# autoconf - package for CORBA support

Ruslan Shevchenko
< *Ruslan@Shevchenko.Kiev.UA* >

August 3, 2000

## 1 Introduction

This is autoconf package for CORBA support.

## 2 autoconf macroses

- `RSSH_CHECK_ORB` - check supported CORBA ORB and define appropriative substitution macroses and preprocessor symbols.
- `RSSH_CHECK_ORBACUS` - check for ORBacus
- `RSSH_CHECK_TAO` - check for TAO
- `RSSH_CHECK_OMNIORB` - check for omniORB

## 3 Makefile.in variables

- ORB Define the name of ORB. Possible names for now are:
  1. TAO
  2. OmniORB
  3. ORBacus
- IDL (deprecated), IDLCXX Define the name of IDL compiler. IDLCXX compiler is used to produce C++ stubs and skeletons. It's defined as:
  1. TAO `$ACE_ROOT/tao/TAO_IDL/taoidl`
  2. OmniORB3 `omniidl -bcxx -I$(OMNI_ROOT)/idl`
  3. ORBacus `idl -I$(OB_PREFIX)/idl -I$(OB_PREFIX)/idl/OB`

  when idl compiler process input file, it's produce few output files:
- – client stub headers,
  – some orb generate additiona helper file for stub headers,

- C++ sources for client stub,
- server skeleton headers
- some ORB generate additional server skeleton headers
- C++ sources for client skeleton

So, the next set of autoconf macroses are defined:

- `IDL_CLN_H_SUFFIX` - suffix for generated client stub headers. (i. e for X.idl client stub header file is `|IDL_CLN_H_SUFFI*`, where `|` – concatenation symbol.
- `IDL_CLN_H1_SUFFIX` – suffix for additional generated client stub headers, or *no*, if one not exists.
- `IDL_CLN_CPP_SUFFIX` – suffix for client stub C++ sources.
- `IDL_CLN_OBJ_SUFFIX` – suffix for client stub object file.
- `IDL_SRV_H_SUFFIX` – suffix for server skeleton header
- `IDL_SRV_H1_SUFFIX` – suffix for additional server skeleton header, or *no* if one not exists
- `IDL_SRV_CPP_SUFFIX` – suffix for server skeleton C++ sources.
- `IDL_SRV_OBJ_SUFFIX` – suffix for server skeleton object file
- `IDL_LIBDIR` – linker options for setting directory of ORB libraries ( something like `-L/opt/ACE/TAO/lib` )
- `ORB_LIBS` – linker options for ORB libraries.
- `ORB_COSNAMING_LIB` - library for CORBA Naming Service. (General rule: for standart CORBA service XX, if ORB implement this service, than `ORB_XX_LIB` is defined.

# 4 Preprocessor variables

1. `CORBA_MODULE_NAMESPACE_MAPPING` defined, when IDL modules are mapped to C++ namespaces.
2. `CORBA_MODULE_CLASS_MAPPING` defined, when IDL modules are mapped to C++ classes.
3. `CORBA_MODULE_C_MAPPING` defined, when IDL modules are mapped to identifiers as in C case.
4. `CORBA_HAVE_POA` defined, when ORB is POA based.
5. `CORBA_HAVE_POA` defined, when ORB is POA based.

6. `CORBA_ORB_INIT_HAVE_3_ARGS` defined, when `CORBA::ORB_init` accept 3 arguments.
7. `CORBA_ORB_INIT_THIRD_ARG` – third argument of `CORBA::ORB_init` The common usage of last 2 macro variables shown in next code snipshet:

```
#ifdef CORBA_ORB_INIT_HAVE_3_ARGS
  CORBA::ORB_var orb = CORBA::ORB_init(argc,argv,CORBA_ORB_INIT_THIRD_ARG);
#else
  CORBA_ORB_var orb = CORBA::ORB_init(argc,argv);
#endif
```

8. `CORBA_SYSTEM_EXCEPTION_IS_STREAMBLE` defined, when operator:

   ```
   ostream& operator<<(ostream&, const CORBA::SystemException&)
   ```

   is defined by ORB.
9. `CORBA_H` – header file for CORBA definitions. common usage:

   ```
   #include CORBA_H
   ```

10. `COSNAMING_H` – header file with definitions of CORBA CosNaming service.
11. `IDL_CLN_H_SUFFIX` – suffix of generated client stubs header.
12. `IDL_SRV_H_SUFFIX` – suffix of generated skeleton header.
    The tupical usage of this symbols are next:

```
#ifndef __CAT2_FF
#define __CAT2_FF(x,y) <##x##y##>
#endif

#ifndef __CAT2_F
#define __CAT2_F(x,y) __CAT2_FF(x,y)
#endif

#define CORBA_STUB_HEADER(x)  __CAT2_F(x,IDL_CLN_H_SUFFIX)
#define CORBA_SKELETON_HEADER(x)  __CAT2_F(x,IDL_SRV_H_SUFFIX)
```

from now we can write:

```
#include CORBA_STUB_HEADER(X)
```

for inclusion of corba client stub header in C++ code.