PolyORB User's Guide

Jérôme Hugues

Table of Contents

About This Guide

This guide describes the use of PolyORB, a middleware that enables the constriction 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, provided by PolyORB.
- Chapter 5 [CORBA], page 17 describes PolyORB's implementation of OMG's CORBA.
- Chapter 6 [GIOP], page 29 describes PolyORB's implementation of GIOP, the protocol defined as part of CORBA.
- Chapter 7 [SOAP], page 33 describes PolyORB's implementation of SOAP.
- Chapter 8 [Tools], page 35 describes PolyORB's tools.
- Appendix B [GNU Free Documentation License], page 39, 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 website: `http://libre.act-europe.fr/polyorb`

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
- 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 95 compiler: GNAT 3.16a1 (or later), GNAT 5.02a (or later), GCC 3.4.0 (or later).

Optional:

- XmlAda (`http://libre.act-europe.fr/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 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. '`doc`') directory to build the examples (resp. the documentation). The build process will only build examples that correspond to the personalities you configured.

Note: you may also install PolyORB's documentation in standard location typing `make install`.

2.4.1 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, SRP

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

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

  Available services: event, ir, naming, time

  e.g. '`--with-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.

- '`--enable-shared`': build shared libraries.
- '`--enable-debug`': enable debugging information generation and supplementary run-time checks.

2.4.2 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

`CXXCPP`, `CXXCPPFLAGS`: the preprocessor used by idlac (only when setting up the CORBA application personality).

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.

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. In addition to the requirements above, they will need autoconf 2.57 or newer, and automake 1.6.3 or newer.

2.5 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.

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 (*[OMG02]*). It is now a well-known and well-established specification, used in a wide range of industrial applications.

PolyORB provides a CORBA-compliant implementation based on mapping of the IDL language version 1.2 described in *[OMG01]* and CORBA core specifications.

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,
- 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.

4.1.2 Middleware tasking policies

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

- `No_Tasking`: There is only one task in middleware, processing all requests.
- `Thread_Per_Sessions`: One task monitors communication entities. One task is spawned for each active connection. This task handles all incoming requests on this connection.
- `Thread_Per_Sessions`: 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.

4.1.3 Object Adapter

TO BE WRITTEN

4.1.4 Linking protocol personalities to executable

TO BE WRITTEN

4.1.5 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.

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.

  A variable `Var.Iable` in section `[Sec]` can be overridden by setting environment variable `"POLYORB_SEC_VAR_IABLE"`. 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 elaboration. Then, proper configuration parameters are selected.

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=disable
binding_data.iiop=disable
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 UIMPC.

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 propagations 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] [--version|-v] [--config] [--libs] [--cflags]
       [--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
              Print PolyORB's installation prefix.

       --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.

--help  Print help message.
```

5 CORBA

5.1 What you should know before Reading this section

This section assumes that the reader is familiar with the CORBA specifications described in *[OMG02a]* and the *IDL-to-Ada* mapping defined in *[OMG01]*.

5.2 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.

5.3 Usage of `idlac`

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

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

SYNOPSIS
       idlac [-E] [-d] [-i] [-k] [-p] [-q] [-noir] 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.

       -k      Keep temporary files.

       -p      Produce source on standard output.

       -q      Be quiet.

       -noir   Don't generate code for interface repository.

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

       -I dir  Shortcut for -cppargs -I dir.
```

`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.

5.4 Resolving names in a CORBA application

PolyORB implements the CORBA COS Naming service.

5.4.1 `po_cos_naming`

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

5.4.2 Using the COS Naming

PolyORB provides a helper package to manipulate the COS Naming in your applications. See Section 5.7 [PolyORB specific APIs], page 23 for more details.

5.5 Building a CORBA application with PolyORB

5.5.1 `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

5.5.1.1 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);
};
```

5.5.1.2 Implementation code for the `echo` object

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

```
with CORBA;
with PortableServer;

package Echo.Impl is

   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;


with Ada.Text_IO;

with Echo.Skel;
pragma Elaborate (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.*

5.5.1.3 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 4.1.5 [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.

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

with Echo;

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

procedure Client is
   use Ada.Command_Line;
   use Ada.Text_IO;

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

begin
   CORBA.ORB.Initialize ("ORB");
   if Argument_Count /= 1 then
      Put_Line ("usage : client <IOR_string_from_server>");
      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));

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;
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.

```
with Ada.Text_IO;

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

with PortableServer.POA;
with PortableServer.POAManager;

with Echo.Impl;

--  Setup server node: use no tasking default configuration

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

procedure Server is
begin
   CORBA.ORB.Initialize ("ORB");

   declare
      Root_POA : PortableServer.POA.Ref;

      Ref : CORBA.Object.Ref;

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

```
   begin
      --  Retrieve Root POA

      Root_POA := PortableServer.POA.To_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))
         & "'");

      --  Launch the server

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

5.5.1.4 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 !
  ```

5.5.2 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.

5.6 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).

5.6.1 Configuring PolyORB

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

5.6.2 Configuring GIOP protocol stack for PolyORB

The GIOP protocol is separated from the CORBA application personality. See Section 6.3 [Configuring the GIOP personality], page 29 for more information on GIOP's configuration.

5.7 PolyORB's specific APIs

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

- Section 5.7.1 [PolyORB.CORBA_P.Naming_Tools], page 24:
  This package defines helper functions to ease interaction with CORBA COS Naming.
- Section 5.7.2 [PolyORB.CORBA_P.Server_Tools], page 26:
  This package defines helper functions to ease set up of a simple CORBA Server.

5.7.1 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) 2001-2003 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.                                              --
--                                                                          --
-- As a special exception,  if other files  instantiate  generics from this --
-- unit, or you link  this unit with other files  to produce an executable, --
-- this  unit  does not  by itself cause  the resulting  executable  to  be --
-- covered  by the  GNU  General  Public  License.  This exception does not --
-- however invalidate  any other reasons why  the executable file  might be --
-- covered by the  GNU Public License.                                      --
--                                                                          --
--                PolyORB is maintained by ACT Europe.                      --
--                    (email: sales@@act-europe.fr)                          --
--                                                                          --
------------------------------------------------------------------------------

--  This package allows an object to be chosen either by its IOR or by
--  its name in the naming service.

--  $Id: //droopi/main/cos/naming/polyorb-corba_p-naming_tools.ads#5 $

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   : in String;
       Ref    : in CORBA.Object.Ref;
       Rebind : in Boolean := False;
       Sep    : in 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   : in String;
       Ref    : in CORBA.Object.Ref;
       Rebind : in Boolean := False;
       Sep    : in 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;
```

5.7.2 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) 2001-2003 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.                                              --
--                                                                          --
-- As a special exception,  if other files  instantiate  generics from this --
-- unit, or you link  this unit with other files  to produce an executable, --
-- this  unit  does not  by itself cause  the resulting  executable  to  be --
-- covered  by the  GNU  General  Public  License.  This exception does not --
-- however invalidate  any other reasons why  the executable file  might be --
-- covered by the  GNU Public License.                                      --
--                                                                          --
--                PolyORB is maintained by ACT Europe.                      --
--                    (email: sales@@act-europe.fr)                          --
--                                                                          --
------------------------------------------------------------------------------

--  Helper functions for CORBA servers.

--  $Id: //droopi/main/src/corba/polyorb-corba_p-server_tools.ads#9 $

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.Ref;
   --  Return the Root_POA attached to the current ORB instance.

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

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

end PolyORB.CORBA_P.Server_Tools;
```

6 GIOP

6.1 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.

6.2 GIOP Instances

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

6.2.1 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.

6.2.2 DIOP

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

6.2.3 MIOP

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

6.3 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.

6.3.1 IIOP Configuration Parameters

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

```
[iiop]

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

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

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

# 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
```

6.3.2 DIOP Configuration Parameters

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

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

# 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
```

6.3.3 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
```

7 SOAP

7.1 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.

7.2 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 port
#polyorb.protocols.soap.default_port=8080
```

8 Tools

8.1 po_catref

`po_catref` is a utility for viewing components of a stringified reference (CORBA IOR, corbaloc or URI).

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

8.2 po_names

`po_names` is a stand-alone name server. It has an interface similar to CORBA COS Naming, without dragging any dependences 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 References

1. [DB98] B. Dobbing and A. Burns. The Ravenscar tasking profile for high integrity real-time programs. In *Proceedings of SigAda'98*, Washington, DC, USA, November 1998.
2. [ISO95] ISO. *Information Technology – Programming Languages – Ada.* ISO, February 1995. ISO/IEC/ANSI 8652:1995.
3. [Obr03] P. Obry. Ada Web Server (AWS) 1.3, 2003.
4. [OMG01] OMG. *Ada Language Mapping Specification, v1.2.* OMG, October 2001. OMG Technical Document formal/2001-10-42.
5. [OMG02a] OMG. *The Common Object Request Broker: Architecture and Specification, revision 3.0.2.* OMG, December 2002. OMG Technical Document formal/2002-12-02.
6. [OMG02b] OMG. *unreliable Multicast InterORB Protocol specification.* OMG, 2002. OMG Technical Document ptc/03-01-11.
7. [SUN99] SUN. Java Message Service, 1999.
8. [W3C00] W3C. *Extensible Markup Language (XML) 1.0*, October 2000. W3C recommandation.
9. [W3C03] W3C. *Simple Object Access Protocol (SOAP) 1.2: primer*, june 2003. W3C recommandation.

Appendix B 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