Problem

Why isn't the channel also destroyed when I invoke Supplier/SimpleSupplier's disconnect method?

Solution

Short Answer:

Please refer to the doxgyen-generated documentation or pydoc for the Supplier/SimpleSupplier class and you will find that the *disconnect* method does not claim to destroy the notification channel it's connected to. Use the protected *destroyNotificationChannel* method instead if you really need to do this.

Detailed Explanation:

Some developers seem to be confused by this probably because they do not realize the underlying notification channel object not only does not reside in their supplier object's process space; it is most likely is located on another PC entirely. Also, *if their supplier object was the first to try to connect to the channel it will create it and then shouldn't it destroy the channel after disconnecting?* The answer to this is no and here's why - one of the very first subjects that comes up in the ACS Notification Channel tutorial is that the ACS Notification Channel framework supports a many-to-many publishing /subscribe mechanism (this is in the ACS Architecture document as well). Logically, how could this be the case if one of N suppliers destroys the channel out from under the other (N-1) suppliers once it's done using the channel?

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?