PolyORB User's Guide

Version GPL 2007-20070405
Date: 20 April 2007

Jérôme Hugues, Thomas Quinot

Table of Contents

About This Guide 1
What This Guide Contains 1
Conventions 2

1 Introduction to PolyORB 3
1.1 Distributed applications and middleware 3
1.2 PolyORB a generic middleware with an instance per distribution model 3

2 Installation 5
2.1 Supported Platforms 5
2.2 Build requirements 5
2.3 Build instructions 5
2.4 Additional instructions for cross platforms 6
2.5 Building the documentation and PolyORB's examples 6
2.5.1 Build Options 6
2.5.2 Compiler, Tools and Run-Time libraries Options 7
2.6 Platform notes 7

3 Overview of PolyORB personalities 9
3.1 Application personalities 9
3.1.1 CORBA 9
3.1.2 Distributed System Annex of Ada (DSA) 9
3.1.3 Message Oriented Middleware for Ada (MOMA) 9
3.1.4 Ada Web Server (AWS) 9
3.2 Protocol personalities 9
3.2.1 GIOP 10
3.2.2 SOAP 10

4 Building an application with PolyORB 11
4.1 Compile-time configuration 11
4.1.1 Tasking run-times 11
4.1.2 Middleware tasking policies 11
4.1.3 Sample files 11
4.2 Run-time configuration 12
4.2.1 Using a configuration file 12
4.2.2 Using environment variables 12
4.2.3 Using the command line 12
4.3 Setting up protocol personalities 13
4.3.1 Activating/Deactivating protocol personalities 13
4.3.2 Configuring protocol personality preferences 13
4.4 Activating debug information 13

4.5 Tracing exceptions 14
4.6 `polyorb-config` 14

5 Tasking model in PolyORB 17
5.1 PolyORB Tasking runtimes 17
5.1.1 Full tasking runtime 17
5.1.2 No tasking runtime 17
5.1.3 Ravenscar tasking runtime 17
5.2 PolyORB ORB Tasking policies 19
5.2.1 No Tasking 19
5.2.2 Thread Pool 19
5.2.3 Thread Per Session 19
5.2.4 Thread Per Request 19
5.3 PolyORB Tasking configuration 19
5.4 PolyORB ORB Controller policies 20
5.4.1 No Tasking 20
5.4.2 Workers 20
5.4.3 Half Sync/Half Async 20
5.4.4 Leader/Followers 20
5.5 PolyORB ORB Controller configuration 20

6 CORBA 23
6.1 What you should know before Reading this section 23
6.2 Installing CORBA application personality 23
6.3 Usage of `idlac` 23
6.4 Resolving names in a CORBA application 24
6.4.1 `po_cos_naming` 24
6.4.2 Registering the reference to the COS Naming server 25
6.4.3 Using the COS Naming 25
6.5 The CORBA Interface Repository 25
6.5.1 `po_ir` 25
6.5.2 Using the Interface Repository 25
6.6 Building a CORBA application with PolyORB 25
6.6.1 `echo` example 25
6.6.1.1 IDL definition of an `echo` object 26
6.6.1.2 Implementation code for the `echo` object 26
6.6.1.3 Test code for client and server nodes 28
6.6.1.4 Compilation and execution 32
6.6.2 Other examples 32
6.7 Configuring a CORBA application 33
6.7.1 Configuring PolyORB 33
6.7.2 Configuring GIOP protocol stack for PolyORB 33
6.7.3 Configuring Security services for PolyORB 33
6.7.3.1 Supported mechasnisms 33
6.7.3.2 Compile-time configuration 33
6.7.3.3 Run-time configuration 33
6.7.4 Command line arguments 36
6.8 Implementation Notes 36

6.8.1 Tasking . . . 36
6.8.2 Implementation of CORBA specifications . . . 36
6.8.3 Additions to the CORBA specifications . . . 36
6.8.4 Interface repository . . . 36
6.8.5 Policy Domain Managers . . . 36
6.8.6 Mapping of exceptions . . . 37
6.8.7 Additional information to `CORBA::Unknown` . . . 37
6.8.8 Internals packages . . . 37
6.9 PolyORB's specific APIs . . . 37
6.9.1 `PolyORB.CORBA_P.CORBALOC` . . . 39
6.9.2 `PolyORB.CORBA_P.Naming_Tools` . . . 40
6.9.3 `PolyORB.CORBA_P.Server_Tools` . . . 42

7 RT-CORBA . . . 45
7.1 What you should know before Reading this section . . . 45
7.2 Installing RT-CORBA . . . 45
7.3 Configuring RT-CORBA . . . 45
7.3.1 `PolyORB.RTCORBA_P.Setup` . . . 45
7.4 `RTCORBA.PriorityMapping` . . . 46
7.5 RTCosScheduling Service . . . 47
7.5.1 Overview . . . 47
7.5.2 `RTCosScheduling::ClientScheduler` . . . 47
7.5.3 `RTCosScheduling::ServerScheduler` . . . 47

8 Ada Distributed System Annex (DSA) . . . 49
8.1 What you should know before Reading this section . . . 49
8.2 Installing DSA application personality . . . 49
8.3 A small example of a DSA application . . . 49
8.4 Building a DSA application with PolyORB . . . 50
8.4.1 Foreword . . . 50
8.4.2 Installing po_gnatdist . . . 50
8.4.3 Using po_gnatdist with PolyORB . . . 50
8.5 Running a DSA application . . . 50
8.6 Configuring a DSA application . . . 51

9 MOMA . . . 53
9.1 What you should know before Reading this section . . . 53
9.2 Installing MOMA application personality . . . 53
9.3 Package hierarchy . . . 53

10 Ada Web Server (AWS) . . . 55

11 GIOP 57
11.1 Installing GIOP protocol personality 57
11.2 GIOP Instances 57
11.2.1 IIOP 57
11.2.2 SSLIOP 57
11.2.3 DIOP 57
11.2.4 MIOP 57
11.3 Configuring the GIOP personality 57
11.3.1 Common configuration parameters 58
11.3.2 IIOP Configuration Parameters 58
11.3.3 SSLIOP Configuration Parameters 59
11.3.3.1 Ciphers name 59
11.3.3.2 SSLIOP Parameters 59
11.3.4 DIOP Configuration Parameters 60
11.3.5 MIOP Configuration Parameters 61
11.4 Code sets 61
11.4.1 Supported code sets 62
11.4.2 Incompatibility in code set support 62
11.4.3 Adding support for new code sets 62
11.4.4 Character data Converter 62
11.4.5 Converters factories 63
11.4.6 Registering new code sets 63

12 SOAP 65
12.1 Installing SOAP protocol personality 65
12.2 Configuring the SOAP personality 65

13 Tools 67
13.1 `po_catref` 67
13.2 `po_dumpir` 67
13.3 `po_names` 67

Appendix A Performance considerations 69

Appendix B Conformance to standards 71
B.1 CORBA standards conformance 71
B.1.1 CORBA IDL-to-Ada mapping 71
B.1.2 CORBA Core 71
B.1.3 CORBA Interoperability 72
B.1.4 CORBA Interworking 72
B.1.5 CORBA Quality Of Service 72
B.1.6 CORBA COS Services 72
B.1.7 CORBA Specialized services 72
B.2 RT-CORBA standards conformance 73
B.3 CSIv2 standards conformance 73
B.4 CORBA/GIOP standards conformance 73
B.5 SOAP standards conformance 74

Appendix C References **75**

Appendix D GNU Free Documentation License **77**

Index **83**

About This Guide

This guide describes the use of PolyORB, a middleware that enables the construction of Ada 95 distributed applications.

It describes the features of the middleware and related APIs and tools, and details how to use them to build Ada 95 applications.

What This Guide Contains

This guide contains the following chapters:

- Chapter 1 [Introduction to PolyORB], page 3 provides a brief description of middleware and PolyORB's architecture.
- Chapter 2 [Installation], page 5 details how to configure and install PolyORB on your system.
- Chapter 3 [Overview of PolyORB personalities], page 9 enumerates the different personalities, or distribution mechanisms, PolyORB provides.
- Chapter 4 [Building an application with PolyORB], page 11 presents the different steps to build a distributed application using PolyORB.
- Chapter 5 [Tasking model in PolyORB], page 17 details the use of tasking constructs within PolyORB.
- Chapter 6 [CORBA], page 23 describes PolyORB's implementation of OMG's CORBA.
- Chapter 7 [RT-CORBA], page 45 describes PolyORB's implementation of RT-CORBA, the real-time extensions of OMG's CORBA.
- Chapter 8 [DSA], page 49 describes PolyORB's implementation of the Ada Distributed System Annex.
- Chapter 9 [MOMA], page 53 describes PolyORB's implementation of MOMA, the Message Oriented Middleware for Ada.
- Chapter 10 [AWS], page 55 describes the integration of the Ada Web Server (AWS) framework into PolyORB.
- Chapter 11 [GIOP], page 57 describes PolyORB's implementation of GIOP, the protocol defined as part of CORBA.
- Chapter 12 [SOAP], page 65 describes PolyORB's implementation of SOAP.
- Chapter 13 [Tools], page 67 describes PolyORB's tools.
- Appendix B [Conformance to standards], page 71 discusses the conformance of the PolyORB's personalities to the CORBA and SOAP standards.
- Appendix C [References], page 75 provides a list of useful references to complete this documentation.
- Appendix D [GNU Free Documentation License], page 77 contains the text of the license under which this document is being distributed.

Conventions

Following are examples of the typographical and graphic conventions used in this guide:

- `Functions`, `utility program names`, `standard names`, and `classes`.
- '`Option flags`'
- '`File Names`', '`button names`', and '`field names`'.
- *Variables*.
- *Emphasis*.
- [optional information or parameters]
- Examples are described by text

```
and then shown this way.
```

Commands that are entered by the user are preceded in this manual by the characters "`$ `" (dollar sign followed by space). If your system uses this sequence as a prompt, then the commands will appear exactly as you see them in the manual. If your system uses some other prompt, then the command will appear with the `$` replaced by whatever prompt character you are using.

Full file names are shown with the "`/`" character as the directory separator; e.g., '`parent-dir/subdir/myfile.adb`'. If you are using GNAT on a Windows platform, please note that the "\" character should be used instead.

1 Introduction to PolyORB

1.1 Distributed applications and middleware

PolyORB aims at providing a uniform solution to build distributed applications; relying either on industrial-strength middleware standards such as CORBA, the Distributed System Annex of Ada 95, distribution programming paradigms such as Web Services, Message Oriented Middleware (MOM), or to implement application-specific middleware.

Middleware provides a framework that hides the complex issues of distribution, and offers the programmer high-level abstractions that allow easy and transparent construction of distributed applications. A number of different standards exist for creating object-oriented distributed applications. These standards define two subsystems that enable interaction between application partitions:

- the API seen by the developer's applicative objects;
- the protocol used by the middleware environment to interact with other nodes in the distributed application.

Middleware implementations also offer programming guidelines as well as development tools to ease the construction of large heterogeneous distributed systems. Many issues typical to distributed programming may still arise: application architectural choice, configuration or deployment. Since there is no "one size fits all" architecture, choosing the adequate distribution middleware in its most appropriate configuration is a key design point that dramatically impacts the design and performance of an application.

Consequently, applications need to rapidly tailor middleware to the specific distribution model they require. A distribution model is defined by the combination of distribution mechanisms made available to the application. Common examples of such mechanisms are Remote Procedure Call (RPC), Distributed Objects or Message Passing. A distribution infrastructure or middleware refers to software that supports one (or several) distribution model, e.g.: OMG CORBA, Java Remote Method Invocation (RMI), the Distributed System Annex of Ada 95, Java Message Service (MOM).

1.2 PolyORB a generic middleware with an instance per distribution model

Typical middleware implementations for one platform support only one set of such interfaces, pre-defined configuration capabilities and cannot interoperate with other platforms. In addition to traditional middleware implementations, PolyORB proposes an original architecture to enable support for multiple interoperating distribution models in a uniform canvas.

PolyORB is a polymorphic, reusable infrastructure for building or prototyping new middleware adapted to specific application needs. It provides a set of components on top of which various instances can be elaborated. These instances (or personalities) are views on PolyORB facilities that are compliant to existing standards, either at the API level (application personality) or at the protocol level (protocol personality). These personalities are mutually exclusive views of the same architecture.

The decoupling of application and protocol personalities, and the support for multiple simultaneous personalities within the same running middleware, are key features required for the construction of interoperable distributed applications. This allows PolyORB to communicate with middleware that implement different distribution standards: PolyORB provides middleware-to-middleware interoperability (M2M).

PolyORB's modularity allows for easy extension and replacement of its core and personality components, in order to meet specific requirements. In this way, standard or application-specific personalities can be created in a streamlined process, from early stage prototyping to full-featured implementation. The PolyORB architecture also allows the automatic, just-in-time creation of proxies between incompatible environments.

You may find more information on PolyORB, including technical and scientific papers on PolyORB, on the project websites: `http://libre.adacore.com/polyorb/` and `http://polyorb.objectweb.org/`.

Note: PolyORB is the project formerly known as DROOPI, a Distributed Reusable Object-Oriented Polymorphic Infrastructure

2 Installation

2.1 Supported Platforms

PolyORB has been compiled and successfully tested on the following platforms:

- FreeBSD
- HP-UX
- Linux
- MacOS X
- Solaris
- Windows

Note: PolyORB should compile and run on every target for which GNAT and the `GNAT.Sockets` *package are available.*

2.2 Build requirements

Ada compiler:

- GNAT Pro 5.04a1 or later
- GNAT GPL 2006 or later
- FSF GCC 4.1.1 or later

For builds for cross targets, both a native and a cross compiler are required, as some tools (idlac...) are meant for use on the build host.

Optional:

- A C++ compiler if you want to build the CORBA application personality (the OMG IDL specification mandates that IDL source files be preprocessed according to standard C++ preprocessing rules, and PolyORB relies on an external preprocessor provided by a suitable C++ compiler to implement this).
- XML/Ada (http://libre.adacore.com/xmlada/) if you want to build the SOAP protocol personality.

Note: per construction, the macro `configure` used to find your GNAT compiler looks first to the executables `gnatgcc`, then `adagcc` and finally to `gcc` to find out which Ada compiler to use. You should be very careful with your path and executables if you have multiple GNAT versions installed. See below explanations on the ADA environment variable if you need to override the default guess.

2.3 Build instructions

To compile and install PolyORB, execute:

```
% ./configure [some options]
% make              (or gmake if your make is not GNU make)
% make install      (ditto)
```

This will install files in standard locations. If you want to choose another prefix than '`/usr/local`', give configure a '`--prefix=whereveryouwant`' argument.

Note: at this time, you MUST use GNU make to compile this software.

2.4 Additional instructions for cross platforms

The '`RANLIB`' environment variable must be set to the path of the cross '`ranlib`' prior to running '`configure`' with the appropriate `--target` option.

Only one PolyORB installation (native or cross) is currently possible with a given `--prefix`. If both a native and a cross installation are needed on the same machine, distinct prefixes must be used.

2.5 Building the documentation and PolyORB's examples

PolyORB's documentation and examples are built separately.

After building PolyORB, simply run make in the '`examples`' (resp. '`docs`') directory to build the examples (resp. the documentation). The build process will only build examples that correspond to the personalities you configured. Note that some examples require the CORBA COS Naming service to be enabled (using `--enable-corba-services="naming"` on the `configure` command line).

You may install PolyORB's documentation in a standard location using `make install`.

2.5.1 Build Options

Available options for the 'configure' script include:

- '`--with-appli-perso="..."`': application personalities to build

  Available personalities: AWS, CORBA, DSA, MOMA

  e.g. '`--with-appli-perso="corba moma"`' to build both the CORBA and MOMA personalities

- '`--with-proto-perso="..."`': personalities to build

  Available personalities: GIOP, SOAP

  e.g. '`--with-proto-perso="giop soap"`' to build both the GIOP and SOAP personalities

- '`--with-corba-services="..."`': CORBA COS services to build

  Available services: event, ir, naming, notification, time

  e.g. '`--with-corba-services="event naming"`' to build only COS Event and COS Naming.

  By default, only the CORBA and GIOP personalities are built, no CORBA Services are built.

- '`--with-openssl`': build SSL support and SSL dependent features, including the IIOP/SSLIOP personality
- '`--help`': list all options available
- '`--enable-shared`': build shared libraries.
- '`--enable-debug`': enable debugging information generation and supplementary runtime checks. Note that this option has a significant space and time cost, and is not recommended for production use.

2.5.2 Compiler, Tools and Run-Time libraries Options

The following environment variables can be used to override configure's guess at what compilers to use:

- `CC`: the C compiler
- `ADA`: the Ada 95 compiler (e.g. gcc, gnatgcc or adagcc)
- `CXXCPP`, `CXXCPPFLAGS`: the preprocessor used by idlac (only when setting up the CORBA application personality). CORBA specifications require this preprocessor to be compatible with the preprocessing rules defined in the *C++* programming language specifications.

For example, if you have two versions of GNAT installed and available in your `PATH`, and configure picks the wrong one, you can indicate what compiler should be used with the following syntax:

```
% ADA=/path/to/good/compiler/gcc ./configure [options]
```

PolyORB will be compiled with GNAT build host's configuration, including run-time library. You may override this setting using `ADA_INCLUDE_PATH` and `ADA_OBJECTS_PATH` environment variables. See GNAT User's Guide for more details.

You can add specific build options to GNAT using the `EXTRA_GNATMAKE_FLAGS` variable:

```
% EXTRA_GNATMAKE_FLAGS=--RTS=rts-sjlj ./configure [options]
```

You can also pass compiler-only flags using the `ADAFLAGS` variable.

NOTE: Developers building PolyORB from the version control repository who need to rebuild the configure and Makefile.in files should use the script support/reconfig for this purpose. This should be done after each update from the repository. In addition to the requirements above, they will need autoconf 2.57 or newer, automake 1.6.3 or newer, and libtool 1.5.8 or newer.

2.6 Platform notes

Solaris 2.8:

- /usr/bin/sed and /usr/ucb/sed will silently chop long lines, and /usr/xpg4/bin/sed will enter an endless loop while processing PolyORB files. GNU sed is required to configure and build PolyORB.
- /usr/ucb/tr does not handle control character escape sequences: it cannot be used to recompute dependencies ('make depend'); /usr/bin/tr or /usr/xpg4/bin/tr must be used.

Tru64 5.1A:

The default maximal data segment size may not be sufficient to compile PolyORB. If a GNAT heap exhausted error message occurs during build, try raising this limit using:

```
ulimit -d unlimited
```

AIX 5.2:

PolyORB must be compiled with the -mminimal-toc compiler switch. This can be achieved by setting the following values in the environment at configure time:

```
ADAFLAGS="-g -O2 -mminimal-toc"
CFLAGS="-g -O2 -mminimal-toc"
```

HP-UX 11.00:

The version of install(1) from /opt/imake/bin on HP-UX is not suitable for installing PolyORB. Make sure that /opt/imake/bin is not on the PATH when building and installing PolyORB.

3 Overview of PolyORB personalities

A personality is an instantiation of specific PolyORB components. It provides the mechanisms specified by a distribution model, e.g. an API, a code generator or a protocol stack.

This section provides a brief overview of existing personalities.

Note: some of these personalities are available only through PolyORB's repository.

3.1 Application personalities

Application personalities constitute the adaptation layer between application components and middleware. They provide APIs and/or code generator to register application entities with PolyORB's core, and interoperate with the core to allow the exchange of requests with remote entities.

3.1.1 CORBA

CORBA is OMG specification of a Distributed Object Computing (DOC) distribution model (*[OMG04]*). It is now a well-known and well-established specification, used in a wide range of industrial applications.

PolyORB provides a CORBA-compatible implementation based on mapping of the IDL language version 1.2 described in *[OMG01]* and CORBA core specifications. PolyORB also proposes an implementation of various additional specifications described by the OMG, including `COS Services : COS Naming, Notification, Event, Time`, additional specifications; `RT-CORBA`, `PortableInterceptors`, `DynamicAny`.

3.1.2 Distributed System Annex of Ada (DSA)

The Distributed System Annex of Ada (DSA) *[ISO95]* is a normative specification part of the language. It describes remote invocation schemes applied to most language constructs.

3.1.3 Message Oriented Middleware for Ada (MOMA)

MOMA (Message Oriented Middleware for Ada) provides message passing mechanisms. It is an Ada adaptation of Sun's Java Message Service (JMS) *[SUN99]*, a standardized API for common message passing models.

3.1.4 Ada Web Server (AWS)

The Web Server personality provides the same API as the Ada Web Server project (AWS) *[Obr03]*. It allows for the implementation of web services, web server applications, or classical web pages. AWS-based servers allow the programmer to directly interact with incoming or outgoing `HTTP` and `SOAP` requests.

3.2 Protocol personalities

Protocol personalities handle the mapping of requests (representing interactions between application entities) onto messages exchanged through a communication network, according to a specific protocol.

3.2.1 GIOP

GIOP is the transport layer of the CORBA specifications. GIOP is a generic protocol. This personality implements GIOP versions from 1.0 to 1.2 along with the CDR representation scheme to map data types between the neutral core layer and CDR streams. It also provides the following dedicated instances:

- IIOP supports synchronous request semantics over TCP/IP,
- IIOP/SSIOP supports synchronous request semantics using SSL sockets,
- MIOP instantiation of GIOP enables group communication over IP multicast,
- DIOP relies on UDP/IP communications to transmit one-way requests only.

3.2.2 SOAP

The SOAP protocol *[W3C03]* enables the exchange of structured and typed information between peers. It is a self-describing XML document *[W3C03]* that defines both its data and semantics. Basically, SOAP with `HTTP` bindings is used as a communication protocol for Web Services.

4 Building an application with PolyORB

4.1 Compile-time configuration

The user may configure some elements of a PolyORB application at compile-time.

4.1.1 Tasking run-times

PolyORB provides different tasking run-times. The user may select the most appropriate one, depending on its application requirements. The tasking run-times determine the constructs PolyORB may use for its internal synchronizations.

- `No_Tasking`: There is no dependency on the Ada tasking run-time, middleware is mono-task.
- `Full_Tasking`: Middleware uses Ada tasking constructs, middleware can be configured for multi-tasking.
- `Ravenscar` : Middleware uses Ada tasking constructs, with the limitations of the Ravenscar profile *[DB98]*. Middleware can be configured for multi-tasking.

See Chapter 5 [Tasking model in PolyORB], page 17 for more information on this point.

4.1.2 Middleware tasking policies

PolyORB provides several tasking policies. A tasking policy defines how tasks are used by the middleware to process incoming requests.

- `No_Tasking`: There is only one task in middleware, processing all requests.
- `Thread_Per_Session`: One task monitors communication entities. One task is spawned for each active connection. This task handles all incoming requests on this connection.
- `Thread_Per_Request`: One task monitors communication entities. One task is spawned for each incoming requests.
- `Thread_Pool`: A set of tasks cooperate to handle all incoming requests.

See Chapter 5 [Tasking model in PolyORB], page 17 for more information on this point.

4.1.3 Sample files

PolyORB proposes a set of pre-defined setup packages. You must with one of them in your application node to activate the corresponding setup.

- `PolyORB.Setup.Client`: a client node, without tasking enabled, configured to use all protocol personalities build with PolyORB.
- `PolyORB.Setup.Ravenscar_TP_Server`: a server node, with tasking enabled, configured to use all protocol personalities build with PolyORB. Middleware tasking runtime follow Ravenscar's profile restrictions. Middleware tasking policies is `Thread_Pool`.
- `PolyORB.Setup.Thread_Per_Request_Server`: a server node, with tasking enabled, configured to use all protocol personalities build with PolyORB. Middleware tasking policies is `Thread_Per_Request`.
- `PolyORB.Setup.Thread_Per_Session_Server`: a server node, with tasking enabled, configured to use all protocol personalities build with PolyORB. Middleware tasking policies is `Thread_Per_Session`.

- `PolyORB.Setup.Thread_Pool_Server`: a server node, with tasking enabled, configured to use all protocol personalities build with PolyORB. Middleware tasking policies is `Thread_Pool`.

To enforce one of these configurations, add a dependency on one of these packages. The elaboration of the application (based on Ada rules) and the initialization of the partition (based on the application personalities mechanisms) will set up properly your application.

4.2 Run-time configuration

The user may configure some elements of a PolyORB application at run-time.

Using the default configurations provided by PolyORB, the parameters are read in the following order: command line, environment variables, configuration file. PolyORB will use the first value that matches the searched parameter.

4.2.1 Using a configuration file

A configuration file may be used to configure a PolyORB node. A sample configuration file may be found in '`src/polyorb.conf`'.

The syntax of the configuration file is:

- empty lines and lines that have a '#' in column 1 are ignored;
- sections can be started by lines of the form `[ SECTION-NAME ']'`;
- variable assignments can be performed by lines of the form `VARIABLE-NAME '=' VALUE`.

  Any variable assignment is local to a section.

  Assignments that occur before the first section declaration are relative to section [environment]. Section and variable names are case sensitive.

  Furthermore, each time a resolved in that section value starts with `"file:"`, the contents of the file is used instead.

Default search path for '`polyorb.conf`' is current directory. Environment variable `POLYORB_CONF` may be used to set up information on configuration file.

PolyORB's configuration file allows the user to

1. enable/disable the output of debug information
2. set up default reference on naming service
3. select the default protocol personality
4. set up each protocol personality

The configuration file is read once when running a node, during initialization.

4.2.2 Using environment variables

A variable `Var.Iable` in section `[Sec]` can be overridden by setting environment variable `"POLYORB_SEC_VAR_IABLE"`.

4.2.3 Using the command line

PolyORB allows to set up configuration variables on the command line. The syntax is close to the one described in configuration files A variable `Var.Iable` in section `[Sec]` can be overridden with flag `--polyorb-<sec>-<var>-<iable>[=<value>]`.

4.3 Setting up protocol personalities

PolyORB allows the user to activate some of the available protocol personalities and to set up preferred protocol. Protocol-specific parameters are defined in their respective sections.

4.3.1 Activating/Deactivating protocol personalities

Protocol activation is controlled by PolyORB's configuration file.

The section `[access_points]` control the initialization of *access points*. An access point is a node entry point that may serve incoming requests.

```
[access_points]
soap=enable
iiop=enable
diop=disable
uipmc=disable
```

This example activates SOAP and IIOP, deactivates DIOP and MIOP.

The section `[modules]` controls the activation/deactivation of some modules within PolyORB. It is used to enable *bindings* to remote entities.

```
[modules]
binding_data.soap=enable
binding_data.iiop=enable
binding_data.diop=disable
binding_data.uipmc=disable
```

This example enables the creation of bindings to remote objects using SOAP or IIOP. Objects cannot be reached using DIOP or UIPMC.

Note: by default, all configured personalities are activated.

4.3.2 Configuring protocol personality preferences

The user may affect a *preference* to each protocol personality. The protocol with the higher preference will be selected among possible protocols to send a request to a remote node.

See `polyorb.binding_data.<protocol>.preference` in section `[protocol]` to set up protocol's preference.

Possible protocols are defined as the protocols available on the remote node, as advertised in its *object reference*. `IOR` or `corbaloc` references may support multiple protocols, `URI` only support one protocol.

Each protocol supports a variety of configuration parameters, please refer to the protocols' sections for more details.

4.4 Activating debug information

To activate the output of debug information, you must first configure and compile PolyORB with debug activate, see help on `--enable-debug` flag in Chapter 2 [Installation], page 5.

To output debug information on a selected package, create a configuration file with a `[log]` section and the name of the packages on which you want debug information:

```
# Sample configuration file, output debug for PolyORB.A_Package
[log]
polyorb.a_package=debug
```

Note that some packages may not provide such information. See sample configuration file the complete list of packages that provide debug.

4.5 Tracing exceptions

To trace exception propagation in PolyORB's source code, it is necessary to:

1. compile PolyORB with debug activated,
2. activate debug information on package `PolyORB.Exceptions`.

4.6 `polyorb-config`

`polyorb-config` returns path and library information on PolyORB's installation.

```
NAME
       polyorb-config  - script to get information about the installed version
       of PolyORB.

SYNOPSIS
       polyorb-config [--prefix[=DIR]] [--exec-prefix[=DIR]] [--version|-v]
       [--config] [--libs] [--cflags] [--idls] [--help]

DESCRIPTION
       polyorb-config  is  a  tool  that is used to determine the compiler and
       linker flags that should be used to compile and link programs that  use
       PolyORB.

OPTIONS
       polyorb-config accepts the following options:

       --prefix[=DIR]
                Output the directory in which PolyORB architecture-independent
               files are installed, or set this directory to DIR.

       --exec-prefix[=DIR]
                Output the directory in which  PolyORB  architecture-dependent
               files are installed, or set this directory to DIR.

       --version
               Print  the  currently installed version of PolyORB on the stan-
               dard output.

       --config
               Print the configuration of the currently installed  version  of
               PolyORB on the standard output.

       --libs  Print  the  linker  flags  that are necessary to link a PolyORB
               program.

       --cflags
               Print the compiler flags that are necessary to compile a  Poly-
               ORB program.

       --idls
               Output flags to set up path to CORBA's IDL for idlac.

       --with-appli-perso=P,P,P
               Restrict output to only those flags relevant to the listed
               applicative personalities.
```

```
--with-proto-perso=P,P,P
        Restrict output to only those flags relevant to the listed
        protocol personalities.

--with-corba-services=S,S,S
       Restrict output to only those flags relevant to the listed
       services.

--help  Print help message.
```

5 Tasking model in PolyORB

5.1 PolyORB Tasking runtimes

PolyORB may use three different tasking runtimes to manage and synchronize tasks, if any. Tasking runtimes capabilities are defined in the Ada reference manual *[ISO95]* and the next revision of this standard (*Ada0Y*).

The choice of a specific tasking runtime is a compile-time parameter, Section 4.1.1 [Tasking run-times], page 11 for more details on their configuration.

5.1.1 Full tasking runtime

Full tasking runtime refers to configuration in which there are some dependencies on the tasking constructs defined in chapter 9 of *[ISO95]*. It makes use of all capabilities defined in this section to manage and synchronize tasks.

In this configuration, a PolyORB application must be compiled and linked with a tasking-capable Ada runtime.

5.1.2 No tasking runtime

No tasking runtime refers to configuration in which there is no semantic dependency on tasking constructs. Thus, no tasking is required.

In this configuration, a PolyORB application may be compiled and linked with a tasking-capable Ada runtime or a no-tasking Ada runtime.

5.1.3 Ravenscar tasking runtime

Ravenscar tasking runtime refers to configuration in which tasking constructs are compliant with the *Ravenscar tasking restricted profile*.

In this configuration, a PolyORB application may be compiled and linked with a tasking-capable Ada runtime or a Ravenscar Ada runtime.

To configure tasking constructs used by PolyORB, one must instanciate the `PolyORB.Setup.Tasking.Ravenscar` package to setup tasks and protected objects used by PolyORB core.

```
------------------------------------------------------------------------------
--                                                                          --
--                           POLYORB COMPONENTS                             --
--                                                                          --
--      P O L Y O R B . S E T U P . T A S K I N G . R A V E N S C A R       --
--                                                                          --
--                                 S p e c                                  --
--                                                                          --
--         Copyright (C) 2002-2004 Free Software Foundation, Inc.           --
--                                                                          --
-- PolyORB is free software; you  can  redistribute  it and/or modify it    --
-- under terms of the  GNU General Public License as published by the  Free --
-- Software Foundation;  either version 2,  or (at your option)  any  later --
-- version. PolyORB is distributed  in the hope that it will be  useful,    --
-- but WITHOUT ANY WARRANTY;  without even the implied warranty of MERCHAN- --
-- TABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public --
```

```
-- License  for more details.  You should have received  a copy of the GNU  --
-- General Public License distributed with PolyORB; see file COPYING. If    --
-- not, write to the Free Software Foundation, 59 Temple Place - Suite 330, --
-- Boston, MA 02111-1307, USA.                                              --
--                                                                          --
--
--
--
--
--
--
--
--
--                                                                          --
--                PolyORB is maintained by ACT Europe.                      --
--                    (email: sales@@act-europe.fr)                          --
--                                                                          --
------------------------------------------------------------------------------

--  You should instanciate this package to set up a ravenscar profile.

with System;

with PolyORB.Tasking.Profiles.Ravenscar.Threads.Annotations;

with PolyORB.Tasking.Profiles.Ravenscar.Threads;
with PolyORB.Tasking.Profiles.Ravenscar.Mutexes;
with PolyORB.Tasking.Profiles.Ravenscar.Condition_Variables;

generic
   Number_Of_Application_Tasks    : Integer;
   --  Number of tasks created by the user.

   Number_Of_System_Tasks         : Integer;
   --  Number of tasks created by the PolyORB run-time library.

   Number_Of_Conditions           : Integer;
   --  Number of preallocated conditions.

   Number_Of_Mutexes              : Integer;
   --  Number of preallocated mutexes.

   Task_Priority                  : System.Priority;
   --  Priority affected to the tasks of the pool.

   Storage_Size                   : Natural;
   --  Stack size of the system tasks.

package PolyORB.Setup.Tasking.Ravenscar is

   package Threads_Package is
      new PolyORB.Tasking.Profiles.Ravenscar.Threads
     (Number_Of_Application_Tasks,
      Number_Of_System_Tasks,
      Task_Priority,
      Storage_Size);

   package Thread_Annotations_Package is new Threads_Package.Annotations;
```

```
      package Conditions_Package is
         new PolyORB.Tasking.Profiles.Ravenscar.Condition_Variables
        (Threads_Package,
         Number_Of_Conditions);

      package Mutexes_Package is
         new PolyORB.Tasking.Profiles.Ravenscar.Mutexes
        (Threads_Package,
         Number_Of_Mutexes);

   end PolyORB.Setup.Tasking.Ravenscar;
```

5.2 PolyORB ORB Tasking policies

PolyORB ORB Tasking policies control the creation of tasks to process all middleware internal jobs, e.g. request processing, I/O monitoring.

Note: there is a dependency between ORB Tasking policies, and the run-time used

5.2.1 No Tasking

Under the No Tasking ORB policy, no task are created within the middleware instance: it uses the environment task to process all jobs. Note that this policy is not thread-safe and is compatible with the No tasking runtime only.

5.2.2 Thread Pool

Under the Thread Pool ORB policy, the middleware creates a pool of thread during the initialization of PolyORB. This pool processes all jobs. The number of tasks in the thread pool can be configured by three parameters in the `[tasking]` configuration section.

- `min_spare_threads` indicates the number of tasks created at startup.
- `max_spare_threads` is a ceiling. When a remote subprogram call is completed, its anonymous task is deallocated if the number of tasks already in the pool is greater than the ceiling. If not, then the task is queued in the pool.
- `max_threads` indicates the maximum number of tasks in the pool.

See Section 5.3 [PolyORB Tasking configuration], page 19, for more information on how to configure the number of tasks in the thread pool.

5.2.3 Thread Per Session

Under the Thread Per Session ORB policy, the middleware creates one task when a new session (one active connection) is opened. The task is finalized when the session is closed.

5.2.4 Thread Per Request

Under the Thread Per Request ORB policy, the middleware creates one task per incoming request. The task is finalized when the request is completed.

5.3 PolyORB Tasking configuration

The following parameters allow the user to set up some of the tasking parameters.

```
##############################################################################
# Parameters for tasking
#

[tasking]
# Default storage size for all threads spawned by PolyORB
#storage_size=262144

# Number of threads by Thread Pool tasking policy
#min_spare_threads=4
#max_spare_threads=4
#max_threads=4
```

5.4 PolyORB ORB Controller policies

The PolyORB ORB Controller policies are responsible for the management of the global state of the middleware: they assign middleware internal jobs, or I/Os monitoring to middleware tasks.

ORB Controller policies grant access to middleware internals and affect one action for each middleware task. They ensure that all tasks work concurrently in a thread-safe manner.

5.4.1 No Tasking

The No Tasking ORB Controller policy is dedicated to no tasking middleware configurations; the middleware task executes the following loop: process internal jobs, then monitor I/Os.

5.4.2 Workers

The Workers ORB Controller policy is a simple controller policy: all tasks are equal, they may alternatively and randomly process requests or wait for I/O sources.

Note: it is the default configuration provided by PolyORB sample setup files, See *Section 4.1.3 [Sample files], page 11.*

5.4.3 Half Sync/Half Async

The Half Sync/Half Async ORB Controller policy implements the "Half Sync/Half Async" design pattern: it discriminates between one thread dedicated to I/O monitoring that queue middleware jobs; another pool of threads dequeue jobs and process them.

Note: this pattern is well-suited to process computation-intensive requests.

5.4.4 Leader/Followers

The Leader/Followers ORB Controller policy implements the "Leader/Followers " design pattern: multiple tasks take turns to monitor I/O sources and then process requests that occur on the event sources.

Note: this pattern is adapted to process a lot of light requests.

5.5 PolyORB ORB Controller configuration

The following parameters allow the user to set up parameters for ORB Controllers.

```
##############################################################################
# Parameters for ORB Controllers
#

[orb_controller]
# Interval between two polling actions on one monitor
#polyorb.orb_controller.polling_interval=0

# Timeout when polling on one monitor
#polyorb.orb_controller.polling_timeout=0
```

6 CORBA

6.1 What you should know before Reading this section

This section assumes that the reader is familiar with the CORBA specifications described in *[OMG04]* and the *IDL-to-Ada* mapping defined in *[OMG01]*.

6.2 Installing CORBA application personality

Ensure PolyORB has been configured and then compiled with CORBA application personality. See Chapter 4 [Building an application with PolyORB], page 11 for more details on how to check installed personalities.

To build the CORBA application personality, see Chapter 2 [Installation], page 5.

6.3 Usage of `idlac`

`idlac` is PolyORB's IDL-to-Ada 95 compiler.

```
NAME
       idlac - PolyORB's IDL-to-Ada compiler

SYNOPSIS
       idlac [-Edikpqv] [-[no]ir] [-gnatW8] [-o DIR] idl_file [-cppargs ...]

DESCRIPTION
       idlac is an IDL-to-Ada compiler, compliant with version 1.2 of the "Ada
       Language Mapping Specification" produced by the OMG.

OPTIONS
       idlac accepts the following options:

       -E      Preprocess only.

       -d      Generate delegation package.

       -i      Generate implementation template.

       -s      Generate server side code.

       -c      Generate client side code.

       -k      Keep temporary files.

       -p      Produce source on standard output.

       -q      Be quiet (default).

       -v      Be verbose.

       -ir     Generate code for interface repository.

       -noir   Don't generate code for interface repository (default).

       -gnatW8
```

```
                Use UTF8 character encoding

        -o DIR Specify output directory

        -cppargs ARGS
                Pass ARGS to the C++ preprocessor.

        -I dir  Shortcut for -cppargs -I dir.

EXIT STATUS
       idlac returns one of the following values upon exit:

       0      Successful completion

       1      Usage error

       2      Illegal IDL specification
```

`idlac` creates several files :

- `myinterface.ads`, `myinterface.adb` : these files contain the mapping for user defined types (client and server side).
- `myinterface-impl.ads`, `myinterface-impl.adb` : these files are to be filled by the user. They contain the implementation of the server. They are generated only if the -i flag is specified.
- `myinterface.ads`, `myinterface.adb` : these files contain the client stubs for the interface.
- `myinterface-skel.ads`, `myinterface-skel.adb` : these files contain the server-side skeletons for the interface.
- `myinterface-helper.ads`, `myinterface-helper.adb` : these files contain subprograms to marshal data into CORBA Any containers.
- `myinterface-ir_info.ads`, `myinterface-ir_info.adb` : these files contain code for registering IDL definitions in the CORBA Interface Repository. They are generated only if the `'-ir'` flag is specified.

6.4 Resolving names in a CORBA application

PolyORB implements the CORBA COS Naming service.

6.4.1 `po_cos_naming`

`po_cos_naming` is a stand alone server that supports CORBA COS Naming specification. When launched, it returns its `IOR` and `corbaloc` that can then be used by other CORBA applications.

If you want `po_cos_naming` to return the same `IOR` or `corbaloc` at each startup, you must set a default listen port for the protocol personalities you use. See Section 4.3.2 [Configuring protocol personality preferences], page 13 for more details.

`po_cos_naming` can output its `IOR` directly to a file using the `-file <filename>` flag. This, in conjonction with the `'file://'` naming scheme provided by `CORBA`, proposes a convenient way to store initial references to the Naming Service.

```
Usage: po_cos_naming
 -file <filename> : output COS Naming IOR to 'filename'
 -help : print this help
 [PolyORB command line configuration variables]
```

6.4.2 Registering the reference to the COS Naming server

You have two ways to register the reference to the root context of the COS Naming server the application will use:

- Setting up the `name_service` entry in the `[corba]` section in your configuration file, `name_service` is the `IOR` or `corbaloc` of the COS Naming server to use. See Section 4.2.1 [Using a configuration file], page 12 for more details.
- Registering an initial reference using the `-ORB InitRef NamingService=<IOR>` or `-ORB InitRef NamingService=<corbaloc>` command-line argument. See the CORBA specifications for more details.
- Registering an initial reference for `NamingService` using the `CORBA.ORB.Register_Initial_Reference` function. See the CORBA specifications for more details.

6.4.3 Using the COS Naming

PolyORB provides a helper package to manipulate the COS Naming in your applications. See Section 6.9 [PolyORB specific APIs], page 37 for more details.

6.5 The CORBA Interface Repository

PolyORB implements the CORBA Interface Repository.

6.5.1 `po_ir`

`po_ir` is a stand alone server that supports the CORBA Interface Repository. When launched, it returns its `IOR` and `corbaloc` that can then be used by other CORBA applications.

If you want `po_ir` to return the same `IOR` or `corbaloc` at each startup, you must set a default listen port for the protocol personalities you use. See Section 4.3.2 [Configuring protocol personality preferences], page 13 for more details.

6.5.2 Using the Interface Repository

`idlac` generates a helper package that allows you to register all entities defined in your IDL specification in the Interface Repository.

6.6 Building a CORBA application with PolyORB

6.6.1 `echo` example

We consider building a simple "Echo" CORBA server and client. This application echoes a string. The source code for this example is located in '`examples/corba/echo`' directory in PolyORB distribution. This applications uses only basic elements of CORBA.

To build this application, you need the following pieces of code:

1. IDL definition of an `echo` object

2. Implementation code for the `echo` object
3. Code for client and server nodes

6.6.1.1 IDL definition of an `echo` object

This interface defines an `echo` object with a unique method `echoString`. Per construction, this method returns its argument.

```
interface Echo {
  string echoString (in string Mesg);
};
```

6.6.1.2 Implementation code for the `echo` object

Package `Echo.Impl` is an implementation of this interface. This implementation follows the *IDL-to-Ada* mapping.

```
------------------------------------------------------------------------------
--                                                                          --
--                            POLYORB COMPONENTS                            --
--                                                                          --
--                             E C H O . I M P L                            --
--                                                                          --
--                                  S p e c                                 --
--                                                                          --
--            Copyright (C) 2002 Free Software Foundation, Inc.             --
--                                                                          --
-- PolyORB is free software; you  can  redistribute  it and/or modify it    --
-- under terms of the  GNU General Public License as published by the  Free --
-- Software Foundation;  either version 2,  or (at your option)  any  later --
-- version. PolyORB is distributed  in the hope that it will be  useful,    --
-- but WITHOUT ANY WARRANTY;  without even the implied warranty of MERCHAN- --
-- TABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public --
-- License  for more details.  You should have received  a copy of the GNU  --
-- General Public License distributed with PolyORB; see file COPYING. If    --
-- not, write to the Free Software Foundation, 59 Temple Place - Suite 330, --
-- Boston, MA 02111-1307, USA.                                              --
--                                                                          --
--
--
--
--
--
--
--
--
--                                                                          --
--                PolyORB is maintained by ACT Europe.                      --
--                    (email: sales@@act-europe.fr)                          --
--                                                                          --
------------------------------------------------------------------------------

with CORBA;
with PortableServer;

package Echo.Impl is
```

```
   --  My own implementation of echo object.
   --  This is simply used to define the operations.

   type Object is new PortableServer.Servant_Base with null record;

   type Object_Acc is access Object;

   function EchoString
     (Self : access Object;
      Mesg : in     CORBA.String)
     return CORBA.String;

end Echo.Impl;
```

```
------------------------------------------------------------------------------
--                                                                          --
--                           POLYORB COMPONENTS                             --
--                                                                          --
--                            E C H O . I M P L                             --
--                                                                          --
--                                 B o d y                                  --
--                                                                          --
--            Copyright (C) 2002 Free Software Foundation, Inc.             --
--                                                                          --
-- PolyORB is free software; you  can  redistribute  it and/or modify it    --
-- under terms of the  GNU General Public License as published by the  Free --
-- Software Foundation;  either version 2,  or (at your option)  any  later --
-- version. PolyORB is distributed  in the hope that it will be  useful,    --
-- but WITHOUT ANY WARRANTY;  without even the implied warranty of MERCHAN- --
-- TABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public --
-- License  for more details.  You should have received  a copy of the GNU  --
-- General Public License distributed with PolyORB; see file COPYING. If    --
-- not, write to the Free Software Foundation, 59 Temple Place - Suite 330, --
-- Boston, MA 02111-1307, USA.                                              --
--                                                                          --
--
--
--
--
--
--
--
--
--                                                                          --
--                PolyORB is maintained by ACT Europe.                      --
--                    (email: sales@@act-europe.fr)                          --
--                                                                          --
------------------------------------------------------------------------------

with Ada.Text_IO;

with Echo.Skel;
pragma Warnings (Off, Echo.Skel);
--  No entity from Echo.Skel is referenced.

package body Echo.Impl is
```

```
   ---------------
   -- EchoString --
   ---------------

   function EchoString
     (Self : access Object;
      Mesg : in     CORBA.String)
     return CORBA.String
   is
      pragma Warnings (Off);
      pragma Unreferenced (Self);
      pragma Warnings (On);
   begin
      Ada.Text_IO.Put_Line
        ("Echoing string:  " & CORBA.To_Standard_String (Mesg)
         & " ");
      return Mesg;
   end EchoString;

end Echo.Impl;
```

Note: `Echo.Impl` *body requires a dependency on* `Echo.Skel` *to ensure the elaboration of skeleton code and the correct setup of PolyORB's internals.*

6.6.1.3 Test code for client and server nodes

Client and server code demonstrate how to make a remote invocation on a CORBA object, and how to setup an object on a server node.

Note: the dependency on `PolyORB.Setup.Client` *or* `PolyORB.Setup.No_Tasking_Server` *enforces compile-time configuration, see* *Section 4.1.3 [Sample files], page 11.*

- Client code tests a simple remote invocation on object. It is a no tasking client. Reference to object is built from stringified reference (or `IOR`), which is passed through command line.

```
------------------------------------------------------------------------------
--                                                                          --
--                            POLYORB COMPONENTS                            --
--                                                                          --
--                               C L I E N T                                --
--                                                                          --
--                                 B o d y                                  --
--                                                                          --
--         Copyright (C) 2002-2004 Free Software Foundation, Inc.           --
--                                                                          --
-- PolyORB is free software; you  can  redistribute  it and/or modify it    --
-- under terms of the  GNU General Public License as published by the  Free --
-- Software Foundation;  either version 2,  or (at your option)  any  later --
-- version. PolyORB is distributed  in the hope that it will be  useful,    --
-- but WITHOUT ANY WARRANTY;  without even the implied warranty of MERCHAN- --
-- TABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public --
-- License  for more details.  You should have received  a copy of the GNU  --
-- General Public License distributed with PolyORB; see file COPYING. If    --
-- not, write to the Free Software Foundation, 59 Temple Place - Suite 330, --
-- Boston, MA 02111-1307, USA.                                              --
--                                                                          --
--
```

```
--
--
--
--
--
--
--                                                                          --
--                PolyORB is maintained by ACT Europe.                      --
--                    (email: sales@@act-europe.fr)                          --
--                                                                          --
------------------------------------------------------------------------------

--   echo client.

with Ada.Command_Line;
with Ada.Text_IO;
with CORBA.ORB;

with Echo;

with PolyORB.Setup.Client;
pragma Warnings (Off, PolyORB.Setup.Client);

with PolyORB.Utils.Report;

procedure Client is
   use Ada.Command_Line;
   use Ada.Text_IO;
   use PolyORB.Utils.Report;

   Sent_Msg, Rcvd_Msg : CORBA.String;
   myecho : Echo.Ref;

begin
   New_Test ("Echo client");

   CORBA.ORB.Initialize ("ORB");
   if Argument_Count /= 1 then
      Put_Line ("usage : client <IOR_string_from_server>|-i");
      return;
   end if;

   --  Getting the CORBA.Object

   CORBA.ORB.String_To_Object
     (CORBA.To_CORBA_String (Ada.Command_Line.Argument (1)), myecho);

   --  Checking if it worked

   if Echo.Is_Nil (myecho) then
      Put_Line ("main : cannot invoke on a nil reference");
      return;
   end if;

   --  Sending message

   Sent_Msg := CORBA.To_CORBA_String (Standard.String'("Hello Ada !"));
   Rcvd_Msg := Echo.echoString (myecho, Sent_Msg);
```

```
   --  Printing result

   Put_Line ("I said : " & CORBA.To_Standard_String (Sent_Msg));
   Put_Line ("The object answered : " & CORBA.To_Standard_String (Rcvd_Msg));

   End_Report;

exception
   when E : CORBA.Transient =>
      declare
         Memb : CORBA.System_Exception_Members;
      begin
         CORBA.Get_Members (E, Memb);
         Put ("received exception transient, minor");
         Put (CORBA.Unsigned_Long'Image (Memb.Minor));
         Put (", completion status: ");
         Put_Line (CORBA.Completion_Status'Image (Memb.Completed));

         End_Report;
      end;
end Client;
```

- Server code setups a no tasking node. Object is registered to the `RootPOA`. Then an `IOR` reference is built to enable interaction with other nodes.

```
------------------------------------------------------------------------------
--                                                                          --
--                            POLYORB COMPONENTS                            --
--                                                                          --
--                               S E R V E R                                --
--                                                                          --
--                                 B o d y                                  --
--                                                                          --
--         Copyright (C) 2002-2007, Free Software Foundation, Inc.          --
--                                                                          --
-- PolyORB is free software; you  can  redistribute  it and/or modify it    --
-- under terms of the  GNU General Public License as published by the  Free --
-- Software Foundation;  either version 2,  or (at your option)  any  later --
-- version. PolyORB is distributed  in the hope that it will be  useful,    --
-- but WITHOUT ANY WARRANTY;  without even the implied warranty of MERCHAN- --
-- TABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public --
-- License  for more details.  You should have received  a copy of the GNU  --
-- General Public License distributed with PolyORB; see file COPYING. If    --
-- not, write to the Free Software Foundation, 51 Franklin Street, Fifth    --
-- Floor, Boston, MA 02111-1301, USA.                                       --
--                                                                          --
--
--
--
--
--
--
--
--
--                                                                          --
--                  PolyORB is maintained by AdaCore                        --
--                     (email: sales@@adacore.com)                           --
```

```
--                                                                          --
------------------------------------------------------------------------------

with Ada.Text_IO;

with CORBA.Impl;
with CORBA.Object;
with CORBA.ORB;

with PortableServer.POA.Helper;
with PortableServer.POAManager;

with Echo.Impl;

with PolyORB.CORBA_P.CORBALOC;

--  Setup server node: use no tasking default configuration

with PolyORB.Setup.No_Tasking_Server;
pragma Warnings (Off, PolyORB.Setup.No_Tasking_Server);

procedure Server is
begin

   declare
      Argv : CORBA.ORB.Arg_List := CORBA.ORB.Command_Line_Arguments;

   begin
      CORBA.ORB.Init (CORBA.ORB.To_CORBA_String ("ORB"), Argv);

      declare
         Root_POA : PortableServer.POA.Local_Ref;

         Ref : CORBA.Object.Ref;

         Obj : constant CORBA.Impl.Object_Ptr := new Echo.Impl.Object;

      begin

         --  Retrieve Root POA

         Root_POA := PortableServer.POA.Helper.To_Local_Ref
           (CORBA.ORB.Resolve_Initial_References
            (CORBA.ORB.To_CORBA_String ("RootPOA")));

         PortableServer.POAManager.Activate
           (PortableServer.POA.Get_The_POAManager (Root_POA));

         --  Set up new object

         Ref := PortableServer.POA.Servant_To_Reference
           (Root_POA, PortableServer.Servant (Obj));

         --  Output IOR

         Ada.Text_IO.Put_Line
           ("'"
            & CORBA.To_Standard_String (CORBA.Object.Object_To_String (Ref))
```

```
            & "'");
         Ada.Text_IO.New_Line;

         --  Output corbaloc

         Ada.Text_IO.Put_Line
           ("'"
            & CORBA.To_Standard_String
            (PolyORB.CORBA_P.CORBALOC.Object_To_Corbaloc (Ref))
            & "'");

         --  Launch the server

         CORBA.ORB.Run;
      end;
   end;
end Server;
```

6.6.1.4 Compilation and execution

To compile this demo,

1. Process the IDL file with `idlac`
   ```
   $ idlac echo.idl
   ```
2. Compile the client node
   ```
   $ gnatmake client.adb `polyorb-config`
   ```
3. Compile the server node
   ```
   $ gnatmake server.adb `polyorb-config`
   ```

Note the use of backticks (`). This means that `polyorb-config` is first executed, and then the command line is replaced with the output of the script, setting up library and include paths and library names.

To run this demo:

- run '`server`', the server outputs its IOR, an hexadecimal string with the <IOR:> prefix.
  ```
  $ ./server
  Loading configuration from polyorb.conf
  No polyorb.conf configuration file.
  'IOR:01534f410d00000049444c3[..]'
  ```
- run '`client`', passing the complete IOR on the command line
  ```
  $ ./client 'IOR:01534f410d00000049444c3[..]'
  Echoing string:  Hello Ada !
  I said : Hello Ada !
  The object answered : Hello Ada !
  ```

6.6.2 Other examples

PolyORB proposes other examples to test other CORBA features. These examples are located in '`example/corba`' directory in PolyORB distribution.

- '`all_functions`' tests CORBA parameters passing mode (`in`, `out`, ..);
- '`all_types`' tests CORBA types;
- '`echo`' is a simple CORBA demo;
- '`random`' is a random number generator;
- '`send`' tests MIOP specific API.

6.7 Configuring a CORBA application

To configure a CORBA application, you need to separately configure PolyORB and the GIOP protocol (or any other protocol personality you wish to use).

6.7.1 Configuring PolyORB

Please, refer to Chapter 4 [Building an application with PolyORB], page 11 for more information on PolyORB's configuration.

6.7.2 Configuring GIOP protocol stack for PolyORB

The GIOP protocol is separated from the CORBA application personality. See Section 11.3 [Configuring the GIOP personality], page 57 for more information on GIOP's configuration.

6.7.3 Configuring Security services for PolyORB

PolyORB provides support for some elements of the CORBA Security mechanisms. This sections lists the corresponding configuration parameters.

6.7.3.1 Supported mechasnisms

PolyORB provides support for the following security mechanisms:

1. SSL/TLS protected transport;
2. GSSUP (user/password) authentication mechanism;
3. identity assertion and backward trust evaluation.

6.7.3.2 Compile-time configuration

To enable security support applications must "with" one of the predefined setup packages:

1. `PolyORB.Setup.Secure_Client` - for client side support only;
2. `PolyORB.Setup.Secure_Server` - for both client and server side support.

6.7.3.3 Run-time configuration

1. Capsule configuration

   This section details the configuration parameters for capsule configuration.

```
[security_manager]
# List of sections for configure client's credentials
#own_credentials=my_credentials
#
# Client requires integrity proteced messages
#integrity_required=true
#
# Client requires confiodentiality protected messages
#confidentiality_required=true
#
# Client requires security association to detect replay (not supported
for now)
#detect_replay_required=true
#
# Client requires security association to detect message sequence
errors (not
# supported for now)
#detect_misordering_required=true
```

```
#
# Client requires target authentication
#establish_trust_in_target_required=true
#
# Client requires client authentication (usually not applicable at
all)
#establish_trust_in_client_required=true
#
# (rare useful)
#identity_assertion_required=true
#
# (rare useful)
#delegation_by_client_required=true
```

2. Credentials configuration

   This section details configuration parameters for defining program's credentials. Depending on used mechanisms for transport and authentication layers, credentials configuration section may define configuration only for one transport mechanism and/or one authentication mechanism.

```
#[my_credentials]
#
# TLS protected transport mechanism used as transport mechanism
#transport_credentials_type=tls
#
# Connection method. Available methods: tls1, ssl3, ssl2
#tls.method=tls1
#
# Certificate file name
#tls.certificate_file=my.crt
#
# Certificate chain file name
#tls.certificate_chain_file=
#
# Private key file name
#tls.private_key_file=my.key
#
# Name of file, at which CA certificates for verification purposes are
#located
#tls.certificate_authority_file=root.crt
#
# Name of directory, at which CA certificates for verification
#purposes are
# located
#tls.certificate_authority_path=
#
# List of available ciphers
#tls.ciphers=ALL
#
# Verify peer certificate
#tls.verify_peer=true
#
# Fail if client don't provide ceritificate (server only)
#tls.verify_fail_if_no_peer_certificate=true
#
# GSSUP (user/password) mechanism as authentication mechanism
#authentication_credentials_type=gssup
#
```

```
# User name
#gssup.username=username@domain
#
# User password
#gssup.password=password
#
# Target name for which user/password pair is applicable
#gssup.target_name=@domain
```

3. POA configuration

   This section details configuration parameters for defining security characteristics of objects managed by POA. POA's name is used as section name.

```
#[MySecurePOA]
#
# Unprotected invocations is allowed
#unprotected_invocation_allowed=true
#
# Section name for configuration of used protected transport mechanism
#(if any)
#transport_mechanism=tlsiop
#
# Section name for configuration of used authentication mechanism (if
#any)
#authentication_mechanism=my_gssup
#
# Target require client authentication at authentication layer (in
#addition
# to authentication at transport layer)
#authentication_required=true
#
# Name of file for backward trust evalutation rules
#backward_trust_rules_file=file.btr
#
# Section name for configuration of authorization tokens authority
#privilege_authorities=
```

4. TLS protected transport mechanism configuration

   This section details configuration parameters for the TLS protected transport mechanism. Section name for mechanism configuration is defined in POA configuration.

```
[tlsiop]
# List of access points
#addresses=127.0.0.1:3456
```

5. GSSUP authentication mechanism

   This section details configuration parameters for the GSSUP auithentication mechanism. Section name for mechanism configuration defined in POA configuration.

```
#[my_gssup]
#
# Authentication mechanism
#mechanism=gssup
#
# Target name
#gssup.target_name=@domain
#
# User name/password mapping file
#gssup.passwd_file=passwd.pwd
```

6.7.4 Command line arguments

The CORBA specifications define a mechanism to pass command line arguments to your application, using the `CORBA::ORB:Init` method.

For now, PolyORB supports the following list of arguments:

- `InitRef` to pass initial reference.

6.8 Implementation Notes

PolyORB strives to support CORBA specifications as closely as possible. However, in rare occasions, the implementation adapts the specifications to actually enable its completion. This section provides information on the various modification we made.

6.8.1 Tasking

PolyORB provides support for tasking and no-tasking, using configuration parameters. Please, refer to Chapter 4 [Building an application with PolyORB], page 11 for more information on PolyORB's configuration.

When selecting a tasking-capable runtime, ORB-related functions are thread safe, following the IDL-to-Ada mapping recommendations.

6.8.2 Implementation of CORBA specifications

In some occasions, the CORBA specifications do not describe the semantics of the interface with sufficient details. We add an `Implementation Notes` tag in the package specification to indicate the modifications or enhancements we made to the standard.

In some occasions, the IDL-to-Ada mapping specifications and the CORBA specifications conflict. We add an `Implementation Notes` tag in the package specification to indicate this issue. Whenever possible, PolyORB follows the CORBA specifications.

6.8.3 Additions to the CORBA specifications

In some occasions, the specifications lack feature that may be useful. We add an `Implementation Notes` tag in the package specification to detail the additions we made to the standard.

Besides, PolyORB follows some of the recommendations derived from the OMG Issues for Ada 2003 Revision Task Force mailing list (see `http://www.omg.org/issues/ada-rtf.html` for more information).

6.8.4 Interface repository

The documentation of the PolyORB's CORBA Interface Repository will appear in a future revision of PolyORB.

6.8.5 Policy Domain Managers

You have two ways to register the reference to the CORBA Policy Domain Manager the application will use:

- Setting up the `policy_domaing_manager` entry in the `[corba]` section in your configuration file, `policy_domaing_manager` is the `IOR` or `corbaloc` of the COS Naming server to use. See Section 4.2.1 [Using a configuration file], page 12 for more details.

- Registering an initial reference using the `-ORB InitRef PolyORBPolicyDomainManager=<IOR>` or `-ORB InitRef PolyORBPolicyDomainManager=<corbaloc>` command-line argument. See the CORBA specifications for more details.
- Registering an initial reference for `PolyORBPolicyDomainManager` using the `CORBA.ORB.Register_Initial_Reference` function. See the CORBA specifications for more details.

6.8.6 Mapping of exceptions

For each exception defined in the CORBA specifications, PolyORB provides the `Raise_<excp_name>` function, a utility function that raises the exception `<excp_name>`, along with its exception member. PolyORB also defines the `Get_Members` function (as defined in the IDL-to-Ada mapping) to provide accessors to retrieve information on the exception.

In addition, for each exception defined in a user-defined IDL specification, '`idlac`' will generate a `Raise_<excp_name>` function in the Helper package. It is a utility function that raises the exception `<excp_name>`, along with its exception member.

6.8.7 Additional information to `CORBA::Unknown`

When a CORBA application raises an Ada exception that is not part of the IDL specifications, or defined by the CORBA specifications, then this exception is translated into a `CORBA::UNKNOWN` exception.

To help debugging CORBA applications, PolyORB supports a specific service context to the GIOP protocol personality that conveys exception information. When displaying exception information, server-side specific exception information are delimited by *"<Invocation Exception Info: ..>"*

Here is an example from the `all_types` example provided by PolyORB.

```
Exception name: CORBA.UNKNOWN
Message: 4F4D0001M
<Invocation Exception Info: Exception name: CONSTRAINT_ERROR
Message: all_types-impl.adb:315 explicit raise
Call stack traceback locations:
0x84d279c 0x84c1e78 0x84b92c6 0x84b8e9>
Call stack traceback locations:
0x81d0425 0x81d0554 0x81d6d8c 0x81fd02b 0x81fc091 0x82eea12 0x83e4c22 0x807b69a 0xb7a15e3e
```

6.8.8 Internals packages

PolyORB sometimes declare internals types and routines inside CORBA packages. In this case, these entities are gathered into an `Internals` child package. You should not use these functions: they are not portable, and may be changed in future releases.

6.9 PolyORB's specific APIs

PolyORB defines packages to help in the development of CORBA programs.

- Section 6.9.1 [PolyORB.CORBA_P.CORBALOC], page 39:

  This package defines a helper function to build a `corbaloc` stringified reference from a CORBA object reference.
- Section 6.9.2 [PolyORB.CORBA_P.Naming_Tools], page 40:

  This package defines helper functions to ease interaction with CORBA COS Naming.

- Section 6.9.3 [PolyORB.CORBA_P.Server_Tools], page 42:
  This package defines helper functions to ease set up of a simple CORBA Server.

6.9.1 PolyORB.CORBA_P.CORBALOC

```
------------------------------------------------------------------------------
--                                                                          --
--                           POLYORB COMPONENTS                             --
--                                                                          --
--             P O L Y O R B . C O R B A _ P . C O R B A L O C              --
--                                                                          --
--                                 S p e c                                  --
--                                                                          --
--         Copyright (C) 2004-2006, Free Software Foundation, Inc.          --
--                                                                          --
-- PolyORB is free software; you  can  redistribute  it and/or modify it    --
-- under terms of the  GNU General Public License as published by the  Free --
-- Software Foundation;  either version 2,  or (at your option)  any  later --
-- version. PolyORB is distributed  in the hope that it will be  useful,    --
-- but WITHOUT ANY WARRANTY;  without even the implied warranty of MERCHAN- --
-- TABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public --
-- License  for more details.  You should have received  a copy of the GNU  --
-- General Public License distributed with PolyORB; see file COPYING. If    --
-- not, write to the Free Software Foundation, 51 Franklin Street, Fifth    --
-- Floor, Boston, MA 02111-1301, USA.                                       --
--                                                                          --
--
--
--
--
--
--
--
--                                                                          --
--                  PolyORB is maintained by AdaCore                        --
--                     (email: sales@@adacore.com)                           --
--                                                                          --
------------------------------------------------------------------------------

--  Helper functions to manage CORBA corbaloc references

with CORBA.Object;

package PolyORB.CORBA_P.CORBALOC is

   function Object_To_Corbaloc
     (Obj : CORBA.Object.Ref'Class)
     return CORBA.String;
   --  Convert reference to corbaloc, return corbaloc of best profile

end PolyORB.CORBA_P.CORBALOC;
```

6.9.2 PolyORB.CORBA_P.Naming_Tools

```
------------------------------------------------------------------------------
--                                                                          --
--                           POLYORB COMPONENTS                             --
--                                                                          --
--        P O L Y O R B . C O R B A _ P . N A M I N G _ T O O L S           --
--                                                                          --
--                                S p e c                                   --
--                                                                          --
--        Copyright (C) 2001-2007, Free Software Foundation, Inc.           --
--                                                                          --
-- PolyORB is free software; you  can  redistribute  it and/or modify it    --
-- under terms of the  GNU General Public License as published by the  Free --
-- Software Foundation;  either version 2,  or (at your option)  any  later --
-- version. PolyORB is distributed  in the hope that it will be  useful,    --
-- but WITHOUT ANY WARRANTY;  without even the implied warranty of MERCHAN- --
-- TABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public --
-- License  for more details.  You should have received  a copy of the GNU  --
-- General Public License distributed with PolyORB; see file COPYING. If    --
-- not, write to the Free Software Foundation, 51 Franklin Street, Fifth    --
-- Floor, Boston, MA 02111-1301, USA.                                       --
--                                                                          --
--
--
--
--
--
--
--
--                                                                          --
--                  PolyORB is maintained by AdaCore                        --
--                     (email: sales@@adacore.com)                           --
--                                                                          --
------------------------------------------------------------------------------

--  Wrappers for the COS Naming service to facilitate retrievel of object
--  references by IOR or by name.

with Ada.Finalization;

with CORBA.Object;
with CosNaming.NamingContext;

package PolyORB.CORBA_P.Naming_Tools is

   function Locate (Name : CosNaming.Name) return CORBA.Object.Ref;

   function Locate
     (Context : CosNaming.NamingContext.Ref;
      Name    : CosNaming.Name) return CORBA.Object.Ref;
   --  Locate an object given its name, given as an array of name components.

   function Locate
     (IOR_Or_Name : String; Sep : Character := '/') return CORBA.Object.Ref;

   function Locate
     (Context     : CosNaming.NamingContext.Ref;
```

```
      IOR_Or_Name : String;
      Sep         : Character := '/') return CORBA.Object.Ref;
   --  Locate an object by IOR or name. If the string does not start with
   --  "IOR:", the name will be parsed before it is looked up, using
   --  Parse_Name below.

   procedure Register
     (Name   : String;
      Ref    : CORBA.Object.Ref;
      Rebind : Boolean := False;
      Sep    : Character := '/');
   --  Register an object by its name by binding or rebinding.
   --  The name will be parsed by Parse_Name below; any necessary contexts
   --  will be created on the name server.
   --  If Rebind is True, then a rebind will be performed if the name
   --  is already bound.

   procedure Unregister (Name : in String);
   --  Unregister an object by its name by unbinding it

   type Server_Guard is limited private;
   procedure Register
     (Guard  : in out Server_Guard;
      Name   : String;
      Ref    : CORBA.Object.Ref;
      Rebind : Boolean := False;
      Sep    : Character := '/');
   --  A Server_Guard object is an object which is able to register a server
   --  reference in a naming service (see Register above), and destroy this
   --  name using Unregister when the object disappears (the program terminates
   --  or the Server_Guard object lifetime has expired).

   function Parse_Name
     (Name : String;
      Sep : Character := '/') return CosNaming.Name;
   --  Split a sequence of name component specifications separated with Sep
   --  characters into a name component array. Any leading Sep is ignored.

private
   --  implementation removed
end PolyORB.CORBA_P.Naming_Tools;
```

6.9.3 PolyORB.CORBA_P.Server_Tools

```
------------------------------------------------------------------------------
--                                                                          --
--                           POLYORB COMPONENTS                             --
--                                                                          --
--        P O L Y O R B . C O R B A _ P . S E R V E R _ T O O L S           --
--                                                                          --
--                                S p e c                                   --
--                                                                          --
--        Copyright (C) 2001-2006, Free Software Foundation, Inc.           --
--                                                                          --
-- PolyORB is free software; you  can  redistribute  it and/or modify it    --
-- under terms of the  GNU General Public License as published by the  Free --
-- Software Foundation;  either version 2,  or (at your option)  any  later --
-- version. PolyORB is distributed  in the hope that it will be  useful,    --
-- but WITHOUT ANY WARRANTY;  without even the implied warranty of MERCHAN- --
-- TABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public --
-- License  for more details.  You should have received  a copy of the GNU  --
-- General Public License distributed with PolyORB; see file COPYING. If    --
-- not, write to the Free Software Foundation, 51 Franklin Street, Fifth    --
-- Floor, Boston, MA 02111-1301, USA.                                       --
--                                                                          --
--
--
--
--
--
--
--
--                                                                          --
--                  PolyORB is maintained by AdaCore                        --
--                     (email: sales@@adacore.com)                           --
--                                                                          --
------------------------------------------------------------------------------

--  Helper functions for CORBA servers. Note that using this unit implies
--  using the Portable Object Adapter.

with CORBA.Object;
with PortableServer.POA;

package PolyORB.CORBA_P.Server_Tools is

   pragma Elaborate_Body;

   type Hook_Type is access procedure;
   Initiate_Server_Hook : Hook_Type;
   --  Access to a procedure to be called upon start up.
   --  See Initiate_Server for more details.

   procedure Initiate_Server (Start_New_Task : Boolean := False);
   --  Start a new ORB, and initialize the Root POA.
   --  If Start_New_Task is True, a new task will be created and
   --  control will be returned to the caller.  Otherwise, the ORB
   --  will be executing in the current context.
   --  If the Initiate_Server_Hook variable is not null, the
   --  designated procedure will be called after initializing the ORB,
```

```
   --  prior to entering the server loop.

   function Get_Root_POA return PortableServer.POA.Local_Ref;
   --  Return the Root_POA attached to the current ORB instance.

   procedure Initiate_Servant
     (S : PortableServer.Servant;
      R : out CORBA.Object.Ref'Class);
   --  Initiate a servant: register a servant to the Root POA.
   --  If the Root POA has not been initialized, initialize it.

   procedure Reference_To_Servant
     (R : CORBA.Object.Ref'Class;
      S : out PortableServer.Servant);
   --  Convert a CORBA.Object.Ref into a PortableServer.Servant.

   procedure Servant_To_Reference
     (S : PortableServer.Servant;
      R : out CORBA.Object.Ref'Class);
   --  Convert a PortableServer.Servant into CORBA.Object.Ref.

   procedure Initiate_Well_Known_Service
     (S    : PortableServer.Servant;
      Name : String;
      R    : out CORBA.Object.Ref'Class);
   --  Make S accessible through a reference appropriate for
   --  generation of a corbaloc URI with a named key of Name.

end PolyORB.CORBA_P.Server_Tools;
```

7 RT-CORBA

7.1 What you should know before Reading this section

This section assumes that the reader is familiar with the Real-Time CORBA specifications described in *[OMG02a]* and *[OMG03]*.

7.2 Installing RT-CORBA

The RT-CORBA library is installed as part of the installation of the CORBA personality. Note that you may have to select specific run-time options to enable full compliance with RT-CORBA specifications and ensure real time behavior.

7.3 Configuring RT-CORBA

This section details how to configure your application to use the RT-CORBA library.

7.3.1 `PolyORB.RTCORBA_P.Setup`

RT-CORBA specifications mandates the implementation provides a mechanism to set up some of its internals.

The package `PolyORB.RTCORBA_P.Setup` provides an API to set up the `PriorityMapping` and `PriorityTransform` objects.

```
------------------------------------------------------------------------------
--                                                                          --
--                           POLYORB COMPONENTS                             --
--                                                                          --
--              P O L Y O R B . R T C O R B A _ P . S E T U P               --
--                                                                          --
--                                 S p e c                                  --
--                                                                          --
--         Copyright (C) 2003-2004 Free Software Foundation, Inc.           --
--                                                                          --
-- PolyORB is free software; you  can  redistribute  it and/or modify it    --
-- under terms of the  GNU General Public License as published by the  Free --
-- Software Foundation;  either version 2,  or (at your option)  any  later --
-- version. PolyORB is distributed  in the hope that it will be  useful,    --
-- but WITHOUT ANY WARRANTY;  without even the implied warranty of MERCHAN- --
-- TABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public --
-- License  for more details.  You should have received  a copy of the GNU  --
-- General Public License distributed with PolyORB; see file COPYING. If    --
-- not, write to the Free Software Foundation, 59 Temple Place - Suite 330, --
-- Boston, MA 02111-1307, USA.                                              --
--                                                                          --
--
--
--
--
--
--
--
--
--                                                                          --
--                PolyORB is maintained by ACT Europe.                      --
```

```
--                    (email: sales@@act-europe.fr)                          --
--                                                                           --
-------------------------------------------------------------------------------

--  Implementation Notes: RTCORBA specifications defines objects that
--  are (Ada) programming language objects rather than CORBA
--  objects. Therefore the normal mechanism for coupling an
--  implementation to the code that uses it (an object reference) does
--  not apply. The implementation must provide specific mechanisms to
--  enable this coupling.
--
--  This package provides accessors to configure them. It supports the
--  following objects:
--  * PriorityMapping
--  * PriorityTransform

with RTCORBA.PriorityMapping;
with RTCORBA.PriorityTransform;

package PolyORB.RTCORBA_P.Setup is

   --  PriorityMapping

   type PriorityMapping_Access is
     access all RTCORBA.PriorityMapping.Object'Class;

   procedure Set_Priority_Mapping
     (Mapping : RTCORBA.PriorityMapping.Object'Class);
   pragma Inline (Set_Priority_Mapping);
   --  Set RT-ORB PriorityMapping object,
   --  overrides previous settings, if any.

   function Get_Priority_Mapping return PriorityMapping_Access;
   pragma Inline (Get_Priority_Mapping);
   --  Return RT-ORB PriorityMapping object.

   --  PriorityTransform

   type PriorityTransform_Access is
     access all RTCORBA.PriorityTransform.Object'Class;

   procedure Set_Priority_Transform
     (Transform : RTCORBA.PriorityTransform.Object'Class);
   pragma Inline (Set_Priority_Transform);
   --  Set RT-ORB global Priority Mapping object,
   --  overrides previous settings, if any.

   function Get_Priority_Transform return PriorityTransform_Access;
   pragma Inline (Get_Priority_Transform);
   --  Return RT-ORB global Priority Mapping object.

end PolyORB.RTCORBA_P.Setup;
```

7.4 RTCORBA.PriorityMapping

PolyORB provides different implementations of this specification:

- `RTCORBA.PriorityMapping.Direct` maps CORBA priorities directly to native priori-

ties. If the CORBA priority is not in `System.Priority'Range`, then the mapping is not possible.

- `RTCORBA.PriorityMapping.Linear` maps each individual native priority to a contiguous range of CORBA priorities, so that the complete CORBA priority range is used up for the mapping. See '`rtcorba-prioritymapping-linear.adb`' for more details.

7.5 RTCosScheduling Service

7.5.1 Overview

PolyORB provides an implementation of the RTCosScheduling service defined in *[OMG02a]*.

PolyORB uses some permission stated in the specifications to allow for an easy configuration of `ClientScheduler` and `ServerScheduler`, defined in the following sections.

Additional information on the use of the API may be found in the RTCosScheduling example in '`examples/corba/rtcorba/rtcosscheduling`'.

7.5.2 `RTCosScheduling::ClientScheduler`

Client side *activities* are defined in a configuration file, than can be loaded using '`RTCosScheduling.ClientScheduler.Impl.Load_Configuration_File`'

On the client side, the user can set up

- current task priority, using registered `PriorityMapping` object.

This file has the following syntax, derived from PolyORB configuration files syntax:

```
# Name of the activity
[activity activity1]

# Activity priority, in RTCORBA.Priority'Range
priority=10000
```

In this example, activity `activity1` is defined with priority `10'000`.

7.5.3 `RTCosScheduling::ServerScheduler`

Server side *POAs* and *objects* are defined in a configuration file, than can be loaded using '`RTCosScheduling.ClientScheduler.Impl.Load_Configuration_File`'

On the server side, the user can set up

- object priority, using registered `PriorityMapping` object.
- all RT-CORBA-specific POA configuration parameters:

This file has the following syntax, derived from PolyORB configuration files syntax:

```
# Name of the object
[object object1]

# Object priority, in RTCORBA.Priority'Range
priority=10000
```

In this example, object `object1` is defined with priority `10'000`.

```
# Name of the POA
[poa poa1]

# PriorityModelPolicy for POA
priority_model=CLIENT_PROPAGATED
default_priority=0 # not meaningful for CLIENT_PROPAGATED

# Threadpools attached to POA
threadpool_id=1

# Name of the POA
[poa poa2]

# PriorityModelPolicy for POA
priority_model=SERVER_DECLARED
default_priority=40

# Threadpools attached to POA
threadpool_id=2

# Name of the POA
[poa poa3]

# POA with no defined policies
```

In this example, Two POAs are defined: POA `poa1` will use the `CLIENT_PROPAGATED` PriorityModel Policy, default value is not meaningful for this configuration, `poa1` will use the Threadpool #1; POA `poa2` will use the `SERVER_DECLARED` PriorityModel Policy, default server priority is 40, `poa2` will use the Threadpool #2. Note that both policies are optional and can be omitted.

8 Ada Distributed System Annex (DSA)

8.1 What you should know before Reading this section

This section assumes the reader is familiar with annex E of the Ada 95 Reference Manual *[ISO95]*. To build DSA applications with PolyORB you will use a tool named `po_gnatdist`. This tool is documented in the GLADE's User Guide *[gla06]*.

8.2 Installing DSA application personality

Ensure PolyORB has been configured and then compiled with DSA application personality. To build the DSA application personality, see Chapter 2 [Installation], page 5.

8.3 A small example of a DSA application

In this section we will write a really simple client-server application using PolyORB DSA. The server will provide a `Remote Call Interface` composed of a single `Echo_String` function that will take a String and return it to the caller.

Here is the code for the server:
'`server.ads`':

```
package Server is
   pragma Remote_Call_Interface;
   function Echo_String (S : String) return String;
end Server;
```

'`server.adb`':

```
package body Server is

   function Echo_String (S : String) return String is
   begin
      return S;
   end Echo_String;

end Server;
```

And here is the code for the client:
'`client.adb`':

```
with Ada.Text_IO; use Ada.Text_IO;
with Server;
procedure Client is
begin
   Put_Line ("The client has started!");
   Put ("Thus spake my server upon me:");
   Put_Line (Server.Echo_String ("Hi!"));
end Client;
```

For more details about the distributed system annex please report yourself to the Ada 95 Reference Manual *[ISO95]*.

8.4 Building a DSA application with PolyORB

8.4.1 Foreword

The preferred way to build distributed application is using `po_gnatdist`. `po_gnatdist` is a tool that provides a configuration language, allowing the user to partition his program and specify various parameters for each partition.

8.4.2 Installing po_gnatdist

`po_gnatdist` is built and installed as part of your PolyORB installation provided the DSA personality is enabled.

8.4.3 Using po_gnatdist with PolyORB

For extensive documentation on the configuration language of `po_gnatdist` and usage of the `po_gnatdist` command, please report yourself to the GLADE User Guide. In this section we will only explain basic usage of `po_gnatdist` to compile the Echo example.

First we need to describe how we want to partition our application. For this we will create a `po_gnatdist` configuration file '`echo.cfg`':

```
configuration Echo is

   --  We declare a server partition that executes the server package ...

   Server_Partition : partition := (Server);

   --  ... and a client partition that executes the client main procedure

   Client_Partition : partition;
   procedure Client is in Client_Partition;

   --  The partitions' executables should be put in ./bin

   for Partition'Directory use "bin";

end Echo;
```

Now we are ready to build our distributed application with the command:

```
po_gnatdist echo.cfg
```

8.5 Running a DSA application

By default `po_gnatdist` will use the Ada starter. So if you have not specified `pragma Starter (None);` in the `po_gnatdist` configuration file, you should have a starter in your build dir, named after your configuration file. In this case you just have to run this program.

If you don't want to use the Starter and have specified `pragma Starter (None);` in your configuration file, then you should have, in your Partition'Directory, one binary for each of your partitions. You'll have to start each of these programs manually.

In both cases you must specify a name server for your application. You can use for example the one included in PolyORB: '`po_cos_naming`'.

Just ensure that you set the global environment variable `POLYORB_DSA_NAME_SERVICE` to an IOR URI referencing the running name server.

8.6 Configuring a DSA application

You can configure some parameters of your DSA applications in the file polyorb.conf. You will find these parameters in the section [dsa]:

`name_service = [IOR/corbaloc]`
: You can set this parameter instead of the environment variable `POLYORB_DSA_NAME_SERVICE`. Though if you use a Starter ensure that this parameter is set for all the partitions, as this is not done automatically as for the `POLYORB_DSA_NAME_SERVICE` environment variable.

`max_failed_requests = [integer]`
: Each partition will attempt a given number of requests to the name server before failing. This allows some time for every partition to register in the name server.

`delay_between_failed_requests = [duration in milliseconds]`
: As above, only this specifies the delay between requests.

`termination_initiator = [true/false]`
: Is this partition a termination initiator.

`termination_policy = [global_termination/deferred_termination/local_termination]`
: The termination policy for this partition.

`tm_time_between_waves = [duration in milliseconds]`
: The delay between termination waves.

`tm_time_before_start = [duration in milliseconds]`
: The delay before the termination manager starts sending waves.

`detach = [true/false]`
: If true, the partition will be detached.

`rsh_options = [string]`
: Options passed to the rsh command when using the module polyorb.dsa_p-remote_launch

`rsh_command = [string]`
: Which command should the module polyorb.dsa_p-remote_launch use to spawn remote programs.

9 MOMA

9.1 What you should know before Reading this section

This section assumes that the reader is familiar with the JMS specifications described in *[SUN99]*. MOMA is a thick adaptation of the JMS specification to the Ada programming language. It preserves most of its concepts.

9.2 Installing MOMA application personality

Ensure PolyORB has been configured and then compiled with MOMA application personality. See Chapter 4 [Building an application with PolyORB], page 11 for more details on how to check installed personalities.

To build the MOMA application personality, see Chapter 2 [Installation], page 5.

9.3 Package hierarchy

Packages installed in '`$INSTALL_DIR/include/polyorb/moma`' hold the MOMA API. MOMA is built around two distinct set of packages:

1. '`MOMA.*`' hold the public MOMA library, all the constructs the user may use.
2. '`POLYORB.MOMA_P.*`' hold the private MOMA library, these packages shall not be used when building your application.

10 Ada Web Server (AWS)

The documentation of this personality will appear in a future revision of PolyORB.

11 GIOP

11.1 Installing GIOP protocol personality

Ensure PolyORB has been configured and then compiled with GIOP protocol personality. See Chapter 4 [Building an application with PolyORB], page 11 for more details on how to check installed personalities.

To enable the configuration of the GIOP protocol personality, see Chapter 2 [Installation], page 5.

11.2 GIOP Instances

GIOP is a generic protocol that can be instantiated for multiple transport stacks. PolyORB proposes three different instances.

11.2.1 IIOP

Internet Inter-ORB Protocol (IIOP) is the default protocol defined by the CORBA specifications. It is a TCP/IP, IPv4, based protocol that supports the full semantics of CORBA requests.

11.2.2 SSLIOP

The SSLIOP protocol provides transport layer security for transmitted requests. Its provides encryption of GIOP requests.

To build the SSLIOP, it is required to activate SSL-related features when building PolyORB. See Chapter 2 [Installation], page 5 for more details.

Enabling security is completely transparent to a preexisting application, it is also possible to phase in secure communications by allowing incoming requests which are unsecured.

11.2.3 DIOP

Datagram Inter-ORB Protocol (DIOP) is a specialization of GIOP for the UDP/IP protocol stack. It supports only asynchronous (`oneway`) requests.

This protocol is specific to PolyORB. DIOP 1.0 is mapping of GIOP on top of UDP/IP. DIOP 1.0 uses GIOP 1.2 message format.

11.2.4 MIOP

Unreliable Multicast Inter-ORB Protocol (MIOP) *[OMG02b]* is a specialization of GIOP for IP/multicast protocol stack. It supports only asynchronous (`oneway`) requests.

11.3 Configuring the GIOP personality

GIOP personality is configured using a configuration file. See Section 4.2.1 [Using a configuration file], page 12 for more details.

Here is a summary of available parameters for each instance of GIOP.

11.3.1 Common configuration parameters

This section details configuration parameters common to all GIOP instances.

```
##############################################################################
# GIOP parameters
#

[giop]

#####################################################################
# Native code sets
#
# Available char data code sets:
#   16#00010001#   ISO 8859-1:1987; Latin Alphabet No. 1
#   16#05010001#   X/Open UTF-8; UCS Transformation Format 8 (UTF-8)
#
# Available wchar data code sets:
#   16#00010100#   ISO/IEC 10646-1:1993; UCS-2, Level 1
#   16#00010109#   ISO/IEC 10646-1:1993;
#                               UTF-16, UCS Transformation Format 16-bit form
#
#giop.native_char_code_set=16#00010001#
#giop.native_wchar_code_set=16#00010100#
#
# The following parameters force the inclusion of fallback code sets
# as supported conversion code sets. This is required to enable
# interoperability with ORBs whose code sets negotiation support is
# broken. See PolyORB's Users Guide for additional information.
#
#giop.add_char_fallback_code_set=false
#giop.add_wchar_fallback_code_set=false
```

11.3.2 IIOP Configuration Parameters

```
##############################################################################
# IIOP parameters
#

[iiop]

#####################################################################
# IIOP Global Settings

# Preference level for IIOP
#polyorb.binding_data.iiop.preference=0

# IIOP's default address
#polyorb.protocols.iiop.default_addr=127.0.0.1

# IIOP's default port
#polyorb.protocols.iiop.default_port=2809

# IIOP's alternate addresses
#polyorb.protocols.iiop.alternate_listen_addresses=127.0.0.1:2810 127.0.0.1:2820

# Default GIOP/IIOP Version
#polyorb.protocols.iiop.giop.default_version.major=1
#polyorb.protocols.iiop.giop.default_version.minor=2
```

```
################################################################
# IIOP 1.2 specific parameters

# Set to True to enable IIOP 1.2
#polyorb.protocols.iiop.giop.1.2.enable=true

# Set to True to send a locate message prior to the request
#polyorb.protocols.iiop.giop.1.2.locate_then_request=true

# Maximum message size before fragmenting request
#polyorb.protocols.iiop.giop.1.2.max_message_size=1000

################################################################
# IIOP 1.1 specific parameters

# Set to True to enable IIOP 1.1
#polyorb.protocols.iiop.giop.1.1.enable=true

# Set to True to send a locate message prior to the request
#polyorb.protocols.iiop.giop.1.1.locate_then_request=true

# Maximum message size before fragmenting request
#polyorb.protocols.iiop.giop.1.1.max_message_size=1000

################################################################
# IIOP 1.0 specific parameters

# Set to True to enable IIOP 1.0
#polyorb.protocols.iiop.giop.1.0.enable=true

# Set to True to send a locate message prior to the request
#polyorb.protocols.iiop.giop.1.0.locate_then_request=true
```

11.3.3 SSLIOP Configuration Parameters

11.3.3.1 Ciphers name

PolyORB's SSLIOP uses the OpenSSL library to support all recommended by CORBA 3.0.3 ciphers. OpenSSL library uses specific names for ciphers. The table below contains CORBA recommended ciphers names and its OpenSSL equivalents:

CORBA recommended ciphers	OpenSSL equivalent
TLS_RSA_WITH_RC4_128_MD5	RC4-MD5
SSL_RSA_WITH_RC4_128_MD5	RC4-MD5
TLS_DHE_DSS_WITH_DES_CBC_SHA	EDH-DSS-CBC-SHA
SSL_DHE_DSS_WITH_DES_CBC_SHA	EDH-DSS-CBC-SHA
TLS_RSA_EXPORT_WITH_RC4_40_MD5	EXP-RC4-MD5
SSL_RSA_EXPORT_WITH_RC4_40_MD5	EXP-RC4-MD5
TLS_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA	EXP-EDH-DSS-DES-CBC-SHA
SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA	EXP-EDH-DSS-DES-CBC-SHA

11.3.3.2 SSLIOP Parameters

```
################################################################################
# SSLIOP parameters
#
```

```
[ssliop]

#################################################################
# SSLIOP Global Settings

# SSLIOP's default port
#polyorb.protocols.ssliop.default_port=2810
# If no SSLIOP default address is provide, PolyORB reuses IIOP's
# address

# Private Key file name
#polyorb.protocols.ssliop.privatekeyfile=privkey.pem

# Certificate file name
#polyorb.protocols.ssliop.certificatefile=cert.pem

# Trusted CA certificates file
#polyorb.protocols.ssliop.cafile=cacert.pem

# Trusted CA certificates path
#polyorb.protocols.ssliop.capath=demoCA/certs

# Disable unprotected invocations
#polyorb.protocols.ssliop.disable_unprotected_invocations=true

#################################################################
# Peer certificate verification mode

# Verify peer certificate
#polyorb.protocols.ssliop.verify=false

# Fail if client did not return certificate. (server side option)
#polyorb.protocols.ssliop.verify_fail_if_no_peer_cert=false

# Request client certificate only once. (server side option)
#polyorb.protocols.ssliop.verify_client_once=false
```

11.3.4 DIOP Configuration Parameters

```
#################################################################
# DIOP Global Settings

# Preference level for DIOP
#polyorb.binding_data.diop.preference=0

# DIOP's default address
#polyorb.protocols.diop.default_addr=127.0.0.1

# DIOP's default port
#polyorb.protocols.diop.default_port=12345

# Default GIOP/DIOP Version
#polyorb.protocols.diop.giop.default_version.major=1
#polyorb.protocols.diop.giop.default_version.minor=2

#################################################################
# DIOP 1.2 specific parameters
```

```
# Set to True to enable DIOP 1.2
#polyorb.protocols.diop.giop.1.2.enable=true

# Maximum message size
#polyorb.protocols.diop.giop.1.2.max_message_size=1000

################################################################
# DIOP 1.1 specific parameters

# Set to True to enable DIOP 1.1
#polyorb.protocols.diop.giop.1.1.enable=true

# Maximum message size
#polyorb.protocols.diop.giop.1.1.max_message_size=1000

################################################################
# DIOP 1.0 specific parameters

# Set to True to enable DIOP 1.0
#polyorb.protocols.diop.giop.1.0.enable=true
```

11.3.5 MIOP Configuration Parameters

```
#############################################################################
# MIOP parameters
#

[miop]

################################################################
# MIOP Global Settings

# Preference level for MIOP
#polyorb.binding_data.uipmc.preference=0

# Maximum message size
#polyorb.miop.max_message_size=6000

# Time To Leave parameter
#polyorb.miop.ttl=15

# Multicast address to use
#polyorb.miop.multicast_addr=239.239.239.18

# Multicast port to use
#polyorb.miop.multicast_port=5678

# Set to True to enable MIOP
#polyorb.protocols.miop.giop.1.2.enable=false

# Maximum message size
#polyorb.protocols.miop.giop.1.2.max_message_size=1000
```

11.4 Code sets

This sections details the various steps required to add the support for new character code sets in PolyORB's GIOP personality. Please, refer to CORBA specifications (*[OMG04]*), par. 13.10 for more details on this topic.

11.4.1 Supported code sets

PolyORB supports the following list of code sets:

1. Available char data code sets:
   1. 16#00010001# ISO 8859-1:1987; Latin Alphabet No. 1
   2. 16#05010001# X/Open UTF-8; UCS Transformation Format 8 (UTF-8)
2. Available wchar data code sets:
   1. 16#00010100# ISO/IEC 10646-1:1993; UCS-2, Level 1
   2. 16#00010109# ISO/IEC 10646-1:1993; UTF-16, UCS Transformation Format 16-bit form

11.4.2 Incompatibility in code set support

Some ORB reports incompatiblity in code sets because fallback converters are not explicitely present in the reference. To work-around this issue, you may use the following parameters:

```
[giop]
giop.add_char_fallback_code_set=true
giop.add_wchar_fallback_code_set=true
```

11.4.3 Adding support for new code sets

PolyORB allows users to extend the set of supported native character code sets. Adding support for new character code set consists of the following steps:

1. Developing sets of Converters - special objects which do marshalling/unmarshalling operations of character data. At least two Converters are required: for direct marshalling character data in native code set and for marshalling/unmarshalling character data in fallback character code set (UTF-8 for char data and UTF-16 for wchar data). Additional Converters may be developed for marshalling character data in conversion code set.
2. Developing converter factory subprogram for each Converter.
3. Registering native code set, its native and fallback converters and optional conversions char sets and it's converters.

11.4.4 Character data Converter

Character data converter do direct marshalling/unmarshalling of character data (char or wchar - depending of `Converter`) into/from PolyORB's buffer. This allows to minimize speed penalty on character data marshalling.

Character data Converter for char data have the following API (from '`PolyORB.GIOP_P.Code_Sets.Converers`' package:

```
type Converter is abstract tagged private;

procedure Marshall
  (C      : Converter;
   Buffer : access Buffers.Buffer_Type;
   Data   : Types.Char;
   Error  : in out Errors.Error_Container)
   is abstract;

procedure Marshall
```

```
   (C      : Converter;
    Buffer : access Buffers.Buffer_Type;
    Data   : Types.String;
    Error  : in out Errors.Error_Container)
    is abstract;

procedure Unmarshall
   (C      : Converter;
    Buffer : access Buffers.Buffer_Type;
    Data   :    out Types.Char;
    Error  : in out Errors.Error_Container)
    is abstract;

procedure Unmarshall
   (C      : Converter;
    Buffer : access Buffers.Buffer_Type;
    Data   :    out Types.String;
    Error  : in out Errors.Error_Container)
    is abstract;
```

Marshall subprograms do marshalling of one character or string of character into the buffer. Unmarshall subprograms do unmarshalling of one character or string of characters from the buffer.

Note: Depending on item size of character data (char/wchar) and GIOP version marshalling/unmarshalling algorithms may vary. For several situations marshalling of string is not equivalent to marshalling its length and marshalling one by one each string's character. Please refere to GIOP specifications for more details.

If marshalling/unmarshalling fails, subprograms must set Error parameter to corresponding error, usually `Data_Conversion_E`.

Note: We recommend to always use Data_Conversion_E error code with Minor status 1.

All `Converters` (native, fallback and conversion) have similar API. Wchar data converters differ only in parameter type.

11.4.5 Converters factories

To create new converters, PolyORB uses special factory subprograms with the following profile:

```
function Factory return Converter_Access;
```

or

```
function Factory return Wide_Converter_Access;
```

This function must allocate a new `Converter` and initialize its state.

11.4.6 Registering new code sets

Registering new native character data code sets begins from registering new native character data code sets and its native and fallback `Converters`. This is done using `Register_Native_Code_Set`:

```
procedure Register_Native_Code_Set
   (Code_Set : Code_Set_Id;
    Native   : Converter_Factory;
    Fallback : Converter_Factory);
```

or

```
procedure Register_Native_Code_Set
  (Code_Set : Code_Set_Id;
   Native   : Wide_Converter_Factory;
   Fallback : Wide_Converter_Factory);
```

If you have additional conversion code sets Converters you may register it by calling Register_Conversion_Code_Set subprogram:

```
procedure Register_Conversion_Code_Set
  (Native     : Code_Set_Id;
   Conversion : Code_Set_Id;
   Factory    : Converter_Factory);
```

or

```
procedure Register_Conversion_Code_Set
  (Native     : Code_Set_Id;
   Conversion : Code_Set_Id;
   Factory    : Wide_Converter_Factory);
```

Note: because of incompatibility in the support of code sets negotiation in some ORB's it is recommend to recognize two boolean PolyORB's parameters:

```
[giop]
giop.add_char_fallback_code_set=false
giop.add_wchar_fallback_code_set=false
```

and also register fallback Converter as conversion Converter if the corresponding parameter set to True.

Finally, define your prefferred native character data code sets by parameters (only integer code sets codes now supported):

```
[giop]
giop.native_char_code_set=16#00010001#
giop.native_wchar_code_set=16#00010100#
```

12 SOAP

12.1 Installing SOAP protocol personality

Ensure PolyORB has been configured and then compiled with SOAP protocol personality. See Chapter 4 [Building an application with PolyORB], page 11 for more details on how to check installed personalities.

To enable the configuration of the SOAP application personality, see Chapter 2 [Installation], page 5.

12.2 Configuring the SOAP personality

SOAP personality is configured using a configuration file. See Section 4.2.1 [Using a configuration file], page 12 for more details.

Here is a summary of available parameters for each instance of SOAP.

```
##############################################################################
# SOAP parameters
#

[soap]

#####################################################################
# SOAP Global Settings

# Preference level for SOAP
#polyorb.binding_data.soap.preference=0

# SOAP's default address
#polyorb.protocols.soap.default_addr=127.0.0.1

# SOAP's default port
#polyorb.protocols.soap.default_port=8080
```

13 Tools

13.1 po_catref

`po_catref` is a utility for viewing components of a stringified reference (CORBA IOR, corbaloc or URI). Reference's component include reference to access an object through multiple protocols (e.g. CORBA IIOP, SOAP) and configuration parameters associated to a reference (e.g. GIOP Service Contexts).

```
Usage:
       po_catref <stringified reference>
```

Note: `po_catref` *can only process protocols PolyORB has been configured with.*

13.2 po_dumpir

`po_dumpir` is a utility for viewing the content of an instance of the CORBA Interface Repository.

```
Usage:
       po_dumpir <stringified reference>
```

Note: `po_dumpir` *will be compiled and installed only if the CORBA personality and the '*`ir`*' service is compiled. Please see Chapter 4 [Building an application with PolyORB], page 11 for more details on how to set up PolyORB.*

13.3 po_names

`po_names` is a stand-alone name server. It has an interface similar to CORBA COS Naming, without dragging any dependencies on CORBA mechanisms. This name server is to be used when the CORBA application personality is not required, e.g. with the DSA or MOMA application personalities.

Appendix A Performance considerations

This section discusses performance when using PolyORB. Many elements can be configured, See Chapter 4 [Building an application with PolyORB], page 11. By carefully selecting them, you can increase the throughput of your application.

We review some parameters that can impact performance.

- **Build options**:
  - You should not build PolyORB with debug activated for production environment.
- **Tasking policies**:
  - You should carefully select the tasking policy to reduce dynamic ressource allocation (tasks, entry points, etc.). See Chapter 5 [Tasking model in PolyORB], page 17.
- **Transport parameters**:
  - Setting `tcp.nodelay` to false will disable Nagle buffering.
- **GIOP parameters**:
  - Setting `polyorb.protocols.iiop.giop.1.X.locate_then_request`, where `X` is the GIOP version in use, to false will disable `Locate_Message`, reducing the number of requests exchanged,
  - Increasing `polyorb.protocols.iiop.giop.1.X.max_message_size`, where `X` is the GIOP version in use, will reduce GIOP fragmentation, reducing middleware processing.

Appendix B Conformance to standards

B.1 CORBA standards conformance

The OMG defines CORBA-compliant ORB as implementations of the CORBA specifications that supports CORBA Core and one mapping of CORBA's IDL.

Here is a summary of PolyORB's conformance issues with the latest CORBA specifications (revision 3.0, formal/02-06-01)

B.1.1 CORBA IDL-to-Ada mapping

PolyORB supports the IDL-to-Ada specification *[OMG01]*, with the following limitations in both CORBA API and the IDL-to-Ada compiler `idlac`:

- no support for abstract interfaces, object-by-value, context data;
- no support for CORBA Components;
- implemented API may present some divergences with current mapping.

Note: generated code is constrained by the limitations of the Ada compiler used. Please refer to its documentation for more information.

B.1.2 CORBA Core

This set encompasses chapters 1-11. Chapters 3 to 11 are normative.

- Chapter 3 describes OMG IDL syntax and semantics. See Section B.1.1 [CORBA IDL-to-Ada mapping], page 71 for a description of non-implemented features;
- Chapter 4 describes the ORB Interface.

  PolyORB partially supports this chapter.
- Chapter 5 describes Value Type Semantics.

  PolyORB does not support this chapter.
- Chapter 6 describes Abstract Interface Semantics.

  PolyORB does not support this chapter.
- Chapter 7 describes Dynamic Invocation Interface (DII)

  PolyORB supports only the following methods: `Create_Request`, `Invoke` and `Delete`.
- Chapter 8 describes Dynamic Skeleton Interface (DSI)

  PolyORB partially supports this chapter: this interface is fully implemented except for context data.
- Chapter 9 describes Dynamic Management of Any Values

  PolyORB partially supports this chapter: this interface is fully implemented except for object references and value types.
- Chapter 10 describes The Interface Repository

  PolyORB supports this chapter, except for the `ExtValueDef` interface, and all CORBA CCM related interfaces.
- Chapter 11 describes The Portable Object Adapter

  PolyORB supports this chapter with the following limitations:

- `USE_SERVANT_MANAGER` policy is partially supported: the `ServantLocator` object is not implemented;
- support for `SINGLE_THREAD` policy is incomplete, reentrant calls may not work;
- `Wait_For_Completion` and `Etherealize_Objects` are not taken into account in `PortableServer.POAManager`;
- `PortableServer.POAManagerFactory` API is not implemented.

B.1.3 CORBA Interoperability

This set encompasses chapters 12-16.

- See Section B.4 [CORBA-GIOP standards conformance], page 73 for more information on this point.

B.1.4 CORBA Interworking

This set encompasses chapters 17-21.

- Chapters 17 to 20 describe interoperability with Microsoft's COM/DCOM.

  PolyORB provides no support for these chapters.
- Chapter 21 describes `PortableInterceptor`.

  PolyORB provides partial support for this chapter.

B.1.5 CORBA Quality Of Service

This set encompasses chapters 22-24.

- Chapter 22 describes CORBA Messaging
- Chapter 23 describes Fault Tolerant CORBA
- Chapter 24 describes Secure Interoperability.

PolyORB provides no support for these chapters.

B.1.6 CORBA COS Services

COS Services are specifications of high level services that are optional extensions to the CORBA specification. They provide helper packages to build distributed applications. PolyORB implement the following COS Services:

- COS Event and TypedEvent;
- COS Naming;
- COS Notification;
- COS Time;

B.1.7 CORBA Specialized services

PolyORB supports the following specialized services:

- Unreliable Multicast (MIOP), proposed 1.0 specification *[OMG02b]*.
- RT-CORBA extensions, see Chapter 7 [RT-CORBA], page 45 for more information on this point.
- CORBA security extensions, see *[OMG]* for more information on this point.

B.2 RT-CORBA standards conformance

RT-CORBA specifications relies on the CORBA application personality. It inherits all its issues, and implementations notes.

In addition, here is a list of issues with the implementation of RT-CORBA static *[OMG02a]* and dynamic scheduling *[OMG03]* specifications.

- RT-CORBA static and dynamic scheduling (Chapter 2)

  Chapter 2 is common to these two specifications. It describes key mechanisms of RT-CORBA that are common to both specifications.

  PolyORB partially implements this chapter from section 2.1 up to section 2.10. PolyORB does not provide support for all connection-related policies.

  See implementation notes in the different package specifications for more details.
- RT-CORBA static scheduling (Chapter 3)

  PolyORB supports this chapter.
- RT-CORBA dynamic scheduling (Chapter 3)

  PolyORB does not support this chapter.

B.3 CSIv2 standards conformance

PolyORB supports IIOP/SSL.

B.4 CORBA/GIOP standards conformance

GIOP supports part of the CORBA Interoperability specification, from chapters 12 to 16 of CORBA specifications.

Chapter 12 defines general concepts about ORB interoperability. It defines *interoperbility-compliant ORB* as ORB that supports:

- API that supports the construction of request-level inter-ORB bridges, Dynamic Invocation Interface, Dynamic Skeleton Interface and the object identity operations described in the Interface Repository. See Section B.1 [CORBA standards conformance], page 71 for more details.
- IIOP protocol as defined in chapter 15.

Support for other components is optional.

- Chapter 13 describes the ORB Interoperability Architecture.

  PolyORB fully supports this chapter.
- Chapter 14 describes how to build Inter-ORB Bridges.

  PolyORB fully supports this chapter.
- Chapter 15 describes the General Inter-ORB Protocol (GIOP).

  PolyORB supports GIOP version 1.0 to 1.2, the CDR representation scheme. Support for IOR and `corbaloc` addressing mechanisms is supported in CORBA personality, see Chapter 6 [CORBA], page 23 for more details.

  PolyORB does not support the optional IIOP IOR Profile Components, Bi-directional GIOP. PolyORB also does not support fragmentation in GIOP 1.1.
- Chapter 16 describes the DCE ESIOP protocol.

  PolyORB does not support this optional chapter.

B.5 SOAP standards conformance

The documentation of the SOAP standards conformance of PolyORB will appear in a future revision of PolyORB.

Appendix C References

Appendix D GNU Free Documentation License

Version 1.1, March 2000

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the

Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties – for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

Heading 6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute

the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright (c) YEAR YOUR NAME.
> Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being LIST"; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Index

A

Application personalities 9
AWS, Ada Web Server 9, 55

C

Code sets, GIOP 61
Configuration, CORBA 33
Configuration, GIOP 57
Configuration, PolyORB 11
Conventions 2
CORBA 9, 23
`CORBA COS Naming` 24
CORBA IDL-to-Ada mapping 71
CORBA, COS Services 9, 72
CORBA, Server-side exception 37
CORBA, Specialized services 72
`CORBA::Unknown` 37

D

Debug information 13
DIOP 10, 57
DSA, Distributed System Annex 9, 49

E

Exceptions 14

F

Free Documentation License, GNU 77

G

GIOP 10, 57, 58
GNU Free Documentation License 77

I

`idlac` 23
IIOP 10, 57

L

License, GNU Free Documentation 77

M

MIOP 10, 57, 72
MOMA, Message Oriented Middleware for Ada 9, 53

P

Personalities 9
`po_catref` 67
`po_cos_naming` 24
`po_dumpir` 67
`po_ir` 25
`po_names` 67
PolyORB 3
`polyorb-config` 14
'`polyorb.conf`' 12
`PolyORB.CORBA_P.CORBALOC` 39
`PolyORB.CORBA_P.Naming_Tools` 40
`PolyORB.CORBA_P.Server_Tools` 42
`PolyORB.RTCORBA_P.Setup` 45
`POLYORB_CONF` 12
Protocol personality 9
Protocol personality, activation 13

R

Ravenscar 11, 17
RT-CORBA 9, 45
`RTCORBA.PriorityMapping` 46
RTCosScheduling Service 47

S

SOAP 10, 65
SSLIOP 10, 57

T

Tasking model 17
Tasking runtime 17
Typographical conventions 2