Problem

How and why does the Python getComponent ZLegacy/ACS.ContainerServices method automatically narrow components?

Solution

ACS has chosen to use *omniORBPy* as the Python ORB ALMA uses. This decision actually has a lot of positive implications and one of them is that under **most** circumstances generic **`CORBA.Object`** references do not need to be narrowed to be used. To find out when **`CORBA.Objects`** must be narrowed, take a look at http://omniorb.sourceforge.net/omnipy2/omniORBpy/omniORBpy003.html#toc10. To be completely safe though, it is highly recommended that **`CORBA.Object`** references be narrowed to their correct type.

All of this sounds fine, but what does this have to do with **my** calls to the **`getComponent`** `ZLegacy/ACS.ContainerServices` method from **my** code ? Well in a never-ending effort to hide CORBA from ALMA developers, the Python **`getComponent`** method tries to automatically import the CORBA stubs for the component (this would be required from your code if **`getComponent`** did not take care of it!) **and** narrow it to the correct type. Immediately a few questions arise from this:

How does `getComponent` determine which IDL interface the component should be narrowed to?

If it's a dynamic component, the IDL type (typically something like "IDL:/alma/someModule/SomeInterface:1.0") **must** be provided by you using the **`comp_idl_type`** keyword parameter. If it's a "normal" component, **`getComponent`** queries manager for the component's IDL type and manager pulls this information up from **`$ACS_CDB/CDB/MACI/Components/`**. **WARNING: if the CDB entry specifies the wrong IDL type, `getComponent` will not be able to narrow the component!**

How does `getComponent` determine which CORBA stub to import?

Once **`getComponent`** has determined the component's IDL type (see previous question), it quite simply assumes that the IDL type follows the standard convention of placing IDL interfaces within a single IDL module. Keeping this and the IDL->Python mapping in mind, it assumes that the second to last alphabetic name in the IDL type is the name of the module and the last alphabetic name in the IDL type is the name of the IDL interface.

What can go wrong?

Unfortunatley lots:

- quite often it's the case that developers incorrectly assume there is something wrong with **`getComponent`** when it cannot retrieve a reference to a component for some reason. More often than not, this is caused because the component name provided is incorrect, the container responsible for the component you're trying to access has not been started or has died trying to activate the component, etc. The situation is made even worse because the Manager IDL interface does not throw remote exceptions describing why it could not give **`getComponent`** a valid reference. Instead, it simply returns a *Nil* CORBA reference which obviously cannot be narrowed to the correct type.
- if the IDL type in the ACS CDB is defined incorrectly or the wrong IDL type is specified as a keyword parameter to **`getComponent`**, it should be immediately apparent that the reference cannot be narrowed
- the IDL type is specified correctly, but the CORBA stubs cannot be imported (see ACS FAQ). One such case is when you've given an IDL module a name identical to a Python keyword (e.g., *exec*). To alleviate this specific problem one must either:
    - depend on omniORB's ability to automatically narrow CORBA references. To do this import the Python CORBA stubs for *exec* before calling *getComponent*. Regardless you will still see an error log because of *getComponents* inability to narrow the interface.
    - narrow the **`CORBA.Object`** returned by *getComponent* to it's correct type yourself. See OMG's IDL->Python mapping document to understand how this is done. Once again, you'll still see the error coming from *getComponent*.

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?