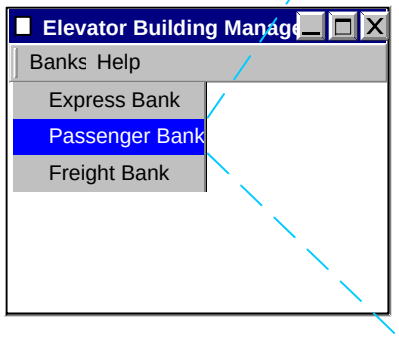
Elevator Application GUI Specification

I have prepared this specification for any GUI developers who would like to contribute a front end to Rev 2.0 of the Elevator Application.

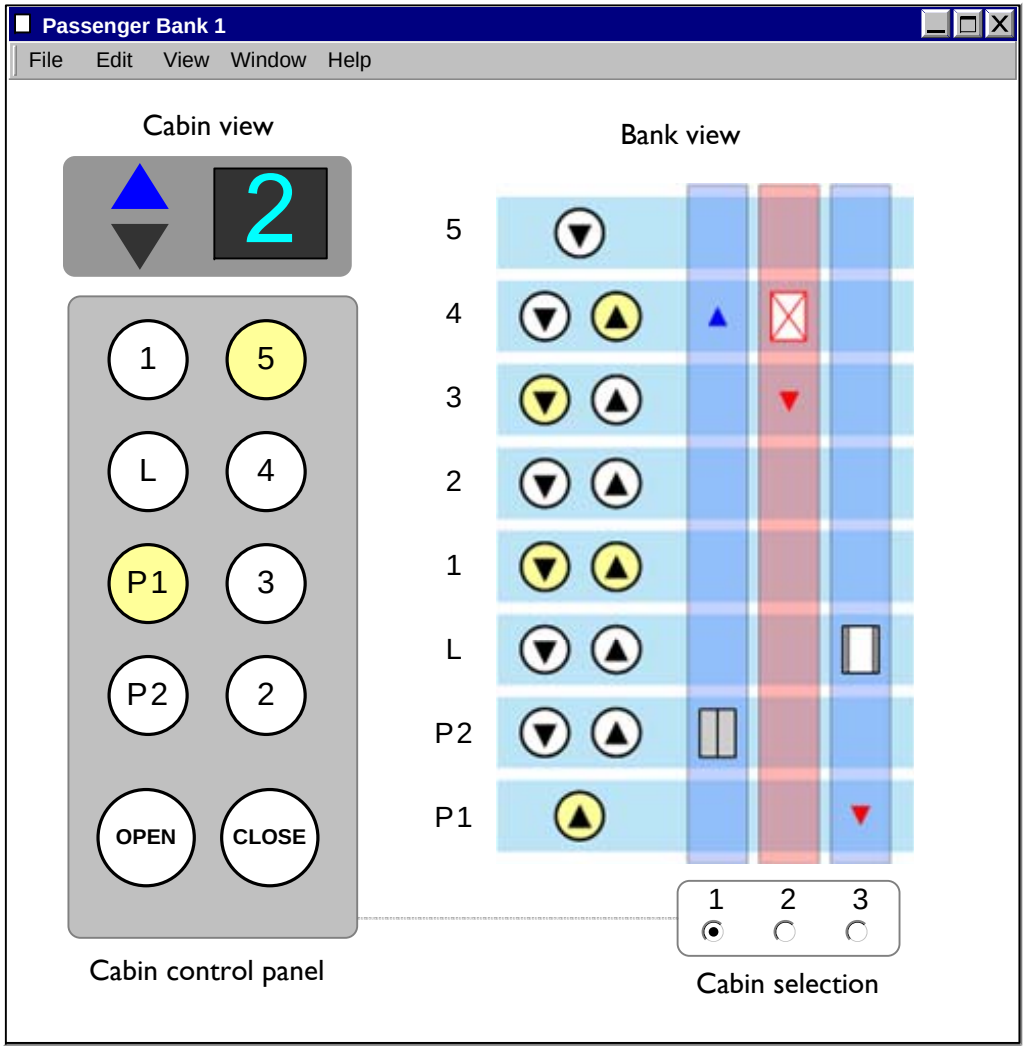
The main panel should list the names of all banks in the building. There will usually be only one or two choices here, but there may be as many as ten. For example, there could be a dropdown menu selection from an Elevator Building Management window as shown:

Buidling Management Window

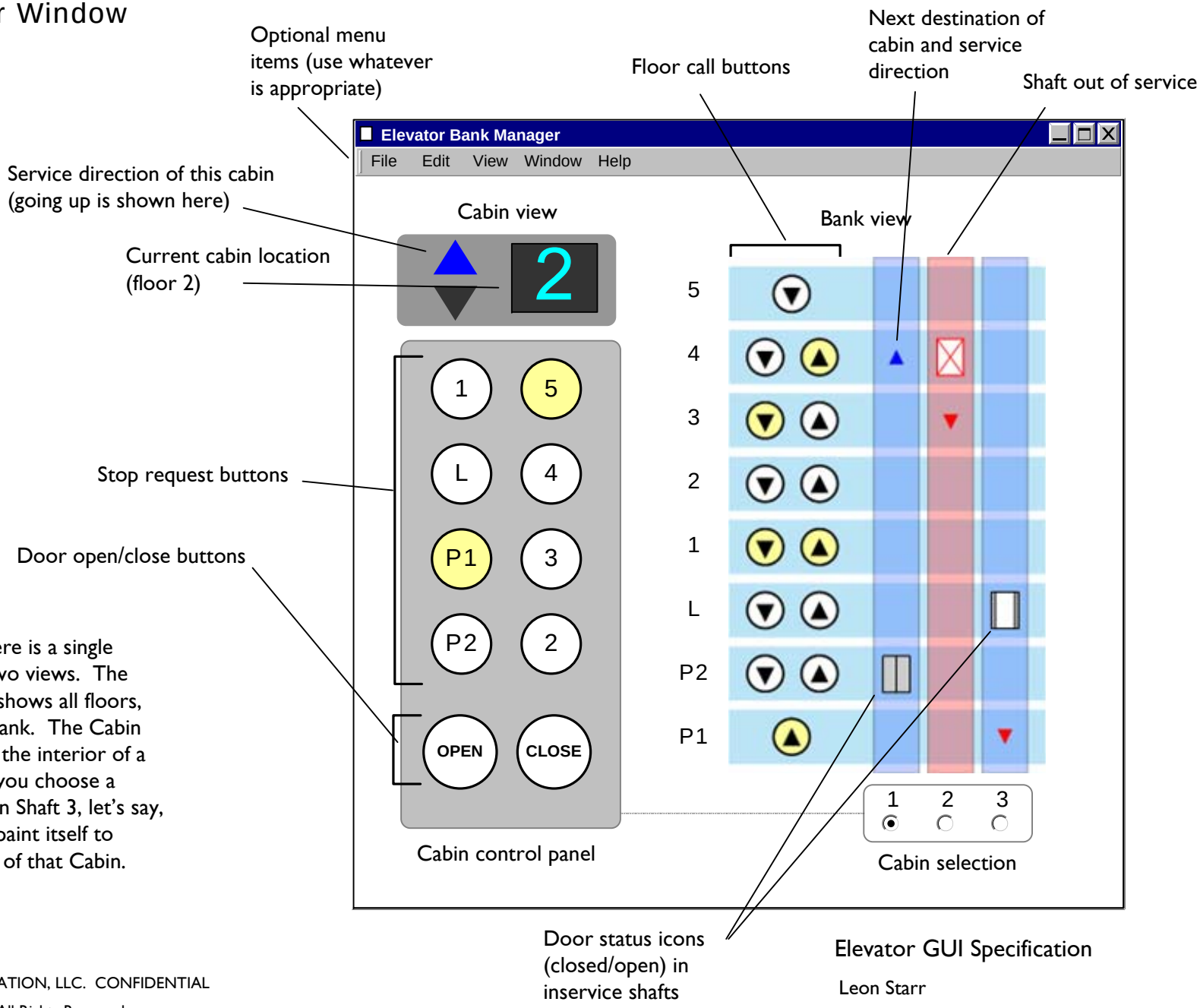


The user selects a bank from the menu. This selection brings up the bank control window shown on the right. The building management window should remain open in case the user wants to bring up multiple bank windows.

Bank Management Window



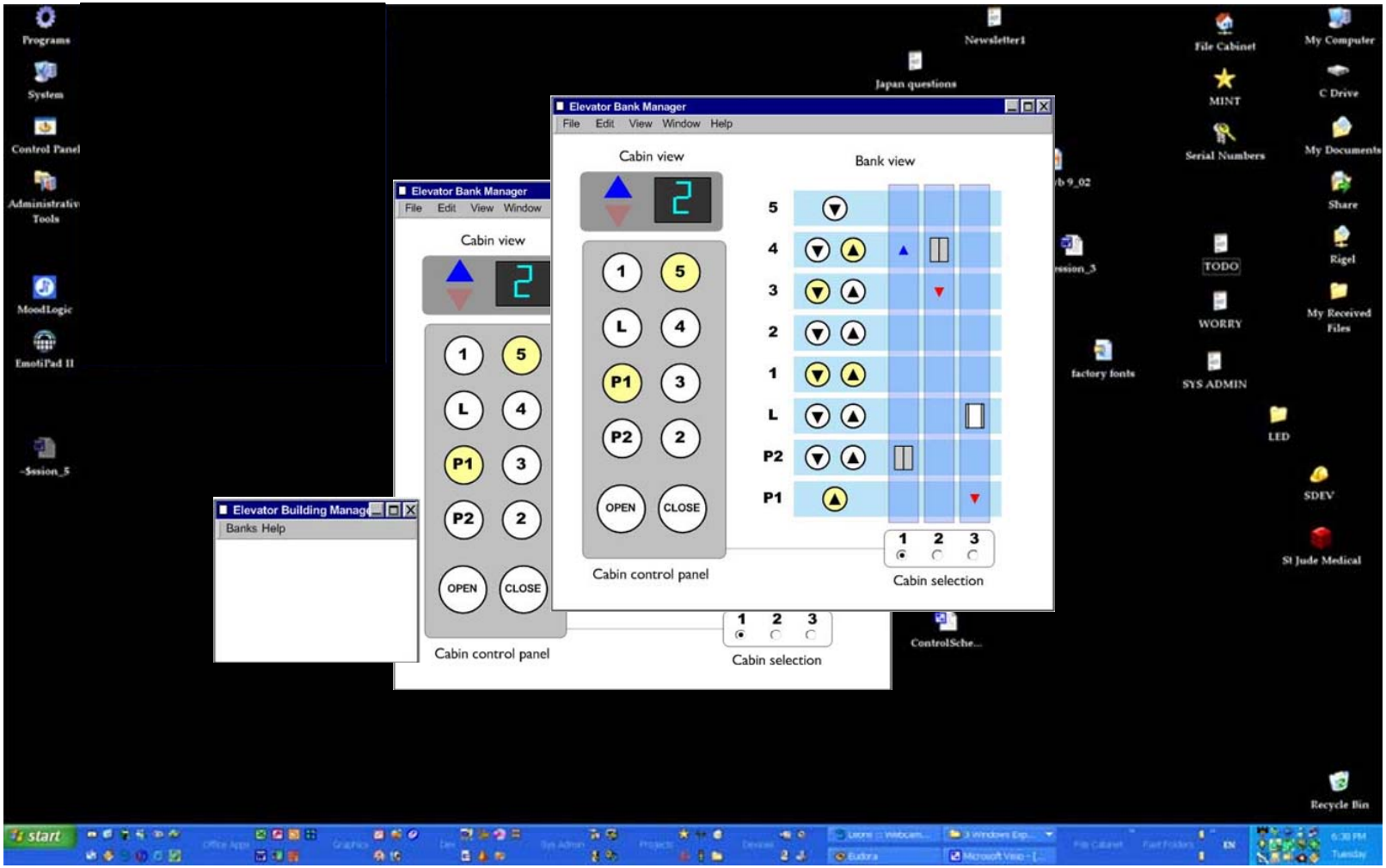
The Bank Manager Window



In the suggested GUI, there is a single window panel showing two views. The Bank View, on the right, shows all floors, shafts and cabins in the Bank. The Cabin View, on the left, mimics the interior of a single selected Cabin. If you choose a different Cabin, the one in Shaft 3, let's say, the Cabin view would repaint itself to reflect the control status of that Cabin.

Multiple Windows may be open

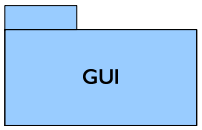
The Elevator Application controls all of the elevators in a single building. The elevators are all organized into one or more Banks within the Building. Each Bank can be viewed with the panel shown on the previous page. If there are multiple Banks, then the user will be able to call up a window for each. These windows can be tiled or overlapped just like any windows on a desktop.



Elevator Management API

This is a summary of the logical commands exchanged between the GUI and the Elevator Application. The final format of these commands will depend on the mechanism (function calls, network commands, etc) chosen for implementation. For now you may just want to stub these out to test your interface. We can worry about the implementation of the command wiring later. The commands are examined in detail on the following pages.

Commands sent to the Elevator Application



Stop_request (Shaft_ID, Floor_name)

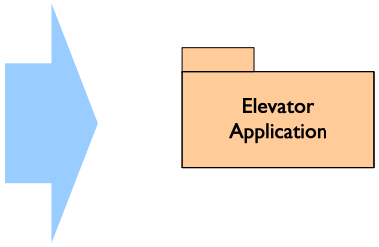
Floor_call (Bank_name, Floor_name, Direction)

Door_open (Shaft_ID)

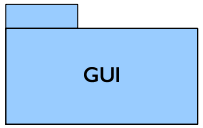
Door_close (Shaft_ID)

Door_release (Shaft_ID)

Take_shaft_outofservice (Shaft_ID)



Commands received from the Elevator Application



Clear_floor_call (Bank_name, Floor_name, Direction)

Clear_stop_request (Shaft_ID, Floor_name)

Cabin_at_floor (Shaft_ID, Floor_name)

Set_service_direction (Shaft_ID, Direction)

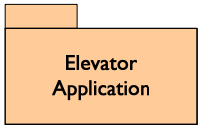
Doors_opened (Shaft_ID)

Doors_closed (Shaft_ID)

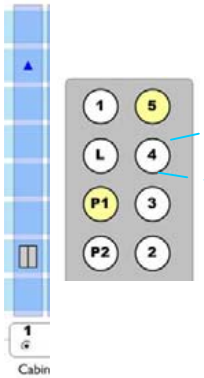
Shaft_outofservice (Shaft_ID)

Set_destination (Shaft_ID, Floor_name)

Cabin_arrived (Shaft_ID)



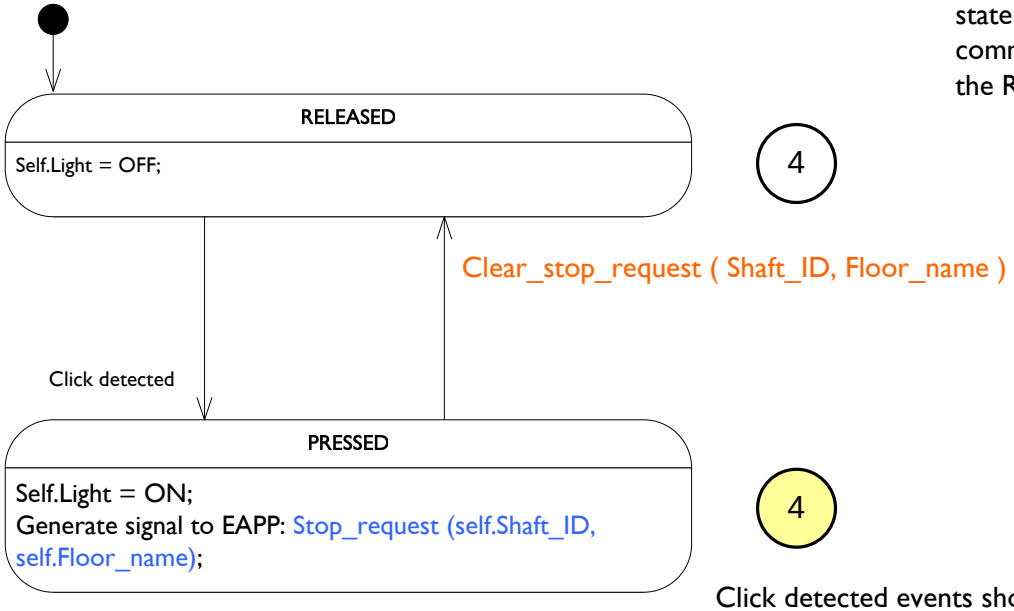
Cabin Panel Floor Stop Buttons



4

Stop_request (Shaft_ID=S1, Floor_name=4)

A Cabin panel stop button should behave according to the state chart shown below. When a button in the "RELEASED" state detects a click, it should light up and generate the Stop_request command to the application. Subsequent clicks will be ignored while in the PRESSED state. Upon receiving the Clear_stop_request command from the application, the button light enters the RELEASED state and turns itself off.



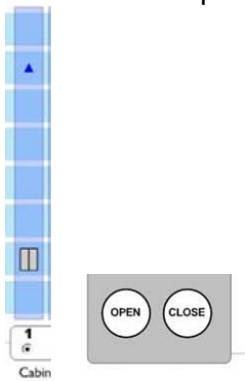
4

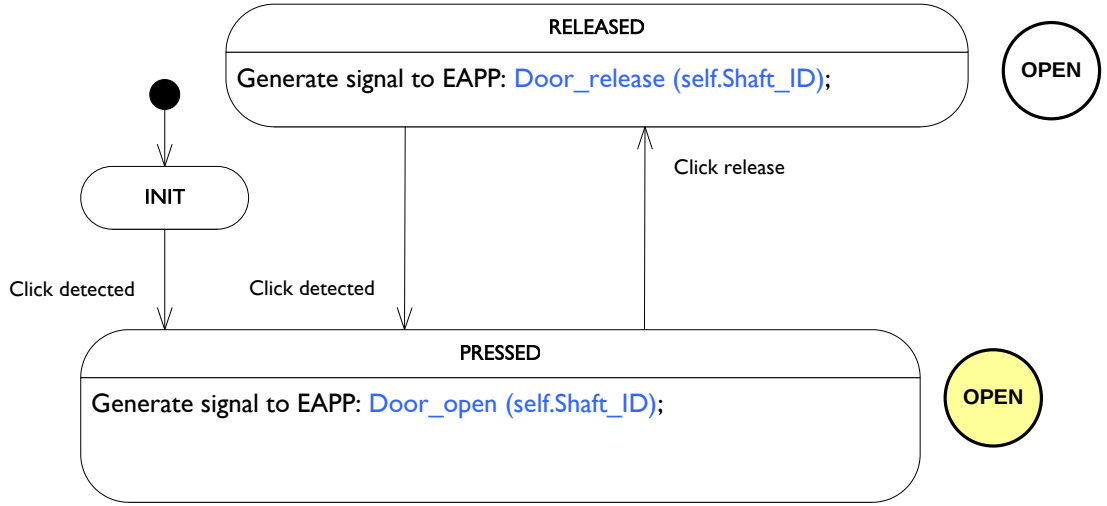
4

Click detected events should be ignored in the PRESSED state.

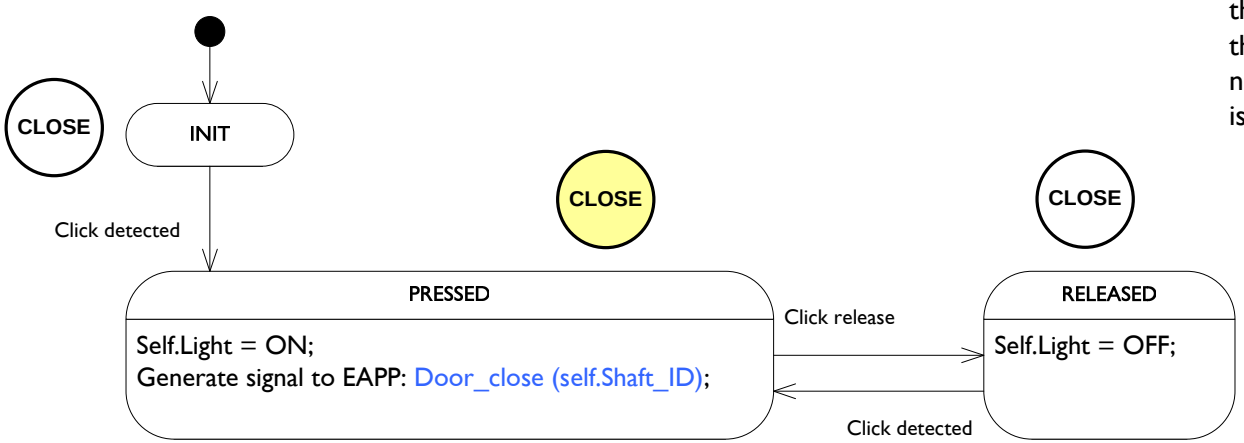
Door Open/Close Buttons

Door Open button behavior





Door Close button behavior

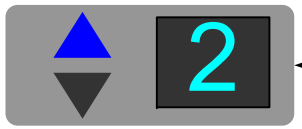


The Door Open and Close buttons look the same, but behave differently. When the Door Open button is pressed, it can be HELD to keep the doors open. SO the button remains pressed until the user lets go with the "click release" input. The elevator application is notified BOTH when the button is pressed and when it is released.

In contrast, the Door Close button has no useful hold function. So the application is signaled only when the button is pressed. The click release input serves only to turn the button light off.

Status info in the Cabin

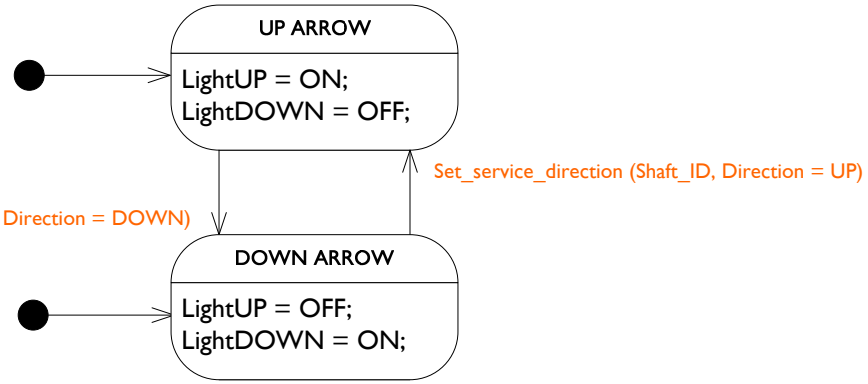
The service direction consists of two lights inside the Cabin. The application will initialize each shaft in either the up or down direction. This setting will be toggled periodically for each Shaft. The current cabin location is represented with a text field. The application will supply a name for the field as the cabin moves.



Cabin_at_floor (Shaft_ID, Floor_name)

Just display the Floor_name

The service direction will be initialized as either up or down.



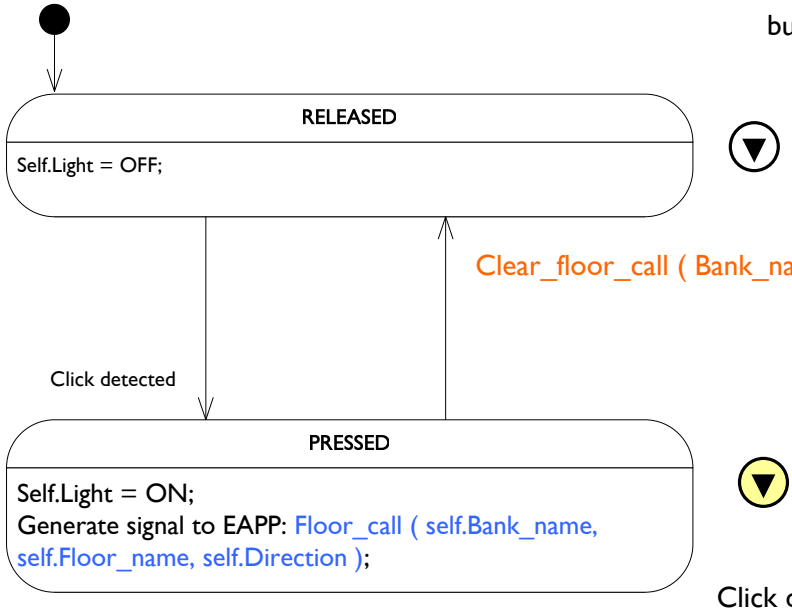
Buttons in the Bank View

4 

Floor_call (Bank_name, Floor_name, Direction)

Depending on the floor, there will be either an up button, down button, or both up and down buttons. These buttons work the same way as the cabin view buttons.

Behavior is the same for both the up and down buttons. (Similar to the cabin floor stop buttons).



Click detected events should be ignored in the PRESSED state.

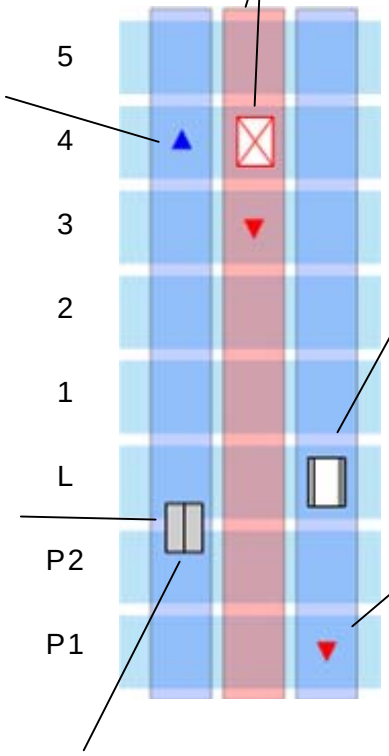
Icons in the Bank View

Shaft_outofservice (Shaft_ID)

The application may decided to take a shaft out of service, for example, if a door is stuck. The GUI should designate the shaft as out-out-of-service and lock out the Cabin control panel so that stop requests and door open/close commands are unavailable for the associated Cabin.

Cabin_arrived (Shaft_ID)

When a Cabin reaches its destination, the application will notify the GUI. The arrow destination/service direction icon should be removed from the shaft symbol. (At this point the Cabin icon will be there).



Doors_opened (Shaft_ID)

When the application successfully opens the doors of a cabin, the GUI is notified. Ideally, the GUI will depict door opening with a little animation. But a simple change of icon will be sufficient.

Doors_closed (Shaft_ID)

See note at Doors_opened

Set_service_direction (Shaft_ID, Direction)

Set_destination (Shaft_ID, Floor_name)

The arrow icon conveys two pieces of information. When the GUI receives a Set_destination command from the application, it should create an arrow icon that points in the same direction as the one in the associated Cabin indicating the service direction. This arrow icon should be placed at the specified shaft-floor intersection. When the Set_service_direction command is received, both the arrow in the cabin view AND the arrow, if any, in the bank view, for the specified Shaft, will be changed to the specified direction. If there is currently no destination set for the shaft, then no action is taken on the bank view.

Update_cabin_position (Shaft_ID, Position)

The numeric position of the Cabin icon should update to the specified floor when this command is received from the application. This should match the floor name displayed in the Cabin View. The position will be a number in motor transport coordinates. The GUI must map this value to the window coordinates. This makes it possible to smoothly move the cabins up and down from floor to floor to reflect the actual accelerated/ decelerated speed computed by motor transport.

Initialization (part 1)

The GUI must be accommodate a variable number of shafts, banks, cabins, etc. So during initialization, before the GUI can draw anything, it must find out how many of everything to display.

There are many ways to implement this initialization. The GUI could read an XML file during startup, or it could initialize a dialog with the application, receive a network stream, whatever. Before solving this problem, we should at least agree on what information will be read regardless of format, protocol or implementation.

I have organized the data into some logical records and we can sort out the implementation later. For each record definition, I show an example of what the resulting instances might look like. Here we start with the Banks and Floors record. This makes it possible for the GUI to figure out the size and number of Bank Views that will be required.

The relative numeric position (motor transport coordinates) expressed in each floor_spec make it possible for the GUI to notice a physical gap between floors as is the case in the express bank. A suitable icon or convention can be used to illustrate this situation.

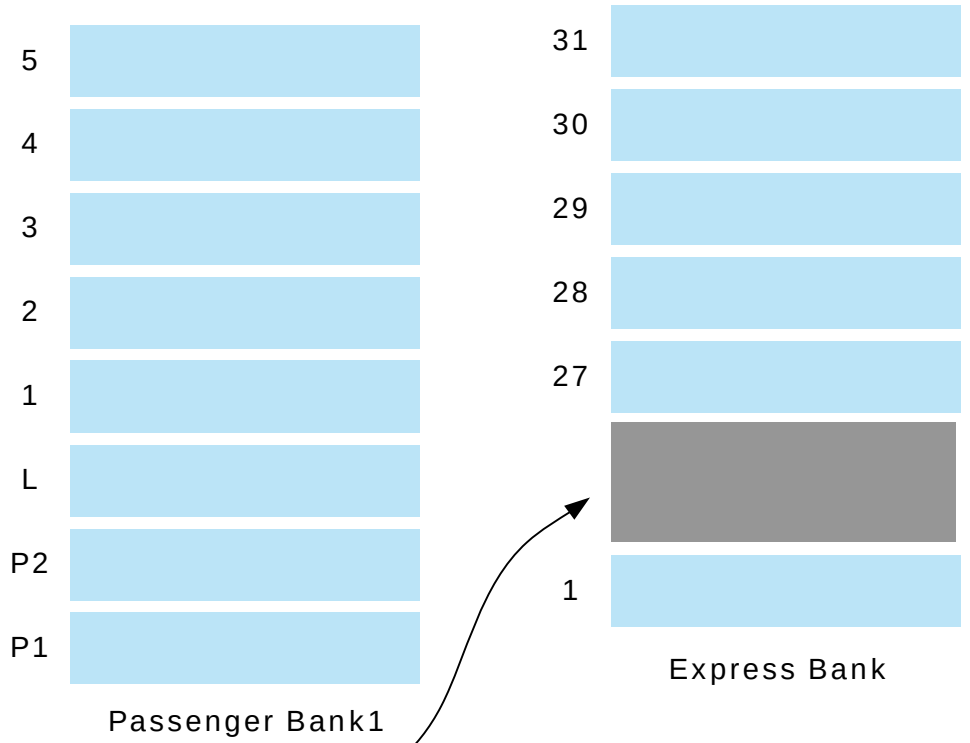
Note: Colored items break down into further detail. Black text is a simple string or numeric value.

Banks and Floors:

bank_list : { bank_name, floor_sequence }

floor_sequence : { bottom_floor_spec, next_highest_floor_spec, … top_floor_spec }

floor_spec : (floor_name, position)



Initialization (part 2)

The following record provides the data about each Shaft. This makes it possible to determine how many shafts are in each bank and to initialize all of the data for each shaft.

The GUI should assume that the topmost floor has one down call button and that the bottom floor has one up call button. All other floors have each a call up and a call down button.

Shaft/Cabin data:

shaft_list : { shaft_ID, service_status, cabin_position, service_direction, door_status, bank_name }