Problem

How do on-change monitors work?

Solution

On-change monitors are used to periodically check the value of a BACI property for changes. Basically they are used in the following manner:

1. Before even starting ACS, the developer should change the **min_timer_trig** characteristic of BACI properties he or she is interested in creating on-change monitors for (found somewhere in $ACS_CDB/CDB/alma/*) to some positive non-zero value. This particular characteristic is the polling rate at which the BACI property will eventually poll its value at to determine if it has changed by some delta value since the last time period. Once the property has been created, there is no way to change this behaviour short of editing the CDB, clearing its cache, and then restarting the container.
2. The first optional step is to change the **default_timer_trig** characteristic of BACI properties within the CDB. This defines a span of time in which a property's value must change by some delta value for the on-change monitor to be triggered. To better explain this, an example is in order. Suppose you have an on-change monitor for a ROlong property where the **min_timer_trig** is two seconds, the **default_timer_trig** is ten seconds, and the delta value is 20. This monitor is triggered under two different conditions the first being that a time period equivalent to the **default_timer_trig**, ten seconds, has passed. Looking at this behaviour, it's identical to the way *normal* monitors work and is not too interesting. However, what's going on behind the scenes is that once every two seconds within that ten second duration the property's value is being retrieved. If the value changes by the delta value, 20, between the beginning of the ten second duration and any time before the end of the ten second duration, the monitor is triggered early. User can check if the value was sent due the value had been changed or timer had expired by examining the type and code value of associated callback value:
    - If type is **`ACSErrTypeMonitor`** (1) and code is **`ACSErrMonitorOnTimer`** (0) than value was sent because the timer had expired.
    - If type is **`ACSErrTypeMonitor`** (1) and code is **`ACSErrMonitorOnValue`** (1) than the value was sent because the value had been changed.
3. The second optional step is to modify the **min_delta_trig** characteristic of BACI properties within the CDB. This makes it impossible at run-time to set a delta value for an on-change monitor smaller than the value of **min_delta_trig**.
4. A developer creates a monitor for a specific BACI property via one of it's IDL methods (typically something like **create_monitor**)
5. Using the **Monitorxyz** interface returned by the BACI property, the developer can *turn on* on-change monitors by utilizing the **set_value_by_trigger** method. The first parameter to this method would be the delta value that was spoken of earlier and the second value is boolean true.
6. To turn off on-change monitors, the developer simply invokes **set_value_by_trigger** again, this time passing boolean false as the second parameter.

- The delta value is ignored for Pattern monitors. Any change from one **min_timer_trig** epoch to the next will trigger the monitor.
- The delta value is ignored for String monitors. Any change from one **min_timer_trig** epoch to the next will trigger the monitor.

Related articles

- How can more people do development with ACS on the same machine without disturbing each other?
- Which ports are used by ACS?
- Problems connecting to ACS servers on a remote machine: bad /etc/hosts
- Why does the getComponent method of ZLegacy/ACS.ContainerServices return an object of type None?
- Why are some of my print statements not showing up in the container output section of acscommandcenter?