Action Step Best Practices

Design Tips

When designing SFCs, the Action step will be the primary workhorse of your charts. There are a few tips to keep in mind while writing your scripts for the action step:

Tip 1: Don't Block/Pause/Wait/Sleep
This is the primary rule for SFCs. You want your scripts to run as quickly as possible. This means that you don't want to use any sort of sleep() or wait() call. Pausing or waiting is the job of transactions in an SFC.

Some sorts of blocking is, of course, unavoidable. For example, running a SQL query that takes some time to execute, or writing to a device over a slow radio connection. These sorts of activities are fine, however, this brings us to tip 2.

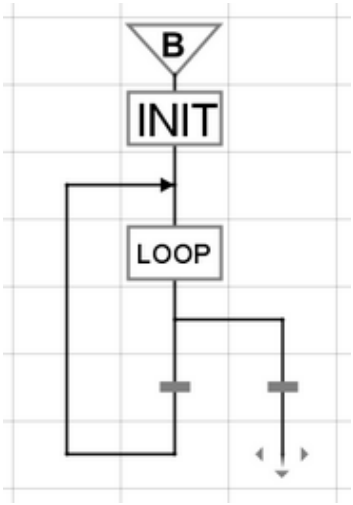
Tip 2: Refractor Loops
This is really just an extension of the don't block rule. Imagine that you have 100 widgets that need processing. Each widget takes some non-trivial amount of time, let's say, 20seconds, to process. The most obvious way to handle this would be with an **On Start** script that had a `while` loop from 1 to 100, processing each widget. This script would take about 33 minutes to run.

Instead of processing the 100 items in a `while` loop, you can solve this problem in two different ways with SFCs:

- Option 1: Timer Action
  In this option, you have a single action step, but instead of having your loop from 1 to 100 in a `while` loop, you initialize a counter to 1 in the On Start action. Then you write a timer action with a rate of 0ms. The timer action processes one widget, and then increments the counter. You place a transition beneath the step whose expression is: `{counter} >= 100`

- Option 2: Chart Loop
  Similar to option 1, you can design the loop in the chart itself. Have one action step do your initialization work, in our example: `chart.counter = 0`

  Have another step do the processing work for one item in its On Start script. Use two transitions, the one on the left set to `{counter} < 100` and the one on the `right` set to `true`. The loop action will run 100 times, and then the flow will continue down the other path.



Rationale

By now you should understand that you want to keep your individual script duration to a minimum. You may be wondering why this is so important. After all, in the example above, it still takes 33 minutes to complete the given work after refactoring the loop as shown.

The reason this is important is to support pausing, persistence, and redundancy. Charts can only be paused after any running script finishes. Similarly, a chart's state only gets synchronized with the redundancy system between script executions. If you had a script that took half an hour to run, you couldn't pause that chart until the script ended, and the redundancy system would be very out of date and not able to recover as well if the Gateway went offline.

As a bonus, breaking your work up into smaller units helps make the chart easier to debug and more visible for monitoring.

Notes ...