



Understanding Rules

CyberHouse software turns your PC into a sophisticated control station, capable of running your home security and automation components according to your requirements and without intervention from you. You and your family can relax as your home is automatically lighted, heated, cooled, secured, opened, closed, and more.

How do I instruct CyberHouse to work the way I like to live?

You instruct CyberHouse by writing “rules” that describe:

- what you want CyberHouse to pay attention to
- what activities around the house you consider to be important
- what you want CyberHouse to do when an important activity occurs
- what you want CyberHouse to do while it is waiting for an important activity to occur

CyberHouse runs on the rules you specify—the “rules of the house.”

CyberHouse Runs on Rules

Software can be designed to follow pre-coded patterns of logic based on the programmer’s requirements or it can be designed to respond to its user’s requirements. CyberHouse responds to your requirements and you define those requirements through rules.

What does a rule look like?

Rules consist of *cause phrases* and *effect phrases*. An example of a rule might be:

|At 8:00| then |Lights on|
Cause → Effect

This rule is very simple to understand—at 8 AM, turn on the lights. A rule with more than one cause and more than one effect might be:

At 8:00 Mon then Lights on, Coffee Pot on

Though still in English, this rule seems more condensed in its phrasing. You read it as—at 8 AM on Mondays, turn on the lights and the coffee pot.

How do I write a rule?

Writing rules for CyberHouse is easy. You can use either the Graphical or classic rule builders to help you describe your requirements quickly and consistently. However, before you begin writing rules, we suggest that you continue reading through this chapter, so you understand what CyberHouse considers a valid rule and what it does with the rules you write.

Are there special formats and codes I need to learn?

No. The rule builders work with simple English words and phrases and provide all of the building blocks you need to create valid rules for everyday requirements—you simply point and click.

The rule builders also allow you to go beyond the basics and create advanced rules that build upon previous rules. As you begin to explore the more advanced capabilities, there are many hints and techniques that we will share with you.

Any new terminology I need to learn?

Yes and no. There are a few terms that will help you write your rules with greater understanding. Terms like:

- cause phrase
- effect phrase
- device-value pair
- Lazy T diagramming

help you define rules more quickly as soon as you learn their meaning. Each of these terms is explained fully—with examples—in the sections that follow.

You don't need to know these terms to begin writing rules, but they will help you become expert at CyberHouse's "shorthand" language enabling you to write the more advanced rules that cause CyberHouse to excel.

How long does it take to write a rule?

Only a minute or so. It can take only seconds when you know exactly what you want CyberHouse to do.

Do rules need to be added in a special order?

Yes and no. Each rule you write is independent and can stand alone. CyberHouse's Rule Agent takes all of the rules you create and manages them collectively, deciding to execute each rule based on what's happening in your home. You can create new rules in any order and assign related rules to different rule sets if you choose.

Note though, that CyberHouse executes its rules in the order they are listed in the rule set. If you have two or more rules that you want to execute in a particular order, you can drag'n'drop the rules into the desired order using one of the rule builders.

We recommend that you establish a method of organizing the contents of your rule sets so you can find a rule when you decide you want to edit it again. Your method for organizing rules doesn't matter to CyberHouse.

How does CyberHouse know when to execute my rules?

CyberHouse is an event-driven system. When an action that has been defined to it occurs, it reviews all the instructions it has to see if it should respond and how.

For example, a driving event could be a motion detector reporting motion. When `Living Room Motion`—a device you registered with CyberHouse—detects motion, it reports it to CyberHouse. This is an event. CyberHouse checks through all the rules you’ve written to see what you have instructed it to do—perhaps turn on a light or a siren.

Another “event” is the reaching of a certain time. CyberHouse manages this action a little differently. When you write a rule instructing CyberHouse to “do” something at a specific time, for example, turn on a lamp at 6 PM, CyberHouse immediately calculates how long it must wait until it needs to turn on the lamp. As the clock ticks, CyberHouse counts down. When the count reaches zero, the lamp is turned on.

The Rule Agent is fine-tuned to keep both its rules and the events occurring in your home in complete synchronization, so you do not need to worry about the details of your environment.

CyberHouse also supports a Startup rule set where you can specify system level settings, such as `Set Voice` and `Set Volume`. CyberHouse always executes this rule set first after any power-up condition.

Is there a limit to the number of rules CyberHouse can manage?

Not really. Create as many rules as you need to feel comfortable in your automated home. Our test labs currently run three separate homes on a single CyberHouse system. We write rules to test hundreds of different possibilities, so our systems are very heavily loaded. Yet they use only a few % (on average) of the PC’s computing power to manage our automation devices.

Can I write more than one rule for a device?

Absolutely. This is where the power of CyberHouse is most useful. You can write many rules to describe the exact behavior you want for a particular device, or you can write just one rule. It is up to you.

Cause and Effect

CyberHouse rules are based on the following simple principle:

A cause generates an effect.

If you understand this principle, you can create all of the rules you need to automate CyberHouse for your specific home requirements. For example, begin thinking of the activities in your home in the following way:

Possible causes: door opens then...
clock strikes 8 AM then...
motion detector triggers then...

Other causes that you want to automate: _____

Possible effects: ...then the lights go on
...then the coffee pot starts brewing
...then the doorbell chimes

Other effects that you want to occur: _____

CyberHouse's rule builders and the Rule Agent are designed to work with this same basic information:

cause then effect

which you can interpret as

When the cause happens, then perform the effect.

In CyberHouse, we refer to each action that makes up a cause or an effect as a *phrase*. A phrase is a deliberate *defined* action or state, such as:

- Light On
- Light Off
- Motion Detector Alarm
- Motion Detector Clear

As you begin to think of CyberHouse phrases, be sure to include the specific action or the desired state you want to recognize or attain.

Device-Value Pairs

When you think in terms of phrases, you can use a simple shorthand language known as *device-value pairs*. Using the cause/effect examples from the previous page, you can simplify these actions/phrases to:

Possible causes: door opens then... Door Open
clock strikes 8 AM then... At 8:00
detector senses then... Motion Detector Alarm

Other causes that you want to automate: _____

Possible effects: ...then the lights go on Light On
...then the coffee pot starts brewing Coffee Pot On
...then the doorbell chimes Chimes On

Other effects that you want to occur: _____

The terms you use for *device* are defined by you when you add a new automation device to your home network. The terms you use for *value* are defined within CyberHouse. You can also create new value options when you begin writing your own advanced rules.

CyberHouse “thinks” simply and requires instructions written in this simple device-value pair style. The rule builders are specially designed to provide point-and-click rule building so you don’t have to be concerned about syntax.

Multiple phrases

If you are like me, you can think in CyberHouse’s simple device-value pair terms, but you’ll probably want CyberHouse to recognize and act on more elaborate causes and effects. For example, you want hot coffee at 6:30 AM on weekdays, but not on weekends. To make your requirements clearer, you can define more than one phrase at a time for both causes and effects.

CyberHouse allows you to define any number of phrases for a cause—that is, any number of actions or states that need to be in place before

CyberHouse performs an effect (or a group of effects). For the effect to occur, *all* of the cause phrases must be in place.

For example You can define the following two rules:

At 6:30 During Mon to Fri then Coffee Pot on

At 9:00 During Sat to Sun then Coffee Pot on

Diagram your rules

Rules can become quite complex and long. To keep perspective while you are creating your rules, use the following graphic technique to “diagram” your rules. It keeps your intentions clear and helps you follow the phrases that are involved in the rule you are creating. As you become more comfortable with your system, you will be creating more precise rules, which require multiple phrases. Diagramming keeps the phrases in a logical order.

The CyberHouse logo is a graphic interpretation of a diagrammed rule.

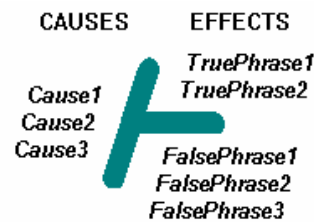


Figure 1: The “Lazy T” technique for diagramming a rule

On the left side of the “lazy T” is a list of the cause phrases that CyberHouse must look for.

On the right side are two lists of effect phrases. The phrases on top are the ones we want to happen when the cause phrases happen, i.e., when all of the cause phrases are “true.”

The phrases on the lower right represent the actions we want CyberHouse to take when an event occurs and the causes are *not* all true. If *any* of the cause phrases is *not* happening when an event occurs and CyberHouse checks this rule, CyberHouse performs the false phrase effects.

The operation of the rule is simple:

- The **True** phrase list is applied whenever all cause phrases become true.

- The **False** phrase list is applied whenever any cause phrase become false.

The following diagrams show some basic rules.



False phrase effects

A False phrase is a direct instruction that CyberHouse must follow, just like each True phrase.

A good example of the use of a false phrase is a hallway light in an office building. You want it on when your staff is working during the day, but at night you want it dimmed to a low level for security. You might also want it to go to full brightness if a motion detector detects someone entering the front door at night. Your rules might look like:



FrontDoorAlarm On HallLight Dim15

The first rule covers your whole week. From 7:30 AM to 6 PM on weekdays, the hall light will be on full brightness. All the rest of the time (including the weekend), it will be dimmed to level 2, saving you energy and “memory” power. You don’t have to remember or leave a note at the front door reminding the last person to dim the lights before they leave!

You can also notify the police that they should investigate whenever they see the hallway light on full brightness after working hours.

Rule Conflicts

The more complex your rules become, the more opportunity you have to create rule conflicts and ambiguous states. Conflicts occur when you tell CyberHouse to perform one action on a device under a specific condition and, in another rule, you specify another action for the device under the same condition or under a condition that might occur at the same time.

In our hall light example above, what would you expect to happen if the `Front Door Alarm` detects motion at exactly 6 PM? It depends on which rule CyberHouse processes first and that is determined by CyberHouse each time it compiles its rules.

It could process the dimming of the light first, then the alarm. The police would then see the light at full brightness after 6 PM and investigate. It might also process the rules in the opposite order—alarm first, then dim the light at 6 PM as it always does. Your thief might trip the alarm again on his way out, but the police wouldn’t notice until he was gone.



Always check your rule phrases to be sure you understand (and fix) any conflicts or ambiguities in your rule phrasing, especially for security devices.

For our hall light example, you could choose to ignore the possibility that someone might enter your building at exactly 6 PM and leave your rules as currently written. Or you could add another action to the

Front Door Alarm rule, such as a siren horn. You always have options with CyberHouse's capabilities.

To prevent conflicts or to recognize them quickly when they affect your plans, you can organize your rules into rule sets. This lets you review similar rules while you are writing new ones.

Reading Rules

Now that you understand the components of a rule and few techniques for thinking about and creating rules, you can begin analyzing some sample rules in the different ways you might see them on your screen or in printed form.

There are a number of different formats that you might see a rule:

- described in text
- diagrammed
- as a rule statement in the rule builders
- in a rule file "dump"
- in a .RUL file

In each of these methods of representing a rule, there are slight syntax differences, but the rule and its meaning stay the same. The following example displays the same complex rule in each method you might see it.

Rules described in text

The rule we are using in this example is similar to the office building rule we used in a previous example.

"While we are on vacation, keep the radio in the kitchen on all day and turn off the flood light in the front walkway. At night, turn the light in the hallway on, but keep it dim to save energy, turn off the radio, and turn the flood light for the front walkway on."

Does this sound like a reasonable request from your spouse and children?

Diagrammed rules

Our new rule diagrams to look like this:

During 6/15 6/30 *KitchenRadio on*
During Sunrise Sunset *HallwayLight dim4*
 WalkwayFlood on

Rule statements in the rule builders

Our new rule looks like this when viewed in the graphical rule builder:

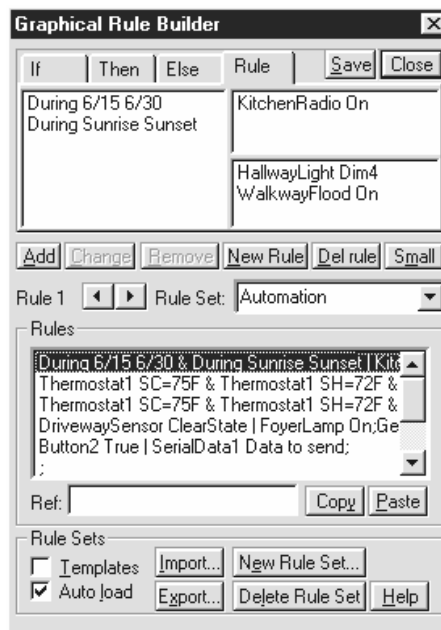


Figure 2: Reading a rule's content in the graphical rule builder

Our new rule looks like this when viewed in the classic rule builder:

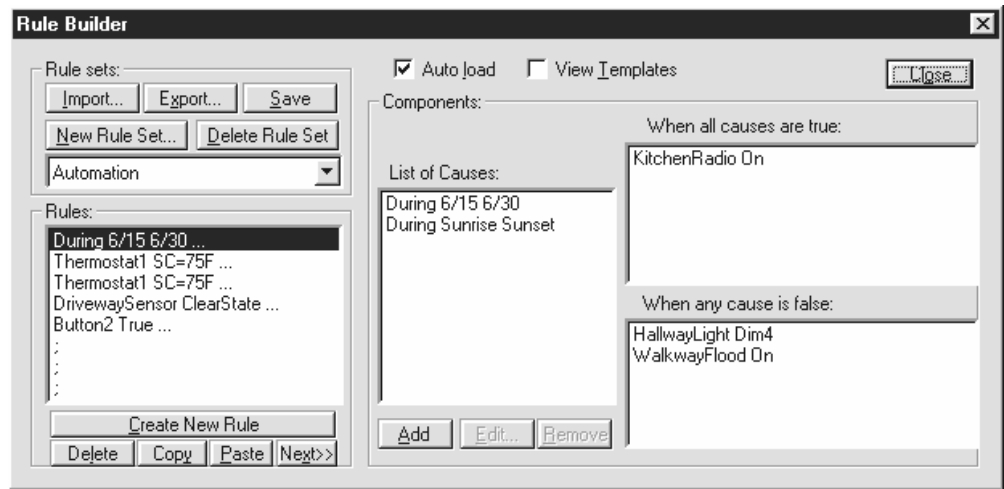


Figure 3: Reading a rule's content in the classic rule builder

Rules in a file dump

You can view your rules in a text file or on paper by “dumping” them out of the rule builders. This is actually done through a rule, one of the first you should add to your permanent collection. Instructions for dumping and reading your rules can be found in the online Help system.

Our rule looks like this in the “dump” file—very similar to what you see in any of the other diagrammed views of the rule:

Vacation:

```
[172695]    During 6/15 6/30 | Kitchen Radio on
[6329]    During Sunrise Sunset | -----
                                     | Hallway Light dim4
                                     | Walkway Flood on
```

Rule syntax in a .RUL file

You will probably never need to view your rules in their “raw” form, but if you do choose to look in the RULES directory, here’s what you will find:

```
During 6/15 6/30 & During Sunrise Sunset |
Kitchen Radio on ~ Hallway Light dim4 & Walkway Flood on;
```

The syntax markers indicate:

- & separates multiple phrases in a cause or effect.
- | separates the cause phrase(s) from the effect phrase(s).
- ~ separates True phrases from False phrases.

A Quick Summary

1. CyberHouse uses rules to understand how you want it to behave. You write rules to define specific events you want the software to watch for and what actions you want it to take when those events occur.
2. Rules are made of *causes* and *effects*. A rule can have multiple causes and/or effects, allowing you to write complex rules that define exactly how you want to live.
3. There are two types of effects:
 - *True effects* contain the instructions you want CyberHouse to follow when all of the causes occur.
 - *False effects* contain the instructions you want CyberHouse to use when any of the causes are not happening.
4. CyberHouse likes rules written in the simplest of language—*device-value pair phrases*.
5. As you write your rules, be aware of the causes and effects you specify. Be sure they do not conflict with a rule that is already in your system.
6. You can group rules together into *rule sets* for easy recall and for “swapping” with other CyberHouse users.
7. The simplest and most common format for viewing a rule is in its diagrammed form. In text format there are a few syntax markers you need to recognize:
 - & separates multiple phrases in a cause or effect.
 - | separates the cause phrase(s) from the effect phrase(s).
 - ~ separates True phrases from False phrases.



Chapter 5

How It All Works

Constructing rules and experimenting with CyberHouse's capabilities is, of course, the fun part. Behind the interface though, there is a well designed computing engine that makes it all possible.

The Rule Agent is a software component within the CyberHouse client-server system which implements the rules-based logic to control devices and coordinate the distributed applications.

The following sections describe how all of your automation network components work together to make the little things (like lighting changes) and the big things (like security system protection) happen.

We'll look at the devices first and then determine how the rules control the devices.

Devices

Automation devices are the electrical or electronic components that you physically install in your home to perform a function. There are many types of automation devices manufactured today by many different companies. You can group the available devices in two basic ways:

1. by how they communicate
 - powerline devices—which plug into your powerlines (wall outlets) to send or receive signals over your electric wires
 - wireless devices—which send or receive infrared or RF (radio frequency) signals through the air

2. by what they do
 - sensors—which detect changes and report them to you
 - controls—which turn things on and off

An example of a physical device is one that manages a light. You simply plug an X-10 module (such as the LM465 Lamp Module)—the device—into a wall socket and then register the name with CyberHouse which can then send instructions to turn the light on, to brighten or dim the light, and to turn the light off.

Other useful devices “Devices” in the CyberHouse system include more than just the physical devices that you can hold in your hand. To ease your rule writing and take advantage of the power of the computer, CyberHouse acknowledges three basic types of devices:

- physical devices, such as the sensors and controls you can hold in your hand
- “logical” devices which are abstract entities used to describe more complex behavior
- “satellite” applications connected to the CyberHouse system, such as Listen

Treating all devices—physical devices, logical devices, and PC applications—in a similar manner provides a degree of simplicity and generality that would be difficult to achieve otherwise.

Logical devices Logical devices are ones that you create. They are generally used to represent groups of devices, or states of the system. For example, you might want to have certain rules depend on whether your children’s school is in session or not. You can create a logical device called `School In Session` and create rules that set `School In Session` to true and/or false. Then create rules where `School In Session True` or `School In Session False` is an input, or cause, phrase.

If we create some rules based on our new logical device, they might look like:

During 9/5/95 6/14/96 **SchoolInSession True**
During Mon Fri
During 8:00 15:00 **SchoolInSession False**

On all weekdays throughout the school year, your son's stereo will be powered off at 8 AM. When your daughter gets a stereo of her own, you can add a rule to turn hers off as well.

For the professional programmer using CyberHouse: A logical device is analogous to a variable in other languages. It has a name and holds a value which can be a complex combination of physical devices, like a logical garage door which includes the button actuator as well as a door open/closed sensor. You use the simple name to invoke the complex combination—and it works.

For the non-programmers: Think of the logical device's name as a placeholder for something more complex. Using the garage door example and diagramming some possible rules might help.

For example, you have a garage door opener with a button in the car and one on the wall in the garage. Both of the buttons simply perform a “toggling” activity. If the door is open, you press either one of the buttons and the door closes. If the door is closed, you press either button and it opens. The buttons do not know what task they are being asked to do when you press them. They simply do the task that needs to be done right now. (The same logic applies to a three-way lighting switch.)

You cannot write a rule to tell one of the buttons to *close* the door because it doesn't know if it is open or not.

CyberHouse can combine the state of a sensor with a button that is pushed to create a garage door control that always does the proper action. When asked to open, it opens **ONLY** if it is currently closed. When asked to close, it closes **ONLY** if it is currently open. This garage door control is a logical device.

Satellite applications Satellite applications are considered devices so that the Rule Agent can instruct them to do things. Conversely, these applications can instruct the Rule Agent to perform functions. Each of these applications—Listen (currently)—can instruct the Rule Agent. Similarly, the Rule Agent can send instructions to the applications for control.

Assertions

Devices deal exclusively with *assertions*. The Rule Agent controls a device by applying an assertion (specifying the device name followed by a function - see below), and similarly, devices report information by applying assertions back to the rule processor.

Similar to rule phrases, all assertions have the following form:

```
Name function argument(s)
```

Assertions are *applied* to devices. Similarly, devices can *apply* assertions.

For example, a physical device like a motion detector applies an assertion of the form

```
Motion Detector Alarm
```

which may cause the Rule Agent to apply an assertion such as

```
Exterior Lights On
```

As another example, to control a satellite application like Listen, the Rule Agent can cause an audio file to play by asserting

```
Listen play c:\cyberhouse\data\file.wav
```

and similarly, when Listen detects noise it might assert

```
Noise On
```

Name Space

All of these assertions are globally distributed across all CyberHouse applications including the House Manager and any satellite applications such as CyberHouse, Listen, and Watch. These applications can be running across a network of interconnected computers and there can be multiple copies of them including the House Manager each of which contains a Rule Agent.

The Rule Agent(s) and all satellite applications share a common name space for devices, but within each application the device may have a unique definition. For example, one House Manager may have `Exterior Lights` defined to be a physical device, whereas another House Manager may have `Exterior Lights` defined as a logical

device. If the latter asserts `Exterior Lights On`, this assertion is received by the former which physically turns on the lights. Assertions, then, are void of any interpretation or knowledge of any device property, and can generally be thought of as text strings, although their structure carries a bit more information.

Arguments

The first word following the name of a device is called the *argument*. Examples of arguments are `On`, `Off`, `True`, `False`, `Green`, `Blue`, etc.

The arguments you can use depend on the type of the device. Physical devices require *keywords*, whereas logical devices can take any string of characters, including keywords.

Keywords are words that CyberHouse recognizes for the “value” in device-value pair phrases. They include words like `On`, `Off`, `True`, `False`, and they exclude words like `Green` and `Blue`. So, while logical devices can be set to any of these, physical devices can only be set to the keywords.

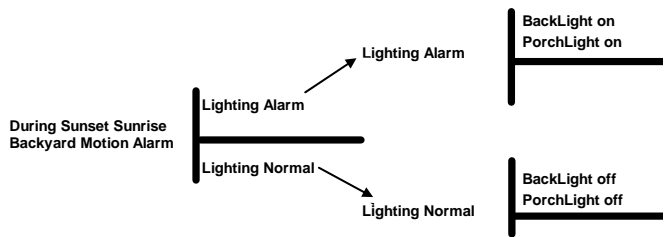
Many keywords are synonyms of others. Specifically, the set `On/True/Open/Alarm` are all equivalent and similarly the set `Off/False/Closed/Clear` are also equivalent.

For example, suppose you turned a lamp on by asserting `Lamp On` and in another rule tested the assertion by `Lamp True`. In this case, when the lamp was turned on, the test assertion would be computed as true, treating both arguments the same. This becomes especially useful in templates (described below) where the type of device is unknown.

Processing the Rules

The Rule Agent is a software component of the CyberHouse Manager that implements the network of rules. The Rule Agent typically receives assertions from devices (as well as a clock) and, according to the rules, it applies assertions back to devices, other Rule Agents, or other satellite applications. In this manner, events from sensors are transformed into actions by actuators, like

```
Motion Detector Alarm then Exterior Lights On
```

Rule Commands

The rule builders are familiar with many English commands. When you use these English commands in rules, CyberHouse recognizes that it must perform specific tasks. The list below represents the English commands that have specific meaning to CyberHouse. These commands are described in greater detail in the online Help system.

Keyword	Description
Archive	Renames the event archive file. Starts a new event file.
At	A cause only becomes true precisely at the time specified. Identifies a time that you want CyberHouse to perform an action.
Beep	An effect that causes the computer running the House Manager to "beep" or make other sound.
Dump	An effect that dumps the current rule set logic, logical device information, and timers into the specified file.
During	As a cause only, identifies a span of time during which this phrase is true.
Exec	An effect that launches the Windows application program specified by the file.

<i>Expire</i>	An effect that expires or terminates the named rule set.
<i>Invoke</i>	An effect that invokes—compiles and executes—a rule set.
<i>Log</i>	An effect that logs a message to the archive file.
<i>Quit</i>	An effect that terminates the House Manager.
<i>Status</i>	An effect that lists the message in the status window of the House Manager.
<i>Timer</i>	<p>Timers count down from a specified value to zero in steps of one second. After they reach zero, they are deleted. Timers can be tested for any positive value including zero.</p> <p>As a cause, <code>Timer</code> tests the value and asserts true or false. This command detects when a timer reaches the value specified by the argument. The assertion is true when they agree and false for all other times.</p> <p>As an effect, <code>Timer</code> creates the named timer and sets it to the value specified by seconds.</p>
<i>Use</i>	<p>Creates a rule set from a template.</p> <p>Templates are rule sets with common logic which can be applied to a variety of situations. Rather than develop a separate rule set for each situation, you develop a template for the logic and use it repeatedly.</p> <p>To be most useful, a template does not include specific device names, but a series of ^N names. When you use the template, you include the correct names for each of the ^N assignments in the <code>Use</code> rule statement. In this way, each rule set created from the same template performs a specific job.</p>



Advanced Rules and Templates

As in any computer application, value is gained through the ease of encoding higher forms of logic from simpler forms—procedures, structures, and object classes all lend themselves to this goal.

Writing Rules for Logical Devices

The Rule Agent does not distinguish between the various types of devices, and so, at the level of names, all devices are treated the same. In addition to physical devices sharing a common name space, non-physical devices are named in the same way. These are software devices, called *logical devices*, that you can create to describe more complex behavior. You can create a logical device from the New Device dialog, selecting Logical devices as the type.

Logical devices can both generate causes as well as receive effects, which can in turn generate more causes. These devices allow for the construction of complex devices out of simple ones, which in turn greatly simplifies the rules that use them.

As an example, electric garage doors pose an interesting problem for automation systems because the same electric control switch is pressed for both OPEN and CLOSE. To do the right thing, you must always know what the current state is, and then either push the button or not. Humans have no problem because they “look” to see what state the door is in and then make the decision.

To get the computer to do the equivalent, we need two controls:

- a sensor to detect whether the door is open or closed. Use a wireless magnetic door detector, such as the Ademco 5816.

- an X-10 relay module to close the garage door control contacts (effectively pressing the button). Use the X-10 UM506, set to momentary mode.

Install the wireless sensor and register it with CyberHouse using the name `Garage Door State`. Install the relay module and register it with the name `Garage Door Button`.

Then create a logical device called `Garage Door`. We will develop a few rules that will allow this logical device to cause the door to open or close whenever a corresponding assertion is applied—that is, if you apply the assertion `Garage Door Open` and the door is already open, nothing will happen; but if it is currently closed, your new controls will open it.

Creating rules for a Garage Door

Open a rule builder and create a rule set named `Garage Door Control`. Create the following rules:

`GarageDoor Open`  **`GarageDoorButton On`**
`GarageDoorState Clear`

`GarageDoor Close`  **`GarageDoorButton On`**
`GarageDoorState Alarm`

You can now use this logical device, `Garage Door`, in other rules throughout the system without dealing with these details. Simply apply an assertion for `Garage Door` and the right thing will be done.

For example, suppose you want the garage door to open for you when you enter the garage between 8:00 and 8:30 AM on weekdays **WHENEVER** it senses motion inside (you enter through another door to go to work).

The rule reads as follows:

During Mon Fri **GarageDoor Open**
During 8:00 8:30
GarageMotion Alarm



If the door is already open, this rule does nothing.

Using a Template

CyberHouse has the capability of using template rule sets to accomplish this goal. A template rule set is similar to a procedure, or macro in other languages—it encodes a set of rules which can subsequently apply to multiple uses. In a template rule set, names can be derived from arguments passed from a client rule set as in the following example:

Consider the following template rule set named `Momentary`. This rule set is included in your default distribution from Savoy Automation.

```
^1 On then Timer ^1Tmr ^2
```

```
Timer ^1Tmr 0 then ^1 Off;
```

This rather cryptic rule set is designed to create a *momentary* behavior for a device, meaning that it turns off a few seconds after it is turned on. Suppose that the physical device named `Relay` is designed only to turn on or off, but the application may require that it only momentarily turn on, let's say for 10 seconds, and then turn off. This trivial complexity would become a burden if each device was individually assigned this behavior. For convenience, we encapsulate this rule set into a template, named `Momentary`. To use the template all you have to do is state the name of the template in a `Use` command followed by device names, numbers, or whatever the template requires.

```
Use Momentary Relay 10
```

This statement is all that's required to transform the conventional relay into a momentary relay. You can easily develop your own templates for performing repetitive functions. To do this, all you need to know is how to encode a template.

Creating Your Own Templates

Templates are created the same way that you create other rule sets with one exception: All device names must be denoted by $\wedge n$, where n is a number corresponding to the n th argument in the `Use` command. In general, the `Use` command has the form:

```
Use TemplateName Argument1 Argument2 Argument3 ...
```

When the template is used to create a rule set, each occurrence of $\wedge n$ is replaced by `ArgumentN`.

For example, our statement `Use Momentary Relay 10` creates the following rule set:

```
Relay On then Timer RelayTmr 10
```

```
Timer RelayTmr 0 then Relay Off
```

Once a template is created, you only need to know how to use it. Observe the format of the `Use` command, and it can then be applied to many devices under many circumstances.

CyberHouse has a number of commonly used templates already installed. Add your own or import new ones from libraries created by others.

Inclusive OR

The Rule Agent is designed to compute the AND of the input (cause) assertions since this is what most applications require. In some cases, you may want to specify the inclusive OR of one or more assertions which can be derived from combinations of AND and the compliment, denoted here by the symbol ' \sim ':

$$A \text{ or } B = \sim(\sim A \text{ and } \sim B)$$

That is, the inclusive OR is obtained by ANDing the compliment of two or more inputs and then complimenting the result. We can diagram the effect as follows:



As an example, suppose you have set up rules for three alarm conditions and have defined three logical devices—`External Alarm`, `Internal Alarm`, and `Periphery Alarm` for each condition, respectively. Now suppose you want a single light, say `Master Light`, to indicate if ANY of these conditions are true.

`ExternalAlarm False`
`InternalAlarm False`
`PeripheryAlarm False`



`MasterLight Off`
`MasterLight On`




Appendix A: **Dumping Rules for Review**

Reviewing your rules in a dump file can be handy. You can use the search functions of a word processor to find every instance you use a particular device when you are about to write new rules for that device. This helps prevent rule conflicts.

Dumping the Rules to a File

You can view your rules in a text file or on paper by “dumping” them. This is actually done through a rule, one of the first you should add to your permanent collection.

I use a logical device named `Rule Dump` to notify CyberHouse that I want a rule dump. I simply click the icon on my Maintenance layout and CyberHouse updates the specified file with the latest rule and device information. You can choose any cause phrase you want to request a rule dump. My rule looks like this:

RuleDump on  *Dump C:{CyberHouse}dumpfile.txt*

While your rules file is small, you can dump it to a .TXT file. When you open it in Notepad, the text aligns correctly because it appears in a mono-spaced font, such as `Courier`. If you open the file in another word processing application and the layout is “wavy,” select the whole file and change to a mono-spaced font.

Reading a Dumped Rules File

A rules dump appears in report format, similar to the example file shown below. Rules are listed in their rule sets, in the order you view them in the rule builders.

The first line shows the time and date of the rules dump.

----- Dumped at 11:56:28, Wednesday, December 20, 1995

The second section lists all rules in diagrammed format under each rule set heading.

```
Maintenance:
    Garage Door Closed | Lighting Night Time
[15752] During Sunset 23:00 | Foyer Lamp On
    Rule Dump On | Dump
    C:\CyberHouse\dumpfile.txt

Lighting Rules:
    Lighting DayTime | Lamp Module Off
    Lighting EveningTime | Lamp Module Dim14
    Lighting NightTime | Lamp Module Dim6

TimedEvents:
[15752] During Sunset 22:00 | Lighting EveningTime
[36212] During 22:00 Sunrise | Lighting NightTime
[15752] During Sunrise Sunset | Lighting DayTime

Vacation:
[15752] During Sunrise Sunset | Kitchen Radio on
    -----
    Hallway Light dim4
    Walkway Flood on
```

The numbers in brackets show the number of seconds before a rule will execute next.

The third section lists each logical device and its current state.

```
Logical Devices:
    Lamp Module = Off
    Listen =
    Lighting = DayTime
```

The last section shows all of the timers that are in the process of counting down at the time of the dump.

----- Timers:

