Allison Bellows
Lab 7 Answers

URLs

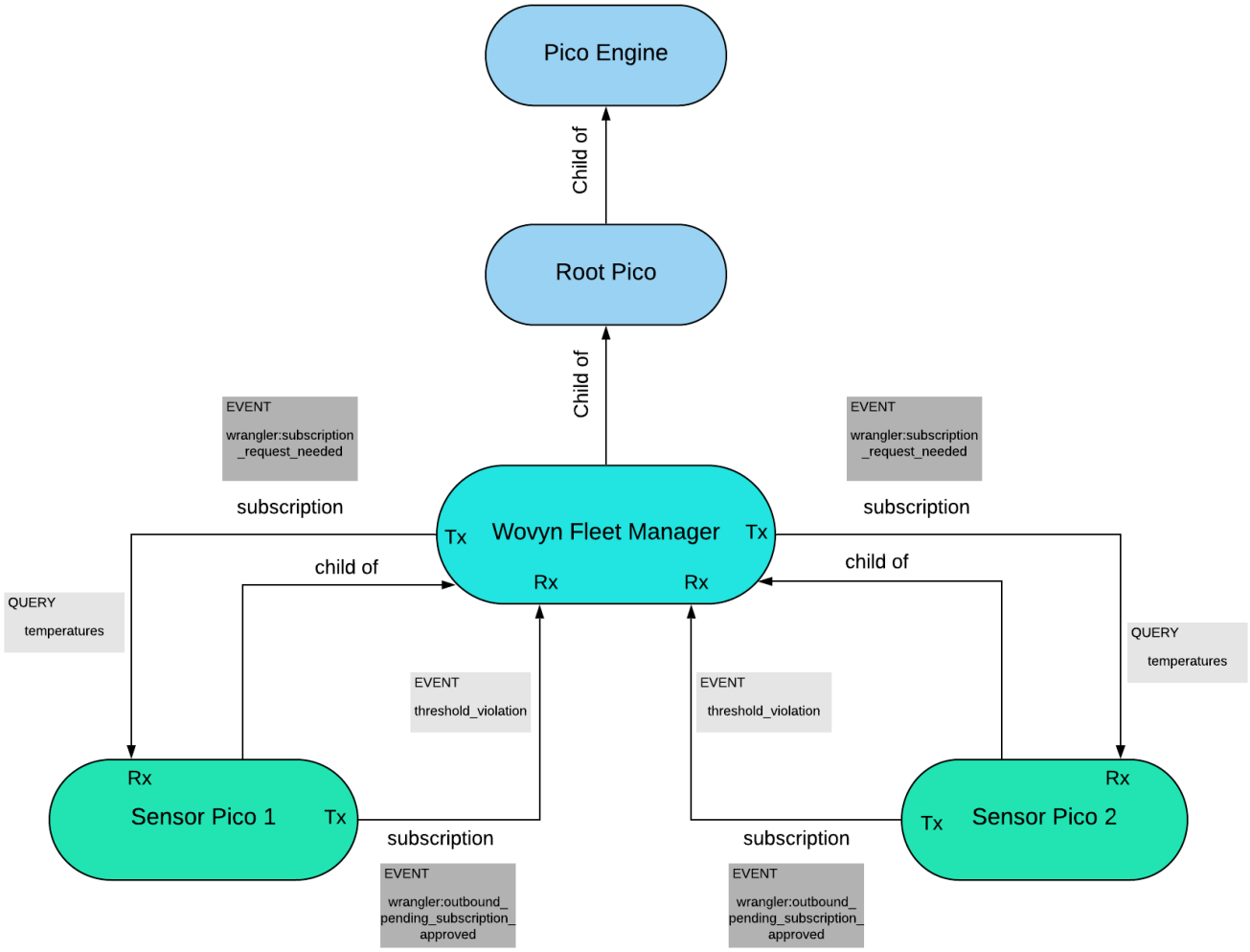
> **New/Modified**
> manage_sensors
> manager_profile
> auto_accept
> wovyn_base

> **Old**
> temperature_store
> sensor_profile

Pico Relationship Diagram

The slightly unappetizing diagram below shows two example sensor picos, as in the instance of connecting both my and a classmate's Wovyn sensors to a manager pico via subscription. (This is a tad visually redundant because both are treated the same by the manager pico, but I figured I would demonstrate understanding of the actual setup introduced by the lab steps).

Dark gray boxes show events sent between picos after an initial `wrangler/subscription` event has been raised. These are part of the inner workings of the subscription ruleset. Once a loop around the directional arrows has been made with the shown dark gray events, the subscriptions have been established as shown. At this point, the events/queries in the light gray boxes (which show explicit events/queries I was sending in this lab) can be successfully sent along those same directional arrows.



Answers to Questions

#1

Auto-approval rules for subscriptions are insecure because if a pico accepts all subscription requests, anyone with access to that pico's ECI can control it with their own picos. In fact, not only can they access that pico's rulesets, but they can install and use their own rulesets. This is obviously insecure because bad actors can use an auto-accept pico for all kinds of uses beyond its intended purposes, including accessing private data.

#2

Yes. You can take another pico with the `manage_sensors` ruleset isntalled and use that ruleset to introduce it to the sensor pico via a new (unique) subscription with a managing Rx_role (I call this role "controller"). Then the sensor pico will have multiple subscriptions whose Tx_roles are "controller" and Rx_roles are "sensor". There is nothing in the current code preventing this.

#3

I could still use one sensor controller pico, but it could have many subscriptions with subscription roles that specify the sensor type. E.g. Some Tx_roles are "temperature_sensor", others are "humidity_sensor", and so on. Or better, given a large amount of sensors, sensor types, and/or unique functionality per sensor type, I would consider having a sensor controller pico for each sensor type and a fleet controller pico that controls all

those controller picos (3 layers total). Under that model, it would be the subscription roles between the sensor controllers and the fleet controller that most crucially specify the sensor type.

#4

As in #3, I could use subscription roles or multiple layers of controllers. Because the functionality for picos on different floors is likely to be nearly the same, I would default to the former. If I knew there would be just one sensor per room, for example, I *could* use subscription names instead of roles, but this would be less scalable.

With this method, I would have a sensor controller pico with subscriptions to all picos in the building; its subscription Tx_roles would specify the sensor location in some subsettable way. (e.g. its subscription to a sensor in room 120 might have a Tx_role "sensor_120"). Then the controller could query specific floors or areas by filtering its subscriptions on Tx_role just as easily as it could query the whole building. (e.g. a temperature-query function could take in an array of location numbers and loop through that array, each i^{th} iteration sending a sky query to all subscriptions with Tx_role "sensor_*i* ", thus building the data structure of overall temperatures to return. To query temperature info from the first floor, one could call this function with the array [100-199]).

#5

Yes! The beauty of heterarchy. In *my* rule handling the `wovyn/threshold_violation` event, I used a `foreach` statement that filters the sensor picos' subscriptions for Tx_roles of "controller", and loops over these subscriptions sending `manager/threshold_violation` events to each. Manager picos use the `twilio:sms` function and stored manager profile info to send a text alert about the violation. Therefore, in my system, if a sensor belongs to more than one collection and has a threshold violation, all of its managing picos will receive a `manager/threshold_violation` event and send a text alert.

Of course, the lab spec did say singular "sensor management pico" in step 6. My guess was that this was unintentional, if not vague. But if I wanted to enforce that a sensor pico only notifies the manager of *one* collection, I could replace my `foreach` implementation with a single event-raising by using `.first()` on the pico's Tx-filtered subscriptions.

#6

I created a new ruleset called `manager_base`. I did this because I wanted to isolate access to private keys, and the ruleset that actually accesses the `twilio:sms` function needs to use my secret set of twilio keys that I store privately in AWS. I could have put it in the `manager_profile` or `manage_sensor` rulesets, but since most of the functionality in those rulesets is separate from the actual text-sending (and Twilio-accessing) functionality, it would be both unnecessary coupling and a theoretical security risk to do so.

#7

I just added one new rule, and modified one other rule. I added the `threshold_notification` rule to my `manager_base` ruleset, and I modified the `threshold_notification` rule in my `wovyn_base` ruleset to raise the former rule to manager picos. I did it this way (and without modifying the `find_high_temps` rule in the `wovyn_base` ruleset) because I think it keeps my rulesets scalable for added functionality, changed functionality, and changed profile info without too much coupling.

`find_high_temps` will always do the same thing - send a `threshold_violation` event if the temp is high enough, which raises the `wovyn/threshold_notification` rule. In the future if I wanted to add more functionality given a high temp, I could create a separate rule that is selected on `wovyn/threshold_violation` events. If I wanted to change functionality within the notification process, I could change the `threshold_notification` rule in either the manager or sensor ruleset (either `manager` or `wovyn` domain), depending on how local the change is to that specific sensor. If I wanted to change notification profile info, I can change it either locally in `sensor_profile` or generally in `manager_profile`, and modify either `wovyn_base` or `manager_base`, respectively, if I needed to specially handle that change (e.g. manager could let sensor profile phone number override "default" manager profile phone number).