Problem

I am having problems with macros in IDL

Solution

The JacORB IDL compiler **does not support preprocessor macros**.

Summary: Always use the **`.midl`** extension for idl files containing macros instead of **`.idl`**

Read below for a detailed explanation:

The JacORB IDL compiler **does not support preprocessor macros**.

This is a bad limitation of the IDL compiler (in our opinion not acceptable).

With ACS 3.1 we have introduced a separate pre-processing step for idl files containing macros.

You now have to put IDL code using macros in files with the **`.midl (macro-idl)`** extension and add the file in the **`IDL_FILES`** section of the Makefile.

The Makefile will automatically expand the macros generating a normal **`.idl`** file and treat it normally, eventually installing it.

For example (see the module ACS/LGPL/CommonSoftware/enumprop), if you create the file myEnumprop.midl:

```
#ifndef _MYENUMPROP_MIDL
#define _MYENUMPROP_MIDL

#include <baci.idl>
#include <enumpropMACRO.idl>

module TEST {
  enum Bool { testFALSE, testTRUE};
  ACS_ENUM(Bool);
};

#endif
```

the Makefile will generate myEnumprop.idl with the macros expanded that you can use in any other file and that will load in the interface repository without troubles.

Before ACS 3.1, to workaround this problem, the ACS Makefile was just preprocessing IDL files with the standard CPP pre-processor before passing them to the JacORB IDL compiler.

This generally worked, but could still lead to problems in special cases.

We have in particular discovered (see SPR ALMASW2003048) that if an IDL file contianing macros is included twice, the JacORB IDL compiler does not behave correctly.

What happens is that the second time the file is included:

1. JacORB encounters the include file guard define that should prevent examining the code for the second time.
2. The parser starts searching for the #endif
3. To do this, it tries to parse all other CPP directives in the file
4. It fails when it encounters the macro, because it cannot handle the redefinition

An easy workaround is to add one layer of include files to make sure JacORB does not try to parse the macros.

For example:

- The original enumpropMACRO.idl was:

  ```
  #ifndef _ENUMPROP_MACRO_IDL_
  #define _ENUMPROP_MACRO_IDL_

  .... macros .......

  #endif /* ENUMPROP_MACRO_IDL_ */
  ```

- Now we have the following enumpropMACRO.idl:

```
#ifndef _ENUMPROP_MACRO_IDL_
#define _ENUMPROP_MACRO_IDL_

#include "enumpropMACRO_included.idl"

#endif /* ENUMPROP_MACRO_IDL_ */
```

- and enumpropMACRO_included.idl is:

```
#ifndef _ENUMPROP_MACRO_INCLUDED_IDL_
#define _ENUMPROP_MACRO_INCLUDED_IDL_
.... here the same macros previously in enumpropMACRO.idl

#endif /* ENUMPROP_MACRO_INCLUDED_IDL_ */
```

- This ensures that JacORB IDL compiler never has to parse the macros twice.

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?