntry
StepNumCmd/ StepNameCmd	<b>State:</b> RunningSingleStep <b>Timer:</b> Restart	StepNum/StepName	<b>OnExit:</b> YES (if Jump write on exit is true) <b>OnEntry:</b> YES

---

## Current Execution State: SingleStepTransitionReady

The following table gives you an overview of how the state, timers, active step, and OnExit/OnEntry execution change depending on the execution state command you issue, when the current execution state is **SingleStepTransitionReady**:

---

Current State: SingleStepTransitionReady

Current Timer: Running

Calculated attributes quality set to BAD: none

Command	Target State Target State Timer	Target Step	Execute OnExit Execute OnEntry
Start	<b>State:</b> Running (StopComplete if in the Final step) <b>Timer:</b> Restart	Next Step (Current step if it was the final step)	<b>OnExit:</b> NO <b>OnEntry:</b> YES (next Step) (NO when this was final step)
Stop	<b>State:</b> Stopped <b>Timer:</b> Stopped	Current	<b>OnExit:</b> NO <b>OnEntry:</b> NO
Reset	<b>State:</b> InitialCommand <b>Timer:</b> Restart	Initial step or first step	<b>OnExit:</b> NO <b>OnEntry:</b> YES (if InitialCommand <> Stop)
Advance	<b>State:</b> RunningSingleStep (StopComplete if in the Final step) <b>Timer:</b> Restart	Next step (Current step if it was the final step)	<b>OnExit:</b> NO (OnExit has already been executed) <b>OnEntry:</b> YES (NO when this was the final step)

---

---

Current State: SingleStepTransitionReady

Current Timer: Running

Calculated attributes quality set to BAD: none

Command	Target State Target State Timer	Target Step	Execute OnExit Execute OnEntry
Confirm	<b>State:</b> RunningSingleStep (StopComplete if in the Final step) <b>Timer:</b> Restart	Next Step (Current step if it was the final step)	<b>OnExit:</b> NO <b>OnEntry:</b> YES (NO when this was the final step)
Hold	<b>State:</b> Running -> RunningHeld <b>Timer:</b> Restart and freeze when reaching RunningHeld	Next step	<b>OnExit:</b> NO <b>OnEntry:</b> YES
StepNumCmd/ StepNameCm d	<b>State:</b> RunningSingleStep <b>Timer:</b> Restart	StepNum/StepName	<b>OnExit:</b> NO (OnExit has already been executed) <b>OnEntry:</b> YES

---

## Current Execution State: RunningHeld

The following table gives you an overview of how the state, timers, active step, and OnExit/OnEntry execution change depending on the execution state command you issue, when the current execution state is **RunningHeld**:

---

Current State: RunningHeld

Current Timer: Frozen

Calculated attributes quality set to BAD: none

Command	Target State Target State Timer	Target Step	Execute OnExit Execute OnEntry
Start	<b>State:</b> Running <b>Timer:</b> Continue	Current	<b>OnExit:</b> NO <b>OnEntry:</b> NO
Stop	<b>State:</b> Stopped <b>Timer:</b> Stopped	Current	<b>OnExit:</b> NO <b>OnEntry:</b> NO
Reset	<b>State:</b> InitialCommand <b>Timer:</b> Restart	Initial step or first step	<b>OnExit:</b> NO <b>OnEntry:</b> YES (if InitialCommand <> Stop)

---

---

Current State: RunningHeld

Current Timer: Frozen

Calculated attributes quality set to BAD: none

Command	Target State Target State Timer	Target Step	Execute OnExit Execute OnEntry
Advance	<b>State:</b> Running Held (StoppedComplete if last step) <b>Timer:</b> Restart	Next step (or current if Final step)	<b>OnExit:</b> YES (if Step write on exit is true) (NO, if last step) <b>OnEntry:</b> YES (NO, if last step)
SingleStep	<b>State:</b> RunningSingleStep <b>Timer:</b> Continue	Current	<b>OnExit:</b> NO <b>OnEntry:</b> NO
Resume	<b>State:</b> State before Hold command <b>Timer:</b> Continue in Timer state before Hold command	Current	<b>OnExit:</b> NO <b>OnEntry:</b> NO
StepNumCmd/ StepNameCm d	<b>State:</b> Running Held <b>Timer:</b> Restart	StepNum/StepName	<b>OnExit:</b> YES (if Jump write on exit is true) <b>OnEntry:</b> YES

## Current Execution State: Stopped

The following table gives you an overview of how the state, timers, active step and OnExit/OnEntry execution changes depending on the execution state command you issue, when the current execution state is **Stopped**:

---

Current State: Stopped

Current Timer: Stopped

Calculated attributes quality set to BAD: all dynamic attributes

Command	Target State Target State Timer	Target Step	Execute OnExit Execute OnEntry
Start	<b>State:</b> Running <b>Timer:</b> Restart	Current	<b>OnExit:</b> NO <b>OnEntry:</b> YES
Reset	<b>State:</b> InitialCommand <b>Timer:</b> Restart	Initial step or first step	<b>OnExit:</b> NO <b>OnEntry:</b> YES (if InitialCommand <> Stop)

---

Current State: Stopped

Current Timer: Stopped

Calculated attributes quality set to BAD: all dynamic attributes

Command	Target State Target State Timer	Target Step	Execute OnExit Execute OnEntry
Advance	<b>State:</b> Stopped <b>Timer:</b> Stopped	Next step (or current if last step)	<b>OnExit:</b> NO <b>OnEntry:</b> NO
SingleStep	<b>State:</b> RunningSingleStep <b>Timer:</b> Restart	Current	<b>OnExit:</b> NO <b>OnEntry:</b> YES
Hold	<b>State:</b> Running -> RunningHeld <b>Timer:</b> Restart and freeze when reaching RunningHeld	Current	<b>OnExit:</b> NO <b>OnEntry:</b> YES
StepNumCmd/ StepNameCm d	<b>State:</b> Stopped <b>Timer:</b> Stopped	StepNum/StepName	<b>OnExit:</b> NO <b>OnEntry:</b> NO

---

## Current Execution State: StoppedComplete

The following table gives you an overview of how the state, timers, active step and OnExit/OnEntry execution changes depending on the execution state command you issue, when the current execution state is **StoppedComplete**:

---

Current State: StoppedComplete

Current Timer: Stopped

Calculated attributes quality set to BAD: all dynamic attributes

Command	Target State Target State Timer	Target Step	Execute OnExit Execute OnEntry
Reset	<b>State:</b> InitialCommand <b>Timer:</b> Restart	Initial step or first step	<b>OnExit:</b> NO <b>OnEntry:</b> YES (if InitialCommand <> Stop)
StepNumCmd/ StepNameCm d	<b>State:</b> Stopped <b>Timer:</b> Stopped	StepNum/StepName	<b>OnExit:</b> NO <b>OnEntry:</b> NO

---

## Current Execution State: StoppedError

The following table gives you an overview of how the state, timers, active step, and OnExit/OnEntry execution change depending on the execution state command you issue, when the current execution state is **StoppedError**:

---

Current State: StoppedError

Current Timer: Stopped

Calculated attributes quality set to BAD: all dynamic attributes

---

Command	Target State Target State Timer	Target Step	Execute OnExit Execute OnEntry
Start	<b>State:</b> Running <b>Timer:</b> Restart	Current	<b>OnExit:</b> NO <b>OnEntry:</b> YES
Reset	<b>State:</b> InitialCommand <b>Timer:</b> Restart	Initial step or first step	<b>OnExit:</b> NO <b>OnEntry:</b> YES (if InitialCommand <> Stop)
Advance	<b>State:</b> Stopped <b>Timer:</b> Stopped	Next step (or current if last step)	<b>OnExit:</b> NO <b>OnEntry:</b> NO
SingleStep	<b>State:</b> RunningSingleStep <b>Timer:</b> Restart	Current	<b>OnExit:</b> NO <b>OnEntry:</b> YES
Hold	<b>State:</b> Running -> RunningHeld <b>Timer:</b> Restart and freeze when reaching RunningHeld	Current	<b>OnExit:</b> NO <b>OnEntry:</b> YES
StepNumCmd/ StepNameCmd	<b>State:</b> Stopped <b>Timer:</b> Stopped	StepNum	<b>OnExit:</b> NO <b>OnEntry:</b> NO

---



# Appendix D

## Error Codes for Run-Time Updating

The following table shows an overview of error codes and descriptions that result from updating the step program, alias configuration and/or settings by the PrgSeqConfig attribute at run time. The attribute group PrgSeqConfigStatus contains error information, such as code, error and description.

Code	Key	Description
2000	FailedToParseXML	Failed to parse XML
2001	FailedToSaveXML	Failed to save XML
2002	InvalidXMLFile	Invalid XML file
2003	InvalidXMLFormat	Invalid XML format
2004	InvalidStepProgramXMLData	Invalid step program XML data
2005	InvalidAliasConfigXMLData	Invalid Alias configuration XML data
2006	InvalidStepConfiguration	Invalid step configuration
2007	InvalidCondition	Invalid condition
2008	MissingStepName	Missing step name
2009	ConditionCodeTooShort	Condition code too short

Code	Key	Description
2010	InvalidStepName	Invalid step name
2011	DuplicateStepName	Duplicate step name
2012	InvalidJumpToStepName	Invalid JumpTo step name
2013	MissingJumpToStepName	Missing JumpTo step name
2014	MissingStepCondition	Missing step condition
2015	MissingStepTrigger	Missing step trigger
2016	MissingTrigger	Missing trigger
2017	TriggerNotConfigured	Trigger not configured
2018	JumpTriggerNotConfigured	Jump trigger not configured
2019	InvalidTimerConfiguration	Invalid timer configuration
2020	InvalidTimerCode	Invalid timer code
2021	InvalidInitialStepName	Invalid initial step name
2022	InvalidFinalStepName	Invalid final step name
2023	InvalidAliasConfiguration	Invalid Alias configuration
2024	InvalidAliasName	Invalid Alias name
2025	DuplicateAliasName	Duplicate Alias name
2026	InvalidIOReference	Invalid Alias Reference
2027	OnEntryExitAliasNotConfigured	OnEntry/OnExit Alias not configured
2028	OnEntryExitValueAliasNotConfigured	OnEntry/OnExit Alias value not configured

# Glossary

<b>Application</b>	A collection of objects in a Galaxy Repository that performs an automation task. Synonymous with Galaxy. There can be one or more applications within a Galaxy Repository.
<b>Application Engine (AppEngine)</b>	A scan-based engine that hosts and executes the run-time logic contained within AutomationObjects.
<b>ApplicationObject</b>	An AutomationObject that represents some element of your application. This can include things like <ul style="list-style-type: none"><li>• an automation process component. For example, a thermocouple, pump, motor, valve, reactor, or tank</li><li>• or associated application component. For example, function block, PID loop, Sequential Function Chart, Ladder Logic program, batch phase, or SPC data sheet</li></ul>
<b>Application view</b>	The Applications view shows the object-related contents of the Galaxy in three different ways: Model view, Deployment view, and Derivation view. The Model view appears when the IDE is opened for the first time.
<b>ArchestrA</b>	The distributed architecture for supervisory control and manufacturing information systems. It is an open and extensible technology based on a distributed, object-based design.
<b>ArchestrA Object Toolkit</b>	A programmer's tool to create new ApplicationObjects and Device Integration Object (DIObjects) Templates, including their configuration and run-time implementations. Includes a tool to build DI Objects and create unique Domain Objects that interact with DI Objects in the ArchestrA environment.
<b>Area</b>	A logical grouping of AutomationObjects that represents an area or unit of a plant. It is used to group related AutomationObjects for alarm, history, and security purposes. It is represented by an Area AutomationObject.
<b>Area Object</b>	The System Object that represents an Area of your plant within a Galaxy. The Area Object acts as an alarm concentrator, and places other Automation Objects into proper context with respect to the actual physical automation layout.

<b>Assignment</b>	The designation of a host for an AutomationObject. For example, an AppEngine AutomationObject is assigned to a WinPlatform AutomationObject.
<b>Attribute</b>	An externally accessible data item of an AutomationObject.
<b>Attribute Reference String</b>	A text string that references an attribute of an AutomationObject.
<b>AutomationObject</b>	An object type that represents permanent things in your plant, such as ApplicationObject or Device Integration Object (DIOjects), with user-defined, unique names within the Galaxy. It provides a standard way to create, name, download, execute, and monitor the represented component.
<b>Automation Object Server (AOS)</b>	A computer that hosts one or more application engines and associated automation objects. An Industrial Application Server Galaxy Namespace can contain several Automation Object Server (AOS), each which requires a Platform.
<b>Backup Application Engine</b>	The object created by the ArchestrA infrastructure when the Primary object is enabled for Redundancy. See Redundancy for further details.
<b>Base Template</b>	A root template at the top of a derived hierarchy. Unlike other Templates, a base template is not derived from another template but developed with the ApplicationObject Toolkit and imported into a Galaxy. All templates names start with a \$.
<b>Block Read Group</b>	A DAGroup that is triggered by the user or another object. It reads a block of data from the external data source and indicates the completion status.
<b>Block Write Group</b>	A DAGroup that is triggered by the user or another object after all the required data items are set. The block of data is sent to the external data device. When the block write is complete, it indicates the completion status.
<b>Bootstrap</b>	The base ArchestrA service which is required on all ArchestrA computers. It provides the base software environment to enable a platform and allows a computer to be included in the Galaxy Namespace.
<b>Change Log</b>	The revision history that tracks the life cycle activities of ArchestrA Objects, such as object creation, Check in/Check out, deployment, and Import/Export.
<b>Change Propagation</b>	The ability to create templates which allows each component template to support changes such that a change in one of the elements can be automatically propagated to all — or select, related — object Instances.
<b>Check in</b>	IDE operation for making a configured Object available for other users to Check out and use.

---

<b>Check out</b>	IDE operation for the purpose of editing an Object. It makes the item unavailable for other users to Check out.
<b>Checkpoint</b>	The act of saving to disk the configuration, state, and all associated data necessary to support automatic restart of a running AutomationObject. The restarted object has the same configuration, state, and associated data as the last checkpoint image on disk.
<b>Compound Object</b>	An ApplicationObject that contains at least one other ApplicationObject.
<b>Contained Name</b>	An alternate naming convention that when combined with the TagName of the root container Object, results in the Hierarchical Name. For example, for a given object, its Hierarchical Name = Line1.Tank1.InletValve and its Contained Name= InletValve.
<b>Containment</b>	A hierarchical grouping that allows one or more AutomationObject to exist within the name space of a parent AutomationObject and be treated like parts of the parent AutomationObject. Allows for relative referencing to be defined at the Template and Instance level.
<b>DAGroup</b>	A data access group associated with Device Integration Object (DIOjects). It defines how communications are achieved with external data sources. It can be a Scan Group, Block Read Group or Block Write Group.
<b>DAServer Manager (DAS Manager)</b>	The System Management Console (SMC) snap-in supplied by the Data Access Server (DAServer) that provides the required interface for activation, configuration, and diagnosis of the DAServer.
<b>Data Access Server (DAServer)</b>	The server executable that handles all communications between field devices of a certain type and client applications. Similar to I/O Servers but with more advanced capabilities.
<b>Data Access Server Toolkit (DAS Toolkit)</b>	A developer tool that can build a Data Access Server (DAServer).
<b>Deployment</b>	The operation which instantiates an AutomationObject Instance in the ArchastrA run time. This action involves installing all the necessary software and instantiating the object on the target platform with the object's default attribute data from Galaxy Repository.
<b>Deployment view</b>	The part of the Application view in the IDE that shows how objects are physically dispersed across Platforms, Areas and Engines. This is a view of how the application is spread across computing resources.
<b>Derivation</b>	The creation of a new Template based on an existing Template.
<b>Derivation view</b>	The part of the Application view in the IDE that shows the parent-child relationship between Base Templates, derived templates and derived instances. A view into the genealogy of the application.

<b>Derived Template</b>	Any template with a parent template. Derived templates inherit the attributes of the parent template. You can changes these attributes in the derived template.
<b>Device Integration Object (DIOjects)</b>	An AutomationObject that represents the communication with external devices or software. DI Objects run on an Application Engine (AppEngine), and include DINetwork Objects and DIDevice Objects.
<b>DIDevice Object</b>	An Object that represents the actual external device (for example, a PLC or RTU) that is associated with a DINetwork Object. It can diagnose and browse data registers of the DAGroups for that device.
<b>DINetwork Object</b>	An Object that represents the network interface port to the device through the Data Access Server (DAServer) or the Object that represents the communications path to another software application. It provides diagnostics and configuration for that specific network card.
<b>Engine Object</b>	An ArcestraA system-enabled Object that contains Local Message Exchange and provides a host for ApplicationObjects, Device Integration Object (DIOjects) and Area Objects.
<b>Event Record</b>	The data that is transferred about the system and logged when a defined event changes state. For example, an analog crosses its high level limit, an acknowledgement is made, or an operator logs in to the system.
<b>Export</b>	The act of generating a Package File (.aaPKG) extension from persisted data in the Galaxy Database. You can Import the resulting .aaPKG file into another Galaxy.
<b>FactorySuite Gateway</b>	FactorySuite Gateway is a Microsoft Windows application program that acts as a communications protocol converter. Built with the ArcestraA DAS Toolkit, FS Gateway links clients and data sources that communicate using different data access protocols.
<b>Galaxy</b>	The entire application. The complete ArcestraA system consisting of a single logical name space (defined by the Galaxy Database) and a collection of Platform Objects, Engine Objects and other Objects. One or more networked PCs that constitute an automation system. This is referred to as the Galaxy Namespace.
<b>Galaxy Database</b>	The relational database containing all persistent configuration information like Templates, Instances, Security, and so on in a Galaxy Repository.
<b>Galaxy Database Manager</b>	The Galaxy Database Manager is a utility to manage your Galaxy. It can back up and restore Galaxies if they become corrupt or to reproduce a Galaxy on another computer. The Galaxy Database Manager is part of the System Management Console (SMC).
<b>GalaxyObject</b>	The Object that represents a Galaxy.

---

<b>Galaxy Repository</b>	The software sub-system consisting of one or more Galaxy Databases.
<b>Hierarchical Name</b>	The name of the Object in the context of its container object. For example, Tank1.OutletValve, where an object called Tank1 contains the OutletValve object.
<b>Historical Storage System (Historian)</b>	The time series data storage system that compresses and stores high volumes of time series data for later retrieval. For the Industrial Application Server, the standard Historian is IndustrialSQL Server.
<b>Host</b>	The parent of a child Instance in the deployment view. Example: a Platform instance is a Host for an Application Engine (AppEngine) instance.
<b>Import</b>	The act of reading a Package File (.aaPKG) and using it to create AutomationObject instances and Templates in the Galaxy Repository.
<b>Industrial Application Server</b>	<p>Industrial Application Server uses existing Wonderware products such as InTouch for visualization, IndustrialSQL Server as its historian, and the device Integration product line like a Data Access Server (DAServer) for device communications. Industrial Application Server uses InTouch or InTouch View for visualization with the addition of Platforms to the visualization node.</p> <p>The Industrial Application Server is sized by:</p> <ul style="list-style-type: none"><li>• the number of Workstation / Server Platforms,</li><li>• by real I/O in the system</li><li>• the number of Terminal Services sessions.</li></ul> <p>The Application Server license is per Galaxy. An Application Server can be distributed across multiple computers as part of a single Galaxy namespace.</p>
<b>Instance</b>	An Object, which is a unique representation of a template that exists in run time.
<b>Instantiation</b>	The creation of a new Instance based on a corresponding Template.
<b>Integrated Development Environment (IDE)</b>	The Integrated Development Environment (IDE) is the interface for the configuration side of Industrial Application Server. In the IDE, you manage Templates, create Instances, deploy and un-deploy Objects, and other functions associated with the development and maintenance of the system.
<b>InTouch View</b>	InTouch View Clients are InTouch run-time clients that solely use of the Industrial Application Server for its data source. In addition, standard InTouch run times can leverage the Industrial Application Server with the addition of a Platform license.

<b>I/O Count</b>	Number of I/O points being accessed into the Galaxy. I/O points are real I/O and are not equivalent to InTouch tags. I/O count is based on the number of I/O points that are configured through an OPC Server, I/O Server, Data Access Server (DAServer) or InTouch Proxy Object, over the whole Application Server namespace, regardless of how many PCs are in the system.
<b>Life Cycle Cost</b>	The cost of a Supervisory Control System attributed to initial development, application changes and on-going maintenance. The Industrial Application Server reduces these costs by using a component object-based development environment and automated change propagation capabilities.
<b>Log Viewer</b>	A Microsoft Management Console (MMC) snap-in that provides a user interface for viewing messages reported to the LogViewer.
<b>Message Exchange</b>	The object to object communications protocol used by ArchestrA and the Industrial Application Server.
<b>Model view</b>	The area in the Application view in the IDE that shows how objects are arranged to describe the physical layout of the plant and supervisory process being controlled.
<b>Object</b>	Any Template or instance in a Galaxy Database. A common characteristic of all objects is they are stored as separate components in the Galaxy Repository.
<b>Object Extensions</b>	The capability to add additional functions to an AutomationObject while not changing the object's original behavior. Can be added to derived Templates and object Instances. They include Scripts, User Defined Attributes (UDAs) and Attribute Extensions.
<b>Object Viewer</b>	A utility in which you can view the attribute values of the selected object in run time. This utility is only available when an Object is deployed. Object Viewer shows you diagnostic information on ApplicationObjects so you can see performance parameters, resource consumption and reliability measurements. In addition to viewing an object's data value, data quality and the communication status of the object, you can also modify some of its attributes for diagnostic testing. Modifications can include adjusting timing parameters and setting objects in an execution or idle mode.
<b>OffScan</b>	The state of an Object that indicates it is idle and not ready to execute its normal run-time processing.
<b>OnScan</b>	The state of an Object in which it is performing its normal run-time processing based on a configured schedule.
<b>Package Definition File (.aaPDF)</b>	The standard description file that contains the configuration data and implementation code for a Base Template. File extension is .aaPDF.



---

<b>Package File (.aaPKG)</b>	The standard description file that contains the configuration data and implementation code for one or more Objects or Templates. File extension is .aaPKG.
<b>Platform Count</b>	<p>Number of PCs in the Galaxy. Each Workstation and/or Server communicating directly with the Application Server requires a platform to be part of the Galaxy Namespace. This includes each InTouch and InTouch View client. Each InTouch Terminal Services Session needs a Industrial Application Server Terminal Services Session License.</p> <p>A Platform License includes a per seat FSCAL2000 with Microsoft 2000 SQL Server CAL. Stand-alone computers only hosting InSQL Servers or a remote Data Access Server (DAServer) do not need a platform license.</p>
<b>Platform Manager</b>	Provides Galaxy application diagnostics by allowing you to view the run-time status of some system objects and to perform actions upon those objects. Actions include setting platforms and engines in an executable or idle mode and starting and stopping platforms and engines. This utility is an extension snap-in to the ArcestrA System Management Console (SMC).
<b>Platform Object</b>	An Object that represents a single computer in a Galaxy, consisting of a system wide message exchange component and a set of basic services. This object hosts all Application Engines.
<b>PLC</b>	Programmable logic controller.
<b>Primary Application Engine</b>	The object created by the ArcestrA infrastructure when the Backup object is created through Redundancy. See Redundancy for further details.
<b>Properties</b>	Data common to all attributes of Objects, such as name, value, quality, and data type.
<b>Proxy Object</b>	An AutomationObject that represents an actual product for the purpose of device integration with the Industrial Application Server or InTouch® HMI. For example, a Proxy Object enables the Industrial Application Server to access an OPC server.
<b>Redundancy</b>	<p>During configuration</p> <ul style="list-style-type: none"><li>• Primary object: The object that is the main or central provider of the functionality in the run time. For AppEngines, it is the object you enable for redundancy. For data acquisition, it is the DIObject you use first as your data source in the run time.</li><li>• Backup object: The object providing the functionality of the Primary object when it fails. For AppEngines, it is the object created by the ArcestrA infrastructure when the Primary object is enabled for redundancy. For data acquisition, it is the Device Integration Object (DIObjects) you do not intend to use first as your data source in the run time.</li></ul>

---

During run time

- **Active object:** The object currently executing desired functions. For AppEngines, it is the object that is hosting and executing ApplicationObjects. For data acquisition, it is the object that is providing field device data through the RedundantDIOObject.
- **Standby object:** The passive object waiting for a failure in the Active object's condition or for a force-failover. For AppEngines, it is the object that monitors the status of the Active AppEngine. For data acquisition, it is the object that is not providing field device data through the RedundantDIOObject.

<b>RedundantDIOObject</b>	The RedundantDIOObject monitors and controls the redundant Device Integration Object (DIOObjects) data sources. Unlike redundant AppEngines, individual DIOObject data sources do not have redundancy-related states. They function as stand-alone objects.
<b>Redundant Message Channel</b>	The Redundant Message Channel (RMC) is a dedicated Ethernet connection which is required between the platforms hosting redundant engines. The RMC is vital to keep both engines synchronized with alarms, history, and checkpoint items from the engine that is in the Active Role. Each engine also uses this Message Channel to provide its health and status information to the other.
<b>Reference</b>	A string that refers to an object or to data within one of its attributes.
<b>Relational Reference</b>	A reference to an Object's attributes that uses a keyword in place of an object's TagName. These keywords allow a reference to be made by an object's relationship to the target attribute. Examples of these keywords are "Me", "MyPlatform", and "MyContainer".
<b>Remote Reference</b>	The ability to redirect ArchedraA object references or references to remote InTouch tags. The new script function that redirects remote references at run time is IOSetRemoteReferences.
<b>Scan Group</b>	A DAGroup that requires only the update interval be defined. The data is retrieved at the requested rate.
<b>Scan State</b>	The Scan State of an object in run time. This can be either OffScan or OnScan.
<b>Security</b>	Industrial Application Server security is applied to IDE, System Management Console (SMC), and the run-time data level. At the run-time data level which centralizes the definition of all permissions to the ApplicationObjects. These ApplicationObjects can be accessed by a variety of clients but the security is centrally defined, allowing ease of maintenance. Users that are allowed to modify these ApplicationObjects at run time are mapped to the objects by user-defined roles. These roles can be mapped directly to existing groups in a Microsoft Domain or workgroup.

---

<b>SmartSymbols</b>	SmartSymbols are objects that integrate object-oriented technology with InTouch graphics to transform them into reusable templates. Changes made to the templates automatically propagate throughout an application—even across multiple networked PC nodes. They are created from a graphic in an InTouch window that is grouped into a cell and converted into a SmartSymbol. Libraries of SmartSymbols can be exported to other applications and plants, allowing companies to standardize on graphics throughout the entire organization.
<b>System Management Console (SMC)</b>	The central run-time system administration/management product where you perform all required run-time administration functions.
<b>System Object</b>	An Object that represents an Area, Platform or Engine.
<b>TagName</b>	The unique name given to an Object. For example, for a given object, its TagName = V1101 and its HierarchicalName = Line1.Tank1.InletValve.
<b>Template</b>	An Object containing configuration information and software templates used to create a Derived Template and/or Instance.
<b>Template Toolbox</b>	The part of the IDE Main Window that hosts Template Toolsets, containing templates. The Template Toolbox shows a tree view of template categories in the Galaxy.
<b>Toolset</b>	A named collection of Templates shown together in the IDE Template Toolbox.
<b>User Defined Attributes (UDA)</b>	Allow you to add new functionality to an object. An attribute is added to an object at configuration time.
<b>UserDefined object</b>	An AutomationObject created from the \$UserDefined Template. This template does not have any application-specific attributes or logic. You must define these attributes and associated logic.
<b>WinPlatform object</b>	An Object that represents a single computer in a Galaxy, consisting of a systemwide message exchange component, a set of basic services, the operating system, and the physical hardware. This object hosts the Application Engine (AppEngine).



# Index

## A

- a numerical output value, assigning 55
- a string output value, assigning 57
- about steps 155
- Adding Comments to Sequencer Objects 34
- advancing a step 103
- alias
  - configuration 158
- alias configuration at run time, changing 115
- alias crossreferences, viewing 68
- alias definitions
  - exporting 82
  - importing 84
- alias references, viewing 67
- aliases 14
- aliases, renaming 60, 64
- Always True, Always False 20
- an alias, creating 53, 63
- an attribute reference to an alias, assigning 64
- an output value to a step, assigning 54
- and changing the configuration of a selected step, viewing 111
- and monitoring the currently active step, viewing 93
- appending steps 28
- assigning
  - a numerical output value 55
  - a string output value 57
  - an attribute reference to an alias 64
  - an output value to a step 54
  - the value of another alias as output value 57

## C

- changing
  - alias configuration at run time 115
  - configuration of a selected step 113
  - the order of aliases 65
  - the position of steps 29
  - the step program configuration at run time 115
- condition and jump condition, step 18
- condition type 157
- conditions that are always true or always false , defining steps 43
- configuration of a selected step
  - changing 113
  - viewing 111
- configuration, alias 158
- configuring
  - security 74
  - steps 24
- confirming a transition when running in single step mode 106
- controlling program flow at run time 95
- creating
  - an alias 53, 63
  - steps 23
- current execution state
  - running 162
  - runningheld 165
  - runningsinglestep 163
  - singlesteptransitionready 164
  - stopped 166
  - stoppedcomplete 167
  - stoppederror 168

cyclic timers 20  
 cyclic timers, defining steps 39

## D

daily timers, defining steps 41  
 defining
 

- event and time combinations 44
- event-based conditions 37
- hourly timers 42
- initial step and final step 31
- minute-based timers 42
- monthly timers 39
- weekly timers 40

 defining steps
 

- conditions that are always true or always false 43
- cyclic timers 39
- daily timers 41
- time delays 38

 delay timers, using 20  
 deleting
 

- aliases 61, 65
- steps 30

 documentation conventions 9

## E

event and time combinations, defining 44  
 event-based conditions, defining 37  
 event-based triggers 19  
 examples, conditions 158  
 exporting
 

- alias definitions 82
- program settings 83
- step program 81

## H

halting program execution on error 72  
 holding
 

- or resuming program execution 100
- program execution 100

 hourly timers, defining 42

## I

importing
 

- alias definitions 84
- program settings 85
- step program 83

 initial step and final step, defining 31

initiating single step mode 105  
 inserting steps 29  
 InTouch control, working with the  
 Sequencer 22

## J

jump
 

- section 24
- to 21

 jumping to a specific step 107

## L

loading
 

- saving 116
- Sequencer configuration at run time 117

 locking
 

- alias definition list 66
- step program 34

## M

minute-based timers, defining 42  
 monitoring the execution of the currently active  
 step 94  
 monthly timers, defining 39

## N

naming the step program 32

## O

object structure, Sequencer 11  
 OnExit output flag 157  
 or resuming program execution, holding 100  
 output values, removing 60  
 outputs 13

## P

program execution
 

- holding 100
- starting 96
- stopping 96

 program settings
 

- exporting 83
- importing 85

 program, step 12

## R

removing, output values 60  
 renaming

- aliases 60, 64
- steps 28
- resetting program execution 99
- restricting
  - step setting 75
- resuming program execution 102
- running in single step mode 105
- running, current execution state 162
- runningheld, current execution state 165
- runningsinglestep, current execution state 163

**S**

- saving Sequencer configuration at run time 116
- section, step 24
- security, configuring 74
- Sequencer
  - object structure 11
  - tank demo xml 159
- settings 16
- simple timer, using 20
- singlesteptransitionready, current execution state 164
- starting, program execution 96
- step
  - condition and jump condition 18
  - entry, step body and step exit 17
  - program 12
  - section 24
  - structure 16
- step program
  - exporting 81
  - importing 83
- step setting, restricting 75
- step/jump condition syntax 156
- steps 12
  - configuring 24
  - creating 23
  - renaming 28
- stopped, current execution state 166
- stoppedcomplete, current execution state 167
- stoppederror, current execution state 168
- stopping program execution 97
- stopping, program execution 96

- structure, step 16

## T

- technical support, contacting 9
- terminology 161
- the configuration of the currently active step, viewing 93
- the order of aliases, changing 65
- the position of steps, changing 29
- the step program configuration at run time, changing 115
- the value of another alias as output value, assigning 57
- time delays, defining steps 38
- time-based (timers) 20
- timer preset 157
- trigger expression 158

## U

- using
  - alarms to report errors and warnings 73
  - the Sequencer object with redundancy 109

## V

- validating the step program 33
- viewing
  - alias crossreferences 68
  - alias references 67
  - and changing the configuration of a selected step 111
  - and monitoring the currently active step 93
  - configuration of a selected step 111
  - the configuration of the currently active step 93

## W

- weekly timers, defining 40
- working with the Sequencer InTouch control 22
- write on exit 21

## X

- xml roots and sub-roots 155

