Support Readiness Document
Java™ 2 Standard Edition, Version 1.4
CORBA, Version 2.3.1, Support

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303
U.S.A. 650-960-1300

March 2002

Table of Contents

Preface

This document provides support readiness information for the Common Object Request Broker Architecture (CORBA), version 2.3.1, as it is implemented in the Java 2 Platform, Standard Edition (J2SE), reference implementation, version 1.4. This document includes support readiness information for both parts of CORBA: Java Interface Definition Language (Java IDL) and Remote Method Invocation over Internet Inter-ORB Protocol (RMI-IIOP). Separate support readiness documents were created for earlier versions of both Java IDL and RMI-IIOP.

This document is not designed to provide comprehensive product training. Instead, it focuses on issues immediately relevant to support, such as changes in this version of the product, using and troubleshooting the product, and installing and configuring the product. For pointers to other documentation, see Section 7 “Reference Information.”

The information contained in this Support Readiness Document (SRD) is current at the time of printing. Since SRDs are typically prepared in advance of the First Customer Ship (FCS) date, there may be more recent or complete information available from the resources mentioned in the SRD.

1 CORBA Overview

CORBA is a framework developed by the Object Management Group (OMG) with the primary goal of collapsing the boundary created by different computer languages and platforms.

In a distributed application, the various functions of the system can be developed using different computer languages, such as Java, C++, C, and so on. These functions can be run on various operating systems, such as the Solaris™ Operating Environment (OE), Microsoft Windows, HP-UX, and so on. Using CORBA, these different language components can interoperate as heterogeneous pieces of functionality.

1.1 Overview

The J2SE, v. 1.4, core contains a CORBA, v. 2.3.1, compliant Object Request Broker (ORB) and other features, such as Common Object Services (COS), Naming Service, and Portable Interceptors.

The J2SE, v. 1.4, CORBA implementation can be programmed in two different ways:

- Using the Java Interface Definition Language (Java IDL)
- Using Remote Method Invocation over Internet Inter-ORB Protocol (RMI-IIOP)

1.1.1 RMI-IIOP

In this model, the distributed object programming will be similar to RMI programming with the flexibility of using either:

- IIOP - the CORBA wire protocol
- Java Remote Method Protocol (JRMP) - the RMI wire protocol

Note – RMI-IIOP is intended to allow interoperability between RMI and CORBA. JRMP, which is the default communication protocol used in RMI, is not supported in CORBA. By providing both protocols, a developer can switch between the IIOP and JRMP protocols within a single program, if required.

Note – The Enterprise JavaBeans™ (EJB™) specification requires RMI-IIOP rather than standard RMI.

1.1.2 Java IDL

In this model the user can define language-neutral interfaces using an OMG-defined IDL. These language-neutral interfaces can be mapped into a particular language; in this case, it can be mapped to Java.

1.2 Features, Advantages, and Benefits

The main advantages of using CORBA are:

- Language-neutral, platform-neutral programming for easy integration with legacy systems
- Portable across various vendor ORBs
- Well-defined, distributed object architecture
- Industry-wide acceptance due to management by OMG, a consortium of over 800 companies
- Interoperability among different vendor ORBs

The important features of the J2SE, v. 1.4, CORBA implementation include:

- Portable Object Adapters (POA) - These help to build object implementations that are portable across different vendor ORBs. The POA framework is a powerful feature; it can be used in various ways to set different policies to handle object implementations. Object implementations are generally referred to as *servants*.
- Portable Interceptors - Portable Interceptors are hooks to the ORB to intercept the ORB requests, replies, creation, and so on.
- Interoperable Naming Service (INS) - INS extends the basic COS Naming service to provide a standard, easy-to-use naming service.
- General Inter-Orb Protocol (GIOP), version 1.2 - This is a standard CORBA protocol for on-the-wire exchange of requests and replies.

1.3 CORBA Tools Provided with J2SE, V. 1.4

The following CORBA tools are provided with J2SE, v. 1.4:

- `idlj` compiler - Generates Java stubs and skeletons from the IDL interfaces.
- `rmic` compiler - Generates IIOP-capable stubs, skeletons, and ties from Java RMI interfaces.
- `tnameserv` (Transient Naming Service) - Supports INS. We recommend using the ORB Daemon's naming service (`orbd`) instead of `tnameserv`. The `tnameserv` tool is provided for backward compatibility.
- `orbd` (ORB Daemon) - Provides a naming service, both transient and persistent, and a server manager to locate and activate persistent servers.
- Server Tool - Used for registering persistent servers with ORBD.

1.4 Features or Services Not Provided

1.4.1 Interoperability Using GIOP Not Tested

One of the main advantages of CORBA is interoperability, using the standard GIOP protocol. We have not done extensive testing to prove that we interoperate with all vendors. Successful testing for interoperability has been done using ORBs from Borland, Hitachi, and IBM.

1.4.2 Some CORBA Services Not Provided

We ship only the COS Naming Service, which is a bare minimum service required to use CORBA. We do not ship other CORBA services, such as CosEventService, CosTraderService, and so on.

1.4.3 No Interface Repository

We do not ship an interface repository, which is useful for Dynamic Invocation Interface (DII) and Dynamic Skeleton Interface (DSI) programming.

1.5 Introduction to CORBA

The following links provide a good introduction to CORBA:

- Information on the OMG's vision and architecture
  http://www.omg.org/oma/
- Introduction to CORBA
  http://www.omg.org/gettingstarted/
- Status of various CORBA interoperability projects
  http://www.omg.org/interoperability_testing/
- Free CORBA downloads that allow you to see CORBA in action
  http://www.omg.org/technology/corba/corbadownloads.htm

1.6 Other Introductory Material

There are tutorials covering all the features shipped as part of the J2SE, v. 1.4, CORBA implementation. Please visit the CORBA webpage for tutorials at:

http://java.sun.com/j2se/1.4/docs/guide/idl/

1.7 Specialized Terminology

Throughout this document, several abbreviations are used. The following is a brief description of the abbreviations:

- OMG - Object Management Group, a consortium of over 800 technology companies managing extensions to the Object Management Architecture (OMA).
- CORBA - Common Object Request Broker Architecture defined by the OMG.
- ORB - Object Request Broker, an important part of the CORBA architecture.
- PI - Portable Interceptors, which are used to intercept ORB requests and replies.
- INS- Interoperable Naming Service, an extension of the basic COS Naming Service.
- DynAny - Dynamic ANY, a convenience API set to build and traverse complex ANY objects.
- GIOP - General Inter-Orb Protocol, which is standard on the wire protocol defined by OMG.
- IIOP - Internet Inter-Orb Protocol, GIOP on TCP/IP.
- JRMP - Java Remote Message Protocol, the standard protocol for RMI.
- `idlj` - IDL to Java compiler tool shipped with J2SE, v. 1.4.
- ORBD - ORB Daemon.
- Servant - Object Implementation.
- IOR - Interoperable Object References, which contains information for contacting the object, such as host, port, object key, and so on. There can be multiple profile information, such as alternate host and port information.

2 Product Changes for the J2SE, V. 1.4, CORBA Implementation

2.1 Changes and New Features Since J2SE, V. 1.3

The changes in CORBA features since J2SE, v. 1.3, are described at:

`http://java.sun.com/j2se/1.4/docs/guide/idl/jidlChanges.html`

2.2 Bugs Fixed in J2SE, Version 1.4, CORBA

TABLE 2-1 lists the bugs and RFEs that have been implemented in J2SE, v. 1.4.

TABLE 2-1 Bugs and RFEs Implemented in J2SE, v. 1.4

Bug/RFE	**BugID**	**Synopsis**
bug	4373899	JTS failure caused by ORB transaction propagation bug.
RFE	4129245	JavaIDL implementation changes required.
bug	4599666	Mismatch in Source File Generation to Class Files in `rt.jar`.
bug	4437784	`JCK13a 3 api/javax_rmi/PortableRemoteObject/failed` with `rmi.RemoteException b59.`
bug	4236554	`RepositoryId.createForAnyType` does not handle some `IDLTypes`.
Bug	4523004	`rmic` generates bad code for local optimization of void return type.
Bug	4385089	Error in registration of transaction `Current` results in null `Current`.
Bug	4396928	PIORB throws `NullPointerException`.

TABLE 2-1 Bugs and RFEs Implemented in J2SE, v. 1.4 *(Continued)*

Bug	4392735	`ServerRequestInfo.get_server_policy` throws `NO_IMPLEMENT` exception.
Bug	4372163	Cannot create instance of stateful session when first instance discarded by server.
Bug	4228125	Cannot make a remote call with a large array of bytes passed as an argument.
RFE	4129272	API changes/additions to `org.omg.CORBA` required for RMI/IIOP.
RFE	4129275	API changes/additions to `org.omg.CORBA` required for JTS and IDLx extensions.
Bug	4119129	Invalid processing of little endian reply or request messages.
Bug	4256038	Implementation package renaming to avoid collisions.
Bug	4266054	Unable to create an instance of non-SUN ORB.
Bug	4296792	Remove non-standard APIs from `org/omg/` package.
Bug	4233362	Bad code generated for argument copies in local stubs.
Bug	4191205	`read_Object(Class stubClass)` returns wrong stubs.
Bug	4430062	Client hangs/server thread dies when server attempts return non-serializable object.
Bug	4452578	RI hangs when a return object is not serializable.
Bug	4485936	`SystemExceptions` raised in postinvoke or sri end points causes memory leak.
Bug	4412097	`ClientDelegate` missing implementation of `CORBA.Object._hash`.
Bug	4419578	`-falltie` option broke when file has multiple interfaces.
Bug	4517874	REGRESSION: RI 1.3_01 failure with J2SE, v. 1.4.
bug	4461743	ORB versioning broken in Merlin/J2EE, v. 1.3, Beta 2.
bug	4360254	Need OMG Issue 3681 fixed in JDK, v. 1.3, ORB.
bug	4372499	Public classes are missing in JDK, v. 1.4, build b32.
bug	4427976	Duplicate source file in JDK workspace breaks the Javadoc build.
bug	4459161	INS assumes `LocateRequest` for `NameService` with `corbaloc` URL.
bug	4184740	Wrong marshalling exception thrown.
Bug	4228093	Application exception not propagated to the client (RMI-IIOP-POA).

TABLE 2-1 Bugs and RFEs Implemented in J2SE, v. 1.4 *(Continued)*

Bug	4187986	Need support for unmarshaling local persistent objects.
Bug	4356662	`BAD_OPERATION` in `examples.hello` interoperation.
Bug	4228097	Transaction not propagated between RMI objects (RMI-IIOP-POA).
Bug	4228100	Incorrect interaction between ORB and JTS for local invocation (RMI-IIOP-POA).
Bug	4262402	Cannot create additional persistent POA if `poaids.db` file is present.
Bug	4236985	`rmic` generates wrong stubs/ties for serializable arguments.
Bug	4262822	`SecureRandom` function implementation is too slow.
Bug	4221548	`TypeCodeImpl.equal` does not handle aliases types.
Bug	4210101	`JavaIDL` does not support `TypeCode` creation in singleton for all types.
Bug	4468349	Wrong `Stub/Tie` code generated when argument type is static nested class.
Bug	4189780	ORB's use of system properties prevent multiple ORBs in same VM.
Bug	4394799	POA-enabled `Ties` should not fail shutdown if no POA is used.
Bug	4275167	Same build flag in Swing and CORBA makefiles can cause build problems on Microsoft Windows.
Bug	4105856	Ignores GIOP Reply message encoded as little-endian.
bug	4105579	Source for package `org.omg.CORBA.ORBPackage` is missing some doc comments.
Bug	4105589	Source for package `org.omg.CosNaming` is missing some documentation `comments`.
Bug	4105584	Source for package `org.omg.CORBA.TypeCodePackage` is missing some doc `comments`.
Bug	4105591	Source for package `org.omg.CosNaming.NamingContextPackage` missing some comments.
Bug	4365188	Bugs in RMI-IIOP Serialization protocol prevents Object Evolution.
Bug	4348378	`Tnameserv` tool is not internationalized.
Bug	4303282	Removal/documenting of `org.omg.CORBA` packages in question from JDK core.

TABLE 2-1 Bugs and RFEs Implemented in J2SE, v. 1.4 *(Continued)*

Bug	4290667	`NullPointerException` thrown at `com.sun.corba.se.internal.core.Profile`.
RFE	4312958	Pure ORB support for `J2EE.next`.
Bug	4321532	Race condition in `IIOPConnection.java`.
Bug	4502971	`ServerRequestInfo::request_id` must be unique.
Bug	4533469	`CORBA BAD OPERATION` server errors seen with CTS appclient/deploy tests.
Bug	4479114	`CORBA.portable.ObjectImpl._non_existent()` throws `OBJECT_NOT_EXIST`.
Bug	4456086	Server thread hangs when fragments don't complete because client-side error.
Bug	4450059	`RequestInfoImpl` uses `Class.forName` incorrectly.
Bug	4430551	`Interceptors.idl` contains module IOP which should be in `IOP.idl`.
Bug	4418740	Co-located `ImplBase` calls are not ORB-mediated.
Bug	4409028	Server side `PICurrent` usage is failing.
Bug	4398869	Exception from PI.
Bug	4395809	`CONN_CLOSE_REBIND` can lead to infinite loops in client ORB.
Bug	4395812	`createSystemExceptionResponse` from `ORB.process` does not work with PI.
Bug	4395814	`COMM_FAILURE` in `ClientDelegate.createRequest` may cause infinite loop.
Bug	4395813	Delete connection if socket open fails in `ConnectionTable.getConnection`.
Bug	4393695	Client interceptors not executed for Object pseudo-ops.
Bug	4393382	`ClienRequestInfo.effective_target()` needs to be optimized.
Bug	4394520	`codec.encode_value` fails when passed an `Any` (from singleton ORB) holds objrefs.
Bug	4379522	PI - Unbalanced call stack.
Bug	4419283	`_non_existent` throws `NullPointerException` in `poa/retain/defaultServant` config.
Bug	4417873	PI: Need to report `TRANSIENT/3` when request cancelled.
Bug	4409926	`ClientRequestInfo::effective_target` should return generic type for performance.

TABLE 2-1 Bugs and RFEs Implemented in J2SE, v. 1.4 *(Continued)*

Bug	4395696	`Util.isLocal` should return false.
Bug	4395811	`ThreadDeath` may cause interceptor ending points not to be called.
Bug	4384985	`InterceptorList.sortInterceptors` gets array out-of-bounds exception.
Bug	4386041	`ClientRequestInfo.get_reply_service_context` gets `NullPointerException` with DII.
RFE	4070259	IOR hostname should be configurable.
Bug	4419994	Problems with GIOP request versioning and location forward.
Bug	4384995	`GIOP.get*ServerPort` should complain if called before endpoints initialized.
Bug	4384988	`DefaultSocketFactory.createServerSocket` should complain if not given `IIOP_CLEAR`.
Bug	4226624	`JavaIDL` orb unnecessarily retries requests that result in system exceptions.
Eou	4486041	`.init()` could provide better failure diagnostics.
Bug	4429899	Replace PI doc ptc/00-08-06 with ptc/2001-03-04 in docs.
Bug	4105571	Source for package `org.omg.CORBA` is missing some documentation comments.
Bug	4397543	GIOP 1.0 reply to a GIOP 1.1 request CORBA `org.omg`.
Bug	4473714	ORB versioning required for JDK 1.3.1_01 and `PutField/GetField`.
Bug	4418763	Stream duplication can cause `INTERNAL`.
Bug	4480483	`javax.rmi.CORBA.Util.writeAny` should include `repId` in `TypeCode`.
Bug	4460764	`ClassDesc` written incorrectly.
Bug	4423950	Class evolution with `PutField/GetField` broken.
Bug	4415491	Fix for 4397033 can cause class loading failures.
Bug	4483833	ORB memory leak when invoking inactive IOR.
Bug	4484193	`ORB.string_to_object` raises `INV_OBJREF` for a valid `corbaloc` URL.
Bug	4403607	Performance Bug: Utility package prefix change.
Bug	4478497	CORBA method invocations work unstable.
Bug	4474942	CORBA makefile references non-existent classic VM.

TABLE 2-1 Bugs and RFEs Implemented in J2SE, v. 1.4 *(Continued)*

Bug	4464481	Incorrect Parsing of `-Dorg.omg.CORBA.ORBInitRefSvcs`.
Bug	4447979	`Svcs` registered using `orb.register_initial_reference()` are unavailable for INS.
Bug	4404956	`to_string` operation incorrectly stringified a `Name` with empty kind field.
Bug	4403619	`Utility.loadClassOfType()` results in Null Pointer Exceptions in some cases.
Bug	4398234	`to_name` returns an incorrect `NameComponent`.
Bug	4398219	`to_url` operation accepting a invalid URL address component.
Bug	4398205	`resolve_str` operation not able to resolve a stringified name.
Bug	4389165	RMI-IIOP `Tie` class should not assume a specific Hieraracy.
Bug	4372194	`org.omg.CORBA.Initializer` is missing a field.
Bug	4325192	`POAORB.init( null, null)` fails.
Bug	4409264	`idlj` generates server classes with variables that clash with parameter names.
Bug	4407349	Missing `InterfaceHelper` Java file from `idlj` (again).
Bug	4374920	`POAImpl` creates listen sockets for client ORBS.
Bug	4409787	Prob w/gen'd Java when using non-complete case-list for union's discriminator.
Bug	4286896	Generated Java code for Discriminated Union doesn't compile.
Bug	4476256	`ObjectKeyTemplate.getAdapterId` slow 'time' performance.
Bug	4473546	Generated union helper class read method must initialize the discriminator.
Bug	4434440	`idlj` generates bad code for non-default unions with `enum` as discriminator.
Bug	4416422	ORB search for `orb.properties` file should be improved.
Eou	4379402	`idlj` needs configurable/compatible output filenames for tie delegates.
RFE	4407304	`idlj`: introduce package name translation, not only package prefixing.
Bug	4394764	Javadocs missing in portable interceptors.

TABLE 2-1 Bugs and RFEs Implemented in J2SE, v. 1.4 *(Continued)*

Bug	4395252	`IORInfo.get_effective_policy` should never throw `INV_POLICY`.
Bug	4393853	PI - Co-located calls overwrite `ServerRequestInfo`.
Bug	4385644	`POA.id()` returning null.
Bug	4384769	PI - `register_initial_reference( LocalObject )` causes `ClassCastException`.
Bug	4382517	PI - `PICurrent` does not function as described in specification.
Bug	4396514	`orbutil.MinorCodes` contains duplicate meaning for INTERNAL 6.
Bug	4395808	`PIORB.registerORBInitializers` use applet `classLoader` if `Class.forName` fails.
Bug	4385945	`classLoader` reference never set in `corba.ORB`.
Bug	4384135	PI - `IORInfo.get_effective_policy` throws incorrect exception.
Bug	4406473	Portable Interceptors REVISITs.
Bug	4399805	Javadoc for `org.omg.CORBA.ORB.create_policy` is out of date.
Bug	4395669	`RequestInfo` operations need to be cached.
Bug	4392779	`object_id` and `adapter_id` should be available in `send_other/exception`.
Bug	4155455	`ORB.string_to_object` throws `MARSHAL` exception when IOR contains `objectkey INIT`.
Bug	4372196	`org.omg.CORBA.ServiceDetail` and `ServiceInformation` have errors in `Helper` classes.
Bug	4318477	`rmic -iiop` fails to use `classpath` properly or to use `jar` files on the `classpath`.
Bug	4373306	`org.omg.CORBA.Initializer` has wrong constructor in SE1.3.
Bug	4424268	Incorrect `TypeCode` behavior for `valuetypes` (CORBA 2.3 compliance issues).
Bug	4419156	Cannot invoke method over RMI/IIOP: its name clashes with an IDL keyword.
Bug	4419495	`rmiiiop/security` problem: unable to marshal `IDLEntity` types across remote `intf`.
Bug	4384167	Marshalling exception for discriminated union `idl` types.

TABLE 2-1 Bugs and RFEs Implemented in J2SE, v. 1.4 *(Continued)*

Bug	4365503	`PortableRemoteObject.narrow()` bug in our J2EE `rmiiiop` implementation from ATG.
Bug	4285443	`Util.unexportObject` throws `NullPointerException`.
Bug	4148483	Java IDL docs fail to document incompleteness of `Any.equal()`.
Bug	4064184	Recursive `TypeCodes` are not implemented.
Bug	4398375	ORB shutdown with wait for completion not implemented.
Bug	4308299	incorrect values in `.../build/solaris/sun/rmi/org/omg/GNUmakefile`.
Bug	4236995	`Anys` cannot be reused under certain conditions.
Bug	4364208	`org.omg.CORBA` package has classes which do not conform to the OMG IDL/Java mappi.
Bug	4360643	`_non_existent` results in null pointer exception in server.
Bug	4386423	ORB does not accept all `AddressingDispositions` as part of GIOP Message Header.
Bug	4149775	JavaIDL ORB interoperability failures.
Bug	4126181	Bug in implementation of `_get_interface()`.
Bug	4318587	Block of RMI-IIOP communication with multithreaded client.
Bug	4294980	`idlj` creates server-side skeletons which do not clone type IDs.
Bug	4328952	`com.sun.corba.se.internal.util.RepositoryId: cache` is non-final public static.
Bug	4267147	Fix Null pointer exception for IORs with 1.0 Profile format.
Bug	4350294	Incorrect CORBA `RepositoryID` calculations.
RFE	4228331	There's no way to get IOR with IP Address instead of Host Name.
Bug	4181568	Java IDL does not call `setTcpNoDelay(true)` on sockets.
Bug	4433966	`<rmic -idl>` or `<rmic -iiop>` fails in JDK1.3/1.3.1 with class loading.
Bug	4531406	`rmic` uses capital `\U` for escaping I18N characters.
Bug	4410058	Performance problem when loading `ties` and `stubs`.
RFE	4328099	Field set to null instead of this by `defaultReadObject()` in RMI-IIOP.
Bug	4463919	`ObjectStreamClass` does not support inherited `writeReplace/readResolve`.

TABLE 2-1 Bugs and RFEs Implemented in J2SE, v. 1.4 *(Continued)*

Bug	4391648	`Anys` created with the `ORBSingleton` throw `ClassCastException` on `insert_Value`.
Bug	4435390	`Util.writeAny` should report a specific error if object not serializable.
Bug	4401044	CORBA `read_Object(interface)` throws CORBA `MARSHAL` exception.
Bug	4385162	`INV_OBJREF` when reading a stringified IOR (re-open # 4322574).
Bug	4324936	Marshaling exception for wstring fields of structure.
RFE	4410060	Allow the user to specify code sets.
Bug	4404982	Accepted a `corbaloc` URL with multiple protocols, one of them being `rir`.
Bug	4362895	`ORB.string_to_object` with keystring causes `OutOfMemoryError` on `tnameserv`.
Bug	4204769	`NotFound` doesn't return proper `rest_of_name`.
Bug	4193117	Character range violation should raise `DATA_CONVERSION`, instead of `MARSHAL`.
Bug	4402934	`to_string` operation throws an incorrect `Exception` for invalid `Name`.
Bug	4378238	Sources for `org.omg.CORBA.PolicyError` do not match `.class-files`.
Bug	4484767	`idlj` gets confused by "(" and/or ")" as IDL `const` string value.
Bug	4330397	Problems with unions there two or more cases points to the same object.
Bug	4257220	RMI-IIOP `idlj` compiler generates bad code for oneway operations.
RFE	4287942	Would you please make a method to delete the instance of ORB.
Bug	4328948	`com.sun.corba.se.internal.corba.ClientDelegate:debug` is non-final public static.
Bug	4294972	`POAImpl.java:activate_object` method returns `object-ids` without cloning.
Bug	4290501	CORBA failure on sending large objects.
RFE	4227148	IDL `javatoidl` compiler 100% Java version.
Bug	4292565	`idlj` creates superfluous folder hierarchy for nested prefixed packages.

TABLE 2-1 Bugs and RFEs Implemented in J2SE, v. 1.4 *(Continued)*

Bug	4278435	Marshalling a CORBA structure with nulls in blows up the client on receipt.
Eou	4379317	`idlj`-generated `readObject` and `WriteObject` methods throw wrong exceptions.
Bug	4067057	Server can't reconnect to client when client side connection table is full.
Bug	4410548	`JDKClassLoader` could abort unnecessarily in the future.
Bug	4294494	DSI does not work with user exceptions.
Bug	4193307	Singleton ORB should not support `object_to_string`.
Bug	4498869	Double slash confuses `idlj` compiler.
Bug	4401627	`OBJ_ADAPTER` exception in POA destroy.
Bug	4309167	`PortableRemoteObject unexportObject` is broken for POA-enabled `Ties`.
Bug	4145490	`org.omg.CORBA.Any.equal(Any)` not fully implemented.
RFE	4227142	ORB singleton optimization.
RFE	4227149	Socket factories for SSL support.
Bug	4517819	`DynAny.current_member_kind()` doesn't throw `InvalidValue` for invalid position.
Bug	4473859	`rmic -iiop` gen. fails when interface throws super class of `impl` exception.
Bug	4393485	`ListenerThread` and server socket not terminated by `ORB.shutdown()`.
Bug	4324049	`TypeCode.equivalent` not implemented.
Bug	4322176	Repeated interface method declarations confuse RMI-IIOP compiler.
Bug	4290049	`org.omg.CORBA.ORB.create_string_tc` should throw `BAD_PARAM` for negative values.
Bug	4274455	`copyObjects` returns wrong `stub` type.
Bug	4274493	`TypeCodeImpl.copy` does not support `tk_wstring`.
Bug	4274588	`CORBA.MARSHAL` error with value types and indirection.
Bug	4176268	`org.omg.CORBA.TypeCode.equal()` method bug.
RFE	4274686	Request to have vendor minor code ID displayed in `SystemException.toString()`.
Bug	4419991	`createRequest GIOPVersion locatedIOR` problem.

TABLE 2-1 Bugs and RFEs Implemented in J2SE, v. 1.4 *(Continued)*

Bug	4386425	ORB uses `byte[]` for object key instead of `ObjectKey` object which is inefficient.
RFE	4414144	GIOP `CancelRequest` processing causes orb worker thread to die.
Bug	4273648	`AnyImplHelper` and `TypeCodeImplHelper` don't compile.
Bug	4267142	Fix `typeId` generation for Interfaces implementing Remote.
Bug	4445431	Exception stack for `LocalObject`.
Bug	4512720	Must use decimal to specify conversion list code sets.
Bug	4407009	`idlj` too lenient with `wchar` and `wstring` literals.
Bug	4201726	`org.omg.CORBA.ORB` loads singleton orb too soon.
Bug	4109166	Command to list the complete path of the server using the tool is not listed.
RFE	4145497	Java IDL does not support long `double`, `wchar`, `wstring`.
Bug	4335580	ORB shutdown does not terminate server thread.
Bug	4288985	Private method `ObjectInputStream.loadClass0()` will be removed soon.
RFE	4344633	Incorrect Repository IDs for `IDLType`, `DefinitionKind`, `ValueMember`.
RFE	4124730	Socket Factory for ORB.
RFE	4413694	Need for unique worker thread names for debugging purposes.

2.3 Previous Versions of CORBA in J2SE

The CORBA implementation in J2SE, v. 1.4, is compliant with CORBA, v. 2.3.1. Prior to J2SE, v. 1.4, we did not have a specific compliance document, but J2SE, v. 1.3, could be considered compliant with CORBA, v. 2.0+.

The documentation for the CORBA implementation in J2SE, v. 1.3, is located at:

`http://java.sun.com/j2se/1.3/docs/guide/idl/index.html`

Support Readiness information for the previous version of RMI-IIOP is available at:

`http://access1.sun.com/SRDs/access1_srds.html`

2.4 Backward and Forward Compatibility

2.4.1 Code Compatibility

Code developed using the CORBA tools in J2SE, v. 1.3, is forwardly compatible with J2SE, v. 1.4; however, code developed using the latest CORBA features in J2SE, v. 1.4, such as POA, PI, INS, is not backwardly compatible.

2.4.2 Wire Compatibility

Clients and servers running on J2SE, v. 1.3, and J2SE, v. 1.4, will interoperate in most cases. There are some corner cases where they will not interoperate. These corner cases are explained in Section 4.1 "Product Limitations."

2.5 Upgrading to the J2SE, v. 1.4, Implementation CORBA From a Previous Version

To use the new features in CORBA, v. 2.3.1, the user needs to understand POA, PI, and INS.

With J2SE, v. 1.4, we recommend using ORBD instead of `tnameserv`. The `NameService` in ORBD provides better quality of service. We also recommend using POA-based programming for CORBA server development instead of old `ImplBase` servants.

For a list of tutorials describing how to write code using these features, please see Section 3.3 "Tutorials."

For information on J2SE, v. 1.4, CORBA compliance with OMG documents, please see:

`http://java.sun.com/j2se/1.4/docs/api/org/omg/CORBA/doc-files/compliance.html`

2.6 Porting Applications to the J2SE, v. 1.4, CORBA Implementation

An application written for J2SE, v. 1.3, CORBA will need to be recompiled with the IDLJ compiler before it can be used with J2SE, v. 1.4. A special IDLJ compiler switch must be set when the code is recompiled. This is explained in the documents listed in Section 3.4 "Tools and Utilities." The user may choose to use only the class files which are precompiled in J2SE, v. 1.3.

3 Using and Supporting the J2SE, V. 1.4, CORBA Implementation

As explained in the previous sections, the J2SE, v. 1.4, CORBA implementation supports two different models:

- RMI-IIOP model
- IDL model

The following paragraphs explain in detail when to use these models.

3.1 When to Use the RMI-IIOP Model

RMI-IIOP is for Java programmers who want to program to the RMI interfaces but also want to use IIOP as the underlying transport. RMI-IIOP provides interoperability with other CORBA objects implemented in various languages, but only if all the remote interfaces are originally defined as Java RMI interfaces. RMI-IIOP is of particular interest to programmers using EJB since the remote object model for EJBs is RMI-based.

RMI-IIOP combines the best features of Java RMI with the best features of CORBA. RMI-IIOP speeds distributed application development by allowing developers to work completely in the Java programming language, writing remote interfaces in the Java programming language and implementing them simply using Java technology and the Java RMI APIs.

When using RMI-IIOP to produce Java technology-based distributed applications, there is no separate Interface Definition Language (IDL) or mapping to learn: the remote interfaces can be implemented in any language that is supported by an OMG mapping and that has a vendor-supplied ORB. Similarly, clients can be written in other languages, using IDL derived from the remote Java technology-based interfaces.

RMI-IIOP provides flexibility by allowing developers to pass any Java object between application components either by reference or by value.

Like CORBA, RMI-IIOP is based on open standards defined with the participation of hundreds of vendors and users in the OMG. IIOP eases legacy application and platform integration by allowing application components written in C++, Smalltalk, and other CORBA supported languages to communicate with components running on the Java platform.

3.2 When to Use the IDL Model

The OMG IDL is a purely declarative language designed for specifying programming language-independent operational interfaces for distributed applications. OMG specifies a mapping from IDL to several different programming languages, including C, C++, Lisp, Python, Smalltalk, COBOL, Ada, and Java. When mapped, each statement in OMG IDL is translated to a corresponding statement in the programming language of choice.

Java IDL is an implementation of the CORBA specification. For example, you could use the Java IDL Compiler, `idlj`, to map an IDL interface to Java and implement the client class in Java. If you map the same IDL to C++, using an IDL-to-C++ compiler and a C++ ORB, and implement the server in that language, the Java client and C++ server interoperate through the ORB as though they were written in the same language.

3.3 Tutorials

The following tutorials provide complete code samples with descriptions:

- Tutorials for RMI-IIOP model can be found at:

  http://java.sun.com/j2se/1.4/docs/guide/rmi-iiop/
- Tutorials for IDL programming can be found at:

  http://java.sun.com/j2se/1.4/docs/guide/idl/

3.4 Tools and Utilities

Description of the following tools can be found at:

`IDLJ`:

http://java.sun.com/j2se/1.4/docs/guide/rmi-iiop/toJavaPortableUG.html

`ORBD`:

http://java.sun.com/j2se/1.4/docs/guide/idl/orbd.html

`ServerTool`:

http://java.sun.com/j2se/1.4/docs/guide/idl/servertool.html

`tnameserv`:

http://java.sun.com/j2se/1.4/docs/guide/idl/tnameserv.html

`rmic`:

http://java.sun.com/j2se/1.4/docs/tooldocs/solaris/rmic.html

3.5 Localization and Internationalization

All the RMI-IIOP and CORBA tools are localized.

4 Troubleshooting

4.1 Product Limitations

4.1.1 Multiple Profile IORs or URLs

The J2SE ORB will use the first profile (Host, Port information) if there are multiple profiles given either in the IOR or INS-based `corbaloc:` or `corbaname:` URLs.

4.1.2 Interoperability With Third-Party ORBs

We have not done extensive testing to prove that we interoperate with all vendors. Successful testing for interoperability has been done using ORBs from Borland, Hitachi, and IBM.

4.1.3 Interoperability With the J2SE, v. 1.3, ORB

The J2SE, v. 1.3 and v. 1.4, ORBs interoperate except for two limitations:

- If a class has evolved from J2SE, v. 1.3, to v. 1.4 and uses the `writeObject()` custom marshaling method, objects cannot be passed by value.
- Using RMI-IIOP with Java `char` arrays and characters greater than 8 bits will result in `DATA_CONVERSION` exceptions. This is fixed in J2SE, v. 1.3.1.

4.1.4 Interoperability With the J2SE, v. 1.3.1, ORB

J2SE, v. 1.3.1, and J2SE, v. 1.4, ORBs interoperate except for one limitation. If a class serial version has changed and the newer version class uses the `putfields()` and `getfields()` methods, then those objects cannot be passed by value. This bug is fixed in the J2SE, v. 1.3.1_01, patch.

4.1.5 Multiple ORBs in Persistent Servers

For persistent servers, that is `ServerTool` registered servers, you cannot have more than one ORB instance running because there will be a `ORBId` collision. There is no CORBA standard property to pass the `ORBId` to avoid this collision.

4.1.6 Java Naming and Directory Interface (JNDI) in Persistent Servers

JNDI's `new InitialContext()` instantiates an ORB. This will not work in persistent servers because of the limitation above. For JNDI to work in a persistent server, pass the `java.naming.corba.orb` property with the ORB instance already in use. This way, JNDI will reuse the same ORB instance to get around the limitation in the J2SE, v. 1.4, ORB.

4.2 Common User Questions

The following FAQ covers most troubleshooting questions:

- Java IDL FAQ

  http://java.sun.com/j2se/1.4/docs/guide/idl/jidlFAQ.html

4.3 Troubleshooting Utilities

The ORB's debugging flag can be turned on by passing the `-ORBDebug` argument to the `ORB.init()` method. The developer can also choose the following debug flags to see debug statements in various areas:

- `transportDebugFlag`
- `subcontractDebugFlag`
- `poaDebugFlag`
- `namingDebugFlag`
- `serviceContextDebugFlag`

- transientObjectManagerDebugFlag
- giopVersionDebugFlag
- shutdownDebugFlag
- giopDebugFlag

These flags can be passed as a comma-separated list to the `-ORBDebug` flag. For example, to see the debug output in the `Transport`, `Subcontract`, and `Shutdown` areas, pass the following `-ORBDebug` flags to `ORB.init()`:

```
-ORBDebug transport,subcontract,shutdown
```

The user can also choose to pass the `-ORBDebug` flags as a property. The property name is `com.sun.CORBA.ORBDebug`, and the property values should list all the debug flags separated by commas. For example, to see the debug output in the areas of `naming` and `giop`, use:

```
com.sun.CORBA.ORBDebug=naming,giop
```

4.4 Common Developer Problems

4.4.1 Internal Exception when Starting ORBD

Problem: Unable to Start `ORBD` because of an `INTERNAL` exception.

Cause: If the last two lines of the stack trace from the `CORBA.INTERNAL` exception are:

```
com.sun.corba.se.internal.iiop.GIOPImpl.createListener(
   GIOPImpl.java)
com.sun.corba.se.internal.iiop.GIOPImpl.getEndpoint(GIOPImpl.java)
```

This means that the specified `ORBInitialPort`/`ActivationPort` is already in use.

Solution: There are two ways to get around this problem:

- Specify a different `ORBInitialPort`/`ActivationPort`
- Kill the process using the specified `ORBInitialPort`/`ActivationPort` if the other process using these ports is no longer required.

4.4.2 Client and Server Unable to Connect to `NamingService`

Problem: The client and server are unable to connect to the `NamingService`.

Cause: If a `COMM_FAILURE` exception occurs, the cause could be:

- `ORBD` or `tnameserv` was not successfully launched.
- The `ORBInitialHost` or `ORBInitialPort` specified to contact `ORBD` or `tnameserv` is incorrect.

Solution: Correct the cause of the problem as listed above.

4.4.3 `Servertool` Registration Failure

Problem: `Servertool` registration fails.

The `Servertool` fails with the following message:

```
server already registered (serverid = <serverId>)
```

Solution: Change the registration for this server. You can do this in one of two ways:

- Shutdown `ORBD` and clean up the `orb.db` directory. This directory contains information about all the registered servers. When you restart the `ORBD`, you should be able to reregister the server.

Note – If you clean up the `orb.db` directory, you may lose all the persistent state information for `ORBD`. This will have the same effect as a fresh `ORBD` startup.

- Choose to use a different `-applicationname` as explained in the `Servertool` documentation.

4.5 Error Message Guide

4.5.1 `CORBA.COMM_FAILURE` Exception

A `CORBA.COMM_FAILURE` exception is raised when the client is unable to contact the server. This may happen for various reasons, such as:

- The server was not successfully started.
- The server was shutdown before the client was invoked.

The solution is to check whether or not the server is up and running correctly. If the server is not running correctly, start the server.

4.5.2 CORBA.INTERNAL Exception

A `CORBA.INTERNAL` exception is raised when there are abnormal situations. This could be a serious error.

4.5.3 Other CORBA Exceptions

Other CORBA exceptions are thrown for various reasons. For details, see:

http://java.sun.com/j2se/1.4/docs/guide/idl/jidlExceptions.html

5 Key Files and Directories

5.1 Configuration Files

5.1.1 The `orb.properties` File

The `orb.properties` file is a read-only file containing `ORB` properties. To modify the `ORB` properties, create a `Properties` object and pass it as a parameter during ORB initialization, using the `ORB.init()` method.

`ORB` looks for the `orb.properties` file first in the user's home directory (the `user.home` property is new for J2SE, v. 1.4). If the `ORB.properties` file is not found, the `ORB` will look for `orb.properties` in the directory `{java.home}/lib`.

Some of the `ORB` properties are explained at:

`http://java.sun.com/j2se/1.4/docs/guide/idl/jidlInitialization.html#systempropertiesobject`

5.2 Directories Created at Installation

The CORBA product is part of J2SE, v. 1.4. The directories created during the J2SE, v. 1.4, installation are used with the J2SE, v. 1.4, CORBA implementation.

6 Installing and Configuring J2SE, V. 1.4 CORBA

The CORBA functionality is included with J2SE, v. 1.4. No additional installation or configuration is required. For information on installing and configuring J2SE, v. 1.4, see the J2SE, v. 1.4, Overview Support Readiness Document, available at:

`http://access1.sun.com/SRDs/access1_srds.html`

7 Reference Information

7.1 Product Information

Product pages from Sun:

- J2SE, v. 1.4, CORBA home page
  http://java.sun.com/j2se/1.4/docs/guide/corba/
- Java IDL home page
  http://java.sun.com/j2se/1.4/docs/guide/idl/
- RMI-IIOP home page
  http://java.sun.com/j2se/1.4/docs/guide/rmi-iiop/

Other sites with product information on CORBA:

- Official OMG website
  http://www.omg.org/

7.2 Technical Documentation

J2SE, v. 1.4, CORBA Programmer's Guides:

- RMI-IIOP Programmer's Guide
  http://java.sun.com/j2se/1.4/docs/guide/rmi-iiop/rmi_iiop_pg.html
- CORBA Concepts
  http://java.sun.com/j2se/1.4/docs/guide/idl/jidlUsingCORBA.html

- Distributed Application Concepts

  http://java.sun.com/j2se/1.4/docs/guide/idl/jidlDistApp.html
- API Guide for J2SE, v. 1.4

  http://java.sun.com/j2se/1.4/docs/api/
  - CORBA packages - `org.omg.*`
  - RMI-IIOP packages - `javax.rmi.*`

Other technical documentation:

- CORBA Exceptions

  http://java.sun.com/j2se/1.4/docs/guide/idl/jidlExceptions.html
- CORBA Initialization

  http://java.sun.com/j2se/1.4/docs/guide/idl/jidlInitialization.html
- COSNaming

  http://java.sun.com/j2se/1.4/docs/guide/idl/jidlNaming.html
- Dynamic Skeleton Interface

  http://java.sun.com/j2se/1.4/docs/guide/idl/jidlDSI.html
- IDL to Java Mapping

  http://java.sun.com/j2se/1.4/docs/guide/idl/mapping/jidlMapping.html

7.3 Frequently Asked Questions

- Java IDL FAQ

  http://java.sun.com/j2se/1.4/docs/guide/idl/jidlFAQ.html
- JDC Forum archive for Java IDL

  http://forum.java.sun.com/forum.jsp?forum=15
- JDC Forum archive for RMI-IIOP

  http://forum.java.sun.com/forum.jsp?forum=59

7.4 Tutorials and Other References

- Getting Started Using RMI-IIOP

  http://java.sun.com/j2se/1.4/docs/guide/rmi-iiop/tutorial.html
- Getting Started with Java IDL

  http://java.sun.com/j2se/1.4/docs/guide/idl/GShome.html
- Introduction to CORBA

  http://developer.java.sun.com/developer/onlineTraining/corba/
- CORBA for beginners

  http://cgi.omg.org/corba/beginners.html
- Information on OMG's vision and architecture

  http://www.omg.org/oma
- Introduction to CORBA

  http://www.omg.org/gettingstarted/index.htm
- Status of various CORBA interoperability projects

  http://www.omg.org/interoperability_testing