

# **EST2**

## **System Programming Manual**

**P/N 270187 • Rev 5.0 • 18AUG00**

<b>DEVELOPED BY</b>	Edwards Systems Technology 6411 Parkland Drive Sarasota, FL 34243 (941) 739-4300
<b>COPYRIGHT NOTICE</b>	<p>Copyright © 2000 Edwards Systems Technology, Inc.</p> <p>This manual is copyrighted by Edwards Systems Technology, Inc. (EST). You may not reproduce, translate, transcribe, or transmit any part of this manual without express, written permission from EST.</p> <p>This manual contains proprietary information intended for distribution to authorized persons or companies for the sole purpose of conducting business with EST. If you distribute any information contained in this manual to unauthorized persons, you have violated all distributor agreements and may be subject to legal action.</p>
<b>TRADEMARKS</b>	<p>IBM is a registered trademark of International Business Machines Corporation.</p> <p>Microsoft and MS-DOS are registered trademarks, and Microsoft Mouse and Windows are trademarks of Microsoft Corporation.</p>
<b>CREDITS</b>	This manual was designed and written by the EST Technical Services - Documentation Department, Sarasota.

#### DOCUMENT HISTORY

Date	Revision	Reason for change
03AUG98	4.5	<p>Revised content to reflect programming using the Systems Definition Utility.</p> <p>Revisions 4.4 and earlier are obsolete.</p>
August 2000	5.0	<p>Updated manual to coincide with release of software version 1.3. Added chapters on input events and output commands. Also added a quick programming reference and a glossary. Corrected minor technical errors and added procedures for the programming of logic groups.</p> <p>Revision 4.5 and earlier revisions are obsolete.</p>

# Content

---

	Important information • iii
	Getting the most out of this manual • iv
	About the Systems Definition Utility • vi
<b>Chapter 1</b>	<b>Overview • 1.1</b>
	Introduction • 1.2
	Programming using rules and objects • 1.3
	Creating a rules file • 1.8
	Compiling the rules file • 1.11
	Developing a labeling plan • 1.12
	Identifying object group relationships • 1.18
<b>Chapter 2</b>	<b>Advanced programming features • 2.1</b>
	Variables • 2.2
	Priorities • 2.5
	Logic groups • 2.7
	Actions and Sequences • 2.9
<b>Chapter 3</b>	<b>Input events • 3.1</b>
	Active events • 3.2
	Alarm events • 3.4
	CallIn events • 3.5
	Confirmation events • 3.6
	Define events • 3.7
	Monitor events • 3.9
	Supervisory events • 3.10
	TestActive events • 3.11
	TestTrouble event • 3.13
	Trouble events • 3.15
<b>Chapter 4</b>	<b>Output commands • 4.1</b>
	Activate command • 4.2
	Cancel command • 4.3
	Close command • 4.4
	CommonAlarmOff command • 4.5
	CommonAlarmOn command • 4.6
	CommonMonitorOff command • 4.7
	CommonMonitorOn command • 4.8
	CommonSupervisoryOff command • 4.9
	CommonSupervisoryOn command • 4.10
	CommonTroubleOff command • 4.11
	CommonTroubleOn command • 4.12
	Delay command • 4.13
	Disable command • 4.14
	Enable command • 4.16
	FanOff command • 4.18
	FanOn command • 4.19

HoldDoor command • 4.20  
LEDOff command • 4.21  
LEDOn command • 4.22  
NSClose command • 4.23  
NSCommonAlarmOff command • 4.24  
NSCommonAlarmOn command • 4.25  
NSCommonMonitorOff command • 4.26  
NSCommonMonitorOn command • 4.27  
NSCommonSupervisoryOff command • 4.28  
NSCommonSupervisoryOn command • 4.29  
NSCommonTroubleOff command • 4.30  
NSCommonTroubleOn command • 4.31  
NSFanOff command • 4.32  
NSFanOn command • 4.33  
NSHoldDoor command • 4.34  
NSOpen command • 4.35  
NSReleaseDoor command • 4.36  
Off command • 4.37  
OffGuard command • 4.39  
On command • 4.40  
OnGuard command • 4.42  
Open command • 4.43  
ReleaseDoor command • 4.44  
Restore command • 4.45

## **Appendix A**

### **Quick reference • A.1**

Event specification summaries • A.2  
Events listed by device • A.6  
Command specification summaries • A.10  
Commands listed by device • A.17

### **Glossary • Y.1**

### **Index • Z.1**

---

## Important information

### Limitation of liability

The content of this manual is proprietary in nature and is intended solely for distribution to authorized persons, companies, distributors and/or others for the sole purpose of conducting business associated with Edwards Systems Technology, Inc. The distribution of information contained within this manual to unauthorized persons shall constitute a violation of any distributor agreements and may result in implementation of legal proceedings.

This product has been designed to meet the requirements of NFPA Standard 72, 1999 Edition; Underwriters Laboratories, Inc., Standard 864, 7th Edition; and Underwriters Laboratories of Canada, Inc., Standard ULC S527. Installation in accordance with this manual, applicable codes, and the instructions of the Authority Having Jurisdiction is mandatory. EST, Inc. shall not under any circumstances be liable for any incidental or consequential damages arising from loss of property or other damages or losses owing to the failure of EST, Inc. products beyond the cost of repair or replacement of any defective products. EST, Inc. reserves the right to make product improvements and change product specifications at any time.

While every precaution has been taken during the preparation of this manual to ensure the accuracy of its contents, EST assumes no responsibility for errors or omissions.

### FCC warning

This equipment can generate and radiate radio frequency energy. If this equipment is not installed in accordance with this manual, it may cause interference to radio communications. This equipment has been tested and found to comply within the limits for Class A computing devices pursuant to Subpart B of Part 15 of the FCC Rules. These rules are designed to provide reasonable protection against such interference when this equipment is operated in a commercial environment. Operation of this equipment is likely to cause interference, in which case the user at his own expense, is required to take whatever measures may be required to correct the interference.

---

## Getting the most out of this manual

### Knowing where to find help

The EST2 System Programming Manual is designed to act as a supplement to the online help for the Systems Definition Utility (SDU). It provides a quick reference to the programming of rules in a paper-based medium. The scope of this manual includes instructions for:

- Developing an effective label plan
- Using advanced programming features
- Writing input events
- Writing output commands

The programming manual also provides a quick reference to assist you when you need to find out what to write for a job specification. If you want to find out what a programming object or term is, you can turn to the glossary for help.

The chapters that follow supply numerous instructions for creating rules. The online help in the SDU offers even more assistance. For example, the manual provides information about AND groups and explains how to write rules for them. The SDU help provides similar instructions, but it also explains how to create AND groups. The programming manual provides instructions for developing labeling plan. The SDU help provides instructions for labeling objects in the Object Configuration. The best practice is to use the programming manual and the SDU help together.

Before you begin the programming process, read the Overview. This will help you develop an effective programming scheme. As you program your system, reference the sections you need. At the same time, use the SDU help to find out how to configure panels and objects.

### Finding EST2 documentation

A library of related documents supports the EST2 product line. Here is a complete list of the EST2 library:

- *EST2 Installation and Service Manual* (P/N 270186)
- *EST2 Network Site Manual* (P/N 270895)
- *EST2 Network Supplement Manual* (P/N 270894)
- *EST2 System Operations Manual* (P/N 270188)
- *EST2 System Programming Manual* (P/N 270187)
- *EST2 Installation Sheets* (P/N 3100060)
- *2-SDU Help* (P/N180902)

Our technical writers constantly update the information in this manual. Your comments during our training classes, technical

support phone calls, and field trips are used to improve this document.

### **Finding related documentation**

The *Signature Series Intelligent Smoke and Heat Detectors Applications Bulletin* (P/N 270145) provides instructions and illustrations for smoke and heat detectors.

The *Signature Series Component Installation Manual* (P/N 270497) supports the installation of the Signature Series detectors and modules.

The *Serial Number Log Book* (P/N 270267) provides a convenient means for recording the serial number of each Signature device connected to the fire alarm system.

The *SAN Annunciator Installation Guide* (P/N 250084) supports the SAN annunciators mentioned in this manual.

The *EST Speaker Application Guide* (P/N 85000-0033) provides information about the placement and layout of speakers for fire alarm signaling and emergency voice communications.

The *EST Strobe Applications Guide* (P/N 85000-0049) provides information for the placement and layout of strobes for fire alarm signaling.

The *Microline 182 Turbo Printer Handbook*, by Okidata provides all the necessary information for the maintenance and configuration of the PT-1S Form Printer. The Okidata handbook comes with the Form Printer.

---

## About the Systems Definition Utility

The Systems Definition Utility (SDU) is a database application used for setting up and programming the fire alarm system.

Using the SDU you can:

- Build setup files using forms to specify system the hardware configuration and operating options for a given project
- Create extensive system controls using advanced rules-based programming.

### Minimum equipment requirements

Before installing the Systems Definition Utility, you should make sure your computer system meets the following minimum equipment requirements:

- IBM-compatible computer
- One or more serial communications (COM) ports for connecting a bar code reader or a download cable
- One parallel printer port (LPT)
- Hard disk drive with at least 40 megabytes or more free disk space
- 32 MB of Random Access Memory (RAM)
- One 3.5-inch floppy drive
- One 2X or faster CD-ROM drive
- SVGA color display, 640 x 480
- Microsoft Mouse or other compatible pointing device
- Windows 3.1x or later

**Note:** The amount of free disk space required varies with the number of projects and the amount of audio messages you plan to save on the hard drive. A general rule of thumb is to have at least twice the amount of hard disk space required by your largest project.

The Systems Definition Utility executes highly disk-intensive functions. For best results, make sure your computer system is configured to achieve optimal performance. To optimize your system, refer to the documentation that came with your equipment.



## Optional equipment

In addition to the basic system described above, you will also need the following optional equipment to make use of some of the advanced features available in the SDU:

- Bar code reader for configuring Signature data circuits
- 300-dpi laser printer or equivalent for printing reports

The following bar code readers are recommended:

Manufacturer	Model
Zebra Technologies VTI, Inc.	SCAN•ONE
Zebra Technologies VTI, Inc.	Barcode Anything SCAN 97

Install the bar code reader per the manufacturer's instructions and configure the reader to interpret Interleaved 2 of 5 bar codes.



**Summary**

Programmers should have a good idea of how the rules work in the Systems Definition Utility before they begin their work. In addition, every project should have a plan for logical and consistent labeling schemes. Finally, an effective plan includes an awareness of object group relationships within the database and the system specifications.

**Content**

- Introduction • 1.2
- Programming using rules and objects • 1.3
  - Rules • 1.3
  - Events • 1.6
  - Device types • 1.6
  - Objects • 1.6
  - Labels • 1.6
- Creating a rules file • 1.8
  - Exceeding rules editor memory limits • 1.8
  - Order is important • 1.8
  - Avoid careless use of wildcards • 1.10
- Compiling the rules file • 1.11
- Developing a labeling plan • 1.12
  - Why use labels • 1.12
  - Formatting labels • 1.13
  - Making descriptive labels • 1.14
  - Using common label modifiers • 1.15
  - Using numbers in labels • 1.15
  - Using labels as messages • 1.17
- Identifying object group relationships • 1.18
  - Linking inputs to outputs • 1.18
  - Sample matrix • 1.18

## Introduction

All process control systems may be divided into three fundamental parts: inputs, controls, and outputs. Examples of fire alarm system inputs are smoke detectors, pull stations, and waterflow switches. Typical fire alarm system outputs include bells, strobes, control relays, and emergency audio amplifiers.

System programming is accomplished by creating a series of rules that specify what action or actions to execute when a specific event occurs. Before you write any rules, take the time to:

- Understand objects, device types, labels, and rules
- Develop a labeling plan
- Identify the objects in the system
- Determine the relationships between the inputs and outputs

## Programming using rules and objects

System programming is accomplished using a set of rules that determine the output responses for given input events. The rules are written, compiled, and then downloaded into the panel. When an input event occurs (a device goes active), the panel connected to the device searches for the device/event type combination in its response tables and, if found, executes the appropriate output commands.

The most basic fire alarm systems can be programmed using one simple rule: “when smoke detector A activates, sound horn B.” As fire alarm systems become more extensive, they require a more sophisticated set of rules to program them properly. Before you begin writing a rule, you should have a thorough understanding of:

- Rules
- Events
- Device types
- Objects
- Labels

### Rules

A rule is a programming statement that specifies which commands to execute when a certain event takes place. Every rule consists of a label, an input statement, and an output statement (or statements).

The basic syntax for a rule is:

```
[Rule_label]
Input_statement:
    Output_statement_1, {comments}
    Output_statement_2, {comments}
    Output_statement_3; {comments}
```

The rule label can be up to 40 characters in length and enclosed in brackets. The rule label can be any ASCII character except: braces “{ }”, the percent symbol “%”, the number symbol “#”, less than and greater than symbols “< >”, and asterisks “\*”.

The input statement ends with a colon and the output statement ends with a semicolon. When more than one output statement is used in a rule, each output statement must end with a comma, except for the last output statement, which must end with a semicolon. A rule may contain up to 32 output statements.

When a rule has multiple output statements, each output command will be executed in the order it is listed in the rule (from first to last). When the event activating the rule restores,

the operations performed by the rule will automatically restore in the reverse order (from last to first).

### Writing comments about rules

You can use {braces} to make comments about your rules or to isolate them for troubleshooting measures. Rules or characters enclosed by braces become invisible to the rules editor. Though comments are optional, it is advisable to include them for future reference. Place the comments wherever they show a clear relationship to the rule without cluttering it.

### Input statements

An input statement is the part of a rule that determines what must happen before the corresponding output statements will be executed. There are two different syntaxes used in rule input statements depending on the input event type selected. The two input statement syntaxes are:

`event_type :`

`event_type device_type 'object_label' :`

where:

<code>event_type</code>	Specifies the type of input event required for the rule to execute. When a system input activates, the resulting change in state creates an event. See <i>Input events</i> for detailed descriptions of event types and their usage.
<code>device_type</code>	Specifies the device type of the input device initiating the event. The <code>device_type</code> parameter is optional when using the <code>'object_label'</code> parameter. When the <code>device_type</code> parameter is not included, all devices whose label matches <code>'object_label'</code> will respond to the command.
<code>'object_label'</code>	Specifies the label of the input device or circuit that must go active for the rule to execute. The <code>'object_label'</code> parameter is optional when using the <code>device_type</code> parameter. When the <code>'object_label'</code> parameter is not included, all devices with the specified device type will trigger the rule.

**Note:** When the `device_type` and `'object_label'` parameters are optional you must specify one or the other or both.

An input statement must be valid for the rule to be successfully compiled. This means that the device type of the object identified by `'object_label'` must match that specified by `device_type`. Also, the input event specified by `event_type` must be applicable for the `device_type`.

## Output statements

An output statement is the part of a rule that determines the commands that will be executed in response to a given input and in what order. There are nine different syntaxes used in rule output statements depending on the output command selected. The nine output statement syntaxes are:

```
command ;
command delay_value ;
command 'cabinet_label' ;
command 'routing_label' ;
command device_type 'object_label' ;
command priority 'object_label' ;
command priority device_type 'object_label' ;
command priority 'amp_label' to 'channel_label' ;
command 'guard_label' Route route_id ;
```

**Note:** When the `device_type` and `'object_label'` parameters are optional you must specify one or the other or both.

where:

command	Specifies the required final state of the output device. See <i>Output commands</i> for a description of output commands and their usage.
priority	Specifies the relative importance this command has over other commands affecting the same output device. See <i>Priorities</i> for a description on priority levels and their usage.
device_type	Specifies the device type of the output device responding to the command. The <code>device_type</code> parameter is optional when using the <code>'object_label'</code> parameter. When the <code>device_type</code> parameter is not included, all devices whose label matches <code>'object_label'</code> will respond to the command.
delay_value	Specifies the length of a delay in seconds.
'object_label'	Specifies the label of the output device or circuit responding to the command. The <code>'object_label'</code> parameter is optional when using the <code>device_type</code> parameter. When the <code>'object_label'</code> parameter is not included, all devices with the specified device type will respond to the command.
'cabinet_label'	Specifies the label of the panel responding to the command (MCM1 for standalone systems).

'routing_label'	Specifies the label of the network routing group responding to the command (not available for standalone systems).
'amp_label'	Specifies the label of an amplifier.
'channel_label'	Specifies the label of an audio channel.
'msg_label'	Specifies the label of a voice message.
'guard_label'	Specifies the label of the Guard Patrol group.
route_id	Specifies the route number of the guard patrol tour.

## Events

An event is the outcome produced by a controller module when an addressable point on the panel changes state. The information contained in an event includes the logical address of the point that changed state, the event type, and the event message.

## Device types

A device type is the classification given to objects created in the database that defines the operating characteristics of the corresponding device. For example, the PULL device type is assigned to objects created for manual pull stations.

See the *Quick Reference* for a list of device types and their use.

## Objects

An object is a database entity that represents an addressable point in the system like a smoke detector, a switch, or a light emitting diodes (LED).

Objects can also be:

- Logic groups
- Voice messages
- Pseudo points

For example, the Signature controller module is an object as are any Signature devices connected to it.

**Note:** You do not have to include the device type with the object label. As a safeguard, however, we suggest that your input statements include the device type and the object label.

## Labels

A label is a descriptive word or set of words that identifies a specific object in the project database to simplify programming. Labels are also used to identify a rule in the rules file. Typically,



object labels describe the physical location of the device that the object represents.

**Tip:** Extensive labels are generally harder to read. Sometimes less is more.

Labels have the following characteristics:

- Labels must be unique. Duplicate labels generate compiler errors and prevent the database from compiling.
- Labels are arbitrary except for labels that are automatically assigned by the system.
- Labels are not case sensitive and may contain up to 40 characters. The characters may be any ASCII character except: braces “{ }”, the percent symbol “%”, the number symbol “#”, less than and greater than symbols “< >”, asterisks “\*”, and blank spaces.

The SDU automatically replaces invalid label characters as shown below to prevent programming errors.

**Character substitution table**

User types	User sees
space	underscore
*	@
%	@
#	@
<	(
>	)
{	(
}	)

---

## Creating a rules file

### Exceeding rules editor memory limits

The SDU provides an editing tool for creating and editing a rules file. The tool is known as the Rules Editor and is limited to editing files of 32 kilobytes or less. Most projects do not approach the memory limits of the rules editor. If it appears, however, that you cannot avoid exceeding the limits, you can use a different text editor.

**Note:** You will no longer be able to use the rules editor if your rules exceed the 32-kilobyte limit.

---

### To use a text editor that can exceed the 32K limit:

1. Enter the rules in the text editor (MS Word®, Wordpad®, or Notepad®).
2. Save the file as PROJRULE.TXT in C:\\FAST\\2-SDU\\PROJECT\\PROJECT\_NAME\\01\_00\_00.
3. Open the project in the 2-SDU.
4. Compile the rules by selecting Compile in the Rules menu.

If you attempt to open the rules editor you will see a warning box, which states that you have exceeded the 32K limit. Nevertheless, you can still compile the rules.

**Tip:** Look for instances where advanced programming techniques may be used to reduce the rule file's size.

### Order is important

When you have more than one rule that uses the same input requirements to trigger separate output responses, the order in which the rules appear in the rules file affects how the rules are executed. The compiler takes multiple rules with common input statements and executes them as though they were a single rule containing multiple output statements. The output statements are executed in the same order that they appear in the rules file.

Suppose, for example, that you have a rules file containing:

```
[Rule 1]
ALARM SMOKE 'LVL5_SMK1' :
      DELAY 30,
      FANON 'STAIRWELL_FAN_2';
```

```
[Rule 2]
ALARM SMOKE 'LVL5_SMK1' :
    LEDON 'FACP_LED28' ;

[Rule 3]
ALARM SMOKE 'LVL5_SMK1' :
    ON AUDIBLE 'LVL5_AMP1' ;
```

After compiling, the rules would be executed as though they were written:

```
[Compiled Rule]
ALARM SMOKE 'LVL5_SMK1' :
    DELAY 30,
    FANON 'STAIRWELL_FAN_2',
    LEDON 'FACP_LED28',
    ON AUDIBLE 'LVL5_AMP1' ;
```

Notice that the placement of Rule 1 before Rule 3 results in a delay before sounding the horn. The outputs scattered over the three rules all have a common input. If you begin by grouping all the outputs to the common input, you can easily spot conflicts and avoid them. You can also economize on rule editor memory.

It makes a difference, however, when you have more than one rule that uses the same device but different event types. To illustrate this suppose you have the following two rules:

```
[Rule 1]
TROUBLE SMOKE 'LVL5_*':
    LEDON 'FACP_LED28',
    DELAY 300,
    ON AUDIBLE 'LVL5_AMP1' ;

[Rule 2]
ALARM SMOKE 'LVL5_*':
    FAST 'CAB1_PNL1_LED1',
    ON AUDIBLE 'LVL5_AMP1' ;
```

If any smoke detector on level 5 goes into trouble, the panel will activate Rule 1. The panel will then turn the specified LED on, wait 5 minutes (300 seconds), and turn on the specified amplifier. If another smoke detector on level 5 goes into alarm during the delay period, the panel will activate Rule 2, which will immediately override any commands in Rule 1.

The panel does not wait for the trouble response (Rule 1) to finish before running the alarm response (Rule 2). In this example however, panel will resume executing Rule 1 if the trouble condition remains after the delay period expires.

**Tip:** When adding rules to a previously compiled rules file, place them at the beginning of the file so they will be checked first. If they need to be in a certain spot in the file, you can move them afterwards.

## Avoid careless use of wildcards

A wildcard is a powerful programming aid that can help reduce the size of your rules file. Remember, however, that the wholesale use of wildcards could have undesirable results. Consider, for example, the following three rules:

```
[Rule 1]
ALARM '*':
    LEDON 'FACP_LED28';

[Rule 2]
ALARM 'SMK_*':
    DELAY 10,
    LEDON 'FACP_LED29';

[Rule 3]
ALARM 'SMK_LVL1_*':
    DELAY 10,
    LEDON 'FACP_LED30';
```

A wildcard in the input statement will cause the compiler to create an output response for each device that matches it. In the rules above, order is still important, and the output response for an alarm-initiating device labeled 'SMK\_LVL1\_1' would result in:

```
[Compiled Rule]
ALARM 'SMK_LVL1_1':
    LEDON 'FACP_LED28',
    DELAY 10,
    LEDON 'FACP_LED29',
    DELAY 10,
    LEDON 'FACP_LED30';
```

The addition of the smoke detector device type (SMOKE) to the input statement would limit the output responses to only smoke detectors with matching labels.

Likewise, you must be careful when using the wildcard character in an output statement. For example, placing a wildcard immediately following the N-variable may return undesirable results. 'LVL<N:1>\*' will select 'LVL1', 'LVL19', 'LVL199', and so on.

---

## Compiling the rules file

You will have to compile the rules after you create them in the rules editor. The compiler checks the rules for improper syntax and other faults before translating the database into a binary setup file. The compiler also checks the database for unlabeled objects and objects with duplicate labels.

The speed at which the compiler checks the rules file depends on the size of the database and the construction of the rules.

Compile speed is relative, but rules that specify:

- Only the device type compile fastest
- Only the object label compile slowest
- Device type and object label compile at medium speed

The rules compiler ignores any characters between opening and closing braces. The rules compiler also ignores tabs, spaces, and line breaks. Run the rules compiler for every change you make in the project database.

**Tip:** As a troubleshooting aid, you can make temporary use of braces to comment out parts of the rules file that are correct.

---

## Developing a labeling plan

### Why use labels

#### The reduction of human effort

Years ago, system designers had to use elaborate addressing schemes when they configured large fire alarm networks. The numeric addresses made sense to the network hardware, but people struggled to interpret and remember them. The tracking of numerous addresses increased the effort required to program a system.

Advancements in system software design now permit referencing system components by a name instead of a number. In programming terms, this name is called a label. A suitable label for a system component will immediately reveal its identity, location, or function.

For example, what can you tell about the following two items?

Address	Label
010324	SMOKE_ELEVLOBBY_LEVEL1

Both items identify the same smoke detector, located in the elevator lobby on the first floor of a multifloor building. The item on the left is a numeric device address that reveals little information about the device. The item on the right is a label that reveals quite a bit more information about the detector.

#### The formation of a consistent plan

System programming requires the assignment of labels to modules and other objects in the database. Presumably, several people will need to know how to interpret the label assignments. Also, the meaning of the labels needs to be clear long after the programmer has assigned them. A sound label plan will ensure logical and useful labels.

**Tip:** Ensure that every label is unique. Any two objects with identical labels will cause a system conflict.

A good plan will include consideration of:

- Label format
- Descriptive label content
- Common label modifiers
- Label numbering
- Labels as messages

## Formatting labels

The person responsible for the labeling plan should remember that labels will be viewed online, on printed reports, and on the system display panel. Formatting considerations may include:

- How to separate label modifiers
- Whether to use uppercase, lowercase, mixed-case characters
- How to abbreviate label modifiers

**Note:** Functionally, it makes no difference whether your labels contain upper or lowercase characters.

### Methods for formatting labels

Suppose that you create labels for multiple cabinets in an industrial park of several buildings. The following illustration shows three different ways to format a label for a cabinet in one of the buildings:

BLDG1CAB                      BLDG1\_CAB                      Bldg1Cab

The left label may be hard to read because it uses all uppercase characters and there is no separation between the label modifiers BLDG1 and CAB. The center label places an underscore between the two modifiers, which makes it easier to read. The right label uses upper and lowercase characters to differentiate between label modifiers.

### Label formatting advice

Be consistent in any label-formatting plan you choose. Consistency is the most important factor in making your labels easy to use and understand. Notice in the example above that each label abbreviates “Building” the same way. You want to avoid using BLDG1, Bldg\_1, and Bldg1 as label modifiers to reference the same building.

Find a comfortable balance between readability and length. Extra underscores may separate label modifiers and make the label more intelligible but they may also add unnecessary length. Consider the label BLDG\_1\_LEVEL\_7\_CAB. Perhaps a better way to label the cabinet is BLDG1\_LVL7\_CAB. The reduction of separators and the abbreviation of level 7 made the label more efficient.

Projects that involve several programmers will require cooperation between everyone who assigns labels. The team should agree on a labeling format and adhere to it.

## Making descriptive labels

The content of the label should include descriptive modifiers that reference the object's location, its function, or its device type. The combination of label modifiers to describe the object's location and function depends on its application.

### Using labels to describe location

Each cabinet in a system is given a numerical designation to identify its position, but not its location. Therefore, a cabinet label should always describe its location. Typical modifiers for cabinet labels in a high-rise application might include:

- LOBBY
- LEVEL7
- EAST\_WING

Typical modifiers for cabinet labels in a campus application might include building names, like:

- BLDG1
- WILSON\_HALL
- ADMIN

### Using labels to describe function

Some devices in a system provide specific functions. Typical modifiers for devices, which provide a specific function, might include:

- DMPR\_CNTRL\_SW (for a damper control switch)
- AMP\_LVL7 (for an amplifier on the 7<sup>th</sup> floor of a building)
- LEDPANEL (for an LED panel)

### Using labels to describe a device type

A system may contain a large number of certain devices, such as detectors. For devices that exist in large quantities, you might want to consider using a label to describe the object's device type. Typical modifiers for these devices include:

- SMK (for smoke detectors)
- PULL (for pull stations)
- LED (for LEDs)
- SW (for switches)

**Tip:** You can also use an object's device type in the input statement of a rule. See *Creating rules* later in this chapter.



Using common label modifiers

As you develop your labeling plan, consider the use of common label modifiers to simplify programming decisions. Common label modifiers quicken the assignment of labels in the Prefabricated Labels editor and allow the use of wildcards.

Using common labels for building levels

The use of Floor numbers as label modifiers can sometimes create extra work for the system designer. The extra work comes from areas, like basements and mezzanines, which do not have floor numbers. Consequently, you may want to use a generic label modifier, like LVL, which can apply to all areas of the building. Figure 1-1 illustrates the use of LVL as a common label modifier in a building with several kinds of areas.

**Note:** You may still want messages that appear on the system display panel to reference floor numbers.

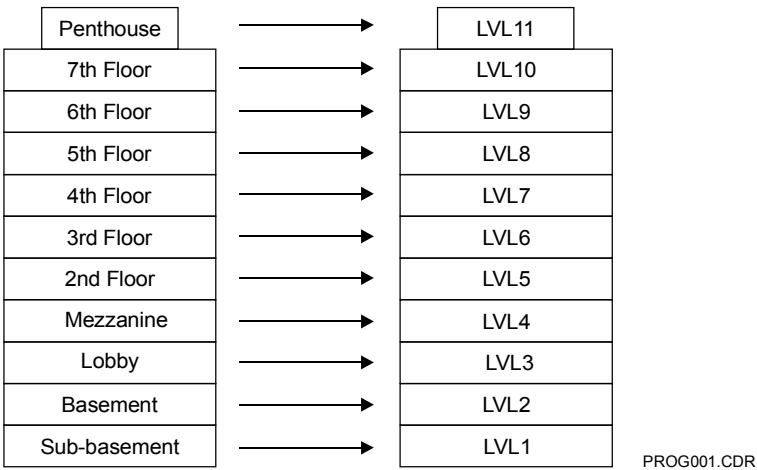


Figure 1-1: Diverse building areas labeled by a common modifier

Using numbers in labels

Another way to make programming easier is to include numbers in your labeling plan. Numbers are particularly useful in labels if several of them have common label modifiers. You will also be able to apply advanced programming techniques to numerically modified labels. For example, AMP\_LVL1, AMP\_LVL2, and AMP\_LVL3 will work well in floor-above and floor-below applications. Devices activated floor-by-floor should include some reference to the floor level number in the device label.

### Using numbers to make a label unique

Unique labels can identify special control functions like elevator lobby, smoke detector, stairwell, and duct applications. For example, the modifier LVL2\_DUCT identifies a duct smoke detector on the second level of a building. Additional duct detectors on the same floor can use a similar label, but they require a unique numeric modifier for identification. For example, LVL2\_DUCT1 is one possible label for a duct smoke detector located on the 2nd level. The only distinction it has from any other duct label on level 2 is the 1 at the end of DUCT. If there were no other duct detectors on level 2, LVL2\_DUCT would be sufficient to make the label unique.

### Using wildcards with numbers

In programming, labels with one or more common modifiers create a tremendous advantage. Common label modifiers allow the use of wildcards (\*). A wildcard acts like a space reserved for any number that follows a common modifier. For example, a database may include the following labels:

- LVL1\_DUCT1
- LVL1\_DUCT2
- LVL1\_DUCT3

In the rules, LVL1\_DUCT\* could easily replace LVL1\_DUCT1, LVL1\_DUCT2, or LVL1\_DUCT3. If LVL1\_DUCT3 becomes active, the 3 will replace the wildcard (\*) in the rule.

**Tip:** For more information about using wildcards, see *Advanced programming features*.

## Using labels as messages

You can program the system to display a different message on the front panel display when each associated object goes active or into alarm. If you design your labels with this function in mind, you may be able use the label as the message instead of creating a separate message in the Object Configuration.

Suppose that a device goes into alarm and you want the front panel display to show its location and type. If you label the smoke detector as ROOM101\_SMOKE, the display will read:

```
SMOKE ACTIVE  
ROOM101_SMOKE
```

Suppose also that the smoke detector is the only device in the room, and a single modifier can uniquely identify it. You might label the detector ROOM101, so the display would read:

```
SMOKE ACTIVE  
ROOM101
```

This strategy limits the redundant information on the display and shortens the label. Remember, however, that this plan will only work if the labels remain unique.

**Note:** Only two 20-character rows appear in any message on the front panel display. The balance of the message text is sent to the printer.

---

## Identifying object group relationships

### Linking inputs to outputs

Whenever you read a job specification, try to get a feeling for the functions (outputs) required of the system. Try also to find the locations and types of devices (inputs) that initiate these functions. The goal is to create user-friendly labels from common functions, areas, or devices, which will also permit efficient programming. The most effective method of recognizing common input and output functions is to create an input-to-output matrix.

### Sample matrix

Figure 1-2 is an abbreviated example of an input-to-output matrix. Place a mark (X) at the intersection of each output activated by the input. When you have entered every input and output on the matrix, inspect it for groups or patterns of marked intersections. The patterns may appear as columns or blocks of marked intersections. When you identify such groups, look at the inputs to see what they have in common.

### Alarm functions

Figure 1-2 illustrates six object groups of marked intersections. Three object groups appear in columns, in which any input will:

- Sound general alarm
- Notify central station
- Lock stairwell Doors

Consequently, the first rule must activate all three functions whenever any device in Figure 1-2 goes into alarm.

### Elevator capture function

The next object group involves the Elevator Capture function. All of the inputs that initiate the elevator capture function are smoke detectors in the elevator lobby. Yet, the system also includes other smoke detectors. Therefore, the label modifier SMK is not the best choice to identify the elevator capture function. You may include SMK as a modifier in the device label, but you should avoid using it to signify the elevator capture function. Better choices might include:

- ELEVATOR
- LOBBY
- ELEVATOR\_SMK1
- LOBBY\_SMK1
- ELEVLOBBY\_SMK

Remember to make the labels unique. Suppose, for example that you need to assign a label to the smoke detector in the elevator lobby on the second floor. LVL2\_SMK1\_ELEVLOBBY would work well with the elevator capture rule. The keyword ELEVLOBBY will become a part of the elevator capture rule. Any label with an ELEVLOBBY modifier will activate an elevator capture.

Input functions	Output functions										
	General alarm	Notify central station	Elevator capture	Alt. elevator capture	Basement HVAC	1st floor HVAC	2nd floor HVAC	3rd floor HVAC	Computer shutdown	1-3 floor exhaust	
3rd floor pull station	X	X								X	LVL3
3rd floor area smoke detector	X	X								X	
3rd floor duct smoke detector	X	X					X		X	X	
3rd floor elevator lobby smoke detector	X	X	X							X	LVL2
2nd floor pull station	X	X						X		X	
2nd floor area smoke detector	X	X						X		X	
2nd floor duct smoke detector	X	X				X		X	X	X	LVL1
2nd floor elevator lobby smoke detector	X	X	X					X		X	
1st floor pull station	X	X								X	
1st floor area smoke detector	X	X								X	LVL0
1st floor duct smoke detector	X	X			X				X	X	
1st floor elevator lobby smoke detector	X	X								X	
Bsmnt floor pull station	X	X								X	LVL0
Bsmnt floor area smoke detector	X	X								X	
Bsmnt floor duct smoke detector	X	X		X						X	
Bsmnt floor elevator lobby smoke detector	X	X	X							X	

[PROG002.CDR]

Figure 1-2: Input-to-output matrix

### Computer shutdown function

Another object group is the computer shutdown function, which can be initiated by any input from the second floor. Accordingly, every device on the second floor should include a label modifier

like LVL2. In fact, the floor or level number should modify every device label in a multifloor structure. Any device with the LVL2 modifier in its label will activate the computer shutdown function. Note that the elevator lobby detector labeled in the previous example will also shutdown the second floor computer. LVL2 is also a keyword in the elevator shutdown rule.

### **Floor exhaust function**

The final object group is the 1-3 floor exhaust function, which is initiated by duct smoke detectors on floors 1, 2, and 3. Include the modifier DUCT with the device label for all duct detectors.

Notice that the input-to-output matrix rules out the activation of the 1-3 Exhaust function by the basement duct detector. This fact should not affect the choice of DUCT as a label modifier in the device label. It will still be useful to the designer in determining the device function.

DUCT is a keyword in the 1-3 exhaust rule. Yet, not all detectors with DUCT in their labels are required to activate the 1-3 exhaust function. Therefore, the additional keywords LVL1, LVL2, and LVL3 will be added to the 1-3 exhaust function rule. Devices that will activate the 1-3 exhaust rule will include the label modifiers:

- LVL1\_DUCT
- LVL2\_DUCT
- LVL3\_DUCT

### **Specific functions and devices**

Notice that there are a few matrix entries, which are not part of any group. A specific device will activate these functions. The rules used to activate these outputs are written for the specific function, rather than use by multiple objects.

The matrix for a major project would be very large and detailed. Many inputs will have only one or two output correlations. The experienced system designer will recognize developing patterns.

In Figure 1-2, note that the first, second, and third floors are basically copies of each other. This is typical of high rise buildings. This duplication of functions permits a designer to recognize that the floor (or level) number will be an important label modifier when creating labels for devices.

Our example also reveals the importance of well-planned label modifiers. The proper selection of device labels permits the designer to account for the common functions throughout the building by writing only a few rules.

## Advanced programming features

### Summary

Advanced programming techniques provide additional programming tools, which can simplify application programming and reduce the size of the rules file.

### Content

- Variables • 2.2
  - Using wildcards • 2.2
  - Using the *N-variable* • 2.2
  - Using *L-variables* • 2.3
  - Using mathematical operators • 2.4
- Priorities • 2.5
  - Priority 0 (Clear) • 2.5
  - Priority 99 (Latch) • 2.5
- Logic groups • 2.7
  - AND groups • 2.7
  - Bell code groups • 2.7
  - Dialer groups • 2.7
  - Guard patrol groups • 2.8
- Actions and Sequences • 2.9
  - Related information • 2.9
  - Actions • 2.9
    - StartAction • 2.10
    - GuardPatrolAlarm • 2.11
    - NetworkClassAFault • 2.11
  - Sequence • 2.11
  - StartSequence • 2.12
  - TimeControl • 2.13

## Variables

Variables add flexibility to the rules and sometimes reduce the number of rules required to program the system.

### Using wildcards

The asterisk (\*) can be used in a rule to conditionally select devices based on the character pattern used in their object labels. The asterisk may be substituted for any single or group of characters anywhere in the label.

Example	Selects
LVL*	any devices whose labels begin with LVL
*_SMK	any devices whose labels end with _SMK
LVL*_SMK	any devices whose labels begin with LVL and end with _SMK
*_SMK_*	any devices that have _SMK_ somewhere in the label

In the following example, any smoke detector on levels 2-10 will activate the horns and strobes on the floor of incident.

```
ALARM SMOKE 'LEVEL<N:2-10>_SMK*':
      ON AUDIBLE 'LEVEL<n>_HORN',
      ON VISUAL 'LEVEL<n>_STROBE';
```

### Using the *N-variable*

The programming variable N can be used in a rule to conditionally select devices based on the numerical indexing used in their object labels.

When using the *N-variable*, the numbers required to make the conditional true are specified in the input statement. When the input statement becomes true, the number is substituted for the variable *n* in the output statement.



Example	Selects
<N: #>	a single number entry
<N: # - #>	a range of numbers
<N: #, #, # - #, # - #>	a combination of single numbers and number ranges

where:

- # may be any number between 0 and 32767
- Wildcards may not be used in place of a number
- The *N-variable* can only be used once in a label

### Example

When the device labeled LVL1\_PHONE\_JACK confirms that it has been energized, the LED labeled LVL1\_PHONE\_CALL is turned off.

```
[PHONE_CONNECT_MADE]
CONFIRMATION FIREPHONE 'LVL<N:1,3-5>_PHONE_JACK': LEDOFF LED 'LVL<N>_PHONECALL';
```

**Note:** Placing a wildcard immediately after the N-variable may return undesirable results. For example, specifying 'LVL<N:1>\*' includes 'LVL1', 'LVL19', 'LVL199', and so on.

### Using *L-variables*

The programming variable L can be used in the output statement of a rule to direct commands to multiple devices based on the numerical indexing used in their object labels.

Example	Selects
<L: #>	a single number entry
<L: # - #>	a range of numbers
<L: #, #, # - #, # - #>	a combination of single numbers and number ranges

where:

- # may be any number between 0 and 32767
- Wildcards may not be used in place of a number
- The *L-variable* can only be used once in a label

**Example**

The following rule lights the same LED on four separate panels when a detector goes active in Building 1.

```
[SAMPLE_RULE]
ACTIVE SMOKE 'BLDG1_SMK_<N:1-4>':      ON LED 'BLDG<L:1-4>_LED<N>';
```

**Using mathematical operators**

Mathematical operators can be used in conjunction with the N-variable in the rule output statement to conditionally select devices based on the numerical indexing used in their object labels.

When using mathematical operators, the numbers required to make the conditional true are specified by the N-variable in the input statement. When the input statement becomes true, the number is substituted for the variable *n* in the output statement and then increased or decreased by the number specified by the #-variable.

Example	Substitutes
<n + #>	a number greater than the number determined by <i>n</i> in the input statement
<n - #>	a number less than the number determined by <i>n</i> in the input statement

where:

- # may be any number between 1 and 32767, inclusive.
- Wildcards may not be used in place of a number.
- <N + #> may not equal greater than 32767.
- <N - #> may not equal less than zero.

**Example**

Any smoke detector on levels 2 through 10 will turn on the fire floor horns and strobes when it goes into alarm. It will also turn on the horns and strobes of the floors above and below fire floor.

```
[FloorAbove_FloorBelow]
ALARM SMOKE'LVL<N:2-10>_SMK*':      ON AUDIBLE 'LEVEL<N>_HORN',
                                      ON VISUAL 'LEVEL<N>_STROBE',
                                      ON AUDIBLE 'LEVEL<N+1>_HORN',
                                      ON VISUAL 'LEVEL<N+1>_STROBE',
                                      ON AUDIBLE 'LEVEL<N-1>_HORN',
                                      ON VISUAL 'LEVEL<N-1>_STROBE';
```

---

## Priorities

**Tip:** If outputs do not require prioritizing, assign the default priorities as 00 in the project preferences.

Priorities specify the order of importance a command has over other commands affecting the same output device. When an output device is turned on (set) or turned off (reset) using a priority, the state of the output device can only be changed by another command with a greater priority. If another command is executed that has a priority of equal or lesser value, then the output device will not change states.

The following priority values may be used in a rule:

- Any number between 0 and 99
- Low (25), medium (50 ), or high (75)
- Clear (0), or latch (99).

The default priority is set for 0, but can be changed in the project preferences in the Systems Definition Utility. Priority 0 and 99 perform special functions as described in the following sections.

### Priority 0 (Clear)

Priority 0 may be used exclusively when outputs do not need to be prioritized. When an output is turned on or off with priority 00, the following occurs:

- The output is forced to the new condition (set or reset) regardless of its previous priority. In this case, priority 00 is considered the highest priority.
- Immediately after the output changes state, it assumes the characteristics of an output with the lowest priority level and may be commanded to the opposite state again by any new priority.

### Priority 99 (Latch)

Priority 99 may be used to latch the output of a device to either the set or the reset state. An output, when latched, will not restore when the input causing the change of state has restored. This type of priority may be useful in the following applications:

- amplifier backup switching
- security circuits
- fan control circuits

The following points must be considered when using relay priority 99:

- An output that is turned on by priority 99 can only be turned off by a priority 0
- An output that is turned off by priority 99 can only be turned on by a priority 99 or 0

Additionally, when priority 0 overrides priority 99, the output is assigned the lowest nonlatching priority level (priority 0).

## Logic groups

Input devices can perform system functions based on their association with logic groups. Logic groups include AND groups, Bell Code groups, Dialer groups, and Guard Patrol groups. Logic groups are defined in the Object Configuration.

**Note:** The programmer must select labeled devices in the Object Configuration tables to include them in a logic group. Also, the device must be appropriate for the logic group. For example, a smoke detector is not compatible with a guard patrol group.

### AND groups

AND groups generate a group response when all devices in the group are active. In addition to configuring the logic group in the database, the AND group response must also be defined in a rule statement.

**Note:** Each active device in a group generates an individual response.

### Example

10 smoke detectors are installed in a computer room. Three of the ten detectors are assigned to AND\_GROUP1 for setting off the automatic release of a fire-extinguishing agent.

The first rule below provides an individual response for each detector in the room. The second rule specifies that the detectors in AND\_GROUP1 will release the agent when it goes active.

```
[COMPUTER_ROOM]
ALARM SMOKE 'COMPROOM_SMK<N:1-10>:      LEDON LED 'COMPUTER_ROOM';

[AND_GROUP1]
DEFINE AND 'AND_GROUP1':                  ON RELAY 'AGENT_RELEASE';
```

### Bell code groups

Bell Code groups provide a coded notification signal, which represents a specific floor or zone. Bell code groups do not require rules.

### Dialer groups

Dialer groups provide automatic notification of an incident on a particular floor or zone to a Central Monitoring Station. Dialer groups do not require rules.

## Guard patrol groups

Patrol groups are sets of input devices, which must be activated within specific time intervals and in sequential order. The typical application is for guard patrols. Guard patrol groups are defined in the Object Configuration. Each route of a guard patrol group lists the number of stations and minimum times to reach each patrol station.

**Note:** The GuardPatrolAlarm response must be defined in a rule statement. See *Define event* and *Activate command*.

### Example

**Tip:** Open the Object Configuration and click tab labeled Guard Patrol Groups to get the full benefit of this example.

A programmer created a guard patrol in the Object Configuration. The guard patrol was labeled GUARD\_PATROL\_GROUP1 and it includes several routes. Route 2 requires the toggling of an SWU-8 switch at a station on each floor. Each station must be reached within ten minutes of the last station. The building consists of six floors and the check-in sequence is in ascending order from LVL1\_SWU8\_SW1 to LVL6\_SWU8\_SW1.

The guard patrol is actually initiated at the fire alarm control panel upon the activation of a switch labeled FACP\_SW17. The switch is located on the second LED/switch module at the fire alarm control panel. The system will issue a GuardPatrolAlarm for any check-in that is overdue or out of sequence. In this case, the GuardPatrolAlarm will light an LED labeled FACP\_LED17. The LED also has a slip-in label next to it to identify the guard patrol route.

To carry out the specifications of the guard patrol group, the system requires two rules. One rule will turn on the guard patrol and the specific route. The other rule will define the system response to a GuardPatrolAlarm when the route is not properly executed.

```
[ON_GUARD_PATROL1_ROUTE2]
ACTIVE SWITCH 'FACP_SW17': ONGUARD GUARDPATROL 'GUARD_PATROL_GROUP1' 2;
```

```
[GUARD_PATROL_ALARM]
DEFINE GUARDPATROLALARM 'MCM1': LEDON LED 'FACP_LED17';
```

## Actions and Sequences

Actions and sequences may be defined in rule statements to execute common output responses. These common output responses can then be activated whenever required without having to duplicate them in each rule.

### Related information

You can find more information about actions and sequences in:

- Activate command
- Define events
- Restore command

### Actions

Actions that perform certain system functions are predefined in the panel controller. For example, the front panel Drill switch calls the Drill action. The programmer may expand these actions by defining additional output responses in a rule statement. The system programmer can also define other actions.

Predefined Actions	Description
AlarmSilence (Action 9004)	Turns off all audibles and, optionally, all visuals depending on panel configuration.
Drill (Action 9003)	Turns on all audibles and, optionally, all visuals depending on panel configuration.
Evacuation (Action 9500)	Turns on all audibles and, optionally, all visuals depending on panel configuration.
SysReset (Action 9002)	Resets the panel controller and all restored alarm initiating devices.
LampTest	Flashes all LEDs on panel controller display and all local display strips.

### Modifying predefined actions

To add additional responses to a predefined action, use the following syntax:

```
[rule_label]  
DEFINE action_name 'object_label':    output_statement;
```

where:

- action\_name is name of the predefined action
- 'object\_label' is the label of the panel controller\*
- output\_statement is the additional system responses

\*'MCM1' for standalone systems

### Creating user-defined actions

To create a user-defined action, use the following syntax:

```
[rule_label]  
DEFINE ACTION 'object_label':        output_statement;
```

where:

- [rule\_label] is the name of the user-defined action
- 'object\_label' is the label of the panel controller\*
- output\_statement is the system response

\*'MCM1' for standalone systems

**Note:** There is no limit to the number of nested actions that can be activated from within an action (i.e., action 1, activates action 2, which activates action 3, etc.)

### StartAction

Use StartAction to initialize the configuration of selected program elements whenever the system is powered up (cold started) or restarted (warm boot). For example, a StartAction may enable the system time control functions or initialize supplementary systems connected to the panel.

To define a StartAction, use the following syntax:

```
[rule_label]  
DEFINE STARTACTION 'object_label':    output_statement;
```

where:

- 'object\_label' is the label of the panel controller\*
- output\_statement is the system response

\*'MCM1' for standalone systems

**Note:** The StartAction does not occur after a panel reset.



## GuardPatrolAlarm

Use GuardPatrolAlarm to define a set of responses that will execute when a guard patrol station is activated out of sequence or not activated within the required time period.

To define a GuardPatrolAlarm, use the following syntax:

```
[rule_label]
DEFINE GUARDPATROLALARM 'object_label': output_statement;
```

where:

- 'object\_label' is the label of the panel controller\*
- output\_statement is the system response

\*'MCM1' for standalone systems

See *Guard patrol groups* for an example.

## NetworkClassAFault

Use NetworkClassAFault to define a set of responses, which will execute upon the detection of an open circuit on the network RS-485 line.

To define a NetworkClassAFault, use the following syntax:

```
[rule_label]
DEFINE NETWORKCLASSAFAULT 'object_label': output_statement;
```

where:

- 'object\_label' is the label of the panel controller\*
- output\_statement is the system response

\*'MCM1' for standalone systems

## Sequence

Use Sequence to define a series of time-delayed output responses. These responses can be commands, actions, or other sequences. Time delays are placed before each response using the Delay command and may be set anywhere between 5 and 4096 seconds, inclusive.

Typical uses for sequences are:

- To automatically change a tone to a pre-recorded message after a time delay
- For staggering the activation and shut down of air supply fans during an alarm

**Note:** Outputs that have been SET by a sequence will not restore when the input causing the sequence has restored. These outputs must be reset by the SysReset function or by a switch.

To define a Sequence, use the following syntax:

```
[rule_label]
DEFINE SEQUENCE 'object_label':      output_statement;
```

where:

- 'object\_label' is the label of the panel controller\*
  - output\_statement is the system response
- \*'MCM1' for standalone systems

The panel controller allocates 1024 blocks of memory space for executing sequences. Each sequence consists of a 1-block definition header and any number of 1-block delay/action pairs. It is possible to define a single sequence with up to 1023 blocks (delay/action pairs.)

```

                                [FANON_SEQ]
1 BLOCK — DEFINE SEQUENCE 'MCM1':
1 BLOCK — [ DELAY 5,
            [ NSFANON FANCONTROL 'EXHFAN_RELAY',
1 BLOCK — [ DELAY 10,
            [ NSFANON FANCONTROL 'SUPFAN_RELAY';
= 3 BLOCKS
```

If more than one sequence is active at the same time, the total number of sequence blocks may not exceed 1024.

**Note:** There is no limit on the number of actions or nested actions that a sequence executes.

### StartSequence

Use StartSequence to initialize the configuration of selected program elements whenever the system is powered up (cold started) or restarted (warm boot.) A start sequence may be used, for example, to stagger power to air supply fans and large motor-driven circuits.

**Note:** The StartSequence does not occur after a panel reset.

## TimeControl

Use TimeControl to define a set of responses that will activate when a configured time control goes active. Time controls are created in the Time controls dialog box.

---

**Caution:** Make sure you create actual time controls before you create rules for them. If the compiler sees a rule that references a nonexistent time control, it may go into a loop error and corrupt the SDU software.

---

### To create a time control:

---

1. Click the following path in the SDU: Configure/Time Controls.
2. In the Select Time Control dialog box, click New.
3. Enter the parameters for the time control in the Time Control dialog box.
4. Click OK.

To define a time control action, use the following syntax:

```
[RULE_LABEL]
DEFINE TIMECONTROL 'object_label':    output_statement;
```

where:

- 'object\_label' is the label of the configured time control
- output\_statement is the system response

**Note:** Time controls may be enabled from the front panel or automatically at startup, but no more than 63 time controls may be enabled at any one time.

### Example

The programmer created a time control that will start every day at 19:30 hours (7:30 PM) and last for 60 minutes. The programmer labeled the time control TCGP1\_2 to signify its function turning on route 2 of GUARD\_PATROL\_GROUP1. By itself, however, the time control will have no effect on the system. The programmer's last task is to define the time control in the rules.

```
[ON_TCGP1_2]
DEFINE TIMECONTROL 'TCGP1_2' : ONGUARD GUARDPATROL 'GUARD_PATROL_GROUP1' 2;
```

**Note:** For more information about guard patrols, see *Guard patrol groups*.

### Summary

Use this chapter to compose input statements in the rules editor.

### Content

- Active events • 3.2
- Alarm events • 3.4
- CallIn events • 3.5
- Confirmation events • 3.6
- Define events • 3.7
- Monitor events • 3.9
- Supervisory events • 3.10
- TestActive events • 3.11
- TestTrouble event • 3.13
- Trouble events • 3.15

---

## Active events

### Purpose

Use the Active input event to trigger the execution of a rule when any device goes active during normal, standby operation.

### Syntax

```
Active device_type 'object_label' :
```

– or –

```
Active device_type :
```

### Device types

Table 3-1 identifies valid devices for Active events.

**Table 3-1: Valid device types and devices**

Device types	Devices
Alarm initiating devices	Duct Heat Pull Smoke SmokeVfy StageOne Waterflow
Latching or non-latching supervisory devices	AuxPowerSupply Gatevalve Power SprinklerSupervisory SupDuct Supervisory Tamper Temperature
Non-latching monitor devices	DamperFeedback DoorFeedback Emergency FanFeedback Monitor
Switch devices	Switch UserDefinedSwitch

### Object labels

The object label specifies the unique label of the device initiating the Active event.

### Examples

Example 3-1 identifies possible syntax for a rule, in which:

- LVL1\_PHONE\_CNTRL turns on LVL1\_PHONE\_JACK
- LVL3\_PHONE\_CNTRL turns on LVL3\_PHONE\_JACK
- LVL4\_PHONE\_CNTRL turns on LVL4\_PHONE\_JACK
- LVL5\_PHONE\_CNTRL turns on LVL5\_PHONE\_JACK

#### Example 3-1: Syntax for active phone switches

```
[PHONE_SWITCH]
ACTIVE SWITCH 'LVL<N:1,3-5>_PHONE_CNTRL' : ON FIREPHONE 'LVL<N>_PHONE_JACK';
```

Example 3-2 identifies possible syntax for a rule that will turn on NAC\_1 and NAC\_2 when a heat detector goes active. The rule will work for every heat detector on the first floor as long as the labels follow a consistent scheme. For example, the labeling scheme could be HEAT\_FLR1F\_1 for a fixed-temperature heat detector or HEAT\_FLR1R\_1 for a rate-of-rise heat detector.

#### Example 3-2: Syntax for active heat detectors

```
[HEAT_FLR1]
ACTIVE HEAT 'HEAT_FLR1*_*' : ON 'NAC_<L:1-2>';
```

---

## Alarm events

### Purpose

Use the Alarm event to trigger the execution of a rule when an alarm-initiating device goes active during normal, standby operation.

### Syntax

```
Alarm device_type 'object_label' :
```

– or –

```
Alarm device_type :
```

### Device types

Valid devices for the Alarm events include:

- Duct
- Heat
- Pull
- Smoke
- SmokeVfy
- StageOne
- Waterflow

### Object labels

The object label specifies the unique label of the device initiating the Alarm event.

### Example

Example 3-3 identifies possible syntax for a rule that will set off a strobe on the respective floor of any pull station on levels 1 through 10. For example, a pull station on level 6 will cause the rule to turn on the strobe in level 6.

#### Example 3-3: Syntax for manual pull stations

```
[MANUAL_PULL]  
ALARM PULL 'LEVEL<N:1-10>_PULL' : ON VISUAL 'LEVEL<N>_STROBE';
```



## CallIn events

### Purpose

Use the CallIn event to trigger the execution of a rule when a firefighter's telephone is plugged into a remote call-in jack.

**Note:** To establish two-way communications, the relay pseudo points at addresses 4106 and 4107 on the audio controller must be turned on using the On command.

### Syntax

CallIn device\_type 'object\_label' :

– or –

CallIn device\_type :

### Device types

Firephone and Telephone are the only valid devices for CallIn events.

### Object labels

The object label specifies the unique label of the device initiating the CallIn event.

### Example

Example 3-4 identifies possible syntax for a rule, in which a call from any firephone on levels 1, 3, 4, or 5 will cause the panel to:

- Turn on an associated LED at the fire alarm control panel
- Close the relay labeled PHONE\_CALL\_IN\_RELAY\_1\_41
- Close the relay labeled CALLIN\_EXISTS\_1\_41

#### Example 3-4: Syntax for Firephone callins

```
[PHONE_CALLIN]
CALLIN FIREPHONE 'LVL<N:1,3-5>_PHONE_JACK' :  ON LED 'LVL<N>_PHONE_CALLME',
                                                ON RELAY 'PHONE_CALL_IN_RELAY_1_41',
                                                ON RELAY 'CALLIN_EXISTS_1_41';
```

## Confirmation events

### Purpose

Use the Confirmation event to trigger the execution of a rule when a supervised output device is activated in response to a system command.

### Syntax

Confirmation device\_type 'object\_label' :

– or –

Confirmation device\_type :

### Device types

Valid devices for Confirmation events include:

- Audible
- AudioAmp
- CommonAlarmOutput
- CommonMonitorOutput
- CommonSupervisoryOutput
- CommonTroubleOutput
- DamperControl
- DigitalMessage
- DoorControl
- FanControl
- Firephone
- Telephone
- Visual

### Object labels

The object label specifies the unique label of the device initiating the Confirmation event.

### Example

Example 3-5 identifies possible syntax for a rule, in which:

- LVL1\_PHONE\_JACK turns on LVL1\_PHONE\_READY
- LVL3\_PHONE\_JACK turns on LVL3\_PHONE\_READY
- LVL4\_PHONE\_JACK turns on LVL4\_PHONE\_READY
- LVL5\_PHONE\_JACK turns on LVL5\_PHONE\_READY

#### Example 3-5: Syntax for confirmation of firephone

```
[PHONE_CONNECT_MADE]
CONFIRMATION FIREPHONE 'LVL<N:1,3-5>_PHONE_JACK' : LED ON LED 'LVL<N>_PHONE_READY';
```

## Define events

### Purpose

Use the Define event to determine system responses that should occur for a given action, sequence, AND group, or time control.

**Note:** By default, the 2-SDU automatically generates rules that define Drill (action 9003), AlarmSilence (action 9004), and Evacuation (action 9500) responses. Deselect the Generate Default Rules in Project Preferences to customize these actions.

### Syntax

Define device\_type 'object\_label' :

### Device types

Table 3-2 identifies valid devices for Define events.

**Table 3-2: Valid device types and devices**

Device types	Devices
Actions	Action AlarmSilence Drill GuardPatrolAlarm LampTest NetworkClassAFault Sequence StartAction SysReset
Sequences	Sequence StartSequence
AND groups	And
Time controls	TimeControl

### Object labels

The object label specifies the unique label of the device initiating the Define event. The object label specifies:

- MCM1 for actions and sequences
- AND group labels for AND groups
- Time control labels for time controls

### Examples

Example 3-6 identifies possible syntax for defining an action that will cause MCM1 to turn on the dialer relay.

#### Example 3-6: Syntax for defining dialer action

```
[DIALER]
DEFINE ACTION 'MCM1' :   ON 'DIALER_RELAY';
```

Example 3-7 identifies possible syntax for defining a guard patrol alarm, which causes MCM1 to light several LEDs.

#### Example 3-7: Syntax for defining a guard patrol alarm

```
[GUARDPATROL_ALARM_RESPONSE]
DEFINE GUARDPATROLALARM 'MCM1' :   LEDON LED 'BLDG_5_LED<L:2-7>;
```

Example 3-8 identifies possible syntax for defining a sequence that will cause the main controller module (MCM1) to:

- Delay for 5 seconds
- Turn on an exhaust fan labeled EXHFAN\_RELAY
- Delay for 10 more seconds
- Turn off SUPFAN\_RELAY

#### Example 3-8: Syntax for defining a sequence

```
[FANON_SEQ]
DEFINE SEQUENCE 'MCM1' :   DELAY 5,
                           NSFANON NSFANCONTROL 'EXHFAN_RELAY',
                           DELAY 10,
                           NSFANOFF NSFANCONTROL 'SUPFAN_RELAY';
```

Example 3-9 identifies possible syntax for defining an AND group that will activate an action labeled EVACUATION.

#### Example 3-9: Syntax for defining an AND group

```
[AND_GROUP1_RESPONSE]
DEFINE AND 'AND_GROUP1' :   ACTIVATE ACTION 'EVACUATION';
```

Example 3-10 identifies possible syntax for defining a time control that will release a door relay.

#### Example 3-10: Syntax for defining a time control

```
[TIME_CNTRL_1_RESPONSE]
DEFINE TIMECONTROL 'TC1' :   RELEASEDOOR DOORCONTROL 'DOOR_RELAY';
```

---

## Monitor events

### Purpose

Use the Monitor event to trigger the execution of a rule when any monitor device goes active while the system is in normal standby operation.

### Syntax

```
Monitor device_type 'object_label' :
```

– or –

```
Monitor device_type :
```

### Device types

Valid devices for Monitor events include:

- DamperFeedback
- DoorFeedback
- Emergency
- FanFeedback
- Monitor

### Object labels

The object label specifies the unique label of the device initiating the Monitor event.

### Example

Example 3-11 identifies possible syntax for a rule that will turn on ANNUNCIATOR\_LED\_10 if WEST\_DOOR is opened.

#### Example 3-11: Syntax for a monitored door

```
[WEST_DOOR]  
MONITOR DOORFEEDBACK 'WEST_DOOR' :   LED ON LED 'ANNUNCIATOR_LED_10';
```

---

## Supervisory events

### Purpose

Use the Supervisory event to trigger the execution of a rule when any supervisory device goes active while the system is in normal standby operation.

### Syntax

```
Supervisory device_type 'object_label' :
```

– or –

```
Supervisory device_type :
```

### Device types

Valid devices for Supervisory events include:

- AuxPowerSupply
- Gatevalve
- Power
- SprinklerSupervisory
- SupDuct
- Supervisory
- Tamper
- Temperature

### Object labels

The object label specifies the unique label of the device initiating the Supervisory event.

### Example

Example 3-12 identifies possible syntax for a rule that will light ANNUNCIATOR\_LED\_5 whenever water flows through GATEVALVE\_1.

#### Example 3-12: Syntax for a supervised gatevalve

```
[GATEVALVE_1_RESPONSE]  
SUPERVISORY GATEVALVE 'GATEVALVE_1' :    LEDON LED 'ANNUNCIATOR_LED_5';
```

## TestActive events

### Purpose

Use the TestActive event to trigger the execution of a rule when any initiating device goes active while the system is in test mode.

### Syntax

TestActive device\_type 'object\_label' :

– or –

TestActive device\_type :

### Device types

Table 3-3 identifies valid devices for TestActive events.

**Table 3-3: Valid device types and devices**

Device types	Devices
Alarm initiating devices	Duct Heat Pull Smoke SmokeVfy StageOne Waterflow
Latching supervisory devices	AuxPowerSupply Gatevalve Power SprinklerSupervisory SupDuct Supervisory, Tamper Temperature
Non-latching monitor devices	DamperFeedback DoorFeedback Emergency FanFeedback Monitor

### Object labels

The object label specifies the unique label of the device initiating the TestActive event.

**Example**

Example 3-13 identifies possible syntax for a rule that causes a pull station to light an associated LED if it goes active during test mode. For example:

- PULL\_1 will light ANNUNCIATOR\_LED\_1
- PULL\_2 will light ANNUNCIATOR\_LED\_2
- PULL\_3 will light ANNUNCIATOR\_LED\_3
- PULL\_4 will light ANNUNCIATOR\_LED\_4
- PULL\_5 will light ANNUNCIATOR\_LED\_5

**Example 3-13: Syntax for a testing pull stations**

```
[TEST_PULL_STATIONS]
TESTACTIVE PULL 'PULL_<N:1-5>' : LED ON LED 'ANNUNCIATOR_LED_<N>';
```



## TestTrouble event

### Purpose

Use the TestTrouble event to trigger the execution of a rule when any initiating device goes into trouble while the system is in test mode.

### Syntax

```
TestTrouble device_type 'object_label' :
```

– or –

```
TestTrouble device_type :
```

### Device types

Table 3-4 identifies valid devices for TestTrouble events.

**Table 3-4: Valid device types and devices**

Device types	Devices
Alarm initiating devices	Duct Heat Pull Smoke SmokeVfy StageOne Waterflow
Latching supervisory devices	AuxPowerSupply Gatevalve Power SprinklerSupervisory SupDuct Supervisory Tamper Temperature
Non-latching monitor devices	DamperFeedback DoorFeedback Emergency FanFeedback Monitor
Supervised output devices	Audible AudioAmp CommonAlarmOutput CommonMonitorOutput CommonSupervisoryOutput CommonTroubleOutput DamperControl DigitalMessage DoorControl FanControl

**Table 3-4: Valid device types and devices**

Device types	Devices
Supervised output devices <i>continued</i>	Firephone Telephone Visual

**Object labels**

The object label specifies the unique label of the device initiating the TestTrouble event.

**Example**

Example 3-14 identifies possible syntax for a rule that will light ANNUNCIATOR\_LED\_2 if FLR\_2\_SPKR goes into a trouble condition during the test mode.

**Example 3-14: Syntax for a speaker in trouble during test mode**

```
[TEST_SPEAKER_TROUBLE]
TESTTROUBLE AUDIBLE 'FLR_2_SPKR' :    LED ON LED 'ANNUNCIATOR_LED_2';
```

## Trouble events

### Purpose

Use the Trouble event to trigger the execution of a rule when any initiating device goes into trouble while the system is in normal standby operation.

### Syntax

Trouble device\_type 'object\_label' :

– or –

Trouble device\_type :

### Device types

Table 3-5 identifies valid devices for Trouble events.

**Table 3-5: Valid device types and devices**

Device types	Devices
Alarm initiating devices	Duct Heat Pull Smoke SmokeVfy StageOne Waterflow
Latching supervisory devices	AuxPowerSupply Gatevalve Power SprinklerSupervisory SupDuct Supervisory Tamper Temperature
Non-latching monitor devices	DamperFeedback DoorFeedback Emergency FanFeedback Monitor
Non-supervised output devices	NonSupervisedOutput NSDamperControl NSDoorControl NSFanControl Relay RemoteTextMessage

**Table 3-5: Valid device types and devices**

Device types	Devices
Supervised Output devices	Audible AudioAmp CommonAlarmOutput CommonMonitorOutput CommonSupervisoryOutput CommonTroubleOutput DamperControl DigitalMessage DoorControl FanControl Firephone Telephone Visual

**Object labels**

The object label specifies the unique label of the device initiating the Trouble event.

**Example**

Example 3-15 identifies possible syntax for a rule that will light ANNUNCIATOR\_LED\_12 if SPRKLR\_1\_TAMP\_2 goes into a trouble condition during normal standby operation.

**Example 3-15: Syntax for a tamper switch in a trouble condition**

```
[TAMPER_TROUBLE_RESPONSE]
TROUBLE TAMPER 'SPRKLR_1_TAMP_2' : LED ON LED 'ANNUNCIATOR_LED_12';
```

### Summary

Use this chapter to write output statements in the rules editor.

### Content

Activate command • 4.2  
Cancel command • 4.3  
Close command • 4.4  
CommonAlarmOff command • 4.5  
CommonAlarmOn command • 4.6  
CommonMonitorOff command • 4.7  
CommonMonitorOn command • 4.8  
CommonSupervisoryOff command • 4.9  
CommonSupervisoryOn command • 4.10  
CommonTroubleOff command • 4.11  
CommonTroubleOn command • 4.12  
Delay command • 4.13  
Disable command • 4.14  
Enable command • 4.16  
FanOff command • 4.18  
FanOn command • 4.19  
HoldDoor command • 4.20  
LEDOff command • 4.21  
LEDOn command • 4.22  
NSClose command • 4.23  
NSCommonAlarmOff command • 4.24  
NSCommonAlarmOn command • 4.25  
NSCommonMonitorOff command • 4.26  
NSCommonMonitorOn command • 4.27  
NSCommonSupervisoryOff command • 4.28  
NSCommonSupervisoryOn command • 4.29  
NSCommonTroubleOff command • 4.30  
NSCommonTroubleOn command • 4.31  
NSFanOff command • 4.32  
NSFanOn command • 4.33  
NSHoldDoor command • 4.34  
NSOpen command • 4.35  
NSReleaseDoor command • 4.36  
Off command • 4.37  
OffGuard command • 4.39  
On command • 4.40  
OnGuard command • 4.42  
Open command • 4.43  
ReleaseDoor command • 4.44  
Restore command • 4.45

---

## Activate command

### Description

Use this command to execute an action or a sequence.

### Syntax

```
Activate [Action | Sequence] 'rule_label';
```

– or –

```
Activate device_type 'object_label';
```

### Device types

Valid devices for the Activate command include:

- Action
- AlarmSilence
- And
- Drill
- Evacuation
- GuardPatrolAlarm
- LampTest
- NetworkClassAFault
- Sequence
- StartAction
- StartSequence
- SysReset
- TimeControl

### Object label

The object label specifies the unique label of the device responding to the Activate command.

### Example

Example 4-1 illustrates possible syntax for a rule that will activate an action labeled STROBE\_SEQ during any alarm event.

#### Example 4-1: Syntax for activating of a strobe sequence

```
[GENERAL_ALARM]  
ALARM '*' :      ACTIVATE ACTION 'STROBE_SEQ';
```

### Related information

You can find information related to this topic in:

- Restore command
- Define event

---

## Cancel command

### Description

Use this command to stop the execution of a sequence.

### Syntax

```
Cancel Sequence 'rule_label';
```

– or –

```
Cancel StartSequence 'MCM_label';
```

### Rule label

The rule label identifies the sequence that responds to the cancel command.

### MCM label

The MCM label specifies the label of the main controller module that responds to the cancel command. Use MCM1 for standalone systems.

### Example

Example 4-2 illustrates possible syntax for a rule that cancels the activation of an action labeled STROBES\_SEQ when STROBE\_SW1 is active.

#### Example 4-2: Syntax for canceling a strobe sequence

```
[CANCEL_TURN_ON_STROBES_SEQ]  
ACTIVE SWITCH 'STROBE_SW1' :   CANCEL SEQUENCE 'STROBE_SEQ';
```

### Related information

You can find information related to this topic in:

- Activate
- Restore

---

## Close command

### Description

Use the Close command to deenergize (reset) supervised control relay modules configured to operate dampers.

### Syntax

```
Close -priority device_type 'object_label';
```

– or –

```
Close -priority device_type;
```

### Priority

Priorities are optional. If you plan to use a priority for this command, see *Priorities* in *Advanced Programming features*.

### Device type

The only valid device type for the Close command is the DamperControl.

### Object label

The object label specifies the unique label of the device responding to the Close command.

**Note:** You do not have to include the device type with the object label. As a safeguard, however, we suggest that your input statements include the device type and the object label.

### Example

During a fire, it may be necessary to close dampers to isolate smoke or flames. Example 4-3 illustrates possible syntax for a rule that will close supervised dampers upon the activation of a switch labeled DAMP\_CNTRL\_SW1.

#### Example 4-3: Syntax for closing a DamperControl

```
[CLOSE_DAMPER]  
ACTIVE SWITCH 'DAMP_CNTRL_SW1' :      CLOSE DAMPERCONTROL 'DAMPER_RELAY';
```

### Related information

You can find information related to this topic in:

- NSClose command
- NSOpen command
- Open command



## CommonAlarmOff command

### Description

Use the CommonAlarmOff command to turn off (reset) a supervised output device configured to automatically activate when the system detects an active alarm point.

### Syntax

```
CommonAlarmOff -priority device_type
'object_label';
```

– or –

```
CommonAlarmOff -priority device_type;
```

### Priority

Priorities are optional. If you plan to use a priority for this command, see *Priorities* in *Advanced Programming features*.

### Device type

The only valid device type for the CommonAlarmOff command is the CommonAlarmOutput.

### Object label

The object label specifies the unique label of the device responding to the CommonAlarmOff command.

### Example

Example 4-4 illustrates possible syntax for a rule that will turn off any active CommonAlarmOutput upon the activation of a switch labeled ALRM\_SW1.

#### Example 4-4: Syntax for resetting a CommonAlarmOutput

```
[TURN_OFF_COMMON_ALARM_OUTPUTS]
ACTIVE SWITCH 'ALRM_SW1':          COMMONALARMOFF COMMONALARMOUTPUT '*';
```

### Related information

You can find information related to this topic in:

- CommonAlarmOn command
- NSCommonAlarmOff command
- NSCommonAlarmOn command

## CommonAlarmOn command

### Description

Use the CommonAlarmOn command to turn on (set) a supervised output device turned off by the CommonAlarmOff command.

### Syntax

```
CommonAlarmOn -priority device_type
'object_label';
```

– or –

```
CommonAlarmOn -priority device_type;
```

### Priority

Priorities are optional. If you plan to use a priority for this command, see *Priorities* in *Advanced Programming features*.

### Device type

The only valid device type for the CommonAlarmOn command is the CommonAlarmOutput.

### Object label

The object label specifies the unique label of the device responding to the CommonAlarmOff command.

### Example

Example 4-5 illustrates possible syntax for a rule that will turn on any CommonAlarmOutput that was turned off by a CommonAlarmOff command. CommonAlarmOn is initiated upon the activation of a switch labeled ALRM\_SW2.

#### Example 4-5: Syntax for setting a CommonAlarmOutput

```
[TURN_ON_COMMON_ALARM_OUTPUTS]
ACTIVE SWITCH 'ALRM_SW2' :      COMMONALARMON COMMONALARMOUTPUT '*';
```

### Related information

You can find information related to this topic in:

- CommonAlarmOff command
- NSCommonAlarmOff command
- NSCommonAlarmOn command

## CommonMonitorOff command

### Description

Use the CommonMonitorOff command to turn off (reset) a supervised output device configured to automatically activate when the system detects an active monitor point.

### Syntax

```
CommonMonitorOff -priority device_type
'object_label';
```

– or –

```
CommonMonitorOff -priority device_type;
```

### Priority

Priorities are optional. If you plan to use a priority for this command, see *Priorities* in *Advanced Programming features*.

### Device type

The only valid device type for the CommonMonitorOff command is the CommonMonitorOutput.

### Object label

The object label specifies the unique label of the device responding to the CommonMonitorOff command.

### Example

Example 4-6 illustrates possible syntax for a rule that will turn off any active CommonMonitorOutput upon the activation of a switch labeled MONTR\_SW1.

#### Example 4-6: Syntax for resetting a CommonMonitorOutput

```
[TURN_OFF_COMMON_MONITOR_OUTPUTS]
ACTIVE SWITCH 'MONTR_SW1' :          COMMONMONITOROFF COMMONMONITOROUTPUT '*';
```

### Related information

You can find information related to this topic in:

- CommonMonitorOn command
- NSCommonMonitorOff command
- NSCommonMonitorOn command

---

## CommonMonitorOn command

### Description

Use the CommonMonitorOn command to turn on (set) a supervised output device turned off by the CommonMonitorOff command.

### Syntax

```
CommonMonitorOn -priority device_type  
'object_label';
```

– or –

```
CommonMonitorOn -priority device_type;
```

### Priority

Priorities are optional. If you plan to use a priority for this command, see *Priorities* in *Advanced Programming features*.

### Device type

The only valid device type for the CommonMonitorOn command is the CommonMonitorOutput.

### Object label

The object label specifies the unique label of the device responding to the CommonMonitorOn command.

### Example

Example 4-7 illustrates possible syntax for a rule that will turn on any CommonMonitorOutput that was turned off by a CommonMonitorOff command. CommonMonitorOn is initiated upon the activation of a switch labeled MONTR\_SW2.

#### Example 4-7: Syntax for setting a CommonMonitorOutput

```
[TURN_ON_COMMON_MONITOR_OUTPUTS]  
ACTIVE SWITCH 'MONTR_SW2' :          COMMONMONITORON COMMONMONITOROUTPUT '*';
```

### Related information

You can find information related to this topic in:

- CommonMonitorOff command
- NSCommonMonitorOff command
- NSCommonMonitorOn command

## CommonSupervisoryOff command

### Description

Use the CommonSupervisoryOff command to turn off (reset) a supervised output device configured to automatically activate when the system detects an active supervisory point.

### Syntax

```
CommonSupervisoryOff -priority device_type
'object_label';
```

– or –

```
CommonSupervisoryOff -priority device_type;
```

### Priority

Priorities are optional. If you plan to use a priority for this command, see *Priorities* in *Advanced Programming features*.

### Device type

The only valid device type for the CommonSupervisoryOff command is the CommonSupervisoryOutput.

### Object label

The object label specifies the unique label of the device responding to the CommonSupervisoryOff command.

**Note:** You do not have to include the device type with the object label. As a safeguard, however, we suggest that your input statements include the device type and the object label.

### Example

Example 4-8 illustrates possible syntax for a rule that will turn off any active CommonSupervisoryOutput upon the activation of a switch labeled SUPV\_SW1.

#### Example 4-8: Syntax for resetting a CommonSupervisoryOutput

```
[TURNOFF_COM_SUPV_OUTPUTS]
ACTIVE SWITCH 'SUPV_SW1' :      COMMONSUPERVISORYOFF COMMONSUPERVISORYOUTPUT '*';
```

### Related information

You can find information related to this topic in:

- CommonSupervisoryOn command
- NSCommonSupervisoryOff command
- NSCommonSupervisoryOn command

## CommonSupervisoryOn command

### Description

Use the CommonSupervisoryOn command to turn on (set) a supervised output device turned off by the CommonSupervisoryOff command.

### Syntax

```
CommonSupervisoryOn -priority device_type
'object_label';
```

– or –

```
CommonSupervisoryOn -priority device_type;
```

### Priority

Priorities are optional. If you plan to use a priority for this command, see *Priorities* in *Advanced Programming features*.

### Device type

The only valid device type for the CommonSupervisoryOn command is the CommonSupervisoryOutput.

### Object label

The object label specifies the unique label of the device responding to the CommonSupervisoryOn command.

### Example

Example 4-9 illustrates possible syntax for a rule that will turn on any CommonSupervisoryOutput that was turned off by a CommonSupervisoryOff command. CommonSupervisoryOn is initiated upon the activation of a switch labeled SUPV\_SW2.

#### Example 4-9: Syntax for setting a CommonSupervisoryOutput

```
[TURNON_COM_SUPV_OUTPUTS]
ACTIVE SWITCH 'SUPV_SW2' :      COMMONSUPERVISORYON COMMONSUPERVISORYOUTPUT '*';
```

### Related information

You can find information related to this topic in:

- CommonSupervisoryOff command
- NSCommonSupervisoryOff command
- NSCommonSupervisoryOn command

## CommonTroubleOff command

### Description

Use the CommonTroubleOff command to turn off (reset) a supervised output device configured to automatically activate when the system detects an active trouble point.

### Syntax

```
CommonTroubleOff -priority device_type
'object_label';
```

– or –

```
CommonTroubleOff -priority device_type;
```

### Priority

Priorities are optional. If you plan to use a priority for this command, see *Priorities* in *Advanced Programming features*.

### Device type

The only valid device type for the CommonTroubleOff command is the CommonTroubleOutput.

### Object label

The object label specifies the unique label of the device responding to the CommonTroubleOff command.

### Example

Example 4-10 illustrates possible syntax for a rule that will turn off any active CommonTroubleOutput upon the activation of a switch labeled TRBL\_SW1.

#### Example 4-10: Syntax for resetting a CommonTroubleOutput

```
[TURNOFF_COM_TRBLE_OUTPUTS]
ACTIVE SWITCH 'TRBL_SW1' :      COMMONTROUBLEOFF COMMONTROUBLEOUTPUT '*';
```

### Related information

You can find information related to this topic in:

- CommonTroubleOn command
- NSCommonTroubleOff command
- NSCommonTroubleOn command

---

## CommonTroubleOn command

### Description

Use the CommonTroubleOn command to turn on (set) a supervised output device turned off by the CommonTroubleOff command.

### Syntax

```
CommonTroubleOn -priority device_type  
'object_label';
```

– or –

```
CommonTroubleOn -priority device_type;
```

### Priority

Priorities are optional. If you plan to use a priority for this command, see *Priorities* in *Advanced Programming features*.

### Device type

The only valid device type for the CommonTroubleOn command is the CommonTroubleOutput.

### Object label

The object label specifies the unique label of the device responding to the CommonTroubleOn command.

### Example

Example 4-11 illustrates possible syntax for a rule that will turn on any CommonTroubleOutput that was turned off by a CommonTroubleOff command. CommonTroubleOn is initiated upon the activation of a switch labeled TRBL\_SW2.

#### Example 4-11: Syntax for setting a CommonTroubleOutput

```
[TURNON_COM_TRBLE_OUTPUTS]  
ACTIVE SWITCH 'TRBL_SW2' :      COMMONTROUBLEON COMMONTROUBLEOUTPUT '*';
```

### Related information

You can find information related to this topic in:

- CommonTroubleOff command
- NSCommonTroubleOff command
- NSCommonTroubleOn command



---

## Delay command

### Description

Use the Delay command to suspend the execution of a command for a specified duration.

### Syntax

Delay delay\_value;

The delay\_value can be any duration between 5 and 4095 seconds.

### Example

Example 4-12 illustrates possible syntax for a rule that will delay the activation of visual notification appliances. The delay of visual NACs on 'MCM\_NAC2' lasts for five seconds. The delay of visual NACs on 'LCX\_NAC2' lasts for 10 seconds.

#### Example 4-12: Syntax for delaying the activation of Visuals

```
[TURN_ON_STROBES_SEQ]
DEFINE SEQUENCE 'MCM1' :
                                DELAY 5,
                                ON VISUAL 'MCM_NAC2',
                                DELAY 10,
                                ON VISUAL 'LCX_NAC2';
```

### Related information

You can find information related to this topic in the Define event topic. See *Define events* in *Input events*.

## Disable command

### Description

Use the Disable command to inhibit the automatic or manual activation of an input device, time control, AND group, action, or sequence.

### Syntax

```
Disable [Action | Sequence] 'rule_label';
```

– or –

```
Disable device_type 'object_label';
```

### Device types

Table 4-1 identifies the device types and the specific devices for which you can use the Disable output statement.

**Table 4-1: Valid device types and devices**

Device types	Devices
Actions or sequences	AlarmSilence Drill Evacuation GuardPatrolAlarm LampTest NetworkClassAFault StartAction StartSequence SysReset
Alarm initiating devices	Duct Heat Pull Smoke SmokeVfy StageOne Waterflow
Latching supervisory devices	Gatevalve Power SprinklerSupervisory SupDuct Supervisory Tamper Temperature
Non-latching monitor devices	DamperFeedback DoorFeedback Emergency FanFeedback Monitor

**Table 4-1: Valid device types and devices**

Device types	Devices
Switch devices	Switch UserDefinedSwitch
Telephone	Firephone Telephone

**Object label**

The object label specifies the unique label of the device responding to the Disable command.

**Examples**

Whenever the fire alarm control panel gives an audible indication of trouble, it is tempting to silence the buzzer and ignore the problem. Fire alarm notifications should never be ignored or silenced. Example 4-13 illustrates possible syntax for a rule that disables all manual controls.

**Example 4-13: Syntax for defining an action that will disable a control switch**

```
[CNTRL_SWITCH_DISABLE_RESPONSE]
DEFINE ACTION 'MCM_SDC_1_01' :      DISABLE SWITCH '*_CNTRL_SW*';
```

Example 4-14 illustrates possible syntax for a rule that will activate the rule in Example 4-13. The condition that will disable all manual controls is a fire alarm.

**Example 4-14: Syntax for activating the action that disables the control switch**

```
[DISABLE_MAN_CNTRLS]
ALARM '*' :      ACTIVATE ACTION 'CNTRL_SWITCH_DISABLE_RESPONSE';
```

**Related information**

You can find information related to this topic in Enable command.

## Enable command

### Description

Use the Enable command to permit the automatic or manual activation of an input device, time control, action, or sequence, inhibited by the Disable command.

### Syntax

```
Enable [Action | Sequence] 'rule_label';
```

– or –

```
Enable device_type 'object_label';
```

### Device types

Table 4-2 identifies the device types and the specific devices for which you can use the Enable output statement.

**Table 4-2: Valid device types and devices**

Device types	Devices
Actions or sequences	AlarmSilence Drill Evacuation GuardPatrolAlarm LampTest NetworkClassAFault StartAction StartSequence SysReset
Alarm initiating devices	Duct Heat Pull Smoke SmokeVfy StageOne Waterflow
Latching supervisory devices	Gatevalve Power SprinklerSupervisory SupDuct Supervisory Tamper Temperature
Non-latching monitor devices	DamperFeedback DoorFeedback Emergency FanFeedback Monitor

**Table 4-2: Valid device types and devices**

Device types	Devices
Switch input devices	Switch UserDefinedSwitch
Telephone inputs	Firephone Telephone

**Object label**

The object label specifies the unique label of the device responding to the Enable command.

**Example**

During a fire, it may become necessary to open dampers manually. Example 4-15 illustrates possible syntax for a rule that will enable the manual damper controls during a fire alarm.

**Example 4-15: Syntax for enabling a damper control switch**

```
[ENABLE_MAN_DAMPER_CNTRL]
ALARM SMOKE 'SMOKE*' :      ENABLE SWITCH 'DAMPER_CNTRL_SW*';
```

**Related information**

You can find information related to this topic in Disable command.

---

## FanOff command

### Description

Use the FanOff command to deenergize (reset) supervised control relay modules configured to operate fans.

### Syntax

```
FanOff -priority device_type 'object_label';
```

– or –

```
FanOff -priority device_type;
```

### Priority

Priorities are optional. If you plan to use a priority for this command, see *Priorities* in *Advanced Programming features*.

### Device type

The only valid device type for the FanOff command is the FanControl.

### Object label

The object label specifies the unique label of the device responding to the FanOff command.

### Example

During a fire, it may be necessary to shut off fans in various areas of a building to prevent the spread of smoke and flames. Example 4-16 illustrates possible syntax for a rule that will shut off the fans upon the activation of a switch labeled FAN\_CNTRL\_SW1.

#### Example 4-16: Syntax for resetting a FanControl

```
[FAN_SHUTDOWN]  
ACTIVE SWITCH 'FAN_CNTRL_SW1' : FANOFF FANCONTROL 'FAN_RELAY';
```

### Related information

You can find information related to this topic in:

- FanOn command
- NSFanOff command
- NSFanOn command

---

## FanOn command

### Description

Use the FanOn command to energize (set) supervised control relay modules configured to operate fans.

### Syntax

```
FanOn -priority device_type 'object_label';
```

– or –

```
FanOn -priority device_type;
```

### Priority

Priorities are optional. If you plan to use a priority for this command, see *Priorities* in *Advanced Programming features*.

### Device type

The only valid device type for the FanOn command is the FanControl.

### Object label

The object label specifies the unique label of the device responding to the Restore command.

### Example

During a fire, it may be necessary to turn fans on that will clear smoke from various parts of a building. Example 4-17 illustrates possible syntax for a rule that will turn a fan on upon the activation of a switch labeled FAN\_CNTRL\_SW2.

#### Example 4-17: Syntax for setting a FanControl

```
[FAN_POWERUP]
ACTIVE SWITCH 'FAN_CNTRL_SW2' : FANON FANCONTROL 'FAN_RELAY';
```

### Related information

You can find information related to this topic in:

- FanOff command
- NSFanOff command
- NSFanOn command

---

## HoldDoor command

### Description

Use the HoldDoor command to energize (set) supervised control relay modules controlling automatic door holders.

### Syntax

```
HoldDoor -priority device_type 'object_label';
```

– or –

```
HoldDoor -priority device_type;
```

### Priority

Priorities are optional. If you plan to use a priority for this command, see *Priorities* in *Advanced Programming features*.

### Device type

The only valid device type for the HoldDoor command is the DoorControl.

### Object label

The object label specifies the unique label of the device responding to the Restore command.

### Example

During a fire, it may become necessary to seal off sections of a building by keeping doors shut. Example 4-18 illustrates possible syntax for a rule that will keep a door shut upon the activation of a switch labeled DOOR\_CNTRL\_SW1.

#### Example 4-18: Syntax for holding a DoorControl in the closed position

```
[DOORS_CLOSED]
ACTIVE SWITCH 'DOOR_CNTRL_SW1' :                HOLDDOOR DOORCONTROL 'DOOR_RELAY';
```

### Related information

You can find information related to this topic in:

- ReleaseDoor command
- NSHoldDoor command
- NSReleaseDoor command



## LEDOff command

### Description

Use the LEDOff command to turn off (reset) an annunciator panel light emitting diode.

### Syntax

```
LEDOff -priority device_type 'object_label';
```

– or –

```
LEDOff -priority device_type;
```

### Priority

Priorities are optional. If you plan to use a priority for this command, see *Priorities* in *Advanced Programming features*.

### Device types

The only valid device types the LEDOff command are LEDs and the UserDefinedLED.

### Object label

The object label specifies the unique label of the device responding to the Restore command.

### Example

An LED may be programmed to turn on in the event of an initiated firephone call. It is also advisable to program a visual confirmation when a firefighter makes a connection at the receiving jack. Example 4-19 illustrates possible syntax for a rule that will turn off an LED when a firefighter makes a connection to the associated firephone jack.

#### Example 4-19: Syntax for confirming a firephone connection

```
[PHONE_CONNECT_MADE]
CONFIRMATION FIREPHONE 'LVL<N:3-5>_PHONEJACK' : LEDOFF LED 'LVL<N>_PHONECALL';
```

### Related information

You can find information related to this topic in LEDOn command.

---

## LEDon command

### Description

Use the LEDOn command to turn on (set) an annunciator panel light emitting diode.

### Syntax

```
LEDon -priority device_type 'object_label';
```

– or –

```
LEDon -priority device_type;
```

### Priority

Priorities are optional. If you plan to use a priority for this command, see *Priorities* in *Advanced Programming features*.

### Device types

The only valid device types for the LEDOn command are LEDs and the UserDefinedLED.

### Object label

The object label specifies the unique label of the device responding to the Restore command.

### Example

Example 4-20 illustrates possible syntax for a rule that will turn on an LED when a firefighter makes a connection to its associated firephone jack.

#### Example 4-20: Syntax for confirming a firephone connection

```
[PHONE_CONNECT_MADE]
CONFIRMATION FIREPHONE 'LVL<N:3-5>_PHONEJACK' : LEDON LED 'LVL<N>_PHONECALL';
```

### Related information

You can find information related to this topic in LEDOff command.

## NSClose command

### Description

Use the NSClose command to deenergize (reset) nonsupervised control relay modules configured to operate dampers.

### Syntax

```
NSClose -priority device_type 'object_label';
```

– or –

```
NSClose -priority device_type;
```

### Priority

Priorities are optional. If you plan to use a priority for this command, see *Priorities* in *Advanced Programming features*.

### Device type

The only valid device type for the NSClose command is the NSDamperControl.

### Object label

The object label specifies the unique label of the device responding to the Restore command.

### Example

During a fire, it may be necessary to close dampers to isolate smoke or flames. Example 4-21 illustrates possible syntax for a rule that will close nonsupervised dampers upon the activation of a switch labeled NSDAMP\_CRNTL\_SW1.

#### Example 4-21: Syntax for closing an NSDamperControl

```
[CLOSE_DAMPER]
ACTIVE SWITCH 'NSDAMP_CRNTL_SW1' :      NSCLOSE NSDAMPERCONTROL 'DAMPER_RELAY';
```

### Related information

You can find information related to this topic in:

- NSOpen command
- Open command

## NSCommonAlarmOff command

### Description

Use the NSCommonAlarmOff command to turn off (reset) a nonsupervised output device configured to activate automatically when the system detects an active alarm point.

### Syntax

```
NSCommonAlarmOff -priority device_type
'object_label';
```

– or –

```
NSCommonAlarmOff -priority device_type;
```

### Priority

Priorities are optional. If you plan to use a priority for this command, see *Priorities* in *Advanced Programming features*.

### Device type

The only valid device type for the NSCommonAlarmOff command is the NSCommonAlarmOutput.

### Object label

The object label specifies the unique label of the device responding to the Restore command.

### Example

Example 4-22 illustrates possible syntax for a rule that will turn off any active NSCommonAlarmOutput upon the activation of a switch labeled NSALRM\_SW1.

#### Example 4-22: Syntax for resetting an NSCommonAlarmOutput

```
[TURN_OFF_NSCOMMON_ALARM_OUTPUTS]
ACTIVE SWITCH 'NSALRM_SW1' :          NSCOMMONALARMOFF NSCOMMONALARMOUTPUT '*';
```

### Related information

You can find information related to this topic in:

- CommonAlarmOff command
- CommonAlarmOn command
- NSCommonAlarmOn command

## NSCommonAlarmOn command

### Description

Use the NSCommonAlarmOn command to turn on (set) a nonsupervised output device turned off by the NSCommonAlarmOff command.

### Syntax

```
NSCommonAlarmOn -priority device_type
'object_label';
```

– or –

```
NSCommonAlarmOn -priority device_type;
```

### Priority

Priorities are optional. If you plan to use a priority for this command, see *Priorities* in *Advanced Programming features*.

### Device type

The only valid device type for the NSCommonAlarmOn command is the NSCommonAlarmOutput.

### Object label

The object label specifies the unique label of the device responding to the Restore command.

### Example

Example 4-23 illustrates possible syntax for a rule that will turn on any NSCommonAlarmOutput that was turned off by a NSCommonAlarmOff command. NSCommonAlarmOn is initiated upon the activation of a switch labeled NSALRM\_SW2.

#### Example 4-23: Syntax for setting an NSCommonAlarmOutput

```
[TURN_ON_NSCOMMON_ALARM_OUTPUTS]
ACTIVE SWITCH 'NSALRM_SW2' :          NSCOMMONALARMON NSCOMMONALARMOUTPUT '*';
```

### Related information

You can find information related to this topic in:

- CommonAlarmOff command
- CommonAlarmOn command
- NSCommonAlarmOff command

---

## NSCommonMonitorOff command

### Description

Use the NSCommonMonitorOff command to turn off (reset) a nonsupervised output device configured to automatically activate when any active monitor point is detected by the system.

### Syntax

```
NSCommonMonitorOff -priority device_type  
'object_label';
```

– or –

```
NSCommonMonitorOff -priority device_type;
```

### Priority

Priorities are optional. If you plan to use a priority for this command, see *Priorities* in *Advanced Programming features*.

### Device type

The only valid device type for the NSCommonMonitorOff command is the NSCommonMonitorOutput.

### Object label

The object label specifies the unique label of the device responding to the Restore command.

### Example

Example 4-24 illustrates possible syntax for a rule that will turn off any active NSCommonMonitorOutput upon the activation of a switch labeled NSMONTR\_SW1.

#### Example 4-24: Syntax for resetting an NSCommonMonitorOutput

```
[TURNOFF_NSCOM_MONTR_OUTPUTS]  
ACTIVE SWITCH 'NSMONTR_SW1' :   NSCOMMONMONITOROFF NSCOMMONMONITOROUTPUT '*';
```

### Related information

You can find information related to this topic in:

- CommonMonitorOff command
- CommonMonitorOn command
- NSCommonMonitorOn command

## NSCommonMonitorOn command

### Description

Use the NSCommonMonitorOn command to turn on (set) a nonsupervised output device turned off by the NSCommonMonitorOff command.

### Syntax

```
NSCommonMonitorOn -priority device_type
'object_label';
```

– or –

```
NSCommonMonitorOn -priority device_type;
```

### Priority

Priorities are optional. If you plan to use a priority for this command, see *Priorities* in *Advanced Programming features*.

### Device type

The only valid device type for the NSCommonMonitorOn command is the NSCommonMonitorOutput.

### Object label

The object label specifies the unique label of the device responding to the Restore command.

### Example

Example 4-25 illustrates possible syntax for a rule that will turn on any NSCommonMonitorOutput that was turned off by a NSCommonMonitorOff command. NSCommonMonitorOn is initiated upon the activation of a switch labeled NSMONTR\_SW2.

#### Example 4-25: Syntax for setting an NSCommonMonitorOutput

```
[TURNON_NSCOM_MONTR_OUTPUTS]
ACTIVE SWITCH 'NSMONTR_SW2' : NSCOMMONMONITORON NSCOMMONMONITOROUTPUT '*';
```

### Related information

You can find information related to this topic in:

- CommonMonitorOff command
- CommonMonitorOn command
- NSCommonMonitorOff command

---

## NSCommonSupervisoryOff command

### Description

Use the NSCommonSupervisoryOff command to turn off (reset) a nonsupervised output device configured to automatically activate when any active supervisory point is detected by the system.

### Syntax

```
NSCommonSupervisoryOff -priority device_type  
'object_label';
```

– or –

```
NSCommonSupervisoryOff -priority device_type;
```

### Priority

Priorities are optional. If you plan to use a priority for this command, see *Priorities* in *Advanced Programming features*.

### Device type

The only valid device type the NSCommonSupervisoryOff command is the NSCommonSupervisoryOutput.

### Object label

The object label specifies the unique label of the device responding to the Restore command.

### Example

Example 4-26 illustrates possible syntax for a rule that will turn off any active NSCommonSupervisoryOutput upon the activation of a switch labeled NSSUPV\_SW1.

#### Example 4-26: Syntax for resetting an NSCommonSupervisoryOutput

```
[TURNOFF_NSCOMSUPVOUT]  
ACTIVE SWITCH 'NSSUPV_SW1' : NSCOMMONSUPERVISORYOFF '*';
```

**Note:** This rule will compile even if the command does not specify the device type. Commands with multiple valid device types require device-type specification.

### Related information

You can find information related to this topic in:

- CommonSupervisoryOff command
- CommonSupervisoryOn command
- NSCommonSupervisoryOn command



## NSCommonSupervisoryOn command

### Description

Use the NSCommonSupervisoryOn command to turn on (set) a nonsupervised output device turned off by the NSCommonSupervisoryOff command.

### Syntax

```
NSCommonSupervisoryOn -priority device_type
'object_label';
```

– or –

```
NSCommonSupervisoryOn -priority device_type;
```

### Priority

Priorities are optional. If you plan to use a priority for this command, see *Priorities* in *Advanced Programming features*.

### Device type

The only valid device type for the NSCommonSupervisoryOn command is the NSCommonSupervisoryOutput.

### Object label

The object label specifies the unique label of the device responding to the Restore command.

### Example

Example 4-27 illustrates possible syntax for a rule that will turn on any NSCommonSupervisoryOutput that was turned off by a NSCommonSupervisoryOff command.

NSCommonSupervisoryOn is initiated upon the activation of a switch labeled NSSUPV\_SW2.

#### Example 4-27: Syntax for setting an NSCommonSupervisoryOutput

```
[TURNON_NSCOMSUPVOUT]
ACTIVE SWITCH 'NSSUPV_SW2' : NSCOMMONSUPERVISORYON '*';
```

**Note:** This rule will compile even if the command does not specify the device type. Commands with multiple valid device types require device-type specification.

### Related information

You can find information related to this topic in:

- CommonSupervisoryOff command
- CommonSupervisoryOn command
- NSCommonSupervisoryOff command

## NSCommonTroubleOff command

### Description

Use the NSCommonTroubleOff command to turn off (reset) a nonsupervised output device configured to automatically activate when the system detects any active trouble point.

### Syntax

```
NSCommonTroubleOff -priority device_type
'object_label';
```

– or –

```
NSCommonTroubleOff -priority device_type;
```

### Priority

Priorities are optional. If you plan to use a priority for this command, see *Priorities* in *Advanced Programming features*.

### Device type

The only valid device type for the NSCommonTroubleOff command is the NSCommonTroubleOutput.

### Object label

The object label specifies the unique label of the device responding to the Restore command.

### Example

Example 4-28 illustrates possible syntax for a rule turn off any active NSCommonTroubleOutput upon the activation of a switch labeled NSTRBL\_SW1.

#### Example 4-28: Syntax for resetting an NSCommonTroubleOutput

```
[TURNOFF_NSCOMTRBLOUT]
ACTIVE SWITCH 'NSTRBL_SW1' :      NSCOMMONTROUBLEOFF NSCOMMONTROUBLEOUTPUT '*';
```

### Related information

You can find information related to this topic in:

- CommonTroubleOff command
- CommonTroubleOn command
- NSCommonTroubleOn command

## NSCommonTroubleOn command

### Description

Use the NSCommonTroubleOn command to turn on (set) a nonsupervised output device turned off by the NSCommonTroubleOff command.

### Syntax

```
NSCommonTroubleOn -priority device_type
'object_label';
```

– or –

```
NSCommonTroubleOn -priority device_type;
```

### Priority

Priorities are optional. If you plan to use a priority for this command, see *Priorities* in *Advanced Programming features*.

### Device type

The only valid device type for the NSCommonTroubleOn command is the NSCommonTroubleOutput.

### Object label

The object label specifies the unique label of the device responding to the Restore command.

### Example

Example 4-29 illustrates possible syntax for a rule turn on any NSCommonTroubleOutput that was turned off by a NSCommonTroubleOff command. NSCommonTroubleOn is initiated upon the activation of a switch labeled NSTRBL\_SW2.

#### Example 4-29: Syntax for setting an NSCommonTroubleOutput

```
[TURNON_NSCOMTRBLOUT]
ACTIVE SWITCH 'NSTRBL_SW2' : NSCOMMONTROUBLEON NSCOMMONTROUBLEOUTPUT '*';
```

### Related information

You can find information related to this topic in:

- CommonTroubleOff command
- CommonTroubleOn command
- NSCommonTroubleOff command

---

## NSFanOff command

### Description

Use the NSFanOff command to deenergize (reset) nonsupervised control relay modules configured to operate fans.

### Syntax

```
NSFanOff -priority device_type 'object_label';
```

– or –

```
NSFanOff -priority device_type;
```

### Priority

Priorities are optional. If you plan to use a priority for this command, see *Priorities* in *Advanced Programming features*.

### Device type

The only valid device type for the NSFanOff command is the NSFanControl.

### Object label

The object label specifies the unique label of the device responding to the Restore command.

### Example

During a fire, it may be necessary to shut off fans in various areas of a building to prevent the spread of smoke and flames. Example 4-30 illustrates possible syntax for a rule will shut off the nonsupervised fans upon the activation of a switch labeled NSFAN\_CRNTL\_SW1.

#### Example 4-30: Syntax for resetting an NSFanControl

```
[FAN_SHUTDOWN]
ACTIVE SWITCH 'NSFAN_CRNTL_SW1' :      NSFANOFF NSFANCONTROL 'FAN_RELAY';
```

### Related information

You can find information related to this topic in:

- FanOff command
- FanOn command
- NSFanOn command

---

## NSFanOn command

### Description

Use the NSFanOn command to energize (set) nonsupervised control relay modules configured to operate fans.

### Syntax

```
NSFanOn -priority device_type 'object_label';
```

– or –

```
NSFanOn -priority device_type;
```

### Priority

Priorities are optional. If you plan to use a priority for this command, see *Priorities* in *Advanced Programming features*.

### Device type

The only valid device type for the NSFanOn command is the NSFanControl.

### Object label

The object label specifies the unique label of the device responding to the Restore command.

### Example

During a fire, it may be necessary to turn fans on that will clear smoke from various parts of a building. Example 4-31 illustrates possible syntax for a rule that will turn a supervised fan on upon the activation of a switch labeled NSFAN\_CRNTL\_SW2.

#### Example 4-31: Syntax for turning on an NSFanControl

```
[FAN_POWERUP]
ACTIVE SWITCH 'NSFAN_CRNTL_SW2' : NSFANON NSFANCONTROL 'FAN_RELAY';
```

### Related information

You can find information related to this topic in:

- FanOff command
- FanOn command
- NSFanOff command

---

## NSHoldDoor command

### Description

Use the NSHoldDoor command to energize (set) supervised control relay modules configured to operate automatic door holders.

### Syntax

```
NSHoldDoor -priority device_type  
'object_label';
```

– or –

```
NSHoldDoor -priority device_type;
```

### Priority

Priorities are optional. If you plan to use a priority for this command, see *Priorities* in *Advanced Programming features*.

### Device type

The only valid device type for the NSHoldDoor command is the NSDoorControl.

### Object label

The object label specifies the unique label of the device responding to the Restore command.

### Example

During a fire, it may become necessary to seal off sections of a building by keeping doors shut. Example 4-32 illustrates possible syntax for a rule that will keep a door shut upon the activation of a switch labeled NSDOOR\_CRNTL\_SW1.

#### Example 4-32: Syntax for holding an NSDoorControl in the closed position

```
[DOORS_CLOSED]  
ACTIVE SWITCH 'NSDOOR_CRNTL_SW1' :   NSHOLDDOOR NSDOORCONTROL 'DOOR_RELAY';
```

### Related information

You can find information related to this topic in:

- HoldDoor command
- ReleaseDoor command
- NSReleaseDoor command

## NSOpen command

### Description

Use the NSOpen command to energize (set) nonsupervised control relay modules configured to operate dampers.

### Syntax

```
NSOpen -priority device_type 'object_label';
```

– or –

```
NSOpen -priority device_type;
```

### Priority

Priorities are optional. If you plan to use a priority for this command, see *Priorities* in *Advanced Programming features*.

### Device type

The only valid device type for the NSOpen command is the NSDamperControl.

### Object label

The object label specifies the unique label of the device responding to the Restore command.

### Example

During a fire, it may become necessary to open dampers to route smoke out of the building. Example 4-33 illustrates possible syntax for a rule that will open the nonsupervised dampers upon the activation of a switch labeled NSDAMP\_CRNTL\_SW2.

#### Example 4-33: Syntax for opening an NSDamperControl

```
[OPEN_DAMPER]
ACTIVE SWITCH 'NSDAMP_CRNTL_SW2' : NSOPEN NSDAMPERCONTROL 'DAMPER_RELAY';
```

### Related information

You can find information related to this topic in:

- Close command
- Open command
- NSClose command

## NSReleaseDoor command

### Description

Use the NSReleaseDoor command to deenergize (reset) nonsupervised control relay modules configured to operate automatic door holders.

### Syntax

```
NSReleaseDoor -priority device_type
'object_label';
```

– or –

```
NSReleaseDoor -priority device_type;
```

### Priority

Priorities are optional. If you plan to use a priority for this command, see *Priorities* in *Advanced Programming features*.

### Device type

The only valid device type for the NSReleaseDoor command is the NSDoorControl.

### Object label

The object label specifies the unique label of the device responding to the NSReleaseDoor command.

### Example

Doors throughout the building may be programmed to shut in the event of a fire. In the case of an evacuation, however, it may become necessary to open the doors. Example 4-34 illustrates possible syntax for a rule that will release an associated, nonsupervised door upon the activation of a switch labeled NSDOOR\_CRNTL\_SW2.

#### Example 4-34: Syntax for releasing an NSDoorControl

```
[DOORS_OPEN]
ACTIVE SWITCH 'NSDOOR_CRNTL_SW2' :    NSRELEASEDOOR NSDOORCONTROL 'DOOR_RELAY';
```

### Related information

You can find information related to this topic in:

- HoldDoor command
- ReleaseDoor command
- NSHoldDoor command



---

## Off command

### Description

Use the Off command to deactivate a point in the system.

### Syntax

```
Off -priority device_type 'object_label';
```

– or –

```
Off -priority device_type;
```

### Priority

Priorities are optional. If you plan to use a priority for this command, see *Priorities* in *Advanced Programming features*.

### Device types

Valid devices for the Off command include:

- Audible
- CommonAlarmOutput
- CommonMonitorOutput
- CommonSupervisoryOutput
- CommonTroubleOutput
- DamperControl
- DoorControl
- FanControl
- Firephone
- LED
- NSDamperControl
- NSDoorControl
- NSFanControl
- NonSupervisedOutput
- Relay
- RemoteTextMessage
- Telephone
- UserDefinedLED
- Visual

### Object label

The object label specifies the unique label of the device responding to the Off command.

### Example

After the evacuation of a building and the containment of a fire, it will be necessary to investigate the cause of the fire alarm. During the investigation, firefighters will want the horns to be silenced. Example 4-35 illustrates possible syntax for a rule that will:

## Output commands

- Turn off LVL1\_HORN when HORN\_SW1 is pressed
- Turn off LVL2\_HORN when HORN\_SW2 is pressed
- Turn off LVL3\_HORN when HORN\_SW3 is pressed
- Turn off LVL4\_HORN when HORN\_SW4 is pressed
- Turn off LVL5\_HORN when HORN\_SW5 is pressed

### Example 4-35: Syntax for resetting Audible devices

```
[DEACTIVATE_AUDIBLES]  
ACTIVE SWITCH 'HORN_SW<N:1-5>' :          OFF AUDIBLE 'LVL<N>_HORN' ;
```

### Related information

You can find information related to this topic in On command.

---

## OffGuard command

### Description

Use the OffGuard command to deactivate a guard patrol route.

### Syntax

```
OffGuard device_type 'group_label' route_id;
```

### Device type

The device type identifies the class of the device named by the object label. The only device type for which you can use the OffGuard command is the GuardPatrol.

### Group label

The group label in the Object Configuration tables specifies which guard patrol group responds to the command.

### Route ID

The route ID specifies the route sequence number of the targeted guard patrol group.

### Example

Guard patrols require personnel to check in at designated points in a timed sequence. Any deviations in time or sequence will set off a programmed GuardPatrolAlarm. In some cases, it may be necessary to deactivate a guard patrol route. Example 4-36 illustrates possible syntax for a rule that will turn off a guard patrol route upon the activation of a switch labeled GUARD\_PRTL\_SW1.

**Note:** The guard patrol group and the designated route are defined separately. See the Guard Patrol Group tab in the Object Configuration.

#### Example 4-36: Syntax for turning off a guard patrol route

```
[DEACTIVATE_GPG1_ROUTE_1]  
ACTIVE SWITCH 'GUARD_PRTL_SW1' : OFFGUARD GUARDPATROL 'GUARD_PATROL_GROUP1' 1;
```

### Related information

You can find information related to this topic in:

- OnGuard command
- Define event (GuardPatrolAlarm)

---

## On command

### Description

Use the On command to activate a point in the system.

### Syntax

```
On -priority device_type 'object_label';
```

– or –

```
On -priority device_type;
```

### Priority

Priorities are optional. If you plan to use a priority for this command, see *Priorities* in *Advanced Programming features*.

### Device types

Valid devices for the On command include:

- Audible
- CommonAlarmOutput
- CommonMonitorOutput
- CommonSupervisoryOutput
- CommonTroubleOutput
- DamperControl
- DoorControl
- FanControl
- Firephone
- LED
- NSDamperControl
- NSDoorControl
- NSFanControl
- NonSupervisedOutput
- Relay
- RemoteTextMessage
- Telephone
- UserDefinedLED
- Visual

### Object label

The object label specifies the unique label of the device responding to the On command.

### Example

During a fire, it may be necessary to evacuate floors selectively. For example, if the fire is on the fourth floor it will be necessary to evacuate floors 3, 4, and 5 immediately. Occupants on floors 1 and 2 should remain where they are to allow occupants of the

fire floors to evacuate. Example 4-37 illustrates possible syntax for a rule that will allow a firefighter to:

- Sound LVL1\_HORN by pressing HORN\_SW6
- Sound LVL2\_HORN by pressing HORN\_SW7
- Sound LVL3\_HORN by pressing HORN\_SW8
- Sound LVL4\_HORN by pressing HORN\_SW9
- Sound LVL5\_HORN by pressing HORN\_SW10

**Example 4-37: Syntax for turning on audible devices**

```
[ACTIVATE_AUDIBLES]  
ACTIVE SWITCH 'HORN_SW<N:6-10>' :      ON AUDIBLE 'LVL<N-5>_HORN';
```

**Related information**

You can find information related to this topic in Off command.

---

## OnGuard command

### Description

Use the OnGuard command to activate a guard patrol route deactivated by the OffGuard command.

### Syntax

```
OnGuard device_type 'group_label' route_id;
```

### Device type

The device type identifies the class of the device named by the object label. The only device type for which you can use the OnGuard command is the GuardPatrol.

### Group label

The group label in the Object Configuration tables specifies which guard patrol group responds to the command.

### Route ID

The route ID specifies the route sequence number of the targeted guard patrol group.

### Example

Guard patrols require personnel to check in at designated points in a timed sequence. Any deviations in time or sequence will set off a programmed GuardPatrolAlarm. Example 4-38 illustrates possible syntax for a rule that will activate a guard patrol upon the activation of a switch labeled GUARD\_PRTL\_SW2.

**Note:** The guard patrol group and the designated route are defined separately. See the Guard Patrol Group tab in the Object Configuration.

#### Example 4-38: Syntax for turning on a guard patrol route

```
[ACTIVATE_GPG1_ROUTE_1]  
ACTIVE SWITCH 'GUARD_PRTL_SW2' :  ONGUARD GUARDPATROL 'GUARD_PATROL_GROUP1' 1;
```

### Related information

You can find information related to this topic in:

- OffGuard command
- Define event (GuardPatrolAlarm)

---

## Open command

### Description

Use the Open command to energize (set) supervised control relay modules configured to operate dampers.

### Syntax

```
Open -priority device_type 'object_label';
```

– or –

```
Open -priority device_type;
```

### Priority

Priorities are optional. If you plan to use a priority for this command, see *Priorities* in *Advanced Programming features*.

### Device type

The only valid device type for the Open command is the DamperControl.

### Object label

The object label specifies the unique label of the device responding to the Open command.

### Example

During a fire, it may become necessary to open dampers to route smoke out of the building. Example 4-39 illustrates possible syntax for a rule that will open the dampers upon the activation of a switch labeled DAMP\_CNTRL\_SW2.

#### Example 4-39: Syntax for opening a DamperControl

```
[OPEN_DAMPER]
ACTIVE SWITCH 'DAMP_CNTRL_SW2' :   OPEN DAMPERCONTROL 'DAMPER_RELAY';
```

### Related information

You can find information related to this topic in:

- NSClose command
- NSOpen command
- Close command

---

## ReleaseDoor command

### Description

Use the ReleaseDoor command to deenergize (reset) supervised control relay modules configured to operate automatic door holders.

### Syntax

```
ReleaseDoor -priority device_type  
'object_label';
```

– or –

```
ReleaseDoor -priority device_type;
```

### Priority

Priorities are optional. If you plan to use a priority for this command, see *Priorities* in *Advanced Programming features*.

### Device type

The only valid device type for the ReleaseDoor command is the DoorControl.

### Object label

The object label specifies the unique label of the device responding to the ReleaseDoor command.

### Example

Doors throughout may be programmed to shut in the event of a fire. In the case of an evacuation, however, it may become necessary to open the doors. Example 4-40 illustrates possible syntax for a rule that will release an associated door upon the activation of a switch labeled DOOR\_CNTRL\_SW2.

#### Example 4-40: Syntax for releasing a DoorControl

```
[DOORS_OPEN]  
ACTIVE SWITCH 'DOOR_CNTRL_SW2' :   RELEASEDOOR DOORCONTROL 'DOOR_RELAY' ;
```

### Related information

You can find information related to this topic in:

- HoldDoor command
- NSHoldDoor command
- NSReleaseDoor command



## Restore command

### Description

Use the Restore command to return a time control, And group, action, or drill to its previous state before being activated by the Activate command.

### Syntax

```
Restore Action 'rule_label';
```

– or –

```
Restore device_type 'object_label';
```

### Device types

Valid devices for the Restore command include:

- Action
- AlarmSilence
- And
- Drill
- Evacuation
- GuardPatrolAlarm
- StartAction
- SysReset
- TimeControl

### Object label

The object label specifies the unique label of the device responding to the Restore command.

### Example

Example 4-41 illustrates possible syntax for a rule that will restore the sequence labeled TURN\_ON\_STROBES\_SEQ upon the activation of the switch labeled STROBE\_SW2.

#### Example 4-41: Syntax for restoring an action

```
[RESTORE_TURNON_STROBES_SEQ]
ACTIVE SWITCH 'STROBE_SW2' :   RESTORE ACTION 'TURN_ON_STROBES_SEQ';
```

### Related information

You can find information related to this topic in:

- Activate command
- Cancel command



### Summary

The Systems Definition Utility (SDU) supports numerous types of devices. The SDU also has a wide array of events and commands. Use the tables in the Quick reference to speed up your search for the appropriate devices to use with events and commands.

### Content

Event specification summaries • A.2

Events listed by device • A.6

Command specification summaries • A.10

Commands listed by device • A.17

## Event specification summaries

While the system is in standby mode, the panel controller waits for inputs from the system components or from an operator. These inputs are classified as events and are summarized in Table A-1. The event\_type parameter is used in a rule to specify the type of input required for the rule to be executed.

For example, the rule below was created to capture an elevator in the event a smoke detector goes into alarm. The input event, in this case, is ALARM.

```
[ElevatorCapture]
    ALARM SMOKE 'BLD2_ELOBBY*':
    ON RELAY 'ELEVRELAY';
```

As the rule above is written, any detector with “BLD2\_ELOBBY” as the first 11 characters of its label will activate the elevator capture relay (labeled ELEVRELAY) when the detector goes into alarm.

**Table A-1: Event specification summaries**

For this input specification...	Use this event type...	With this device type...
Alarm initiating device goes into alarm indicating the presence of a fire	Active, Alarm	Alarm Duct Heat Pull Smoke SmokeVfy StageOne Waterflow
Latching or non-latching supervisory device goes active indicating a condition that would prevent normal system operation	Active	AuxPowerSupply Gatevalve Power SprinklerSupervisory SupDuct Supervisory Tamper Temperature
Non-latching monitor device goes active	Active	DamperFeedback DoorFeedback Emergency FanFeedback Monitor
Local or remote panel switch is pressed	Active	Switch UserDefinedSwitch
Firefighter telephone is plugged into a call-in jack	CallIn	Firephone Telephone

**Table A-1: Event specification summaries**

<b>For this input specification...</b>	<b>Use this event type...</b>	<b>With this device type...</b>
Supervised output device activated in response to a system command	Confirmation	Audible AudioAmp CommonAlarmOutput CommonMonitorOutput CommonSupervisoryOutput CommonTroubleOutput DamperControl DigitalMessage DoorControl FanControl Firephone Telephone Visual
Alarm initiating device goes into alarm while in system test mode	TestActive	Alarm Duct Heat Pull Smoke SmokeVfy StageOne Waterflow
Latching supervisory device goes active while in system test mode	TestActive	Gatevalve Power SprinklerSupervisory SupDuct Supervisory Tamper Temperature
Non-latching monitor device goes active while in system test mode	TestActive	DamperFeedback DoorFeedback Emergency FanFeedback Monitor
Alarm initiating device goes into trouble while in system test mode	TestTrouble	Alarm Duct Heat Pull Smoke SmokeVfy StageOne Waterflow

**Table A-1: Event specification summaries**

<b>For this input specification...</b>	<b>Use this event type...</b>	<b>With this device type...</b>
Latching supervisory device goes into trouble while in system test mode	TestTrouble	Gatevalve Power SprinklerSupervisory SupDuct Supervisory Tamper Temperature
Non-latching monitor device goes into trouble while in system test mode	TestTrouble	DamperFeedback DoorFeedback Emergency FanFeedback Monitor
Supervised output device goes into trouble while in system test mode	TestTrouble	Audible AudioAmp CommonAlarmOutput CommonMonitorOutput CommonSupervisoryOutput CommonTroubleOutput DamperControl DoorControl FanControl Firephone Telephone Visual
Alarm initiating device goes into trouble	Trouble	Alarm Duct Heat Pull Smoke SmokeVfy StageOne Waterflow
Latching supervisory device goes into trouble	Trouble	Gatevalve Power SprinklerSupervisory SupDuct Supervisory Tamper Temperature
Non-latching monitor device goes into trouble	Trouble	DamperFeedback DoorFeedback Emergency FanFeedback Monitor

**Table A-1: Event specification summaries**

<b>For this input specification...</b>	<b>Use this event type...</b>	<b>With this device type...</b>
Non-supervised output device goes into trouble	Trouble	NonSupervisedOutput NSDamperControl NSDoorControl NSFanControl Relay RemoteTextMessage
Supervised output device goes into trouble	Trouble	Audible AudioAmp CommonAlarmOutput CommonMonitorOutput CommonSupervisoryOutput CommonTroubleOutput DamperControl DoorControl FanControl Firephone Telephone Visual

## Events listed by device

Table A-2 provides a quick reference of the valid events for each device type.

**Table A-2: Valid events by device**

Device type	Event(s)
Action (user-programmed)	Define
AlarmSilence (Action 9004)	Define
AND	Define
Audible	Confirmation TestTrouble Trouble
AudioAmp	Confirmation TestTrouble Trouble
AuxPowerSupply	Active
CommonAlarmOutput	Confirmation TestTrouble Trouble
CommonMonitorOutput	Confirmation TestTrouble Trouble
CommonSupervisoryOutput	Confirmation TestTrouble Trouble
CommonTroubleOutput	Confirmation TestTrouble Trouble
DamperControl	Confirmation TestTrouble Trouble
DamperFeedback	Active TestActive TestTrouble Trouble
DoorControl	Confirmation TestTrouble Trouble
DoorFeedback	Active TestActive TestTrouble Trouble
DigitalMessage	Confirmation



**Table A-2: Valid events by device**

Device type	Event(s)
Drill (Action 9003)	Define
Duct	Active Alarm TestActive TestTrouble Trouble
Emergency	Active TestActive TestTrouble Trouble
FanControl	Confirmation TestTrouble Trouble
FanFeedback	Active TestActive TestTrouble Trouble
Firephone	CallIn Confirmation TestTrouble Trouble
Gatevalve	Active TestActive TestTrouble Trouble
GuardPatrol (Action 9000)	Define
GuardPatrolAlarm	Define
Heat	Active Alarm TestActive TestTrouble Trouble
LampTest	Define
Monitor	Active TestActive TestTrouble Trouble
NetworkClassAFault	Define
NonSupervisedOutput	Trouble
NSDamperControl	Trouble
NSDoorControl	Trouble

**Table A-2: Valid events by device**

Device type	Event(s)
NSFanControl	Trouble
Power	Active TestActive TestTrouble Trouble
Pull	Active Alarm TestActive TestTrouble Trouble
Relay	Trouble
RemoteTextMessage	Trouble
Sequence	Define
Smoke	Active Alarm TestActive TestTrouble Trouble
SmokeVfy	Active Alarm TestActive TestTrouble Trouble
SprinklerSupervisory	Active TestActive TestTrouble Trouble
StageOne	Active Alarm TestActive TestTrouble Trouble
StartAction	Define
StartSequence	Define
SupDuct	Active TestActive TestTrouble Trouble
Supervisory	Active TestActive TestTrouble Trouble

**Table A-2: Valid events by device**

Device type	Event(s)
Switch <i>See also</i> UserDefinedSwitch	Active
SysReset	Define
Tamper	Active TestActive TestTrouble Trouble
Telephone	CallIn Confirmation TestTrouble Trouble
Temperature	Active TestActive TestTrouble Trouble
TimeControl	Define
UserDefinedSwitch	Active
Visual	Confirmation TestTrouble Trouble
Waterflow	Active Alarm TestActive TestTrouble Trouble

---

## Command specification summaries

The output command parameter specifies the required final state of the targeted device. Output commands are summarized in Table A-3.

In our example, it has been assumed that the elevator capture relay was to be turned ON when the smoke detectors went into alarm. When writing a rule, the output command must be specified.

The rule used in the elevator capture example includes the “ON” output state:

```
[ElevatorCapture]
    ALARM SMOKE 'BLD2_ELOBBY*':
    ON RELAY 'ELEVRELAY';
```

As the rule above is now written, any smoke detector with the label modifier “BLD2\_ELOBBY” at the beginning of its label will turn ON the elevator capture relay (labeled ELEVRELAY) when it goes into alarm.

Multiple output commands may be placed on the right side of a rule using a comma (,) as a divider. In our elevator capture example, the requirements have changed to require the activation of an elevator capture strobe in addition to activating the elevator capture relay.

Rewriting the rule used in the elevator capture example to include two output commands:

```
[ElevatorCapture]
    ALARM SMOKE 'BLD2_ELOBBY*':
    ON RELAY 'ELEVRELAY',
    ON RELAY 'ELEVSTROBE';
```

**Note:** Up to 32 commands may be used in a single rule.

**Table A-3: Command Summary**

<b>For this output specification...</b>	<b>Use this command...</b>	<b>With this device type...</b>
Start the execution of an action or sequence	Activate	Action AlarmSilence And Drill Evacuation GuardPatrolAlarm LampTest NetworkClassAFault Sequence StartAction StartSequence SysReset TimeControl
Stop the execution of a sequence	Cancel	Sequence StartSequence
Turn off a supervised common alarm output device	CommonAlarmOff	CommonAlarmOutput
Turn on a supervised common alarm output device	CommonAlarmOn	CommonAlarmOutput
Turn off a supervised common monitor output device	CommonMonitorOff	CommonMonitorOutput
Turn on a supervised common monitor output device	CommonMonitorOn	CommonMonitorOutput
Turn off a supervised common supervisory output device	CommonSupervisoryOff	CommonSupervisoryOutput
Turn on a supervised common supervisory output device	CommonSupervisoryOn	CommonSupervisoryOutput
Turn off a common trouble output device	CommonTroubleOff	CommonTroubleOutput
Turn on a common trouble output device	CommonTroubleOn	CommonTroubleOutput
Deenergize a damper control device	Close	DamperControl
Energize a damper control device	Open	DamperControl
Delay the execution of a command in a rule	Delay	none

**Table A-3: Command Summary**

<b>For this output specification...</b>	<b>Use this command...</b>	<b>With this device type...</b>
Inhibit the automatic or manual control of a device	Disable	Action Alarm AlarmSilence And DamperFeedback DoorFeedback Drill Duct Emergency Evacuation FanFeedback Firephone Gatevalve GuardPatrolAlarm Heat LampTest Monitor NetworkClassAFault Power Pull Sequence Smoke SmokeVfy SprinklerSupervisory StageOne StartAction StartSequence SupDuct Supervisory Switch SysReset Tamper Telephone Temperature TimeControl UserDefinedSwitch Waterflow

**Table A-3: Command Summary**

<b>For this output specification...</b>	<b>Use this command...</b>	<b>With this device type...</b>
Allow the automatic or manual control of a device	Enable	Action Alarm AlarmSilence And DamperFeedback DoorFeedback Drill Duct Emergency Evacuation FanFeedback Firephone Gatevalve GuardPatrolAlarm Heat LampTest Monitor NetworkClassAFault Power Pull Sequence Smoke SmokeVfy SprinklerSupervisory StageOne StartAction StartSequence SupDuct Supervisory Switch SysReset Tamper Telephone Temperature TimeControl UserDefinedSwitch Waterflow
Deenergize a supervised fan control device	FanOff	FanControl
Energize a supervised fan control device	FanOn	FanControl
Energize a supervised door control device	HoldDoor	DoorControl
Denenergize a supervised door control device	ReleaseDoor	DoorControl
Turn LED off	LEDOff	LED UserDefinedLED

**Table A-3: Command Summary**

<b>For this output specification...</b>	<b>Use this command...</b>	<b>With this device type...</b>
Turn LED on	LEDon	LED UserDefinedLED
Energize a non-supervised damper control device	NSClose	NSDamperControl
Deenergize a non-supervised damper control device	NSOpen	NSDamperControl
Turn off a non-supervised common alarm output device	NSCommonAlarmOff	NSCommonAlarmOutput
Turn on a non-supervised common alarm output device	NSCommonAlarmOn	NSCommonAlarmOutput
Turn off a non-supervised common monitor output device	NSCommonMonitorOff	NSCommonMonitorOutput
Turn on a non-supervised common monitor output device	NSCommonMonitorOn	NSCommonMonitorOutput
Turn off a non-supervised common supervisory output device	NSCommonSupervisoryOff	NSCommonSupervisoryOutput
Turn on a non-supervised common supervisory output device	NSCommonSupervisoryOn	NSCommonSupervisoryOutput
Turn off a non-supervised common trouble output device	NSCommonTroubleOff	NSCommonTroubleOutput
Turn on a non-supervised common trouble output device	NSCommonTroubleOn	NSCommonTroubleOutput
Deenergize a non-supervised fan control device	NSFanOff	NSFanControl
Energize a non-supervised fan control device	NSFanOn	NSFanControl
Energize a non-supervised door control device	NSHoldDoor	NSDoorControl
Deenergize a non-supervised door control device	NSReleaseDoor	NSDoorControl
Deactivate a guard patrol	OffGuard	GuardPatrol
Activate a guard patrol	OnGuard	GuardPatrol



**Table A-3: Command Summary**

<b>For this output specification...</b>	<b>Use this command...</b>	<b>With this device type...</b>
Deactivate an output device	Off	Audible CommonAlarmOutput CommonMonitorOutput CommonSupervisoryOutput CommonTroubleOutput DamperControl DoorControl FanControl Firephone LED NSDamperControl NSDoorControl NSFanControl NonSupervisedOutput Relay RemoteTextMessage Telephone UserDefinedLED Visual
Activate a device	On	Audible CommonAlarmOutput CommonMonitorOutput CommonSupervisoryOutput CommonTroubleOutput DamperControl DoorControl FanControl Firephone LED NSDamperControl NSDoorControl NSFanControl NonSupervisedOutput Relay RemoteTextMessage Telephone UserDefinedLED Visual

**Table A-3: Command Summary**

<b>For this output specification...</b>	<b>Use this command...</b>	<b>With this device type...</b>
Returns device to state before changed by the execution of an action of sequence	Restore	Action AlarmSilence And Drill Evacuation GuardPatrolAlarm LampTest NetworkClassAFault StartAction SysReset TimeControl

## Commands listed by device

Table A-4 provides a quick reference of the valid commands for each device type.

**Table A-4: Valid commands by device type**

Device type	Command(s)
Action (user-programmed)	Activate Disable Enable Restore
AlarmSilence (Action 9004)	Activate Disable Enable Restore
AND	Activate Disable Enable Restore
Audible	Off On
CommonAlarmOutput	CommonAlarmOff CommonAlarmOn Off On
CommonMonitorOutput	CommonMonitorOff CommonMonitorOn Off On
CommonSupervisoryOutput	CommonSupervisoryOff CommonSupervisoryOn Off On
CommonTroubleOutput	CommonTroubleOff CommonTroubleOn Off On
DamperControl	Close Open Off On
DamperFeedback	Disable Enable

**Table A-4: Valid commands by device type**

<b>Device type</b>	<b>Command(s)</b>
DoorControl	HoldDoor ReleaseDoor Off On
DoorFeedback	Disable Enable
Drill (Action 9003)	Activate Disable Enable Restore
Duct	Disable Enable
Evacuation (Action 9500)	Activate Disable Enable Restore
Emergency	Disable Enable
FanControl	FanOff FanOn Off On
FanFeedback	Disable Enable
Firephone	Disable Enable Off On
Gatevalve	Disable Enable
GuardPatrol	OffGuard OnGuard
GuardPatrolAlarm	Activate Disable Enable Restore
LampTest	Activate Disable Enable Restore

**Table A-4: Valid commands by device type**

Device type	Command(s)
LED <i>See also</i> UserDefinedLED	LEDOff LEDOn Off On
Monitor	Disable Enable
NetworkClassAFault	Activate Disable Enable Restore
NonSupervisedOutput	Off On
NSCommonAlarmOutput	NSCommonAlarmOff NSCommonAlarmOn Off On
NSCommonMonitorOutput	NSCommonMonitorOff NSCommonMonitorOn Off On
NSCommonSupervisoryOutput	NSCommonSupervisoryOff NSCommonSupervisoryOn Off On
NSCommonTroubleOutput	NSCommonTroubleOff NSCommonTroubleOn Off On
NSDamperControl	NSClose NSOpen Off On
NSDoorControl	NSHoldDoor NSReleaseDoor Off On
NSFanControl	NSFanOff NSFanOn Off On
Power	Disable Enable

**Table A-4: Valid commands by device type**

Device type	Command(s)
Pull	Disable Enable
Relay	Off On
RemoteTextMessage	Off On
Sequence	Activate Cancel Disable Enable Restore
Smoke	Disable Enable
SmokeVfy	Disable Enable
SprinklerSupervisory	Disable Enable
StageOne	Disable Enable
StartAction	Activate Disable Enable Restore
StartSequence	Activate Cancel Disable Enable
SupDuct	Disable Enable
Supervisory	Disable Enable
Switch <i>See also</i> UserDefinedSwitch	Disable Enable
SysReset	Activate Disable Enable Restore
Tamper	Disable Enable

**Table A-4: Valid commands by device type**

<b>Device type</b>	<b>Command(s)</b>
Telephone	Disable Enable Off On
Temperature	Disable Enable
TimeControl	Activate Disable Enable Restore
UserDefinedLED	LEDOff LEDOn Off On
UserDefinedSwitch	Disable Enable
Visual	Off On
Waterflow	Disable Enable





Accessory database	A tab on the Communication Functions dialog box that accesses download prompts for LSRAs and SMDNs. See Communications in the Tools menu.
Actual data	Device data read from the data circuit and stored in the controller during the mapping process.
Address	An electronic number assigned by the SDC controller to locate and poll a device.
Alarm verification	The shortest time span, during business hours, that a device has to report an alarm condition before the system verifies it (05 to 55 seconds).
Alternate alarm verification	The shortest time span, outside of normal business hours, that a device has to report an alarm condition before the system verifies it (05 to 55 seconds).
AND statement	A logic function, which requires two or more conditions to be met before it initiates a response.
AND group	A labeled group consisting of two or more devices selected from the object configuration table to form an AND statement.
Audible	An assignment made in the SDC Configuration table for a supervised module output wired to a notification appliance circuit that building occupants can hear.
Audible notification appliances	Indicating appliances designed to produce a signal that building occupants can hear. Examples include bells, horns, chimes, electronic horns, or buzzers.
AudioAmp	An assignment made in made in SDC Configuration table for a supervised output relay on a 30 or 50 Watt amplifier.
AuxPowerSupply	An assignment made in SDC Configuration table for a supervised power supply that supports appliances not tied to the primary power supply.
Barcode	Vertical bars of varying width that identify the serial numbers of Signature series components.
Bell code group	A label that identifies devices selected from the object configuration table and grouped as members of a set that activates a coded alarm signal.
CommonAlarmOff	Command used to turn off a supervised notification signal output circuit that a panel automatically activates when an Alarm event occurs.
CommonAlarmOn	Command used to turn on a supervised notification signal output circuit that a panel automatically activates when an Alarm event occurs.

## Glossary

CommonAlarmOutput	Classification used for a supervised notification signal output circuit that a panel automatically activates when an Alarm event occurs.
CommonMonitorOff	Command used to turn off a supervised notification signal output circuit that a panel automatically activates when a Monitor event occurs.
CommonMonitorOn	Command used to turn on a supervised notification signal output circuit that a panel automatically activates when a Monitor event occurs.
CommonMonitorOutput	Classification used for a supervised notification signal output circuit that a panel automatically activates when a Monitor event occurs.
CommonSupervisoryOff	Command used to turn off a supervised notification signal output circuit that a panel automatically activates when a Supervisory event occurs.
CommonSupervisoryOn	Command used to turn on a supervised notification signal output circuit that a panel automatically activates when a Supervisory event occurs.
CommonSupervisoryOutput	Classification used for a supervised notification signal output circuit that a panel automatically activates when a Supervisory event occurs.
CommonTroubleOff	Command used to turn off a supervised notification signal output circuit that a panel automatically activates when a Trouble event occurs.
CommonTroubleOn	Command used to turn on a supervised notification signal output circuit that a panel automatically activates when a Trouble event occurs.
CommonTroubleOutput	Classification used for a supervised notification signal output circuit that a panel automatically activates when a Trouble event occurs.
DamperControl	A module assignment made in the SDC Configuration for a supervised output relay that automatically turns on when a damper's status goes active.
DamperFeedback	A module assignment made in the SDC Configuration for a monitor that reports damper status.
Deenergize	To place a relay in its nonoperating position where the normally closed contact is closed and the normally open contact is open.
DigitalMessage	A module assignment made in the SDC Configuration for a supervised output relay that automatically turns on when a digital message module goes active.
Digital message module	A voice-quality, audio message, record and playback module that can store one or two 30-second messages.
Dialer	A digital alarm communicator transmitter or receiver that automatically calls up the central monitoring station in the event of a fire alarm.

Dialer group	A logic group, which consists of two or more zones associated with a dialer.
DoorControl	A module assignment made in the SDC configuration for a supervised output relay that automatically turns on when a door's status goes active.
DoorFeedback	A module assignment made in the SDC Configuration for a monitor that reports door status.
Duct	A module assignment made in the SDC Configuration for an alarm input that detects a fire by sampling air from piping or tubing for products of combustion
Emergency	An assignment made in the SDC Configuration for a module that monitors an emergency input used in distress notification applications.
Energize	To place a relay in its operating position where the normally-closed contact is open and the normally-open contact is closed.
EOL (end-of-line)	Any device or resistor at that is last on a data line.
Expected data	Device data programmed in the database through Systems Definition Utility and downloaded into the panel controller afterwards.
Fan control	A module assignment made in the SDC Configuration for a supervised output relay that automatically turns on when a fan goes active.
Fan feedback	A module assignment made in the SDC Configuration for a monitor that reports fan status.
Firephone	A module assignment made in the SDC Configuration for a supervised firephone circuit.
Gatevalve	A module assignment made in the SDC configuration for a sprinkler supervisory input that reports valve status.
General	A setting that causes time controls to take place every day.
Guard patrol	A series of locations, from which an active security patrol initiates devices to indicate that they are following a designated route at the proper intervals.
Heat	A module assignment made in the SDC Configuration for an alarm input that detects a fire by its thermal properties.
Holiday	A setting that causes time controls to take place only during a specified holiday.
IDC (initiating device circuit)	A supervised input circuit connected to alarm initiating devices like smoke detectors, manual pull stations, and waterflow switches.
Label	Descriptive text assigned by the system designer to an object in the database.
Latching	Any device that remains locked on until it is reset.
LED (light emitting diode)	A device that lights up to indicate activity on its associated device or function.

## Glossary

Mapping	The process the data circuit controller uses to determine the electrical relative positions between devices on the data circuit.
Monitor	A module assignment made in the SDC Configuration for a monitor input that activates system common monitor functions.
NAC (Notification Appliance Circuit)	A circuit directly connected to notification appliances. <i>See also</i> Notification appliance.
Node	A single control panel that is part of a network.
Non-latching	A device that will activate and reset to its normal position according to its own input condition.
NonSupervisedOutput	A module assignment made in the SDC Configuration for relays that do not monitor the integrity of the circuit they are controlling.
Notification appliance	Any alarm device that provides a visual, audible, or tactile (touchable) warnings of fire alarm conditions.
NSCommonAlarmOff	Command used to turn off a nonsupervised notification signal output circuit that a panel automatically activates when an Alarm event occurs.
NSCommonAlarmOn	Command used to turn on a nonsupervised notification signal output circuit that a panel automatically activates when an Alarm event occurs.
NSCommonAlarmOutput	Classification used for a nonsupervised notification signal output circuit that a panel automatically activates when an Alarm event occurs.
NSCommonMonitorOff	Command used to turn off a nonsupervised notification signal output circuit that a panel automatically activates when a Monitor event occurs.
NSCommonMonitorOn	Command used to turn on a nonsupervised notification signal output circuit that a panel automatically activates when a Monitor event occurs.
NSCommonMonitorOutput	Classification used for a nonsupervised notification signal output circuit that a panel automatically activates when a Monitor event occurs.
NSCommonSupervisoryOff	Command used to turn off a nonsupervised notification signal output circuit that a panel automatically activates when a Supervisory event occurs.
NSCommonSupervisoryOn	Command used to turn on a nonsupervised notification signal output circuit that a panel automatically activates when a Supervisory event occurs.
NSCommonSupervisoryOutput	Classification used for a nonsupervised notification signal output circuit that a panel automatically activates when a Supervisory event occurs.
NSCommonTroubleOff	Command used to turn off a nonsupervised notification signal output circuit that a panel automatically activates when a Trouble event occurs.

NSCommonTroubleOn	Command used to turn on a nonsupervised notification signal output circuit that a panel automatically activates when a Trouble event occurs.
NSCommonTroubleOutput	Classification used for a nonsupervised notification signal output circuit that a panel automatically activates when a Trouble event occurs.
NSDamperControl	A module assignment made in the SDC configuration for a nonsupervised output that automatically turns on when a damper's status goes active.
NSDoorControl	A module assignment made in the SDC configuration for a nonsupervised output that automatically turns on when a door's status goes active.
NSFanControl	A module assignment made in the SDC configuration for a nonsupervised output that automatically turns on when a fan's status goes active.
Object	An entity in the controller database, which represents an addressable device, a circuit, or a point in the system.
Power	A module assignment made in the SDC Configuration for a sprinkler supervisory input that reports a loss of power to electrically powered sprinkler equipment.
Priority	Numbers assigned to commands in output statements to specify their importance over other commands.
Pseudo point	An input generated by a nonphysical device in the system.
Pull	A module assignment made in the SDC Configuration for an alarm input that is activated using a mechanical switch, as in a manual pull station.
Relay	A module assignment made in the SDC Configuration for a device that switches positions to activate other devices or outputs.
RemoteTextMessage	An object that displays an active system point on a remote alphanumeric annunciator. Select the LSRA or the SMDN accessory in the Object Configuration table and look for a RemoteTextMessage in the Device Type column.
Revision	A set of changes made and saved in a single project database.
Rule	A programming statement that consists of a label, an input statement, and one or more output statements that specify what the system will do during a specific event.
SDC (Signature data circuit)	The wiring that connects detectors and modules to each other and to the main controller module.
SDU	The Systems Definition Utility (SDU), which consists of everything necessary to program a fire alarm system, check its status, and generate reports on it.
Smoke	A module assignment made in the SDC Configuration for an alarm input that goes active when a Signature series smoke detector senses a prescribed level of obscuration.

## Glossary

SmokeVfy	A module assignment made in the SDC Configuration for an alarm input that goes active when a smoke detector is in the verification mode.
SprinklerSupervisory	A module assignment made in the SDC Configuration for a sprinkler supervisory input.
StageOne	A module assignment made in the SDC Configuration for the pre-alarm stage (first address) of a two-stage pull station.
SupDuct	A module assignment made in the SDC Configuration for a duct input that reports off-normal supervisory duct status.
Supervisory	A module assignment made in the SDC Configuration for an IDC that monitors the integrity of its circuit.
Switch	A module assignment made in the SDC Configuration for a control/display module switch.
Tamper	A module assignment made in the SDC Configuration for a supervisory input that reports off-normal sprinkler system status.
Telephone	A module assignment made in the SDC Configuration for a supervised telephone circuit.
Temperature	A module assignment made in the SDC Configuration for a sprinkler supervisory input that reports freezing temperatures in the vicinity of sprinkler system components.
Time control	Any programmed command that takes place during a specified day type at a prescribed hour and minute for a chosen duration. See also General, Holiday, Weekend, and Weekday.
UserDefinedLED	A light emitting diode (LED) on the front panel display that the user may program to light for any system function.
UserDefinedSwitch	A switch on the front panel display that the user may program to perform any system function.
Visible notification appliances	Indicating appliances designed to produce a signal building occupants can see. Examples include strobes, lamps, target, or meter deflection.
Visual	An assignment made in the SDC Configuration table for a supervised module output wired to a notification appliance circuit that building occupants can see.
Waterflow	A module assignment made in the SDC configuration for an alarm input that detects a fire by the movement of water through the fire-protection sprinkler system.
Weekday	A setting that causes time controls to take place only during weekdays.
Weekend	A setting that causes time controls to take place only during weekends.
Zone	A distinct physical area, in which closely associated devices are located.

\*

\*. See wildcards

{

{. See comments, rule

## A

action 9000. See GuardPatrolAlarm

action 9002. See SysReset

action 9003. See Drill

action 9004. See AlarmSilence

action 9500. See Evacuation

actions

described • 2.9–2.10

in Activate commands • 4.2

in Define events • 3.7

in Disable commands • 4.14

in Enable commands • 4.16

in Restore commands • 4.45

Activate command • 4.2

Active event • 3.2–3.3

Alarm event • 3.4

alarm initiating devices

in Active events • 3.2

in Alarm events • 3.4

in Disable commands • 4.14

in Enable commands • 4.16

in TestActive events • 3.11

in TestTrouble events • 3.13

in Trouble events • 3.15

AlarmSilence • 2.9

AND groups • 2.7, 3.7

## B

bar code readers • vii

bell code groups • 2.7

## C

CallIn event • 3.5

Cancel command • 4.3

Close command • 4.4

commands

Activate • 4.2

commands (*continued*)

Cancel • 4.3

Close • 4.4

CommonAlarmOff • 4.5

CommonAlarmOn command • 4.6

CommonMonitorOff • 4.7

CommonMonitorOn • 4.8

CommonSupervisoryOff • 4.9

CommonSupervisoryOn • 4.10

CommonTroubleOff • 4.11

CommonTroubleOn • 4.12

Delay • 4.13

Disable • 4.14–4.15

Enable • 4.16–4.17

FanOff • 4.18

FanOn • 4.19

HoldDoor • 4.20

LEDOff • 4.21

LEDOn • 4.22

listed by device • A.17–A.21

NSClose • 4.23

NSCommonAlarmOff • 4.24

NSCommonAlarmOn • 4.25

NSCommonMonitorOff • 4.26

NSCommonMonitorOn • 4.27

NSCommonSupervisoryOff • 4.28

NSCommonSupervisoryOn • 4.29

NSCommonTroubleOff • 4.30

NSCommonTroubleOn • 4.31

NSFanOff • 4.32

NSFanOn • 4.33

NSHoldDoor • 4.34

NSOpen • 4.35

NSReleaseDoor • 4.36

Off • 4.37–4.38

Offguard • 4.39

On • 4.40–4.41

OnGuard • 4.42

Open • 4.43

ReleaseDoor • 4.44

Restore • 4.45

comments, rule • 1.4

CommonAlarmOff command • 4.5

CommonAlarmOn • 4.6

CommonMonitorOff command • 4.7

CommonMonitorOn command • 4.8

CommonSupervisoryOff command • 4.9

CommonSupervisoryOn command • 4.10

CommonTroubleOff command • 4.11  
 CommonTroubleOn command • 4.12  
 compiling the rules file • 1.11  
 Confirmation event • 3.6

## D

default priorities • 2.5  
 Define event • 3.7–3.8  
 Delay command • 4.13  
 device type lists, valid  
   for Activate commands • 4.2  
   for Active events • 3.2  
   for Alarm events • 3.4  
   for Confirmation events • 3.6  
   for Define events • 3.7  
   for Disable commands • 4.14  
   for Enable commands • 4.16  
   for Monitor events • 3.9  
   for Off commands • 4.37  
   for On commands • 4.40  
   for Restore commands • 4.45  
   for Supervisory events • 3.10  
   for TestActive events • 3.11  
   for TestTrouble events • 3.13  
   for Trouble events • 3.15  
 dialer groups • 2.7  
 Disable command • 4.14–4.15  
 documentation  
   EST2 • iv  
   related • v  
 Drill • 2.9

## E

Enable command • 4.16–4.17  
 equipment, optional • vii  
 EST2 documentation • iv  
 Evacuation • 2.9  
 events  
   Active • 3.2–3.3  
   Alarm • 3.4  
   CallIn • 3.5  
   Confirmation • 3.6  
   Define • 3.7–3.8  
   listed by device • A.6–A.9  
   Monitor • 3.9  
   Supervisory • 3.10  
   TestActive • 3.11–3.12  
   TestTrouble • 3.13–3.14  
   TimeControl • 2.13  
   Trouble • 3.15–3.16  
 examples, programming. *See under* sample  
   command rules; sample event rules  
 exceeding rules editor memory limits • 1.8

## F

FanOff command • 4.18  
 FanOn command • 4.19  
 Firephone  
   in Callin events • 3.5  
   in Disable commands • 4.15  
   in Enable commands • 4.17  
   in Off commands • 4.37  
   in On commands • 4.40  
 formats, label • 1.13

## G

glossary of terms • Y.1–Y.6  
 guard patrol groups  
   defined and illustrated • 2.8  
   in Offguard commands • 4.39  
   in Onguard commands • 4.42  
 GuardPatrolAlarm • 2.8, 2.11

## H

HoldDoor command • 4.20

## I

inputs. *See specific entries under* events  
 input-to-output matrix • 1.19

## L

labels  
   characteristics of • 1.7  
   developing a plan for • 1.12–1.17  
 LampTest • 2.9  
 latching supervisory devices  
   in Active events • 3.2  
   in Disable commands • 4.14  
   in Enable commands • 4.16  
   in Supervisory events • 3.10  
   in TestActive events • 3.11  
   in TestTrouble events • 3.13  
   in Trouble events • 3.15  
 LEDOff command • 4.21  
 LEDOn command • 4.22  
 limit in rules editor, memory • 1.8  
 logic groups • 2.7  
 L-variable • 2.3

## M

mathematical operators • 2.4  
 matrix, input-to-output • 1.19  
 memory limit, rules editor • 1.8  
 modifiers, label • 1.14–1.15



Monitor event • 3.9

## N

NetworkClassAFault • 2.11

non-latching monitor devices

in Active events • 3.2

in Disable commands • 4.14

in Enable commands • 4.16

in Monitor events • 3.9

in TestActive events • 3.11

in TestTrouble events • 3.13

in Trouble events • 3.15

non-latching supervisory devices • 3.2

nonsupervised output devices

in Off commands • 4.37

in On commands • 4.40

in Trouble events • 3.15

NSClose command • 4.23

NSCommonAlarmOff command • 4.24

NSCommonAlarmOn command • 4.25

NSCommonMonitorOff command • 4.26

NSCommonMonitorOn command • 4.27

NSCommonSupervisoryOff command • 4.28

NSCommonSupervisoryOn command • 4.29

NSCommonTroubleOff command • 4.30

NSCommonTroubleOn command • 4.31

NSFanOff command • 4.32

NSFanOn command • 4.33

NSHoldDoor command • 4.34

NSOpen command • 4.35

NSReleaseDoor command • 4.36

numbers and unique labels • 1.16

N-variable • 2.2

## O

objects

defined • 1.6

identification of by groups • 1.18

Off command • 4.37–4.38

OffGuard command • 4.39

On command • 4.40–4.41

OnGuard command • 4.42

Open command • 4.43

operators, mathematical • 2.4

output commands. *See specific entries under commands*

## P

plan, label • 1.12–1.17

predefined actions • 2.9

priorities • 2.5

programming examples. *See under* sample command rules; sample event rules

## Q

quick reference • A.1–A.21

## R

readers, bar code • vii

related documentation • v

ReleaseDoor command • 4.44

requirements, minimum equipment • vi

Restore command • 4.45

rules

examples of. *See under* sample command rules; sample event rules

factors affecting compile speed of • 1.11

memory limit for in rules editor • 1.8

using L-variables in • 2.3

using mathematical operators in • 2.4

using N-variables in • 2.2

using wildcards in • 2.2

## S

sample command rules

Activate • 4.2

Cancel • 4.3

Close • 4.4

CommonAlarmOff • 4.5

CommonAlarmOn • 4.6

CommonMonitorOff • 4.7

CommonMonitorOn • 4.8

CommonSupervisoryOff • 4.9

CommonSupervisoryOn • 4.10

CommonTroubleOff • 4.11

CommonTroubleOn • 4.12

Delay • 4.13

Disable • 4.15

Enable • 4.17

FanOff • 4.18

FanOn • 4.19

HoldDoor • 4.20

LEDOff • 4.21

LEDOn • 4.22

NSClose • 4.23

NSCommonAlarmOff • 4.24

NSCommonAlarmOn • 4.25

NSCommonMonitorOff • 4.26

NSCommonMonitorOn • 4.27

NSCommonSupervisoryOff • 4.28

NSCommonSupervisoryOn • 4.29

NSCommonTroubleOff • 4.30

NSCommonTroubleOn • 4.31

sample command rules (*continued*)

- NSFanOff • 4.32
- NSFanOn • 4.33
- NSHoldDoor • 4.34
- NSOpen • 4.35
- NSReleaseDoor • 4.36
- Off • 4.38
- OffGuard • 4.39
- On • 4.41
- OnGuard • 4.42
- Open • 4.43
- ReleaseDoor • 4.44
- Restore • 4.45

sample event rules

- Active • 3.3
- Alarm • 3.4
- CallIn • 3.5
- Confirmation • 3.6
- Define • 3.7–3.8
- Monitor • 3.9
- Supervisory • 3.10
- TestActive • 3.12
- TestTrouble • 3.14
- Trouble • 3.16

sequences

- described • 2.11–2.12
- in Activate commands • 4.2
- in Define events • 3.7
- in Disable commands • 4.14
- in Enable commands • 4.16

StartAction • 2.10

StartSequence • 2.12

summaries

- command specification • A.10–A.16
- event specification • A.2–A.5

supervised output devices

- in Confirmation events • 3.6
- in Off commands • 4.37
- in On commands • 4.40
- in TestTrouble events • 3.13–3.14
- in Trouble events • 3.16

Supervisory event • 3.10

switches

- in Active events • 3.2
- in Disable commands • 4.15
- in Enable commands • 4.17

syntax, basic

- for every rule • 1.3
- for input statements • 1.4
- for output statements • 1.5–1.6

syntax, command

- Activate • 4.2
- Cancel • 4.3
- Close • 4.4
- CommonAlarmOff • 4.5

syntax, command (*continued*)

- CommonAlarmOn • 4.6
- CommonMonitorOff • 4.7
- CommonMonitorOn • 4.8
- CommonSupervisoryOff • 4.9
- CommonSupervisoryOn • 4.10
- CommonTroubleOff • 4.11
- CommonTroubleOn • 4.12
- Delay • 4.13
- Disable • 4.14
- Enable • 4.16
- FanOff • 4.18
- FanOn • 4.19
- HoldDoor • 4.20
- LEDOff • 4.21
- LEDOn • 4.22
- NSClose • 4.23
- NSCommonAlarmOff • 4.24
- NSCommonAlarmOn • 4.25
- NSCommonMonitorOff • 4.26
- NSCommonMonitorOn • 4.27
- NSCommonSupervisoryOff • 4.28
- NSCommonSupervisoryOn • 4.29
- NSCommonTroubleOff • 4.30
- NSCommonTroubleOn • 4.31
- NSFanOff • 4.32
- NSFanOn • 4.33
- NSHoldDoor • 4.34
- NSOpen • 4.35
- NSReleaseDoor • 4.36
- Off • 4.37
- Offguard • 4.39
- On • 4.40
- Onguard • 4.42
- Open • 4.43
- ReleaseDoor • 4.44
- Restore • 4.45

syntax, event

- Active • 3.2
- Alarm • 3.4
- CallIn • 3.5
- Confirmation • 3.6
- Define • 3.7
- Monitor • 3.9
- Supervisory • 3.10
- TestActive • 3.11
- TestTrouble • 3.13
- Trouble • 3.15

SysReset • 2.9

Systems Definition Utility (SDU) • vi

## T

Telephone

- in Callin events • 3.5

Telephone (*continued*)  
    in disable commands • 4.15  
    in Enable commands • 4.17  
    in Off commands • 4.37  
    in On commands • 4.40  
TestActive event • 3.11–3.12  
TestTrouble event • 3.13–3.14  
time controls  
    in Activate commands • 4.2  
    in Define events • 3.7  
    programming of • 2.13  
Trouble event • 3.15–3.16

## U

user-defined LEDs  
    in LEDOff command • 4.21  
    in LEDOn command • 4.22

user-defined LEDs (*continued*)  
    in Off commands • 4.37  
    in On commands • 4.40  
user-defined switch  
    in Disable commands • 4.15  
    in Enable commands • 4.17  
using labels as messages • 1.17

## V

variables, L • 2.3  
variables, N • 2.2

## W

wildcards (\*) • 1.10, 1.16, 2.2  
writing comments about rules • 1.4

