Why SKAO does not want to use Tango's polling loop

Tango Polling Loop SIG workshop

Thomas Ives and Thomas Juerges

2026-04-14

Outline

SKAO's problems with the polling loop

a) Our commissioners are not Tango experts
b) It's just not fair
c) The algorithm has strict timing requirements

SKAO's desires for a new polling loop

a) We often don't want to poll
b) Decoupling responsibilities
c) Acquiring data does not have strict timing requirements
d) Mockable time would be nice

SKAO’s problems with the polling loop

a) Our commissioners are not Tango experts

- Successful tuning requires:
  - Knowledge from Tango device development team in e.g. UK
  - Knowledge from commissioning team in e.g. Australia
- The timezone difference makes it hard and slow to tune
- If we get the tuning wrong it can be quite disastrous because…

b) It’s just not fair

- If we don’t allocate enough time, polls get discarded
  - Often for the same work item every time
- This is a disaster because we heavily rely on events in our control system

Opinionated aside: The event we send to the client when we discard a poll is useless. What are you supposed to do with this? This should be reported to the operator by the Tango device instead.

- We run into this a lot at SKAO because…

c) The algorithm has strict timing requirements

- The algorithms (both “before 9” and “after 9”) interpret a specified polling period as a strict requirement to send events at this rate
- We are not using a real time OS
  - The polling loop cannot guarantee this timing requirement
  - In practice, on systems with low CPU load the polling loop manages its aims
  - In practice, on a kubernetes cluster, where we get CPU throttled, the polling loop often fails
- We also see this problem in cppTango CI pipeline

SKAO’s desires for a new polling loop

a) We often don’t want to poll

- We have fancy modern hardware that natively supports event-driven protocols e.g. OPC UA
  - We would like to have our control system be event driven too to take advantage of this
- We see event-driven Tango w/o the polling loop as an afterthought
  - We have to give up the polling buffer
  - push_change_event etc. requiring a lock makes them hard to work with from external threads
  - if I want to be truly event-driven I sometimes don’t want to provide an attribute read function, I just want to give Tango values and have it deal with clients

b) Decoupling responsibilities

- The current polling system does three things:
  1. Manages a data cache to service clients – including sending events
  2. Acquires data for the cache by calling `read_attributes` periodically
  3. Performs periodic tasks such as invoking commands and periodic events
- These should handled by separate subsystems of Tango that can be used independently by each device
- It would nice if we could swap out 2. with whatever we want

c) Acquiring data does not have strict timing requirements

- When we are polling to acquire data, what we mean by “polling period = <x>” is “don’t try to acquire data more often that once every <x>”
- When we use periodic events, what we mean by “polling period = <x>” is “send me an event every <x>”
  - We often wouldn’t mind receiving a cached value
- These cannot be not best served by the same mechanism

d) Mockable time would be nice

- If we are going to rely on the polling loop, it would be nice to be able to mock time for testing
- Not just for cppTango tests but for our SKAO device tests too

Any comments/questions?
Any other issues with the polling loop?

https://confluence.skatelescope.org/display/SEC/Tango+device+server+polling+loop+considered+harmful