Reference Guide

Netscape Internet Service Broker for Java

Netscape Communications Corporation ("Netscape") and its licensors retain all ownership rights to the software programs offered by Netscape (referred to herein as "Software") and related documentation. Use of the Software and related documentation is governed by the license agreement accompanying the Software and applicable copyright law.

Your right to copy this documentation is limited by copyright law. Making unauthorized copies, adaptations, or compilation works is prohibited and constitutes a punishable violation of the law. Netscape may revise this documentation from time to time without notice.

THIS DOCUMENTATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND. IN NO EVENT SHALL NETSCAPE BE LIABLE FOR ANY LOSS OF PROFITS, LOSS OF BUSINESS, LOSS OF USE OR DATA, INTERRUPTION OF BUSINESS, OR FOR INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY KIND, ARISING FROM ANY ERROR IN THIS DOCUMENTATION.

The Software and documentation are copyright © 1995-1997 Netscape Communications Corporation. All rights reserved.

Portions of the software and documentation are copyright © 1996-1997 Visigenic Software, Inc.

Netscape, Netscape Communications, the Netscape Communications Corporation Logo, and other Netscape product names are trademarks of Netscape Communications Corporation. These trademarks may be registered in other countries. Other product or brand names are trademarks of their respective owners.

Any provision of the Software to the U.S. Government is with restricted rights as described in the license agreement accompanying the Software.

The downloading, export or reexport of the Software or any underlying information or technology must be in full compliance with all United States and other applicable laws and regulations as further described in the license agreement accompanying the Software.


Recycled and Recyclable Paper

Version 1.0

©Netscape Communications Corporation 1997
All Rights Reserved

Printed in USA
99 98 97 10 9 8 7 6 5 4 3 2 1

Netscape Communications Corporation 501 East Middlefield Road, Mountain View, CA 94043

Contents

Introduction

This reference provides information on the classes supplied with Netscape Internet Service Broker for Java (ISB for Java). ISB for Java allows you to develop and deploy applications and applets that use distributed objects, as defined in the Common Object Request Broker (CORBA) specification.

This chapter describes typographical and syntax conventions used throughout the guide and provides references for more information about CORBA. It presents the following topics.

- Organization of this Guide
- Typographic Conventions
- Platform Conventions
- Where to Find Additional Information

Organization of this Guide

This guide includes the following sections:

- "Commands" describes the commands available for creating, starting, and managing applications.
- "The IDL to Java Mapping" describes the IDL-to-Java mapping used by ISB for Java.
- "Generated Classes" describes the classes generated by the `idl2java` compiler.
- "Core Interfaces and Classes" describes all of the interfaces in the `org.omg.CORBA` package.
- "Dynamic Interfaces and Classes" describes the dynamic interfaces and classes in the `org.omg.CORBA` package.
- "Interface Repository" describes the interfaces and classes in the `org.omg.CORBA` package that are used with the interface repository.

- "Exceptions Classes" describes the exception classes used in ISB for Java.

Typographic Conventions

This manual uses the following typographic conventions:

Convention	Used for
boldface	In syntax notation, bold type indicates information that the user or application provides, such as variable or parameter values.
italics	Italics are also used to introduce new terms.
`computer`	Computer typeface is used for sample command lines and code.
[ ]	Brackets indicate optional items.
{ }	Curly brackets (also called braces) are used in the more complex syntax statements to show a required item.
...	In syntax notation, an ellipsis indicates that the previous argument can be repeated. In ac ode example, it indicates the continuation of previous lines of code.
|	A vertical bar separates two mutually exclusive choices.

Platform Conventions

This manual uses the following symbols to indicate that information is platform-specific:

W All Windows platforms including Windows 3.1, Windows NT, and Windows 95 .

NT Windows NT only.

95 Windows 95 only.

U All Unix platforms.

Where to Find Additional Information

For more information about Netscape ISB for Java, refer to the following source:

- *Netscape Internet Service Broker for Java Programmer's Guide*. This guide contains the information on the developing applications using Netscape ISB for Java.

For more information about the CORBA specification, refer to the following source:

- The CORBA 2.0 Specification - 96-03-04. This document is available from the Object Management Group at http://www.omg.org and describes the architectural details of CORBA.
- The OMG IDL/Java Language Mapping Specification. This document is available at http://www.visigenic.com and describes the IDL to Java mapping specifications.

Chapter

1

Commands

This chapter covers commands used for creating, executing and managing applications developed with Netscape ISB for Java. It includes the following major sections:

- idl2ir
- idl2java
- irep
- java2idl
- java2iiop

To view a list of options for a command, enter the command name with no option and a list of options appears. For example:

```
prompt> idl2ir
```

A list of options appears. For example:

```
Usage: idl2ir [-options] files...

where options include:
      -ir <name>  The name of the interface repository
      -replace    Replace definitions, instead of updating them
```

idl2ir

This command allows you to populate an interface repository with objects defined in an Interface Definition Language source file.

Syntax

idl2ir [*options*] *infile*.**idl**

Description

The `idl2ir` command takes an IDL file as input, binds itself to an interface repository server and populates the repository with the IDL constructs contained in *infile*. If the repository already contains an item with the same name as an item in the IDL file, the old item will not be updated unless you use the `-replace` option.

Note The `idl2ir` command does not handle anonymous arrays or sequences properly. To work around this problem, typedefs must be used for all sequences and arrays.

Option	Description
-ir *name*	Specifies the instance name of the interface repository to which `idl2ir` will attempt to bind. If *name* is not specified, `idl2ir` will bind itself to the interface repository server found in the current domain. The current domain is defined by the OSAGENT_PORT environment variable.
-replace	Replaces definitions instead of updating them.

Example

```
idl2ir -ir my_repository -replace java_examples/bank/Bank.idl
```

idl2java

This command generates Java source code from an IDL source file.

Syntax

idl2java [*options*] *filename*.**idl**

Description

The `idl2java` command, a Java-based preprocessor, compiles an IDL source file and creates a directory structure containing the Java mappings for the IDL declarations. Typically, one `.IDL` file will be mapped to many `.java` files because Java allows only one public interface or class per file. IDL file names must end with the .IDL extension.

Option	Description
-no_comments	Supresses comments in the generated code.
-no_example	Supresses the generation of example classes.
-no_tie	Supresses the generation of _tie classes.
-package <name>	Generates code for the specified package. If a directory with the specified package name does not exist, it will be created. If the package directory exists, its contents will be updated.
-portable	Generates portable stubs using DII.
-verbose	Turns verbose mode on.

The supported preprocessor arguments, standard for most preprocessors, are shown in the table that follows. Refer to your preprocessor's documentation to find more information about arguments.

Supported args	Description
-C	Retains comments from IDL file when the Java code is generated. Otherwise, the comments will not appear in the Java code.
-D	Defines a macro. Recursive macro definitions are not supported.
-H	Prints the full paths of included files on the standard error output.
-I	Takes a directory as input.
-P	Supresses the generation of line number information.
-U	Undefines a macro.

Example

```
idl2java -verbose bank.idl
```

irep

This command starts an interface repository server.

Syntax

irep [*options*] *irep_name* [*IDL_storage_file*]

Description

The `irep` command starts an interface repository server that manages a database containing detailed descriptions of IDL interfaces. The required argument *irep_name* specifies the instance name of the repository server. The optional argument *IDL_storage_file* specifies the storage file to use. The repository's database includes interface names, inheritance structure, supported operations, and arguments. The interface repository can consist of multiple databases and may be loaded using the `idl2ir` command.

Option	Description
-console	Runs `irep` as a console application.

Example

```
irep -console my_server
```

java2idl

This command creates an IDL file from a Java class file (in Java byte code). You can enter one or more Java class file names. If you enter more than one filename, include spaces between the file names.

Note To use this command, you must have a virtual machine supporting JDK 1.1.

Syntax

java2idl [*options*] *filename*

Description

Use this command to create an IDL file from Java byte code stored in one or more `.class` files represented by *filename*. You might want to use this when you have existing Java byte code and want to create an IDL file from it so it can be used with some other programming language like C++, Cobal, or Smalltalk.

Option	Description
-verbose	Turns verbose mode on.

Example

The following command compiles class files named `Account.class`, `Client.class`, and `Server.class`.

```
java2idl Account Client Server
```

java2iiop

This command (also called the Caffeine compiler) allows you to use the Java language to define IDL interfaces instead of using IDL. You can enter one or more Java class file names (in Java byte code). If you enter more than one file name, include spaces between the file names. Do not include the `.class` extension. For example, to compile a file named `Bank.class`, enter

```
java2iiop Bank
```

Note To use this command, you must have a virtual machine supporting JDK 1.1.

Syntax

java2iiop [*options*] *filename*

Description

Use `java2iiop` if you have existing Java byte code that you wish to adapt to use distributed objects or if you do not have the time to learn IDL. The *filename* argument specifies the file or files from which to generate files using IIOP. By using `java2iiop`, you can generate the necessary container classes, client stubs, and server skeletons from Java byte code.

Option	Description
-no_comments	Suppresses comments in generated code.
-no_examples	Suppesses the generation of example code.
-no_tie	Suppresses the generation of tie code.

Option	Description
-portable	Generates portable stubs using DII.
-verbose	Turns verbose mode on.

Example

The following command compiles class files named `Account.class`, `Client.class`, and `Server.class` without generating tie code.

```
java2iiop -no_tie Account Client Server
```

Chapter

2

The IDL to Java Mapping

This chapter describes ISB for Java's IDL-to-Java language mapping as implemented by the `idl2java` compiler. For each IDL construct there is a section that describes the corresponding Java construct, along with code samples. ISB for Java conforms with the *OMG IDL/Java Language Mapping Specification*. See the latest version of this specification for complete information, and especially, for information about the following:

- Mapping pseudo-objects to Java
- Server-side mapping
- Java ORB portability issues

This chapter includes the following major sections:

- Names
- Reserved Names
- Reserved Words
- Modules
- Global Scope
- Constants
- Basic Types
- Helper Classes
- Holder Classes
- Constructed Types
- Interfaces

- Mapping for Certain Nested Types
- Mapping for Typedef
- Mapping for Exceptions

Names

In general, IDL names and identifiers are mapped to Java names and identifiers with no change. If a name collision might be generated in the mapped Java code, the name collision is resolved by prepending an underscore (_) to the mapped name.

In addition, because of the nature of the Java language, a single IDL construct may be mapped to several (differently named) Java constructs. The “additional” names are constructed by appending a descriptive suffix. For example, the IDL interface AccountManager is mapped to the Java interface AccountManager and additional Java classes AccountManagerHelper and AccountManagerHolder.

In those exceptional cases where the “additional” names could conflict with other mapped IDL names, the resolution rule described above is applied to the other mapped IDL names. In other words, the naming and use of required “additional” names takes precedence. For example, interfaces named `fooHelper` and `fooHolder` are mapped to `_fooHelper` and `_fooHolder`, respectively, regardless of whether an interface named `foo` exists. The Helper and Holder classes for interface `fooHelper` are named `_fooHelperHelper` and `_fooHelperHolder`.

IDL names that would normally be mapped unchanged to Java identifiers that conflict with Java reserved words will have the collision rule applied.

Reserved Names

The mapping reserves the use of several names for its own purposes. The use of any of these names for a user-defined IDL type or interface (assuming it is also a legal IDL name) will result in the mapped name having an (_) prepended. Reserved names are as follows:

- The Java class <type>Helper, where <type> is the name of an IDL user-defined type.

- The Java class <type>Holder, where <type> is the name of an IDL user-defined type (with certain exceptions such as typedef aliases).
- The Java classes <basicJavaType>Holder, where <basicJavaType> is one of the Java primitive datatypes that is used by one of the IDL basic datatypes.
- The nested scope Java package name <interface>Package, where <interface> is the name of an IDL interface.

Reserved Words

The use of any of these words for a user-defined IDL type or interface (assuming it is also a legal IDL name) will result in the mapped name having an (_) prepended. The reserved keywords in the Java language are as follows:

abstract	boolean	break	byte
case	catch	char	class
const	continue	default	do
double	else	extends	final
finally	float	for	goto
if	implements	import	instanceof
int	interface	long	native
new	package	private	protected
public	return	short	static
super	switch	synchronized	this
throw	throws	transient	try
void	volatile	while	

Modules

The IDL module construct is mapped to a Java package and the package is given the same name as the IDL module. For all IDL types within the module that are mapped to Java classes or Java interfaces, the corresponding Java class

or interface is declared inside the Java package that is generated. IDL declarations that are not enclosed in any modules are mapped into the unnamed Java global scope.

The following code example shows the Java code generated for an enumeration declared within an IDL module.

Listing 2.1 Mapping an IDL module to a Java package

```
/* IDL code from Example.idl: */
module Example { .... };
};

// Generated Java code:
package Example;
...
}
```

Global Scope

Any IDL construct declared in the global scope will be placed in a java package which is named IDL_GLOBAL. This name can be changed to a user defined one through the use of compiler options.

All the IDL examples used in this chapter are from the IDL module Example. Therefore, the generated Java declarations will all be placed inside the Java package Example.

Constants

Constants are mapped depending upon the scope in which they appear.

Constants within an Interface

Constants declared within an IDL interface are mapped to `public static final` fields in the corresponding Java interface.

Listing 2.2 Mapping an IDL constant within a module to a Java class.

```
/* IDL code from Example.idl: */
```

```
module Example
    interface Foo{
        const long aLongerOne = -321;
    };
};
```

```
// Generated Java code:
package Example;
public interface Foo {
    public static final int aLongerOne = (int) (-321L);
}
```

Constants NOT within an Interface

Constants declared within an IDL module are mapped to a public interface with the same name as the constant and containing a public static final field named value. This field holds the constant's value.

Note The Java compiler normally inlines the value when the class is used in other Java code.

Listing 2.3 Mapping an IDL constant within a module to a Java class.

```
/* IDL code from Example.idl: */
module Example {
    const long aLongOne = -123;
};
```

```
// Generated Java code:
package Example;
public interface aLongOne {
    public static final int value = (int) (-123L);
}
```

Basic Types

The following table shows how the defined IDL types map to basic Java types. When there is a potential mismatch between an IDL type and its mapped Java type, the EXCEPTIONS column lists the standard CORBA exception that can be raised. For the most part, exceptions are in two categories:

- The range of the Java type is larger than the IDL type. The value must be checked at runtime when it is marshalled as an IN parameter (or an input for an INOUT parameter). For example, Java chars are a superset of IDL chars.
- Because there is no support in Java for unsigned types, the developer is responsible for ensuring that large unsigned IDL type values are handled correctly as negative integers in Java.

Note The Java null may only be used to represent the null object reference. For example, a zero length string, rather than null must be used to represent the empty string. This is also true for arrays.

IDL Type	Java Type	Exceptions
boolean	boolean	
char	char	CORBA::DATA_CONVERSION
wchar	char	
octet	byte	
string	java.lang.String	CORBA::MARSHAL CORBA::DATA_CONVERSION
WString	java.lang.String	CORBA::MARSHAL
short	short	
unsigned short	short	
long	int	
unsigned long	int	
long long	long	
unsigned long long	long	
float	float	
double	double	

Boolean

The IDL type `boolean` is mapped to the Java type `boolean`. The IDL constants TRUE and FALSE are mapped to the Java constants true and false.

char and wchar

IDL characters are 8-bit quantities representing elements of a character set, while Java characters are 16-bit unsigned quantities representing Unicode characters. To enforce type safety, the Java CORBA runtime asserts range validity of all Java `char`s mapped from IDL `char`s when parameters are marshaled during method invocation. If the `char` falls outside the range defined by the character set, a CORBA::DATA_CONVERSION exception is thrown. The IDL `wchar` maps to the Java `char` type.

Octet

The IDL type `octet`, an 8-bit quantity, is mapped to the Java type `byte`.

String

The IDL type `string` is mapped to the Java type `java.lang.String`. Range checking for characters in the string as well as bounds checking of the string is done at marshal time.

WString

The IDL type `wstring`, used to represent Unicode strings, is mapped to the Java type `java.lang.String`. Bounds checking of the string is done at marshal time.

Integer Types

IDL `short` and `unsigned short` map to Java type `short`. IDL `long` and `unsigned long` map to Java's `int`.

Note Because there is no support in Java for unsigned types, the developer is responsible for ensuring that negative integers in Java are handled correctly as large unsigned values.

Floating Point Types

The IDL floating point types float and double map to a Java class containing the corresponding data type.

Helper Classes

All user-defined IDL types have an additional “helper” Java class with the suffix Helper appended to the type name generated. Several static methods needed to manipulate the type are supplied:

- Any insert and extract operations for the type
- getting the repository id
- getting the typecode
- reading and writing the type from and to a stream

For any user-defined IDL type, <typename>, the following Java code is generated for the type. In addition, both the helper class associated with an IDL interface and the helper class for a mapped IDL interface have a `narrow` operation defined for them.

```
// generated Java helper
public class <typename>Helper {
    public static void
        insert(org.omg.CORBA.Any a, <typename> t);
    public static <typename> extract(Any a);
    public static org.omg.CORBA.TypeCode type();
    public static String id();
    public static <typename> read(org.omg.CORBA.portable.InputStream istream)
    { ... };
    public static void write(org.omg.CORBA.portable.OutputStream ostream, <typename>
value)
    { ... };

    // only for interface helpers
    public static <typename> narrow(org.omg.CORBA.Object obj);
}
```

The helper class for a mapped IDL interface has a narrow operation defined for it. The helper class for a mapped IDL enum also has a from_int operation defined for it.

Listing 2.4 Mapping of a named type to Java helper class

```
// IDL - named type
struct st {long f1, String f2};

// generated Java
public class stHelper {
    public static void insert(org.omg.CORBA.Any any, st s) {...}
    public static st extract(Any a) {...}
    public static org.omg.CORBA.TypeCode type() {...}
    public static String id() {...}
    public static st read(org.omg.CORBA.InputStream is) {...}
    public static void write(org.omg.CORBA.OutputStream os, st s) {...}
}
```

Listing 2.5 Mapping of a typedef sequence to Java helper class

```
// IDL - typedef sequence
typedef sequence <long> IntSeq;

// generated Java helper
public class IntSeqHelper {
    public static void insert(org.omg.CORBA.Any any,int[] seq);
    public static int[] extract(Any a){...}

    public static org.omg.CORBA.TypeCode type(){...}
    public static String id(){...}
    public static int[] read(org.omg.CORBA.portable.InputStream is)
        {...}
    public static void write(
                            org.omg.CORBA.portable.OutputStream os,
                            int[] seq)
        {...}
}
```

Holder Classes

Holder classes support OUT and INOUT parameter passing modes and are available for all the basic IDL data types in the `org.omg.CORBA` package. Holder classes are generated for all named user-defined types except those defined by typedefs. For more information about Holder classes, see "Generated Classes."

For user-defined IDL types, the holder class name is constructed by appending Holder to the mapped Java name of the type.

For the basic IDL data types, the holder class name is the Java type name (with its first letter capitalized) to which the datatype is mapped with an appended `Holder`, for example `IntHolder`.

Each holder class has a a default constructor and constructor from an instance, and a public instance member, `value`, which is the typed value. The default constructor sets the `value` field to the default value for the type as defined by the Java language:

- `false` for boolean
- 0 for numeric and char types
- `null` for strings
- `null` for object references

In order to support portable stubs and skeletons, Holder classes for use-defined types also implement the `org.omg.CORBA.portable.Streamable` interface.

The holder classes for the basic types are defined below. Note that they do not implement the Streamable interface. They are in the `org.omg.CORBA` package.

```
// Java
package org.omg.CORBA;

    final public class ShortHolder {
        public short value;
        public ShortHolder() {}
        public ShortHolder(short initial) {
            value = initial;
        }
    }

    final public class IntHolder {
        public int value;
        public IntHolder() {}
        public IntHolder(int initial) {
            value = initial;
        }
    }

    final public class LongHolder {
        public long value;
        public LongHolder() {}
        public LongHolder(long initial) {
            value = initial;
        }
    }

    final public class ByteHolder {
        public byte value;
        public ByteHolder() {}
        public ByteHolder(byte initial) {
```

```
        value = initial;
    }
}

final public class FloatHolder {
    public float value;
    public FloatHolder() {}
    public FloatHolder(float initial) {
        value = initial;
    }
}

final public class DoubleHolder {
    public double value;
    public DoubleHolder() {}
    public DoubleHolder(double initial) {
        value = initial;
    }
}

final public class CharHolder {
    public char value;
    public CharHolder() {}
    public CharHolder(char initial) {
        value = initial;
    }
}

final public class BooleanHolder {
    public boolean value;
    public BooleanHolder() {}
    public BooleanHolder(boolean initial) {
        value = initial;
    }
}

final public class StringHolder {
    public java.lang.String value;
    public StringHolder() {}
    public StringHolder(java.lang.String initial) {
        value = initial;
    }
}

final public class ObjectHolder {
    public org.omg.CORBA.Object value;
    public ObjectHolder() {}
    public ObjectHolder(org.omg.CORBA.Object initial) {
        value = initial;
    }
}

final public class AnyHolder {
```

```
        public Any value;
        public AnyHolder() {}
        public AnyHolder(Any initial) {
            value = initial;
        }
    }

    final public class TypeCodeHolder {
        public TypeCode value;
        public typeCodeHolder() {}
        public TypeCodeHolder(TypeCode initial) {
            value = initial;
        }
    }

    final public class PrincipalHolder {
        public Principal value;
        public PrincipalHolder() {}
        public PrincipalHolder(TypeCode initial) {
            value = initial;
        }
    }
}
```

The Holder class for a user defined type <foo> is shown below:

```
// Java
final public class <foo>Holder
    implements org.omg.CORBA.portable.Streamable {
    public <foo> value;
    public <foo>Holder() {}
    public <foo>Holder(<foo> initial) {}
    public void _read(org.omg.CORBA.portable.InputStream i)
        {...}
    public void _write(org.omg.CORBA.portable.OutputStream o)
        {...}
    public org.omg.CORBA.TypeCode _type() {...}
    }
```

Constructed Types

IDL constructed types include enum, struct, union, sequence and array. The types sequence and array are both mapped to the Java array type. The IDL constructed types enum, struct, and union are mapped to a Java class that implements the semantics of the IDL type. The Java class generated will have the same name as the original IDL type.

Enum

An IDL `enum` is mapped to a Java `final class` bearing the same name as the enum type. The final class declares a value method and two static data members per label, an integer conversion method, and a private constructor. An example follows:

Listing 2.6 An IDL enum mapped to a Java final class.

```
// Generated java

public final class <enum_name> {
    //one pair for each label in the enum
    public static final int _<label> = <value>;
    public static final <enum_name> <label> =
        new <enum_name>(_<label>);

    public int value() {...}

    // get enum with specified value
    public static <enum_name> from_int (int value);

    //constructor
    private <enum_name>(int) {...}
}
```

One of the members is a public static final that has the same name as the IDL enum label. The other has an underscore(_) prepended and is intended to be used in switch statements.

The value method returns the integer value. Values are assigned sequentially starting with 0. If the enum has a label named value, there is no conflict with the value() method in Java.

There will be only one instance of an enum. Since there is only one instance, pointer equality tests will work correctly. That is, the default `java.lang.Object` implementation of `equals` and `hash` will automatically work correctly for an enum's singleton object. The Java class for the enum also has an additional method, `fromint`, which returns the enum with the specified value.

The holder class for the enum is also generated. Its name is the enum's mapped Java classname with Holder appended to it as follows:

Listing 2.7 Mapping an enum

```
public class <enum_name>Holder implements
        org.omg.CORBA.portable.Streamable {
```

```
    public <enum_name> value;
    public <enum_name>Holder() {}
    public <enum_name>Holder(<enum_name> initial) {...}
    public void _read(org.omg.CORBA.portable.InputStream i)
        {...}
    public void _write(org.omg.CORBA.portable.OutputStream o)
        {...}
    public org.omg.CORBA.TypeCode _type() {...}
}

// IDL
enum EnumType {a, b, c};

// generated Java
public final class EnumType {
    public static final int _a = 0;
    public static final EnumType a = new EnumType(_a);

    public static final int _b = 1;
    public static final EnumType b = new EnumType(_b);

    public static final int _c = 2;
    public static final EnumType c = new EnumType(_c);

    public int value() {...}
    public static EnumType from_int(int value) { ... };

    // constructor
    private EnumType(int) {...}
};
```

Struct

An IDL struct is mapped to a final Java class with the same name that provides instance variables for the fields in IDL member ordering and a constructor for all values. A null constructor is also provided that allows the structure's fields to be initialized later. The Holder class for the struct is also generated. Its name is the struct's mapped Java classname with Holder appended to it as follows:

Listing 2.8 Mapping an IDL struct to Java.

```
final public class <class>Holder implements
        org.omg.CORBA.portable.Streamable {
    public <class> value;
    public <class>Holder() {}
    public <class>Holder(<class> initial) {...}
    public void _read(org.omg.CORBA.portable.InputStream i)
        {...}
    public void _write(org.omg.CORBA.portable.OutputStream o)
```

```
        {...}
    public org.omg.CORBA.TypeCode _type() {...}
}

/* From Example.idl: */
struct StructType {
    long field1;
    string field2;
};

// generated Java
final public class StructType {
    // instance variables
    public int field1;
    public String field2;
    // constructors
    public StructType() {}
    public StructType(int field1, String field2)
        {...}
}

final public class StructTypeHolder
        implements org.omg.CORBA.portable.Streamable {
    public StructType value;
    public StructTypeHolder() {}
    public StructTypeHolder(StructType initial) {...}
    public void _read(org.omg.CORBA.portable.InputStream i)
        {...}
    public void _write(org.omg.CORBA.portable.OutputStream o)
        {...}
    public org.omg.CORBA.TypeCode _type() {...}
}
```

Union

An IDL union is given the same name as the final Java class and mapped to it; it provides the following:

- a default constructor,
- an accessor method for the union's discriminator, named discriminator()
- an accessor method for each branch,
- a modifier method for each branch,
- a modifier method for each branch having more than one case label, and
- a default modifier method, if needed.

If there is a name clash with the mapped union type name or any of the field names, the normal name conflict resolution rule is used: prepend an underscore for the discriminator.

The methods for setting and retrieving a union branch have the same name but differ by their signature. The constructor creates the union without initializing it. The union can be initialized using the methods provided.

The holder class for the union is also generated. Its name is the union's mapped Java classname with Holder appended to it as follows:

Listing 2.9 Mapping an IDL union to Java.

```
final public class <union_class>Holder
            implements org.omg.CORBA.portable.Streamable {
    public <union_class> value;
    public <union_class>Holder() {}
    public <union_class>Holder(<union_class> initial) {...}
    public void _read(org.omg.CORBA.portable.InputStream i)
        {...}
    public void _write(org.omg.CORBA.portable.OutputStream o)
        {...}
    public org.omg.CORBA.TypeCode _type() {...}
}

/* From Example.idl: */
        union UnionType switch (EnumType) {
            case first:   long win;
            case second:  short place;
            case third:
            case fourth:  octet show;
            default:      boolean other;
        };

// Generated java
final public class UnionType {
            //constructor
            public UnionType() {...}

            //discriminator accessor
            public int discriminator() { ... }

            //win
            public int    win() { ... }
            public void   win(int value) { ... }

            //place
            public short  place() { ... }
            public void   place(short value) { ... }
```

```
            //show
            public byte   show() { ... }
            public void   show(byte value) { ... }
            public void   show(int discriminator, byte value) { ... }

            //other
            public boolean  other() {...}
            public void     other(boolean value) { ... }
}
final public class UnionTypeHolder {
                    implements org.omg.CORBA.portable.Streamable {
            public UnionType value;
            public UnionTypeHolder() {}
            public UnionTypeHolder(UnionType initial) {...}
            public _void read(org.omg.CORBA.portable.InputStream is)
        {...}
    public void _write(org.omg.CORBA.portable.OutputStream os)
        {...}
    public org.omg.CORBA.TypeCode type() {...}
}
```

Sequence

An IDL sequence is mapped to a Java array with the same name. In the mapping, anywhere the sequence type is needed, an array of the mapped type of the sequence element is used.

The holder class for the sequence is also generated. Its name is the sequence's mapped Java classname with Holder appended to it as follows:

Listing 2.10Mapping an IDL sequence to Java.

```
final public class <sequence_class>Holder {
    public <sequence_element_type>[] value;
    public <sequence_class>Holder() {};
    public <sequence_class>Holder(
                        <sequence_element_type>[] initial) {...};
    public void _read(org.omg.CORBA.portable.InputStream i)
        {...}
    public void _write(org.omg.CORBA.portable.OutputStream o)
        {...}
    public org.omg.CORBA.TypeCode _type() {...}
}
```

```
// IDL
sequence<long>UnboundedData;
sequence<long, 42> BoundedData;
```

```
// generated Java

final public class UnboundedDataHolder
                implements org.omg.CORBA.portable.Streamable {
    public int[] value;
    public UnboundedDataHolder() {};
    public UnboundedDataHolder(int[] initial) {...};
    public void _read(org.omg.CORBA.portable.InputStream is)
        {...}
    public void _write(org.omg.CORBA.portable.OutputStream os)
        {...}
    public org.omg.CORBA.TypeCode type() {...}
}

final public class BoundedDataHolder
        implements org.omg.CORBA.portable.Streamable {
    public int[] value;
    public BoundedDataHolder() {};
    public BoundedDataHolder(int[] initial) {...};
    public void _read(org.omg.CORBA.portable.InputStream is)
        {...}
    public void _write(org.omg.CORBA.portable.OutputStream os)
        {...}
    public org.omg.CORBA.TypeCode type() {...}
}
```

Array

An IDL array is mapped the same way as an IDL bounded sequence. In the mapping, anywhere the array type is needed, an array of the mapped type of array element is used. In Java, the natural Java subscripting operator is applied to the mapped array. The length of the array can be made available in Java, by bounding the array with an IDL constant, which will be mapped as per the rules for constants.

The holder class for the array is also generated. Its name is the array's mapped Java classname with Holder appended to it as follows:

Listing 2.11Mapping for an array

```
final public class <array_class>Holder
        implements org.omg.CORBA.portable.Streamable {
    public <array_element_type>[] value;
    public <array_class>Holder() {}
    public <array_class>Holder(
                        <array_element_type>[] initial) {...}
    public void _read(org.omg.CORBA.portable.InputStream i)
```

```
        {...}
    public void _write(org.omg.CORBA.portable.OutputStream o)
        {...}
    public org.omg.CORBA.TypeCode _type() {...}
}

// IDL
const long ArrayBound = 42;
long larray[ArrayBound];

// generated Java

final public class larrayHolder
                implements org.omg.CORBA.portable.Streamable {
    public int[] value;
    public larrayHolder() {}
    public larrayHolder(int[] initial) {...}
    public void _read(org.omg.CORBA.portable.InputStream is)
        {...}
    public void _write(org.omg.CORBA.portable.OutputStream os)
        {...}
    public org.omg.CORBA.TypeCode type() {...}
}
```

Interfaces

IDL interfaces are mapped to public Java interfaces and given the same names. An additional “helper” Java class with the suffix Helper is appended to the interface name. The Java interface extends the mapped, base `org.omg.CORBA.Object` interface.

The Java interface contains the mapped operation signatures. Methods can be invoked on an object reference to this interface.

The helper class holds a static narrow method that allows an instance of `org.omg.CORBA.Object` to be narrowed to the object reference of a more specific type. The IDL exception CORBA::BAD_PARAM is thrown if the narrow fails.

There are no special “nil” object references. Java `null` can be passed freely wherever an object reference is expected.

Attributes are mapped to a pair of Java accessor and modifier methods. These methods have the same name as the IDL attribute and are overloaded. There is no modifier method for IDL readonly attributes.

The holder class for the interface is also generated. Its name is the interface's mapped Java classname with Holder appended to it as follows:

Listing 2.12Mapping an IDL interface to Java.

```
final public class <interface_class>Holder
        implements org.omg.CORBA.portable.Streamable {
    public <interface_class> value;
    public <interface_class>Holder() {}
    public <interface_class>Holder(
                        <interface_class> initial) {
        value = initial;
    public void _read(org.omg.CORBA.portable.InputStream i)
        {...}
    public void _write(org.omg.CORBA.portable.OutputStream o)
        {...}
    public org.omg.CORBA.TypeCode _type() {...}
}
```

```
/* From Example.idl: */
module Example {
    interface Foo {
        long method(in long arg) raises(e);
        attribute long assignable;
        readonly attribute long nonassignable;
    }
}

// Generated java

package Example;

public interface Foo extends org.omg.CORBA.Object {
    int method(int arg) throws Example.e;
    int assignable();
    void assignable(int i);
    int nonassignable();
}

public class FooHelper {

    // ... other standard helper methods

    public static Foo narrow(org.omg.CORBA.Object obj)
        {...}
}

final public class FooHolder
        implements org.omg.CORBA.portable.Streamable {
    public Foo value;
    public FooHolder() {}
```

```
    public FooHolder(Foo initial) {...}
    public void _read(org.omg.CORBA.portable.InputStream is)
        {...}
    public void _write(org.omg.CORBA.portable.OutputStream os)
        {...}
    public org.omg.CORBA.TypeCode type() {...}
}
```

Passing Parameters

IDL defines three parameter passing modes: in, out, and inout.

The `in` parameters implement call-by-value semantics, so IDL `in` parameters are supplied by passing the actual parameter when a method is invoked. The results of IDL operations are returned as the result of the corresponding Java method.

The IDL `out` and `inout` parameters implement call-by-result and call-by-value/result semantics, so they cannot be mapped directly into the Java parameter passing mechanism. The mapping mechanism defines holder classes for all the IDL basic and user-defined types which are used to implement the parameter modes in Java. The client supplies an instance of the appropriate Java holder class that is passed by value for each IDL `out` or `inout` parameter. The contents of the holder instance, but not the instance itself, is modified by the invocation and the client uses the changed contents, if any changes were made, after the invocation returns.

Listing 2.13IN parameter mapping to Java actual parameters.

```
/* From Example.idl: */
module Example {
   interface Modes {
      long operation(in long inArg, out long outArg, inout long inoutArg);
   };
};

// Generated Java:
package Example;
public interface Modes {
   int operation(int inArg, IntHolder outArg, IntHolder inoutArg);
}
```

In the example above, the result comes back as an ordinary result and the `in` parameter is an ordinary value. But for the `out` and `inout` parameters, an appropriate holder must be constructed. A typical use case might look as follows:

```
// user Java code
// select a target object
Example.Modes target = ...;
// get the in actual value
int inArg = 57;
// prepare to receive out
IntHolder outHolder = new IntHolder();
// set up the in side of the inout
IntHolder inoutHolder = new IntHolder(131);
// make the invocation
int result =target.operation(inArg, outHolder, inoutHolder);
// use the value of the outHolder
... outHolder.value ...
// use the value of the inoutHolder
... inoutHolder.value ...
```

Before the invocation, the input value of the inout parameter must be set in the holder instance that will be the actual parameter. The inout holder can be filled in either by constructing a new holder from a value, or by assigning to the value of an existing holder of the appropriate type. After the invocation, the client uses the outHolder.value to access the value of the out parameter, and the inoutHolder.value to access the output value of the inout parameter. The return result of the IDL operation is available as the result of the invocation.

Server Implementation with Inheritance

Using inheritance is the simplest way to implement a server because server objects and object references look the same, behave the same, and can be used in exactly the same contexts. If a server object happens to be in the same process as its client, method invocations are an ordinary java function call with no transport, indirection, or delegation of any kind.

Each IDL interface is mapped to a Java skeleton abstract class that implements the Java version of the IDL interface. User defined server classes are then linked to the ORB by extending the skeleton class, as shown below.

Listing 2.14Server implementation in Java using inheritance.

```
/* From Example.idl: */
module Example {
```

```
   interface Account {
   };
};

// Generated java
package Example;
abstract public class _sk_Account
   extends org.omg.CORBA.Skeleton
   implements Account { ... }


// Linking an implementation to the ORB :
class AccountImpl extends Example._sk_Account { ... }
```

Server Implementation with Delegation

The use of inheritance to implement a server has one drawback: The server class extends the skeleton class, so it cannot use implementation inheritance for other purposes because Java only supports single inheritance. If the server class needs to use the inheritance link for another purpose, the delegation approach must be used.

When server classes are implemented using delegation, some extra code is generated:

- Each interface is mapped to a Tie class that extends the skeleton and provides the delegation code.
- Each interface is also mapped to an Operations interface that is used to defined the type of object the Tie class is delegating.

The Operations interface does not extend CORBA.Object and, as a result, its instances cannot be treated directly as ORB objects. Instead, they have to be stored in a Tie class instance. Storing the instance of the Operation interface in the Tie object is done through a constructor provided by the Tie class. The following code example shows an example of how delegation is used.

Listing 2.15Server implementation in Java using delegation.

```
/* From Example.idl: */
module Example {
    interface Account
        float balance();
    };
};
```

```
// Generated java
package Example;
public interface AccountOperations {
    public float balance();
};


package Example;
// Generated java
public class _tie_Account extends _sk_Account {
private AccountOperations delegate;
    public _tie_Account(AccountOperations d) { ... }
    public _tie_Account(AccountOperations d, String name) { ... }
    public float balance() {
        return delegate.balance();
    }
}


// Linking an implementation to the ORB :
class AccountImpl implements Example.AccountOperations extends Whatever { ...}
...
Example.Account a_server = new Example._tie_Account(new AccountImpl());
...
```

Interface Scope

Java does not allow declarations to be nested within an interface scope nor does it allow packages and interfaces to have the same name. Accordingly, interface scope is mapped to a package with the same name with an underscore suffix.

Mapping for Certain Nested Types

IDL allows type declarations nested within interfaces. Java does not allow classes to be nested within interfaces. Hence those IDL types that map to Java classes and that are declared within the scope of an interface must appear in a special “scope” package when mapped to Java.

IDL interfaces that contain these type declarations generate a scope package to contain the mapped Java class declarations. The scope package name is constructed by appending Package to the IDL type name.

Listing 2.16Mapping for certain nested types

```
// IDL
module Example {
    interface Foo {
        exception e1 {};
    };
}

// generated Java
package Example.FooPackage;
final public class e1 extends org.omg.CORBA.UserException {...}
```

Mapping for Typedef

Java does not have a typedef construct.

IDL types that are mapped to simple Java types may not be subclassed in Java. Hence any typedefs that are type declarations for simple types are mapped to the original (mapped type) everywhere the typedef type appears. For simple types, Helper classes are generated for all typedefs.

Typedefs for non arrays and sequences are “unwound” to their original type until a simple IDL type or user-defined IDL type (of the non typedef variety) is encountered.

Holder classes are generated for sequence and array typedefs.

```
// IDL
struct EmpName {
    string firstName;
    string lastName;
};
typedef EmpName EmpRec;

// generated Java
   // regular struct mapping for EmpName
   // regular helper class mapping for EmpRec

final public class EmpName {
    ...
}
```

```
public class EmpRecHelper {
    ...
}
```

Mapping for Exceptions

IDL exceptions are mapped very similarly to structs. They are mapped to a Java class that provides instance variables for the fields of the exception and constructors.

CORBA system exceptions are unchecked exceptions. They inherit (indirectly) from `java.lang.RuntimeException`.

User-defined exceptions are checked exceptions. They inherit (indirectly) from `java.lang.Exception`.

User-defined Exceptions

User-defined exceptions are mapped to Java classes that extend `org.omg.CORBA.UserException` and are otherwise mapped just like the IDL struct type, including the generation of Helper and Holder classes.

If the exception is not defined within a nested IDL scope (essentially within an interface), its Java class name is defined within a special scope. Otherwise, its Java class name is defined within the scope of the Java package that corresponds to the exception's enclosing IDL module.

Listing 2.17Mapping User-defined Exceptions

```
// IDL
module Example {
    exception ex1 {string reason;}
}

// Generated Java

package Example;

final public class ex1 extends org.omg.CORBA.UserException {
    public String reason;         // instance
    public ex1() {...}            // default constructor
```

```
    public ex1(String r) {...}    // constructor
}

final public class ex1Holder
                implements org.omg.CORBA.portable.Streamable {
    public ex1 value;
    public ex1Holder() {}
    public ex1Holder(ex1 initial) { ... }
    public void _read(org.omg.CORBA.portable.InputStream is)
        {...}
    public void _write(org.omg.CORBA.portable.OutputStream os)
        {...}
    public org.omg.CORBA.TypeCode type() {...}
}
```

Chapter

3

Generated Classes

This chapter describes the classes that can be generated by the `idl2java` compiler. It contains the following sections:

- Overview
- <class_name>Helper
- <class_name>Holder
- <interface_name>Operations
- _st_<class_name>
- _sk_<class_name>
- _tie_<class_name>

Overview

The Helper and Holder classes are provided for all the classes in the `org.omg.CORBA` package. They are also generated by the `idl2java` compiler for user-defined types and are given the name of the class that is generated for the type with an additional Holder or Helper suffix. Given a user-defined type named Foo, the `idl2java` compiler will generate the following:

- public class Foo
- public class FooHolder
- public class FooHelper

The Helper Class

An abstract Helper class is generated by the `idl2java` compiler and contains the utility methods for operating on the associated object. The motivation for the Helper class is to avoid loading the methods that the class offers if they are not needed. The Holder class, also generated by the `idl2java` compiler, is used when an object needs to be passed as a parameter for an operation request.

For objects like structures, enumerations, and unions, the Helper class provides methods for reading and writing the object to a stream and returning the object's repository identifier. The Helper classes generated for interfaces contain additional methods, like bind and narrow.

The Holder Class

Because the Java language only allows parameters to be passed by value and not by reference, Holder classes are used to support the passing of `out` and `inout` parameters associated with operation requests. The interface of the Holder class is consistent for all types.

The Stub Class

The stub class provides stub implementation for <class_name> which the client calls.

The Skeleton Class

A skeleton class is used to derive an implementation class for <class_name>.

The Tie Class

This class enherits from the skeleton class and delegates every call to the real implementation class. For an example of the tie mechanism, see “The Tie Mechanism” in the *Netscape Internet Service Broker for Java Programmer's Guide*.

The Operations Class

This class defines all of the methods that must be implemented by the object implementation. This class acts as the delegate object for the associated _tie class when the tie mechanism is used. For an example of the tie mechanism, see “The Tie Mechanism” in the *Netscape Internet Service Broker for Java Programmer's Guide*.

Examples Class

This class provides code you can fill in to implement the object. After adding code, save the file using a different file name. This preserves your work if you run the compiler again and generate another (empty) example file.

<class_name>Helper

```
abstract public class <class_name>Helper
```

A Helper class is provided for most classes in the `org.omg.COBRA` package. Helper classes are also generated by the `idl2java` compiler for all user-defined types. The suffix `Helper` is added to the class name that is generated for the type. A variety of static methods are provided to manipulate the class.

Class Methods	Interface Methods
extract	bind
id	narrow
insert	

Class Methods	Interface Methods
read	
type	
write	

```
public static int[ ] extract(org.omg.CORBA.Any any)
```

This method extracts the type from the specified Any object.

Parameter	Description
any	The Any object to contain the object.

```
public static String id()
```

This method gets the repository id for this object.

```
public static void insert(org.omg.CORBA.Any any, int[ ] seq)
```

This method inserts a type into the specified Any object.

Parameter	Description
any	The Any object to contain the type.
seq	The type to insert.

```
public static int[ ] read(org.omg.CORBA.portable.InputStream is)
```

This method reads a type from the specified input stream.

Parameter	Description
is	The input stream from which the type is read.

```
public static org.omg.CORBA.TypeCode type()
```

This method returns the TypeCode associated with this object. For a list of possible return values see “TCKind.”

```
public static void write(org.omg.CORBA.portable.OutputStream os, int[ ]
```

seq)

This method writes a type to the specified output stream.

Parameter	Description
os	The output stream to which the object is written.
seq	The type to be written to the output stream.

Methods Generated for Interfaces

```
public static <class_name> narrow(org.omg.CORBA.Object object)
```

This method attempts to narrow a org.omg.CORBA.Object reference to an object of type <class_name>. If the object reference cannot be narrowed, a null value is returned.

Parameter	Description
object	The object to be narrowed to the type <class_name>.

```
public static <class_name> bind(org.omg.CORBA.ORB orb)
```

This method attempts to bind to any instance of an object of type <class_name>.

```
public static <class_name> bind(org.omg.CORBA.ORB orb, String name)
```

This method attempts to bind to an object of type <class_name> that has the specified instance name.

Parameter	Description
name	The instance name of the desired object.

```
public static <class_name> bind(org.omg.CORBA.ORB orb, String name, String host)
```

This method attempts to bind to an object of type <class_name> that has the specified instance name and which is located on the specified host.

Parameter	Description
name	The instance name of the desired object.
host	The host name where the desired object is located.

```
public static <class_name> bind(org.omg.CORBA.ORB orb, String name,
String host, org.omg.CORBA.BindOptions options)
```

This method attempts to bind to an object of type <class_name> that has the specified instance name and which is located on the specified host, using the specified BindOptions.

Parameter	Description
name	The instance name of the desired object.
host	The optional host name where the desired object is located.
options	The bind options for this object.

<class_name>Holder

```
public class <class_name>Holder
```

A Holder class is provided for all basic IDL types in the `org.omg.CORBA` package. Holder classes are also generated by the `idl2java` compiler for all user-defined types. The suffix `Holder` is added to the class name that is generated for user-defined types. Each Holder has a set of constructors and a value member, which is the typed value.

The holder classes for the basic types are defined below. They are in the `org.omg.CORBA` package.

- public class ShortHolder
- public class IntHolder
- public class LongHolder
- public class ByteHolder
- public class FloatHolder
- public class DoubleHolder

- public class CharHolder
- public class BooleanHolder
- public class StringHolder
- public class ObjectHolder
- public class AnyHolder
- public class TypeCodeHolder
- public class PrincipalHolder

The Holder class for a user defined type <class_name> is shown below:

```
// Java
final public class <class_name>Holder implements
org.omg.CORBA.portable.Streamable {
    public <class_name> value;
    public <class_name>Holder() {};
    public <class_name>Holder(<class_name> initial) {};
    public void _read(org.omg.CORBA.portable.InputStream is)
        {...};
    public void _write(org.omg.CORBA.portable.OutputStream os)
        {...};
    public org.omg.CORBA.TypeCode _type() {...};
}
```

Member Data

```
public <class_name> value;
```

This value represents the type contained by this object.

Methods

```
public <class_name>Holder()
```

This default constructor is useful for `out` parameters. The default constructor sets the value field to the default value for the type as defined by the Java language. The value is set to `false` for boolean types, 0 for integral and char types, `null` for strings, and `null` for object references.

public **<class_name>Holder**(<class_name> **inital**)

The inital value constructor is useful for `inout` parameters. The value of **inital** is copied from the `value` field of the specified <class_name> object.

Parameter	Description
inital	The inital value of the `inout` parameter.

<interface_name>Operations

```
abstract public interface <interface_name>Operations
```

A Operations class is generated by the `idl2java` compiler and is meant to contain the implementations of the method for <interface_name>.

_st_<class_name>

```
abstract public class _st_<class_name>
```

A stub class is generated by the `idl2java` compiler to provide a stub implementation for <class_name> which the client calls. This class provides the implementation for transparently acting on an object implementation.

_sk_<class_name>

```
abstract public class _sk_<class_name>
```

A skeleton class is generated by the `idl2java` compiler and is used to derive an implementation class for <class_name>.

tie<class_name>

```
abstract public class <class_name>
```

A tie class is generated by the `idl2java` compiler for creating a delegator class for <class_name>.

Chapter

4

Core Interfaces and Classes

This chapter describes the core interfaces and classes in the `org.omg.CORBA` package. It contains the following sections.

- ARG_IN
- ARG_INOUT
- ARG_OUT
- BOA
- CompletionStatus
- Context
- InvalidName
- netscape.WAI.Naming
- Object
- ORB
- Principal

ARG_IN

```
final public class ARG_IN
```

ARG_IN is used to designate parameters for dynamic invocation interface requests that are only used for input purposes and will not be modified by the server. See also: Request and NVList.

Variables

```
final public static int value = (int) 1;
```

ARG_INOUT

```
final public class ARG_INOUT
```

ARG_INOUT is used to designate parameters for dynamic invocation interface requests that are used for input purposes, but may also be modified by the server upon return to the client. See also: Request and NVList.

Variables

```
final public static int value = (int) 3;
```

ARG_OUT

```
final public class ARG_OUT
```

ARG_OUT is used to designate parameters for dynamic invocation interface requests that are used for input purposes, but may also be modified by the server upon return to the client. See also: Request and NVList.

Variables

```
final public static int value = (int) 2;
```

BOA

```
public interface BOA
```

The Basic Object Adapter is used by object implementations to activate and deactivate the objects they offer to clients. An object implementation invokes the `obj_is_ready` method to make the implementation visible to clients on the network. The `deactivate_obj` method is used to makes an object implementation unavailable to clients.

The BOA also provides methods for obtaining the Principle associated with a client request and entering an event loop to wait for the receipt of client requests.

The BOA instance is obtained by using the method `ORB.BOA_init()`. For example:

```
try {
            org.omg.CORBA.ORB orb = CORBA.ORB.init();
            org.omg.CORBA.BOA boa = orb.BOA_init();
            // create a new implementation object
            Bank bank = new Bank();
            // make the implementation Net visible
            boa.obj_is_ready(bank);
            // wait for incoming requests...
            boa.impl_is_ready();
            ...
}
```

For multiple calls to `BOA_init`, you can create multiple orbs with one BOA per orb. See also the Object interface.

IDL Definition

```
interface BOA {
        void deactivate_obj(in CORBA::Object object);
        CORBA::Principal get_principal(in CORBA::Object object);
        void impl_is_ready();
        void obj_is_ready(in CORBA::Object object);
};
```

Methods

```
public void deactivate_obj(Object object)
```

This method makes a server's implementation object *invisible* to the network. After invoking this method, requests received for this object will cause a CORBA.NO_IMPLEMENT exception to be raised.

Parameter	Description
object	The server's implementation object to be deactivated.

```
public Principal get_principal(Object object)
```

This method returns the `Principal` object associated with the current operation request or null if the specified object is not the target of the current invocation.

Parameter	Description
object	The server object on which the current operation request is being called. Normally, this object reference is to this object.

```
public void impl_is_ready()
```

This method is invoked by a server after all the objects that it implements have been activated using object_is_ready method. This method enters a loop waiting for client operation requests to arrive and does not return. Calling this method is not required if the server has some other non-daemon thread running. For example, if there is a GUI running, the GUI will control the lifetime of the program and this method need not be used.

```
public void obj_is_ready(Object object)
```

This method makes an object implementation provided by a server visible on the network. Servers that implement more than one object must make a separate call to this method for each object that they offer.

Parameter	Description
object	The server's implementation object to be activated.

CompletionStatus

```
public final class CompletionStatus extends Object
```

This class works with SystemException and indicates whether the operation completed before the exception was raised.

IDL definition

```
enum CompletionStatus { COMPLETED_YES, COMPLETED_NO, COMPLETED_MAYBE };
```

Methods

For more information about these methods, see “Enum.”

```
public final static int _COMPLETED_YES
public final static int _COMPLETED_NO
public final static int _COMPLETED_MAYBE
public final static CompletionStatus COMPLETED_YES
public final static CompletionStatus COMPLETED_NO
public final static CompletionStatus COMPLETED_MAYBE
public int value()
public static CompletionStatus from_int(int value)
```

Context

```
public interface Context
```

The Context interface contains a property list for a client. This property list is propagated to the server when a client makes a request. Because the CORBA specification does not define the contents of a Context, the uses of these properties are left to the user and implementor to define. Context objects are organized as a tree with each containing a pointer to its parent context. The root context is the *global default context*, whose parent is null. The default context is obtained by using the ORB get_default_context method.

IDL Definition

```
interface Context {
      CORBA::Identifier context_name();
      CORBA::Context create_child(in CORBA::Identifier context_name);
      void delete_values(in CORBA::Identifier prop_name);
      CORBA::NVList get_values(in CORBA::Identifier start_scope,
                               in boolean restrict_scope,
                               in CORBA::Identifier prop_name);
      CORBA::Context parent();
      void set_one_value(in CORBA::Identifier prop_name, in any value);
      void set_values(in CORBA::NVList values);
    };
```

Methods

`public String` **`context_name`**`()`

This method returns the name of this `Context`.

`public Context` **`create_child`**`(String` **`context_name`**`)`

This method creates a child (leaf) `Context` with the specified parent `Context`. This method returns the newly created child context.

Parameter	Description
context_name	The name of the child Context to be created.

`public void` **`delete_values`**`(String` **`prop_name`**`)`

This method removes all properties with the specified name from the current Context. You can use an asterisk as a wildcard character at the end of the prop_name.

Parameter	Description
name	The name of the property to be removed.

`public org.omg.CORBA.NVList` **`get_values`**`(String` **`start_scope`**`, boolean` **`restrict_scope`**`, String` **`prop_name`**`)`

This method returns the properties associated with the current Context as an NVList of name-value pairs. Scope of the Context search may be limited using the start_scope and restrict_scope parameters. You can use an asterisk as a wildcard at the end of the prop_name. This method returns a name value list for the specified search. If the start_scope is not null and the corresponding context is not found, this method throws BAD_PARAM.

Parameter	Description
start_scope	The name of the Context where the search is to begin.
restrict_scope	True indicates that the search is only for the current context or the scope matching start_scope, if not null. False indicates that the search includes the current context as well as its ancestors.
prop_name	The name of the property to be returned.

```
public org.omg.CORBA.Context parent()
```

This method returns the parent Context for this object. If this object is the default global context, NULL is returned.

```
public void set_one_value(String prop_name, org.omg.CORBA.Any value)
```

This method adds a new property to the current Context. The value of the property is represented by the Any class.

Parameter	Description
prop_name	The name of the new property.
value	An Any object that contains the value of the new property.

```
public void set_values(org.omg.CORBA.NVList values)
```

This method sets the properties of the current Context using the supplied NVList, containing one or more name-value pairs.

Parameter	Description
values	The list of properties for the Context.

InvalidName

```
public class org.omg.CORBA.ORBPackage.InvalidName extends org.omg.CORBA.UserException
```

The exception is raised by the ORB resolve_initial_references method.

Helper and Holder versions of this class are also provided. See “Generated Classes” for more information on these classes and the methods they offer.

netscape.WAI.Naming

The Naming class uses URL's of the form `http://host/path/object_name` to resolve and register objects by name.

Methods

```
public static org.omg.CORBA.Object resolve(String url) throws SystemException
```

This method resolves a URL of the form `http://host[:port]/path/name`.

Parameter	Description
url	The URL to resolve.

```
public static void register(String url, org.omg.CORBA.Object obj) throws SystemException
```

This method uses WebNaming to register an object with a URL of the form `http://host[:port]/path/name`. The caller's machine must be allowed to do an HTTP 'PUT' on the designated host.

Parameter	Description
url	The URL at which to register.
obj	The object to register.

Object

```
public interface Object
```

The Object interface is the root of the CORBA inheritance hierarchy. All interfaces defined in IDL inherit from this interface. This interface provides platform independent run-time type information and object reference equivalence testing.

A Holder version of this class is also provided, as described in “Generated Classes.”

IDL Definition

```
interface Object {
       attribute ::CORBA::BindOptions _bind_options;
       attribute ::CORBA::Principal _principal;

      CORBA::BOA _boa();
      CORBA::Object _clone();
      CORBA::Request _create_request(
        in CORBA::Context ctx,
        in CORBA::Identifier operation,
        in CORBA::NVList arg_list,
        in CORBA::NamedValue result
      );
      CORBA::Request _create_request(
        in CORBA::Context ctx,
        in CORBA::Identifier operation,
        in CORBA::NVList arg_list,
        in CORBA::NamedValue result,
        in CORBA::ExceptionList exceptions,
        in CORBA::ContextList contexts
      );
      CORBA::Object _duplicate();
      CORBA::InterfaceDef _get_interface();
      CORBA::ImplementationDef _get_implementation();
       unsigned long _hash(in unsigned long maximum);
       boolean _is_a(in CORBA::RepositoryId repId);
       boolean _is_bound();
```

```
    boolean _is_equivalent(in CORBA::Object other_object);
    boolean _is_local();
    boolean _is_persistent();
    boolean _is_remote();
    boolean _non_existent();
   CORBA::Identifier _object_name();
   CORBA::ORB _orb();
    void _release();
   CORBA::RepositoryId _repository_id();
   CORBA::Request _request(in CORBA::Identifier operation);
};
```

Methods

`public org.omg.CORBA.BindOptions` **`_bind_options`**`()`

This method returns the bind options associated with this object.

`public void` **`_bind_options`**`(org.omg.CORBA.BindOptions` **`_bind_options`**`)`

This method is used by client applications and sets the bind options associated with this object.

Bind options cannot be set on an implementation object. They can only be set on a client's proxy object.

Parameter	Description
_bind_options	The bind options to be set for this object.

`public org.omg.CORBA.BOA` **`_boa`**`()`

This method returns the singleton Basic Object Adapter. If the BOA has not been initialized, a CORBA.INITIALIZE exception is raised.

`public org.omg.CORBA.Object` **`_clone`**`()`

This method creates a copy of this Object, which then has its own TCP connection to the server. If the object implementation is local, the pointer of the implementation object is returned.

`public org.omg.CORBA.Request` **`_create_request`**`(org.omg.CORBA.Context` **`ctx`**`, nString` **`operation`**`, org.omg.CORBA.NVList` **`arg_list`**`, org.omg.CORBA.NamedValue` **`result`**`)`

This method creates an dynamic invocation request initialized with the specified parameters. For information on obtaining the default context, see "get_default_context."

Parameter	Description
ctx	The Context to use for the dynamic invocation request.
operation	The name of the operation to be invoked.
arg_list	A list of NamedValue items. There is one NamedValue for each argument that is to be passed to the operation.
result	The type of the return value.

```
public org.omg.CORBA.Request _create_request(org.omg.CORBA.Context ctx, String operation,
org.omg.CORBA.NVList arg_list, org.omg.CORBA.NamedValue result, org.omg.CORBA.Typecode[]
exceptions, String[] contexts)
```

This method creates an dynamic invocation request initialized with the specified parameters, including a list of exceptions that the request may raise. For information on obtaining the default context, see "get_default_context."

Parameter	Description
ctx	The Context to use for the dynamic invocation request.
operation	The name of the operation to be invoked.
arg_list	A list of NamedValue items. There is one NamedValue for each argument that is to be passed to the operation.
exceptions	A list of Typecode objects representing the exceptions that this request might raise.
contexts	A list of Context objects. The list of Context objects supports type checking on context names.

```
public org.omg.CORBA.InterfaceDef _get_interface()
```

This method returns the interface definition of this object. An interface repository must be available if this method is to be used. For information about starting an interface repository, see "Commands."

```
public int _hash(int maximum)
```

This method computes a hash value for this object in the range of [0 - `maximum`]. The value returned is always positive.

Parameter	Description
maximum	The maximum hash value to return.

```
public boolean _is_a(String repid)
```

This method queries an object to see if it implements the specified interface. This method returns true if the implementation object supports the interface. Otherwise, false is returned.

Parameter	Description
repId	A String containing the repository identifier of the desired interface.

Invoking this method may result in a call to the implementation object, since it is possible for a given server to simultaneously implement multiple interfaces via multiple inheritance, as shown in the following IDL example.

Listing 4.1 Interface C inherits from both interface A and interface B.

```
module M {
    interface A {
        void opA();
    };
    interface B {
        void opB();
    };
    interface C : A, B {
    };
};
```

Given an interface A in module M, (i.e., M::A) the corresponding repository identifier will generally be “IDL:M/A:1.0” (omitting the quotation marks). The repository identifier can also be set to an arbitrary string, using #pragmas in the IDL file.

```
public boolean _is_bound()
```

This method returns `true` if a TCP connection has been established with the implementation object. Otherwise, `false` is returned.

```
public boolean _is_equivalent(org.omg.CORBA.Object other_object)
```

This method compares this object with the specified object and returns `true` if they both refer to the same implementation object. Otherwise, `false` is returned.

This method does not simply compare pointers. Two object references may be locally distinct yet still refer to the same object implementation.

Parameter	Description
other_object	A reference to the object to be compared with this object.

```
public boolean _is_local()
```

This method returns `true` if this object refers to an object implemented in the local address space. Otherwise, `false` is returned.

```
public boolean _is_persistent()
```

This method returns `true` if this object reference is valid beyond the lifetime of the process that implements the object. Otherwise, `false` is returned.

```
public boolean _is_remote()
```

This method returns `true` if this object refers to an object implemented in a remote address space. Otherwise, `false` is returned.

```
public boolean _non_existent()
```

This method attempts to *ping* the implementation object to determine if it is active. This method returns `false` if the implementation object is currently active (possibly after causing the server to be activated). Otherwise, `true` is returned. This method does not cause a client's proxy object to rebind to another server. In other words, it does force a bind, but it does not force a rebind.

```
public String _object_name()
```

This method returns name of the object implementation. If the object is not named, NULL is returned. Transient objects and foreign objects are not named.

```
public org.omg.CORBA.BOA _orb()
```

This method returns a handle to an Object Request Broker. If the ORB has not been initialized, a CORBA.INITIALIZE exception is raised.

```
public Principal _principal()
```

This method is used by object implementations to obtain the principal associated with a request on this object. The principal can be associated with a client's proxy object or with the target of an active operation request. A NULL value is returned if this object is not currently processing an operation request.

Client applications and object implementations may set the principal using the `_principal` method, shown below.

```
public void _principal(org.omg.CORBA.Princpal principal)
```

This method sets the principal associated with this object. A principal can either be associated with a proxy object or with the target of an active invocation. Attempting to obtain the principal of an implementation while not in an invocation results in a NULL. The principal is stored by reference, so be careful when modifying the value.

Parameter	Description
principal	A principal to set for this object.

```
public String _repository_id()
```

This method returns the repository identifier of the object implementation's most derived interface.

```
public org.omg.CORBA.Request _request(String operation)
```

This method creates an empty dynamic invocation request. Both in and inout parameters must be initialized prior to sending the request. The types must also be initialized for out parameters and return values. See “Request” for more information initializing and sending dynamic invocation requests.

Parameter	Description
operation	The name of the operation to be invoked.

ORB

```
abstract public class ORB
```

This class provides a method for initializing the CORBA infrastructure, as shown in the following code example. The Object Request Broker, along with the Basic Object Adaptor, provides a variety of methods used by both clients and servers.

Listing 4.2 Example client usage of the ORB class.

```
 public class SimpleClientProgram {
    public static void main(String args[]) {
        try {
            org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init();
            org.omg.CORBA.Object object = orb.string_to_object(args[0]);
            System.out.println("Contacted object: " + object);
        }
        catch(org.omg.CORBA.SystemException se) {
            System.out.println("Failure: " + se);
        }
    }
}
```

Following is the IDL definition of the ORB interface.

```
pseudo interface ORB {
   void connect(Object obj);
   TypeCode create_alias_tc ( in RepositoryId id,
     in Identifier name,
     in TypeCode original_type);
   Any create_any();
   TypeCode create_array_tc ( in unsigned long length,
     in TypeCode element_type);
   ContextList create_context_list();
   Environment create_environment();
   TypeCode create_enum_tc ( in RepositoryId id,
     in Identifier name,
     in EnumMemberSeq members);
   ExceptionList create_exception_list();
   TypeCode create_exception_tc ( in RepositoryId id,
     in Identifier name,
     in StructMemberSeq members);
   TypeCode create_interface_tc ( in RepositoryId id,
     in Identifier name);
   NVList create_list(in long count);
   NamedValue create_named_value(in String name, in Any value, in Flags flags);
   NVList create_operation_list(in OperationDef oper);
   OutputStream create_output_stream();
```

```
  TypeCode create_recursive_sequence_tc( in unsigned long bound,
    in unsigned long offset);
  TypeCode create_sequence_tc ( in unsigned long bound,
    in TypeCode element_type);
  TypeCode create_string_tc ( in unsigned long bound);
  TypeCode create_struct_tc ( in RepositoryId id,
    in Identifier name,
    in StructMemberSeq members);
  TypeCode create_union_tc ( in RepositoryId id,
    in Identifier name,
    in TypeCode discriminator_type,
    in UnionMemberSeq members);
  void disconnect(Object obj);
  Current get_current();
  Context get_default_context();
  Request get_next_response();
  TypeCode get_primitive_tc( in TCKind tcKind);
exception InvalidName {};
typedef string ObjectId;
  typedef sequence<ObjectId> ObjectIdList;
  ObjectIdList list_initial_services();
  string object_to_string(in Object obj);
  boolean poll_next_response();
  Object resolve_initial_references(in ObjectId object_name)
    raises(InvalidName);
  void send_multiple_requests_deferred(in RequestSeq req);
  void send_multiple_requests_oneway(in RequestSeq req);
  Object string_to_object(in string str);
```

Methods

`public static org.omg.CORBA.BOA` **`BOA_init`**`()`

This method initializes the BOA singleton and returns a reference to the BOA singleton. Like the init method, BOA_init can be called repeatedly and at anytime to obtain a reference to the BOA. Returns the BOA singleton.

`public static org.omg.CORBA.BOA` **`BOA_init`**`(java.applet.Applet` **`applet`**`)`

This method initializes the BOA singleton from an applet and returns a reference to the BOA. This method can be called repeatedly and at anytime to obtain a reference to the BOA.

Parameter	Description
applet	The repository identifier that specifies the type in IDL.

```
abstract public org.omg.CORBA.TypeCode create_alias_tc(String repository_id,
                                                       String type_name,
                                                       org.omg.CORBA.TypeCode
original_type)
```

This method creates and returns a TypeCode describing an IDL alias.

Parameter	Description
repository_id	The repository identifier that specifies the type in IDL.
type_name	The un-scoped type name of the type.
original_type	The aliased type.

```
abstract public org.omg.CORBA.Any create_any()
```

This method creates an empty Any object with a NULL type code.

```
abstract public org.omg.CORBA.TypeCode create_array_tc(int length,
                org.omg.CORBA.TypeCode element_type)
```

This method creates and returns a TypeCode describing an IDL array.

Parameter	Description
length	The length of the array.
element_type	The type of the elements contained in the array.

```
public abstract ContextList create_context_list()
```

This method creates and returns an empty ContextList.

```
abstract public org.omg.CORBA.TypeCode create_enum_tc(String repository_id,
                                                      String type_name,
                                                      String members[])
```

This method creates and returns a TypeCode describing an IDL enumeration.

Parameter	Description
repository_id	The repository identifier that specifies the type in IDL.
type_name	The un-scoped type name of the type.
members	An array of strings defining the members of the type.

```
abstract public org.omg.CORBA.Environment create_environment()
```

This method creates and returns an empty Environment.

```
abstract public org.omg.CORBA.TypeCode create_estruct_tc(String repository_id, String
type_name, org.omg.CORBA.TypeCode base org.omg.CORBA.StructMember members[])
```

This method creates and returns a TypeCode describing an IDL estruct.

Parameter	Description
repository_id	The repository identifier that specifies the type in IDL.
type_name	The un-scoped type name of the type.
base	The estruct from which the current estruct inherits. Use NULL to indicate that this is the root structure.
members	An array of structures defining the members of the type.

```
abstract public org.omg.CORBA.TypeCode create_exception_tc(String repository_id,
                                                           String type_name,
                                                           org.omg.CORBA.StructMember
members[])
```

This method creates and returns a TypeCode describing an IDL exception.

Parameter	Description
repository_id	The repository identifier that specifies the type in IDL.
type_name	The un-scoped type name of the type.
members	An array of structures defining the members of the type.

```
abstract public org.omg.CORBA.InputStream create_input_stream(org.omg.CORBA.OutputStream
```

ostream)

This method creates an IIOP input stream from an IIOP output stream. All bytes written to the output stream will be available to be read from the input stream.

Parameter	Description
ostream	The output stream from which the input stream is created.

```
abstract public org.omg.CORBA.TypeCode create_interface_tc(
String repository_id, String type_name)
```

This method creates and returns a TypeCode describing an IDL interface.

Parameter	Description
repository_id	The repository identifier that specifies the type in IDL.
type_name	The un-scoped type name of the type.

```
abstract public org.omg.CORBA.NVList create_list(int length)
```

This method creates and returns an NVList of the specified length.

Parameter	Description
length	The length of the list to be created.

```
abstract public org.omg.CORBA.NamedValue create_named_value(String name
org.omg.CORBA.Any value, int flags)
```

This method creates and returns a new NamedValue for the Dynamic Invocation Interface.

Parameter	Description
name	The name for the NamedValue.
value	The value for the NamedValue.
flags	The flags for the NamedValue: IN, OUT, or INOUT.

```
abstract public org.omg.CORBA.NVList create_operation_list(
org.omg.CORBA.OperationDef operationDef)
```

This method creates and returns a new NVList for use with a Dynamic Invocation Interface request.

Parameter	Description
operationDef	The operation description that must be specified.

```
abstract public org.omg.CORBA.OutputStream create_output_stream()
```

This method creates an IIOP output stream. An array of bytes constituting an IIOP buffer can be extracted from the stream.

```
abstract public org.omg.CORBA.TypeCode create_recursive_sequence_tc(
int length, int offset)
```

This method creates and returns a TypeCode describing an IDL sequence.

Parameter	Description
length	The length of the sequence to be created. A length of zero indicates an unbounded sequence is desired.
offset	The offset into the type code's (recursive) definition.

```
abstract public org.omg.CORBA.TypeCode create_sequence_tc(int length,
org.omg.CORBA.TypeCode element_type)
```

This method creates and returns a TypeCode describing an IDL sequence.

Parameter	Description
length	The length of the sequence to be created. A length of zero indicates an unbounded sequence is desired.
element_type	The type of the elements contained by the sequence.

```
abstract public org.omg.CORBA.TypeCode create_string_tc(int length)
```

This method creates and returns a TypeCode describing an IDL String.

Parameter	Description
length	The length of the String to be created. A length of zero indicates an unbounded string is desired.

```
abstract public org.omg.CORBA.TypeCode create_wstring_tc(int length)
```

This method creates and returns a TypeCode describing an IDL wString, or Unicode string.

Parameter	Description
length	The length of the String to be created. A length of zero indicates an unbounded string is desired.

```
abstract public org.omg.CORBA.TypeCode create_struct_tc(String
repository_id, String type_name, org.omg.CORBA.StructMember members[])
```

This method creates and returns a TypeCode describing an IDL struct.

Parameter	Description
repository_id	The repository identifier that specifies the type in IDL.
type_name	The un-scoped type name of the type.
members	An array of structures defining the members of the type.

```
abstract public org.omg.CORBA.TypeCode create_union_tc(String repository_id,
                                                       String type_name,
                                                       org.omg.CORBA.TypeCode
discriminator_type,
                                                       org.omg.CORBA.UnionMembers
members[])
```

This method creates and returns a TypeCode describing an IDL union.

Parameter	Description
repository_id	The repository identifier that specifies the type in IDL.
type_name	The un-scoped type name of the type.
discriminator_type	The type of the discriminator. The discriminator is the type used in the switch statement.
members	An array of structures defining the members of the type.

```
abstract public org.omg.CORBA.BindOptions default_bind_options()
```

This method returns the global default bind options, if one has been set; otherwise, NULL is returned. The bind options are stored by reference, so be careful when modifying the value.

```
abstract public void default_bind_options( org.omg.CORBA.BindOptions options)
```

This method sets the global, default bind options. The global default bind options are stored by reference. Use care when modifying the value.

Parameter	Description
options	The new bind options that are being specified.

```
abstract public Principal default_principal()
```

This method returns the global, default Principal, if one has been set; otherwise, NULL is returned.

```
abstract public Principal default_principal(byte principal[])
```

This method sets the global default principal. The bind options are stored by reference, so be careful when modifying the value. The global, default Principal is stored by reference. Use care when modifying the value.

Parameter	Description
principal	The new principal that is being specified.

```
abstract public org.omg.CORBA.Context get_default_context()
```

This method returns the global default Context. Because the default context is a shared resource, any updates to it should be synchronized.

```
abstract public org.omg.CORBA.Request get_next_response()
```

This blocking method waits until a response to a deferred operation request in available. The completed Request is returned. See also: send_multiple_requests_deferred.

```
abstract public org.omg.CORBA.TypeCode get_primitive_tc(TCKind kind)
```

This method returns the primitive type code associated with the kind. An `org.omg.CORBA.BAD_PARAM` exception is raised if *kind* is out of range or is not for a primitive data type

Parameter	Description
kind	The type code kind, as defined in TCKind.

```
abstract public String[] list_initial_services()
```

This method returns a list of names of any object services, such as a Name Service or Event Service, initially available to the process.

```
abstract public String object_to_string(org.omg.CORBA.Object obj)
```

This method converts an object reference to a String, which is returned. The String is valid for the lifetime of the server or, if the implementation is registered with the activation daemon, for the lifetime of the registration and activation daemons. This method returns a stringified Internet Object Reference.

Parameter	Description
obj	The object reference to be converted.

```
abstract public boolean poll_next_response()
```

This method returns true if a response to a deferred operation request is available. Otherwise, false is returned. See also: send_multiple_requests_deferred.

```
abstract public org.omg.CORBA.Object resolve_initial_references(String
identifier) throws org.omg.CORBA._ORB.InvalidName
```

This method resolves one of the names returned by the `list_initial_services` method to its corresponding implementation object. If specified name is not found, an `org.omg.CORBA.InvalidName` exception is raised.

Parameter	Description
identifier	The resolved object, which can be narrowed to the appropriate server type.

```
abstract public void send_multiple_requests_deferred(org.omg.CORBA.Request reqs[])
```

This non-blocking method sends a number of operation requests. Return values may then be obtained using the `poll_next_response` and `get_next_response` methods.

Parameter	Description
reqs	The operation requests.

```
abstract public void send_multiple_requests_oneway(org.omg.CORBA.Request reqs[])
```

This method sends a number of *oneway* operation requests. Return values are not provided for *oneway* requests

Parameter	Description
reqs	The server's implementation object to be activated.

```
abstract public org.omg.CORBA.Object string_to_object(String ior)
```

This method converts a String to an object reference. The Object that is returned can be narrowed to a specific interface. If the `ior` parameter refers to an implementation object in the local address space, the resulting object will be a direct pointer reference to the implementation object. An `org.omg.CORBA.INV_OBJREF` exception will be raised if the `ior` parameter is invalid.

Parameter	Description
ior	An Internet Object Reference that was previously created with the `object_to_string` method.

Principal

```
abstract public class Principal
```

The Principal contains a sequence of bytes that a client application may associate with an operation request. Client applications can set a default principal for a proxy object by using the `orb.default_principal method`. The principal may also be retrieved using the `boa.get_principal` method.

IDL Definition

```
interface Principal {
   attribute ::CORBA::OctetSequence name;
};
```

Methods

```
abstract public void name(byte[] newName)
```

This method sets the name of the Principal.

Parameter	Description
newName	The name to be set.

```
abstract public byte[] name()
```

This method returns the name of the Principal.

Chapter

5

Dynamic Interfaces and Classes

This chapter describes the dynamic interfaces and classes in the `org.omg.CORBA` package. InputStream and OutputStream are in the `org.omg.CORBA.portable` package. All of these interfaces and classes are used in the creation of client requests and object implementations at run time.

- Any
- ContextList
- Environment
- ExceptionList
- InputStream
- NamedValue
- NVList
- OutputStream
- Request
- ServerRequest
- TCKind
- TypeCode
- UnknownUserException

Any

```
abstract public class Any
```

The Any class is used to store a value of any type in a type-safe manner and is used in the Dynamic Invocation Interface. The type stored in an Any is defined by a TypeCode. An Any can store a String, an interface object, or even another Any. Methods are provided to set and retrieve the contained value. By default, an Any contains a NULL until it is initialized.

To create an Any, use `ORB.create_any`.

A Holder version of this class is also provided.

Methods

create_input_stream
create_output_stream
equal
read_value
type
write_value
Extraction Methods
Insertion Methods

```
public abstract InputStream create_input_stream()
```

This static method creates an input stream containing the `Any`'s value.

```
public abstract OutputStream create_output_stream()
```

This static method creates an empty output stream.

```
abstract public boolean equal(Any rhs)
```

This method returns true if the value contained by the Any is the same as the value contained by the specified Any. Otherwise, false is returned.

Parameter	Description
rhs	The Any whose value is compared with the value of this Any.

```
public abstract void read_value(InputStream input, Typecode type)
```

This method reads an Any's value from an input stream, given a type code. Only the Any's value is read. To read the complete Any definition, including the type code, use `org.omg.CORBA.portable.InputStream.read_any`.

Parameter	Description
input	A GIOP input stream from which the specified type's value will be read.
type	The type to read from the input stream. See “TCKind” for the possible values for this parameter.

```
abstract public TypeCode type()
```

This method returns the TypeCode representing the type contained in this Any.

```
abstract public void type(org.omg.CORBA.TypeCode type)
```

This method sets the TypeCode representing the type contained in this Any.

Parameter	Description
type	The type to be set for this Any object. See “TCKind” for the possible values for this parameter.

```
public abstract void write_value(OutputStream output)
```

This method writes an Any's value to an output stream. Only the Any's value is written. To write the complete Any definition, including the type code, use `org.omg.CORBA.portable.OutputStream.write_any`.

Parameter	Description
output	A GIOP output stream into which the specified type's value will be written.

Extraction Methods

A set of methods is provided which return the type contained in this Any. The following code listing shows the name of each of the extraction methods. A BAD_PARAM exception is raised if the value contained in this Any does not match the expected return type for the extraction method used.

Listing 5.1 The extraction methods offered by the Any class.

```
abstract public org.omg.CORBA.Any    extract_any()
abstract public boolean              extract_boolean()
abstract public char                 extract_char()
abstract public double               extract_double()
abstract public float                extract_float()
abstract public int                  extract_long()
abstract public byte                 extract_octet()
abstract public long                 extract_longlong()
abstract public org.omg.CORBA.Object extract_Object()
abstract public byte                 extract_octet()
abstract public org.omg.CORBA.Principal extract_Principal()
abstract public short                extract_short()
abstract public String               extract_string()
abstract public org.omg.CORBA.TypeCode extract_TypeCode()
abstract public int                  extract_ulong()
abstract public long                 extract_ulonglong()
abstract public short                extract_ushort()
abstract public char                 extract_wchar()
abstract public String               extract_wstring()
```

Insertion Methods

A set of methods is provided that copies a particular type of value to this Any. The following code example shows the list of methods provided for inserting various types. With one exception, all of the methods accept a single parameter that represents the type to be inserted.

- The first `insert_Object` method inserts an Object.
- The second `insert_Object` method inserts an Object with a particular TypeCode, effectively narrowing the object to a more specialized type. The second method will raise a BAD_PARAM exception if the TypeCode kind is not `TCKind.tk_objref`.

Listing 5.2 The insertion methods offered by the Any class.

```
abstract public void insert_any(org.omg.CORBA.Any a)
abstract public void insert_boolean(boolean b)
abstract public void insert_char(char c)
abstract public void insert_double(double d)
abstract public void insert_float(float f)
abstract public void insert_long(int i)
abstract public void insert_longlong(long l)
abstract public void insert_Object(org.omg.CORBA.Object o)
abstract public void insert_Object(org.omg.CORBA.Object o, TypeCode t)
abstract public void insert_octet(byte b)
abstract public void insert_Principal(org.omg.CORBA.Principal p)
abstract public void insert_short(short s)
abstract public void insert_Streamable(org.omg.CORBA.portable.Streamable s)
abstract public void insert_string(String s)
abstract public void insert_TypeCode(org.omg.CORBA.TypeCode t)
abstract public void insert_ulong(int i)
abstract public void insert_ulonglong(long l)
abstract public void insert_ushort(short s)
abstract public void insert_wchar(char c)
abstract public void insert_wstring(String s)
```

ContextList

```
public class ContextList extends Object
```

A ContextList maintains a modifiable list of context strings used . To create an instance of ContextList, use `create_context_list()` provided by `org.omg.CORBA.ORB`.

IDL Definition

```
interface ContextList {
  void add(in String ctx);
  readonly attribute unsigned long count;
```

```
  ::CORBA::Context item(in unsigned long index) raises(::CORBA::Bounds);
  void remove(in unsigned long index) raises(::CORBA::Bounds);
};
```

Methods

`public abstract void` **`add`**`(String` **`ctx`**`)`

This method adds a string to the context list.

Parameter	Description
ctx	The name of the context list.

`public abstract int` **`count`**`()`

This method returns the number of elements in the context list.

`public abstract Context` **`item`**`(int` **`index`**`) throws org.omg.CORBA.Bounds`

This method returns an item in the context list. Bounds is raised if index number is not valid.

Parameter	Description
index	The index number of the item.

`public abstract void` **`remove`**`(int` **`index`**`) throws org.omg.CORBA.Bounds`

This method deletes an item from the context list. Bounds is raised if index number is not valid.

Parameter	Description
index	The index number of the item.

DynamicImplementation

`abstract public class` **`DynamicImplementation`** `extends org.omg.CORBA.portable.Skeleton`

The DynamicImplementation is an abstract class that provides a way to deliver requests from an ORB to any object implementation, even object implementations that do not have compile-time knowledge of the type of the objects they are implementing. This differs with the type-specific, IDL-based skeletons; however, they both serve the same function. The DynamicImplementation implements all requests on a particular object by having the ORB invoke an upcall to the implementation via the invoke method.

The ORB upcalls to the DynamicImplementation, passing a ServerRequest object. The ServerRequest pseudo object captures the explicit state of a request for the DynamicImplementation.

Methods

DynamicImplementation
invoke

```
protected DynamicImplementation(String object_name, String repository_id)
```

This constructor assumes that the interface has no other derived interfaces. If the interface has base interfaces, use the other constructor.

Parameter	Description
object_name	The name of the instance. If null, the instance is transient (anonymous).
repository_id	The interface's repository identifier.

```
protected DynamicImplementation(String object_name, String[] repository_ids)
```

Parameter	Description
object_name	The name of the instance. If null, the instance is transient (anonymous).
repository_id	An array of repository identifiers, one for each interface. The most derived interface should be element zero in the array.

```
abstract public void invoke(ServerRequest request)
```

This method provides the functionality of the server.

Parameter	Description
request	A description of the request which the server is to perform.

Environment

```
public abstract class Environment extends Object
```

The Environment interface encapsulates an exception. It is used in conjunction with the dynamic invocation interface to provide a place for exceptions raised by asynchronous DII requests. To create an instance of Environment, use `create_environment()` provided by `org.omg.CORBA.ORB`.

Methods

clear
exception

```
public abstract void clear()
```

This method clears any Exception that may have been raised in the current Environment. This is the same as setting the exception to null.

```
public abstract void exception(java.lang.Exception exception)
```

This method sets the current exception. When setting, any previously stored exception will be lost.

Parameter	Description
exception	The exception to be set for the current Environment.

```
public abstract Exception exception()
```

This method returns the current Exception set for this Environment. If no Exception has been set, NULL is returned.

ExceptionList

```
public class ExceptionList extends Object
```

An ExceptionList is used in the DII (Dynamic Invocation Interface) to describe exceptions that can be raised by IDL operations. It maintains a modifiable list of type codes. To create an instance of ExceptionList, use `create_exception_list()` provided by `org.omg.CORBA.ORB`.

IDL definition

```
interface ExceptionList {
   void add(in CORBA::TypeCode exc);
   readonly attribute unsigned long count;
   CORBA::TypeCode item(in unsigned long index) raises(CORBA::Bounds);
   void remove(in unsigned long index) raises(CORBA::Bounds);
 };
```

Methods

```
public abstract void add(TypeCode exc)
```

This method adds a type code to the exception list.

Parameter	Description
exc	The exception to add to the list.

```
public abstract int count()
```

This method returns the number of items in the exception list.

```
public abstract TypeCode item(int index) throws org.omg.CORBA.Bounds
```

This method returns an item from the list. Bounds is raised if the index number is not valid.

Parameter	Description
index	The index number of the item to be returned.

```
public abstract void remove(int index) throws org.omg.CORBA.Bounds
```

This method removes an item from the exception list. Bounds is raised if the index number specified is not valid.

Parameter	Description
index	The index number of the item to be removed from the list.

InputStream

```
public interface InputStream
```

The interface represents an Internet Inter-ORB Protocol input stream. Objects of this type are created with the `ORB.create_input_stream` method. All bytes written to an output stream can be read using the input stream methods. Several methods are provided for reading various data types. See also ORB.create_input_stream and ORB.create_output_stream.

Methods

The methods shown below are provided for reading data from an InputStream object. Each method returns a particular data type. See also read_estruct and read_byte_array.

Listing 5.3 The read methods offered by the InputStream class.

```
public abstract boolean read_boolean();
public abstract char read_char();
public abstract char read_wchar();
public abstract byte read_octet();
public abstract short read_short();
public abstract short read_ushort();
public abstract int read_long();
public abstract int read_ulong();
```

```
public abstract long read_longlong();
public abstract long read_ulonglong();
public abstract float read_float();
public abstract double read_double();
public abstract String read_string();
public abstract String read_wstring();
public abstract void read_boolean_array(boolean[] value,
                                        int offset, int length);
public abstract void read_char_array(char[] value,
                                        int offset, int length);
public abstract void read_wchar_array(char[] value,
                                        int offset, int length);
public abstract void read_octet_array(byte[] value,
                                           int offset, int length);
public abstract void read_short_array(short[] value,
                                        int offset, int length);
public abstract void read_ushort_array(short[] value,
                                        int offset, int length);
public abstract void read_long_array(int[] value,
                                        int offset, int length);
public abstract void read_ulong_array(int[] value,
                                        int offset, int length);
public abstract void read_longlong_array(long[] value,
                                        int offset, int length);
public abstract void read_ulonglong_array(long[] value,
                                        int offset, int length);
public abstract void read_float_array(float[] value,
                                        int offset, int length);
public abstract void read_double_array(double[] value,
                                        int offset, int length);
public abstract org.omg.CORBA.Object read_Object();
public abstract org.omg.CORBA.TypeCode read_TypeCode();
public abstract org.omg.CORBA.Any read_any();
public abstract org.omg.CORBA.Principal read_Principal();
```

```
public Object     read_estruct(String expected_type)
```

This method reads an IDL extensible struct

Parameter	Description
expected_type	The expected type of the object being read.

```
public byte[]     read_byte_array(int length)
```

This method reads an array of bytes from the stream.

Parameter	Description
length	The number of bytes to be read.

NamedValue

```
public class NamedValue extends Object
```

The NamedValue interface is used by a client, alone or in conjunction with the NVList, to specify parameters and return values for a Dynamic Invocation Interface request. In Java, it includes a name, a value (such as an Any), and an integer representing a set of flags. NamedValue items can only be created using the NVList interface. To create an instance of NamedValue, use `create_named_value(String name, Any value, int flags)` provided by `org.omg.CORBA.ORB`.

IDL Definition

```
interface NamedValue {
      readonly attribute ::CORBA::Flags flags;
      readonly attribute ::CORBA::Identifier name;
      readonly attribute any value;
    };
```

Methods

```
public abstract int flags()
```

This method returns the flags for this NamedValue. See ARG_IN, ARG_INOUT, and ARG_OUT for more information.

```
public abstract String name()
```

This method returns the name set for this NamedValue. If no name has been set, NULL is returned.

```
public abstract org.omg.CORBA.Any value()
```

This method returns an Any representing the current value set for this NamedValue. If no value has been set, NULL is returned. The value may be modified in place.

NVList

```
public interface NVList
```

The NVList interface contains a set of NamedValue objects. It is used by client applications to pass the parameters associated with a Dynamic Invocation Interface request and in the context routines to describe context values. In Java, it maintains a modifiable list of NamedValues. You can create an NVList using the ORB.create_list method.

IDL Definition

```
interface NVList {
  void add(in CORBA::Flags flags);
  void add_item(in CORBA::Identifier name,
                in CORBA::Flags flags);
  void add_value(in CORBA::Identifier name,
                 in any value,
                 in CORBA::Flags flags);
  unsigned long count();
  CORBA::NamedValue item(in unsigned long index);
  void remove(in unsigned long index);
};
```

Methods

```
public abstract org.omg.CORBA.NamedValue add(int flags)
```

This method adds a NamedValue item to this list without initializing the name or the value associated with the item.

Parameter	Description
flags	The mode of the parameter to be added. Allowed values are in, inout, or out.

```
public abstract org.omg.CORBA.NamedValue add_item(String item_name, int flags)
```

This method adds a NamedValue item to this list without initializing the value associated with the item.

Parameter	Description
item_name	The name of the item to be added.
flags	The mode of the parameter to be added. Allowed values are in, inout, or out.

```
public abstract org.omg.CORBA.NamedValue add_value(String item_name,
                                    org.omg.CORBA.Any value,
                                    int flags)
```

This method adds a NamedValue item to this NVList that has the specified name, value and flags.

Parameter	Description
item_name	The name of the NamedValue to be added.
value	The value of the NamedValue, represented as an Any.
flags	The mode of the parameter to be added.

```
public abstract int count()
```

This method returns the number of NamedValue items in this NVList.

```
public abstract org.omg.CORBA.NamedValue item(int index) throws org.omg.CORBA.Bounds
```

This method returns the NamedValue item from this list that has the specified index. If the index is out of range, Bounds is raised.

Parameter	Description
index	The index of the NamedValue to be returned from this list.

```
public void remove(int index)
```

This method removes the NamedValue with the specified index from this list. If the index is out of range, Bounds is raised.

Parameter	Description
index	The index of the NamedValue item to be removed from this list.

OutputStream

```
public interface OutputStream
```

The interface represents an Internet Inter-ORB Protocol output stream. Objects of this type are created by using the `ORB.create_output_stream` method. All bytes written to an output stream will be available to be read using the input stream. Several methods are provided for writing various data types.

See also ORB.create_input_stream and ORB.create_output_stream.

Methods

The methods shown below are provided to write a particular type to this OutputStream. See also write_estruct and write_byte_array.

Listing 5.4 The write methods offered by the OutputStream class.

```
public abstract InputStream create_input_stream();
public abstract void write_boolean  (boolean value);
public abstract void write_char     (char value);
public abstract void write_wchar    (char value);
public abstract void write_octet    (byte value);
public abstract void write_short    (short value);
public abstract void write_ushort   (short value);
```

```
public abstract void write_long     (int value);
public abstract void write_ulong    (int value);
public abstract void write_longlong (long value);
public abstract void write_ulonglong (long value);
public abstract void write_float     (float value);
public abstract void write_double    (double value);
public abstract void write_string    (String value);
public abstract void write_wstring   (String value);
public abstract void write_boolean_array (boolean[] value, int offset, int length);
public abstract void write_char_array    (char[] value, int offset, int length);
public abstract void write_wchar_array   (char[] value, int offset, int length);
public abstract void write_octet_array   (byte[] value, int offset, int length);
public abstract void write_short_array   (short[] value, int offset, int length);
public abstract void write_ushort_array  (short[] value, int offset, int length);
public abstract void write_long_array    (int[] value, int offset, int length);
public abstract void write_ulong_array   (int[] value, int offset, int length);
public abstract void write_longlong_array (long[] value, int offset, int length);
public abstract void write_ulonglong_array (long[] value, int offset, int length);
public abstract void write_float_array   (float[] value, int offset, int length);
public abstract void write_double_array  (double[] value, int offset, int length);
public abstract void write_Object        (org.omg.CORBA.Object value);
public abstract void write_TypeCode      (org.omg.CORBA.TypeCode value);
public abstract void write_any           (org.omg.CORBA.Any value);
public abstract void write_Principal     (org.omg.CORBA.Principal value);
```

The write methods offered by the OutputStream class.

```
public org.omg.CORBA.Object   write_estruct(org.omg.CORBA.Object value, String
expected_type)
```

This method writes an Object to the stream as an IDL extensible struct. Extensible structures are used to represent objects that are passed as operation request parameters.

Parameter	Description
value	The extensible struct to be written to the stream.
expected_type	The type of object to be written.

```
public byte[]     write_byte_array(byte value[], int offset, int length)
```

This method writes an array of bytes to the stream.

Parameter	Description
value	The array of bytes to be written.
offset	The position within the stream where the bytes are to be written. This value is usually set to zero.
length	The number of bytes to be written.

Request

```
public interface Request
```

The Request interface represents a dynamic invocation request and provides methods for initializing and sending the request as well as receiving the response. An operation request may be sent synchronously, asynchronously, or as a *oneway* request for which no response is expected. Replies to invocations can be polled or obtained synchronously. The ORB interface can also be used to perform multiple simultaneous invocations, allowing for higher parallelism and reduced latency.

This object includes the following state information:

- The target object.
- The operation name.
- The argument types and values.
- The return type and value.
- The Environment, containing exceptions.
- The Context.

See the Object methods _create_request and _request for information on creating a Request.

IDL Definition

```
interface Request {
      readonly attribute CORBA::NVList arguments;
      readonly attribute CORBA::ContextList contexts;
      attribute CORBA::Context ctx;
```

```
        readonly attribute CORBA::Environment env;
        readonly attribute CORBA::ExceptionList exceptions;
        readonly attribute CORBA::Identifier operation;
        readonly attribute CORBA::NamedValue result;
        readonly attribute CORBA::Object target;

        any add_in_arg();
        any add_inout_arg();
        any add_out_arg();
        any add_named_in_arg(in string name);
        any add_named_inout_arg(in string name);
        any add_named_out_arg(in string name);
        void get_response();
        void invoke();
        boolean poll_response();
        any return_value();
        void send_oneway();
        void send_deferred();
        void set_return_type(in ::CORBA::TypeCode tc);
};
```

Methods

`public abstract Any` **`add_in_arg()`**

Adds an IN argument to the request.

`public abstract Any` **`add_inout_arg()`**

Adds an INOUT argument to the request.

`public abstract Any` **`add_named_in_arg`**`(String` **`name`**`)`

Adds a named IN argument to the request.

Parameter	Description
name	The name of the argument associated with this request.

`public abstract Any` **`add_named_inout_arg`**`(String` **`name`**`)`

Adds a named INOUT argument to the request.

Parameter	Description
name	The name of the argument associated with this request.

```
public abstract Any add_named_out_arg(String name)
```

Adds a named OUT argument to the request.

Parameter	Description
name	The name of the argument associated with this request.

```
public abstract Any add_out_arg()
```

Adds a OUT argument to the request.

```
public abstract org.omg.CORBA.NVList arguments()
```

This method returns the list of arguments for this request. These arguments must be initialized prior to sending the request.

```
public abstract String[] contexts()
```

This method returns the context list. The list will be empty if the operation does not specify any contexts.

```
public abstract org.omg.CORBA.Context ctx()
```

This method returns the context list associated with this request. See "ContextList" for more information.

```
public abstract void ctx(org.omg.CORBA.Context ctx)
```

This method sets the context for this request. See also: ORB.get_default_context.

Parameter	Description
ctx	The Context.

```
public abstract org.omg.CORBA.Environment env()
```

This method returns the Environment in which the request is invoked. Exceptions raised by the server will be put into the request's Environment. For more information, see “Environment.”

```
public abstract org.omg.CORBA.ExceptionList exceptions()
```

This method returns the list of user exception type codes. The list will be empty if no user exceptions can be raised by the operation.

```
public abstract void get_response()
```

This blocking method waits for the results of a dynamic invocation request that was sent with the send_deferred method. All inout, out, and return values are updated by this method.

The non-blocking poll_response method can be used to determine if a response is available prior to invoking this method.

```
public abstract void invoke()
```

This method sends the request and blocks waiting for a response. If the client does not wish to block waiting for a response, the `send_deferred` method may be used instead.

```
public abstract String operation()
```

This method returns the name of the operation, or method name, associated with this request.

```
public abstract boolean poll_response()
```

This method returns true if a response for an request is currently available. Otherwise, false is returned. This method is used after the send_deferred method has been invoked and prior to invoking the get_response method, which actually reads in the result values. See also: poll_next_response.

```
public abstract org.omg.CORBA.NamedValue result()
```

This method returns the results, or return value, of the request. If the type of the result is not specified, the type defaults to `void`. If the return type is not `void`, the type must be initialized prior to sending the request.

```
public abstract Any return_value()
```

This method gets the return value of the operation as an Any.

```
public abstract void send_deferred()
```

This method sends this request, but does not block awaiting a response. The poll_response and get_response methods are then used to determine when a response is available and to receive the results. See also: send_multiple_requests_deferred.

```
public abstract void send_oneway()
```

This non-blocking method sends this request as a *oneway* request. Oneway requests do not result in a response from the server to which they are sent. See also: send_multiple_requests_oneway.

```
public abstract void set_return_type(TypeCode tc)
```

This method sets the type expected to be returned prior to invoking the operation.

Parameter	Description
tc	The type code to set.

```
public abstract org.omg.CORBA.Object target()
```

This method returns the target Object to which this request will be sent. The target Object is specified when the Request is created, using the Object method _create_request.

ServerRequest

```
abstract public interface ServerRequest
```

The ServerRequest interface, an important element when using dynamic skeletons, represents a request received by a server from a client. It provides methods for obtaining the Context, operation name, and operation parameters as well as a method for reflecting any Exception that may be raised during the processing of the request. This interface works with DynamicImplementation to provide dynamic skeletons. For more information about using dynamic skeletons, see “The Dynamic Skeleton Interface” chapter in the *Netscape Internet Service Broker for Java Programmer's Guide*.

IDL definition

```
interface ServerRequest {
  readonly attribute ::CORBA::Context ctx;
  readonly attribute ::CORBA::Identifier op_name;

  void except(in any except);
  void params(in ::CORBA::NVList params);
  void result(in any result);
};
```

Methods

`abstract public org.omg.CORBA.Context` **`ctx`**`()`

This method returns the current Context associated with this ServerRequest.

`abstract public void` **`except`**`(org.omg.CORBA.Any` **`except`**`)`

This method is used by the server to set an Exception that occurred during the processing of the request so that it may be reflected to the client.

Parameter	Description
exception	The Exception that was raised.

`abstract public String` **`op_name`**`()`

This method returns the operation name associated with this request.

`abstract public void` **`params`**`(org.omg.CORBA.NVList` **`params`**`)`

This method sets the parameters for this operation request. For more information about using this method, see “The Dynamic Skeleton Interface” chapter in the *Netscape Internet Service Broker for Java Programmer's Guide*.

Parameter	Description
params	The NVList where the parameters are to be stored.

`abstract public void` **`result`**`(org.omg.CORBA.Any` **`result`**`)`

This method sets the result for this operation request.

Parameter	Description
result	The result to be set for the operation request.

TCKind

```
public class TCKind extends Object
```

The TCKind class contains the constants used in conjunction with TypeCode objects, which define an object's type. There are a set of integer constants, prefixed with `tk_`, that correspond to all the possible type codes. For example, the type code for float is `TCKind.tk_float`.

There is also a set of TypeCode globals, prefixed with `tc_`, that correspond to the built-in type codes. The TypeCode constant for `float`, for example, is `TCKind.tc_float`. A complete list of the enums for TCKind can be found in the IDL definition. For each enum, an equivalent TypeCode and TCKind constant exists. See also the methods `value` and `from_int`.

Helper and Holder versions of this class are also provided.

See also ORB.create_list.

IDL Definition

```
enum TCKind {
    tk_null, tk_void,
    tk_short, tk_long, tk_ushort, tk_ulong,
    tk_float, tk_double, tk_boolean, tk_char,
    tk_octet, tk_any, tk_TypeCode, tk_Principal, tk_objref,
    tk_struct, tk_union, tk_enum, tk_string,
    tk_sequence, tk_array, tk_alias, tk_except,
    tk_longlong, tk_ulonglong, tk_longdouble,
    tk_wchar, tk_wstring, tk_fixed
};
```

Methods

from_int
value

```
public static TCKind from_int(int value)
```

This method returns an enum instance for the value you specify. For more information about enum mapping, see “Enum.”

Parameter	Description
value	The enum value.

```
public int value()
```

This method returns an integral value representing the constant.

TypeCode

```
public interface TypeCode
```

The TypeCode interface describes the various types that are defined in IDL and allows them to be created and examined at run time. Type codes are most often used to describe the type of value being stored in an Any object. Type codes may also be passed as parameters to method invocations.

Type codes are created using the various ORB.create_<type>_tc methods. Type codes for all built-in types are provided by the TCKind class.

A Holder version of this class is also provided.

IDL Definition

```
interface TypeCode {
      exception Bounds {};
      exception BadKind {};

      CORBA::TypeCode content_type() raises(CORBA::TypeCode::BadKind);
```

```
        long default_index() raises(CORBA::TypeCode::BadKind);
        CORBA::TypeCode discriminator_type() raises(CORBA::TypeCode::BadKind);
        boolean equal(in CORBA::TypeCode tc);
        CORBA::RepositoryId id() raises(CORBA::TypeCode::BadKind);
        CORBA::TCKind kind();
        unsigned long length() raises(CORBA::TypeCode::BadKind);
        unsigned long member_count()  raises(CORBA::TypeCode::BadKind);
        any member_label(in unsigned long index)
                         raises(CORBA::TypeCode::BadKind, CORBA::TypeCode::Bounds);
        CORBA::Identifier member_name(in unsigned long index)
                         raises(CORBA::TypeCode::BadKind, CORBA::TypeCode::Bounds);
        CORBA::TypeCode member_type(in unsigned long index)
                         raises(CORBA::TypeCode::BadKind, CORBA::TypeCode::Bounds);
        CORBA::Identifier name() raises(CORBA::TypeCode::BadKind);
        long param_count();
        any parameter(in long index) raises(CORBA::TypeCode::Bounds);
};
```

Methods

```
public org.omg.CORBA.TypeCode content_type()
                throws org.omg.CORBA._TypeCode.BadKind
```

This method returns the type code of the element contained in container types or an aliased type. This method is only valid for the following type codes.

- tk_sequence
- tk_array
- tk_alias

A BAD_PARAM exception will be raised if the type code is not one of these types.

```
public int default_index() throws org.omg.CORBA._TypeCode.BadKind
```

This method returns the default index of a union. This method is only valid for type codes tk_union, otherwise a BAD_PARAM exception will be raised.

```
public TypeCode discriminator_type() throws org.omg.CORBA._TypeCode.BadKind
```

This method returns the type code of the discriminator of a union. This method is only valid when invoked on object with the type code of tk_union, otherwise a BAD_PARAM exception will be raised.

```
public boolean equal(org.omg.CORBA.TypeCode tc)
```

This method returns true if this object is equivalent to tc. Otherwise, false is returned. Type equivalence is determined by the structure of the types, not by their names. Two structures with the same fields, declared in the same order, are considered type equivalent.

Parameter	Description
tc	The TypeCode to be compared to this object's type.

```
public String id() throws org.omg.CORBA._TypeCode.BadKind
```

This method returns the repository identifier of the type code. This string is used by IDL to define the type.

```
public TCKind kind()
```

This method returns the kind of type associated with this type code. Type codes kind constants are defined by TCKind.

```
public int length() throws org.omg.CORBA._TypeCode.BadKind
```

This method returns the number of elements contained by the type. Zero is returned if the number of elements is unbounded, such as for strings and sequences. This method is only valid for the following type codes.

- tk_string
- tk_sequence
- tk_array

A BAD_PARAM exception will be raised if the type code is not one of these types.

```
public int member_count() throws org.omg.CORBA._TypeCode.BadKind
```

This method returns the number of members contained by the type. This method is only valid for the following type codes.

- tk_struct
- tk_union
- tk_enum
- tk_except

A BAD_PARAM exception will be raised if the type code is not one of these types.

```
public Any member_label(int index)
                    throws org.omg.CORBA._TypeCode.BadKind,
                           org.omg.CORBA._TypeCode.Bounds
```

This method returns the label of the case statement associated with the member that has the specified index. This method is only valid for the type code tk_union, otherwise a BAD_PARAM exception will be raised. If the index is out of the bounds of the String, a Bounds exception will be raised.

Parameter	Description
index	The index of the member whose label is to be returned.

```
public String member_name(int index)
                    throws org.omg.CORBA._TypeCode.BadKind,
                           org.omg.CORBA._TypeCode.Bounds
```

This method returns the name of the member that has the specified index. This method is only valid for the following type codes:

- tk_struct
- tk_union
- tk_enum
- tk_except

A BAD_PARAM exception will be raised if the type code is not one of these types. If the index is out of the bounds of the String, a Bounds exception will be raised.

Parameter	Description
index	The index of the member whose name is to be returned.

```
public org.omg.CORBA.TypeCode member_type(int index)
                     throws org.omg.CORBA._TypeCode.BadKind,
                            org.omg.CORBA._TypeCode.Bounds
```

This method returns the type code of the member that has the specified index. This method is only valid for the following type codes:

- tk_struct

- tk_union
- tk_except

A BAD_PARAM exception will be raised if the type code is not one of these types. If the index is out of the bounds of the String, a Bounds exception will be raised.

Parameter	Description
index	The index of the member whose type code is to be returned.

```
public String name() throws org.omg.CORBA._TypeCode.BadKind
```

This method returns the unscoped type name. This method is only valid for the following type codes.

- tk_objref
- tk_struct
- tk_union
- tk_enum
- tk_alais
- tk_except

A BAD_PARAM exception will be raised if the type code is not one of these types.

```
public long param_count()
```

This method returns the number of parameters for the method.

```
public any parameter(in long index) raises(CORBA::TypeCode::Bounds)
```

This method returns the parameter at the specified index. An exception is raised if the index is out of bounds.

UnknownUserException

```
public class UnknownUserException extends org.omg.CORBA.UserException
```

When a client issues a DII request and a user exception is raised, the specific exception cannot be reflected to the client. This exception is used instead.

Chapter

6

Interface Repository

This chapter describes the interfaces and classes in the `org.omg.CORBA` package that are used with the interface repository.

- AliasDef
- ArrayDef
- AttributeDef
- AttributeDescription
- AttributeMode
- ConstantDef
- ConstantDescription
- Contained
- ContainedPackage.Description
- Container
- ContainerPackage.Description
- DefinitionKind
- EnumDef
- EstructDef
- ExceptionDef
- ExceptionDescription
- IDLType
- InterfaceDef
- InterfaceDefPackage.FullInterfaceDescription

- InterfaceDescription
- IRObject
- ModuleDef
- ModuleDescription
- OperationDef
- OperationDescription
- OperationMode
- ParameterDescription
- ParameterMode
- PrimitiveDef
- PrimitiveKind
- Repository
- SequenceDef
- StringDef
- StructDef
- TypedefDef
- TypeDescription
- UnionDef
- UnionMember
- VersionSpec
- WstringDef

AliasDef

```
public interface AliasDef extends org.omg.CORBA.TypedefDef
```

The interface is used to represent an alias for a typedef that is stored in the Interface Repository. This interface provides methods for setting and obtaining the IDLType of the original typedef.

For more information, see the TypedefDef interface.

Helper and Holder versions of this class are also provided.

Methods

- original_type_def

```
public void original_type_def(org.omg.CORBA.IDLType original_type_def)
```

This method sets the IDLType of this object.

Parameter	Description
original_type_def	The IDLType of this object.

```
public org.omg.CORBA.IDLType original_type_def()
```

This method returns the IDLType of the original typedef for which this object is an alias.

ArrayDef

```
public interface ArrayDef extends org.omg.CORBA.IDLType
```

The interface is used to represent an array that is stored in the Interface Repository. This interface provides methods for setting and obtaining the type of elements in the array as well as the length of the array.

Helper and Holder versions of this class are also provided.

Methods

- element_type
- element_type_def
- length

```
public org.omg.CORBA.TypeCode element_type()
```

This method returns the TypeCode of the array's elements.

```
public void element_type_def(org.omg.CORBA.IDLType element_type_def)
```

This method sets the IDLType of the elements stored in the array.

Parameter	Description
original_type_def	The IDLType of the elements in the array.

```
public org.omg.CORBA.IDLType element_type_def()
```

This method returns the IDLType of the elements stored in this array.

```
public void length(int length)
```

This method sets the number of elements in the array.

Parameter	Description
length	The number of elements in the array.

AttributeDef

```
public interface AttributeDef extends org.omg.CORBA.Contained
```

The interface is used to represent an interface attribute that is stored in the Interface Repository. This interface provides methods for setting and obtaining the attribute's mode, type and typedef.

Helper and Holder versions of this class are also provided.

Methods

- type
- type_def

```
public org.omg.CORBA.TypeCode type()
```

This method returns the TypeCode representing the attribute's type.

```
public void type_def(org.omg.CORBA.IDLType type_def)
```

This method sets the IDLType for which this object.

Parameter	Description
type_def	The IDLType of this object.

```
public org.omg.CORBA.IDLType type_def()
```

This method returns this object's IDLType.

AttributeDescription

```
public class AttributeDescription
```

The AttributeDescription class describes an attribute that is stored in the interface repository.

Helper and Holder versions of this class are also provided.

Variables

name	The name of the attribute.
id	The repository id of the attribute.
defined_in	The name of the module or interface in which this attribute is defined.
version	The attribute's version.
type	The attribute's IDL type.
mode	The mode of the attribute.

Methods

- AttributeDescription

```
public AttributeDescription()
```

This method is the default constructor for an AttributeDescription.

```
public AttributeDescription(String name, String id, String defined_in,
                    org.omg.CORBA.VersionSpec version,
                    org.omg.CORBA.TypeCode type,
                    org.omg.CORBA.AttributeMode mode)
```

This method constructs an AttributeDescription, using the supplied parameters.

Parameter	Description
name	The name of this attribute.
id	The repository id for this attribute.
defined_in	The module or interface in which this attribute is defined.
version	The object's version.
type	The attribute's IDL type code.
mode	The mode of this attribute; read-only or read-write. See AttributeMode.

AttributeMode

```
public class AttributeMode
```

The class is used to represent the *mode* of an attribute; either read-only or normal (read-write).

Helper and Holder versions of this class are also provided.

Variables

Constant	Represents
_ATTR_NORMAL	A read-write attribute.
_ATTR_READONLY	A read-only attribute.

Constant	Represents
ATTR_NORMAL	An attribute definition as Normal.
ATTR_READONLY	A attribue definition as read-only.

Methods

- _value

```
public int _value()
```

This method returns the value of this object, which is either _ATTR_NORMAL or ATTR_READONLY.

ConstantDef

```
public interface ConstantDef extends org.omg.CORBA.Contained
```

The interface is used to represent a constant definition that is stored in the interface repository. This interface provides methods for setting and obtaining the constant's type, value and typedef.

Helper and Holder versions of this class are also provided.

Methods

- type
- type_def
- value

```
public org.omg.CORBA.TypeCode type()
```

This method returns the TypeCode representing the object's type.

```
public org.omg.CORBA.IDLType type_def()
```

This method returns this object's IDLType.

```
public void type_def(org.omg.CORBA.IDLType type_def)
```

This method sets the IDLType of the original typedef for this object.

Parameter	Description
type_def	The IDLType of this object.

```
public org.omg.CORBA.Any value()
```

This method returns an Any object representing this object's value.

```
public void value(org.omg.CORBA.Any value)
```

This method sets the value for this constant.

Parameter	Description
value	An Any object that represents this object's value.

ConstantDescription

```
public class ConstantDescription
```

The ConstantDescription class describes a constant that is stored in the interface repository.

Variables

`public String name`	The name of the constant.
`public String id`	The repository id of the constant.
`public String defined_in`	The name of the module or interface in which this constant is defined.
`public org.omg.CORBA.VersionSpec version`	The constant's version.
`public org.omg.CORBA.TypeCode type`	The constant's IDL type.
`public org.omg.CORBA.Any value`	The value of this constant.
`public String name`	The name of the constant.

Methods

- ConstantDescription

```
public ConstantDescription()
```

This method is the default constructor for a ConstantDescription.

```
public ConstantDescription(String name, String id, String defined_in,
                   org.omg.CORBA.VersionSpec version,
                   org.omg.CORBA.TypeCode type,
                   org.omg.CORBA.Any value)
```

This method constructs an ConstantDescription, using the supplied parameters.

Parameter	Description
name	The name of this constant.
id	The repository id for this constant.
defined_in	The module or interface in which this constant is defined.
version	The object's version.
type	The constant's IDL type code.
value	The value of this constant.

Contained

```
public interface Contained extends org.omg.CORBA.IRObject
```

The interface is used to represent Interface Repository objects that are, themselves, contained within another Interface Repository object. This interface provides methods for:

- Setting and retrieving the object's name and version.
- Determining the Container that contains this object.
- Obtaining the object's absolute name, containing repository, and description.
- Moving an object from one container to another.

Helper and Holder versions of this class are also provided.

IDL Definition

```
interface Contained : CORBA::IRObject {
      readonly attribute CORBA::ScopedName absolute_name;
      readonly attribute CORBA::Repository containing_repository;
      readonly attribute CORBA::Container defined_in;
      CORBA::Contained::Description describe();
      struct Description {
         CORBA::DefinitionKind kind;
         any value;
      };
      attribute CORBA::RepositoryId id;
      void move(
         in CORBA::Container new_container,
         in CORBA::Identifier new_name,
         in CORBA::VersionSpec new_version
      );
      attribute CORBA::Identifier name;
      attribute CORBA::VersionSpec version;
};
```

Methods

`public String` **`absolute_name`**`()`

This method returns this object's absolute name.

`public org.omg.CORBA.Repository` **`containing_repository`**`()`

This method returns the repository that contains this object.

`public org.omg.CORBA.Container` **`defined_in`**`()`

This method returns the Container where this object is defined.

`public org.omg.CORBA.ContainedPackage.Description` **`describe`**`()`

This method returns this object's description.

`public String` **`id`**`()`

This method returns this object's repository identifier.

`public void` **`id`**`(String` **`id`**`)`

This method sets the repository identification that uniquely identifies this object.

Parameter	Description
id	The repository identifier for this object.

```
public string name()
```

This method returns this object's name.

```
public void name(String name)
```

This method sets the name for this object.

Parameter	Description
name	The object's name.

```
public org.omg.CORBA.VersionSpec version()
```

This method returns this object's version.

```
public void version(org.omg.CORBA.VersionSpec version)
```

This method sets the version for this object.

Parameter	Description
version	The object's version.

```
public void move(org.omg.CORBA.Container new_container,
                 String new_name,
                 org.omg.CORBA.VersionSpec new_version)
```

This method moves this object to another container.

Parameter	Description
new_container	The Container to which the object is to be moved.
new_name	The new name for the object.
new_version	The new version specification for the object.

ContainedPackage.Description

```
public class Description
```

This class provides a generic description for items in the interface repository that are derived from the Contained interface.

Helper and Holder versions of this class are also provided.

Variables

`public org.omg.CORBA.DefinitionKind kind`	The kind of the item.
`public org.omg.CORBA.Any value`	The value of the item.

Methods

- Description

```
public Description()
```

This method is the default constructor for a Description.

```
public Description(org.omg.CORBA.DefinitionKind kind,
                   org.omg.CORBA.Any value)
```

This method constructs an Description, using the supplied parameters.

Parameter	Description
kind	This item's kind. See DefinitionKind for more information.
value	An Any object that represents the value for this item.

Container

```
public interface Container extends org.omg.CORBA.IRObject
```

The Container interface is used to create a containment hierarchy in the Interface Repository. A Container object holds object definitions derived from the Contained class. All object definitions derived from the Container class, with the exception of the Repository class, also inherit from the Contained class.

Helper and Holder versions of this class are also provided.

IDL Definition

```
interface Container : CORBA::IRObject {

        CORBA::AliasDef create_alias(
        in CORBA::RepositoryId id,
        in CORBA::Identifier name,
        in CORBA::VersionSpec version,
        in CORBA::IDLType original_type
      );
      CORBA::ConstantDef create_constant(
        in CORBA::RepositoryId id,
        in CORBA::Identifier name,
        in CORBA::VersionSpec version,
        in CORBA::IDLType type,
        in any value
      );
   CORBA::EnumDef    create_enum(
        in CORBA::RepositoryId id,
        in CORBA::Identifier name,
        in CORBA::VersionSpec version,
        in CORBA::EnumMemberSeq members
      );
      CORBA::EstructDef create_estruct(
        in CORBA::RepositoryId id,
        in CORBA::Identifier name,
        in CORBA::VersionSpec version,
        in CORBA::EstructDef base,
        in CORBA::StructMemberSeq members
      );
      CORBA::ExceptionDef create_exception(
        in CORBA::RepositoryId id,
        in CORBA::Identifier name,
```

```
  in CORBA::VersionSpec version,
  in CORBA::StructMemberSeq members
);
CORBA::InterfaceDef create_interface(
  in CORBA::RepositoryId id,
  in CORBA::Identifier name,
  in CORBA::VersionSpec version,
  in CORBA::InterfaceDefSeq base_interfaces
);
CORBA::ModuleDef create_module(
  in CORBA::RepositoryId id,
  in CORBA::Identifier name,
  in CORBA::VersionSpec version
);
CORBA::StructDef create_struct(
  in CORBA::RepositoryId id,
  in CORBA::Identifier name,
  in CORBA::VersionSpec version,
  in CORBA::StructMemberSeq members
);
CORBA::UnionDef create_union(
  in CORBA::RepositoryId id,
  in CORBA::Identifier name,
  in CORBA::VersionSpec version,
  in CORBA::IDLType discriminator_type,
  in CORBA::UnionMemberSeq members
);

  CORBA::ContainedSeq contents(
  in CORBA::DefinitionKind limit_type,
  in boolean exclude_inherited
);
  CORBA::Contained lookup(
  in CORBA::ScopedName search_name
);
  CORBA::ContainedSeq lookup_name(
  in CORBA::Identifier search_name,
  in long levels_to_search,
  in CORBA::DefinitionKind limit_type,
  in boolean exclude_inherited
);
struct Description {
```

```
        CORBA::Contained contained_object;
        CORBA::DefinitionKind kind;
        any value;
    };
    typedef sequence <CORBA::Container::Description> DescriptionSeq;
    CORBA::Container::DescriptionSeq describe_contents(
      in CORBA::DefinitionKind limit_type,
      in boolean exclude_inherited,
      in long max_returned_objs
    );
    CORBA::ContainedSeq private_lookup(
      in CORBA::Identifier search_name,
      in long levels_to_search,
      in CORBA::DefinitionKind limit_type,
      in boolean exclude_inherited,
      in long max_returned_objs
    );
    void private_add_name(
      in CORBA::Contained cont
    );
    void private_remove_name(
      in CORBA::Contained cont
    );
    CORBA::InterfaceDef forward_declare_interface(
      in CORBA::RepositoryId id,
      in CORBA::Identifier name,
      in CORBA::VersionSpec version
    );
};
```

Methods

```
public org.omg.CORBA.Contained[] contents(
                        org.omg.CORBA.DefinitionKind limit_type,
                        boolean exclude_inherited)
```

This method sets the repository identification that uniquely identifies this object.

Parameter	Description
limit_type	The interface object types to be returned.
exclude_inherited	If set to true, inherited objects will not be returned.

```
public AliasDef create_alias    (String id,
                        String name,
                        org.omg.CORBA.VersionSpec version,
                        org.omg.CORBA.IDLType original_type)
```

This method creates a AliasDef object in this Container with the specified attributes and returns a reference to the newly created object.

Parameter	Description
id	The alias' repository id.
name	The alias' name.
version	The alias' version.
original_type	The IDL type of the original object for which this is an alias.

```
public org.omg.CORBA.ConstantDef create_constant(String id,
                        String name,
                        org.omg.CORBA.VersionSpec version,
                        org.omg.CORBA.IDLType type,
                        org.omg.CORBA.Any value)
```

This method creates a ConstantDef object in this Container with the specified attributes and returns a reference to the newly created object.

Parameter	Description
id	The constant's repository id.
name	The constant's name.
version	The constant's version.

Parameter	Description
type	The constant's IDL type.
value	The constant's value, represented by an Any object.

```
public org.omg.CORBA.EnumDef create_enum(String id,
                        String name,
                        org.omg.CORBA.VersionSpec version,
                        String members[])
```

This method creates a EnumDef object in this Container with the specified attributes and returns a reference to the newly created object.

Parameter	Description
id	The enumeration's repository id.
name	The enumeration's name.
version	The enumeration's version.
members	A list of the enumeration's values.

```
public org.omg.CORBA.EstructDef create_estruct(String id,
                        String name,
                        org.omg.CORBA.VersionSpec version,
                        org.omg.CORBA.EstructDef base,
                        org.omg.CORBA.StructMember members[])
```

This method creates an EstructDef object in this Container with the specified attributes and returns a reference to the newly created object.

Parameter	Description
id	The extensible structure's repository id.
name	The extensible structure's name.
version	The extensible structure's version.
base	The base class from which the extensible struct inherits. Use NULL to indicate that this is the root structure.
members	The values for the extensible structure's fields.

```
public org.omg.CORBA.ExceptionDef create_exception(String id,
                        String name,
                        org.omg.CORBA.VersionSpec version,
                        org.omg.CORBA.StructMember members[])
```

This method creates a ExceptionDef object in this Container with the specified attributes and returns a reference to the newly created object.

Parameter	Description
id	The exception's repository id.
name	The exception's name.
version	The exception's version.
members	A list of all the types of the members of the exception, if any.

```
public org.omg.CORBA.InterfaceDef create_interface(String id,
                        String name,
                        org.omg.CORBA.VersionSpec version,
                        org.omg.CORBA.InterfaceDef base_interfaces)
```

This method creates a InterfaceDef object in this Container with the specified attributes and returns a reference to the newly created object.

Parameter	Description
id	The interface's repository id.
name	The interface's name.
version	The interface's version.
base_interfaces	A list of all interfaces from which this interface inherits.

```
public org.omg.CORBA.ModuleDef create_module(String id,
                        String name,
                        org.omg.CORBA.VersionSpec version)
```

This method creates a ModuleDef object in this Container with the specified attributes and returns a reference to the newly created object.

Parameter	Description
id	The module's repository id.
name	The module's name.
version	The module's version.

```
public org.omg.CORBA.StructDef create_struct(String id,
                        String name,
                        org.omg.CORBA.VersionSpec version,
                        org.omg.CORBA.StructMember members[])
```

This method creates a StructDef object in this Container with the specified attributes and returns a reference to the newly created object.

Parameter	Description
id	The structure's repository id.
name	The structure's name.
version	The structure's version.
members	The values for the structure's fields.

```
public org.omg.CORBA.UnionDef create_union(String id,
                        String name,
                        org.omg.CORBA.VersionSpec version,
                        org.omg.CORBA.IDLType discriminator_type,
                        org.omg.CORBA.UnionMember members[])
```

This method creates a UnionDef object in this Container with the specified attributes and returns a reference to the newly created object.

Parameter	Description
id	The union's repository id.
name	The union's name.
version	The union's version.
discriminator_type	The IDL type of the union's discriminant value.
members	A list of the types of each of the union's fields.

```
public org.omg.CORBA.ContainerPackage.Description[] describe_contents(
                        org.omg.CORBA.DefinitionKind limit_type,
                        boolean exclude_inherited,
                        int max_returned_objs)
```

This method returns a description for all definitions directly contained by, or inherited into this container.

Parameter	Description
limit_type	The interface object types to be returned.
exclude_inherited	If set to true, inherited objects will not be returned.
max_returned_objs	The maximum number of object to be returned. Setting this parameter to -1 will return all objects.

```
public org.omg.CORBA.Contained lookup(String search_name)
```

This method locates a definition relative to this container, given a scoped name. An absolute scoped name, one beginning with "::", may be specified to locate a definition within the enclosing repository. If no object is found, a NULL value is returned.

Parameter	Description
search_name	The name of the object to be located.

```
public org.omg.CORBA.Contained[] lookup_name(String search_name,
                        int levels_to_search,
                        org.omg.CORBA.DefinitionKind limit_type,
                        boolean exclude_inherited)
```

This method locates an object by name within a particular object. The search can be constrained by the number of levels in the hierarchy to be searched, the type of object, and whether or not inherited objects should be returned.

Parameter	Description
search_name	The name of the object or objects to be located.
levels_to_search	The number of levels in the hierarchy to search. Setting this parameter to a value of -1 will cause all levels to be searched. Setting this parameter to 1 will search only this object.
limit_type	The interface object types to be returned.
exclude_inherited	If set to true, inherited objects will not be returned.

ContainerPackage.Description

```
public class Description
```

This class provides a generic description for items in the interface repository that are derived from the Contained interface.

Helper and Holder versions of this class are also provided.

Variables

public org.omg.CORBA.Contained **contained_object**	The contained item.
public org.omg.CORBA.DefinitionKind **kind**	The kind of the item.
public org.omg.CORBA.Any **value**	The value of the item.

Methods

- Description

```
public Description()
```

This method is the default constructor for a Description.

```
public Description(org.omg.CORBA.Contained contained_object,
                   org.omg.CORBA.DefinitionKind kind,
```

```
org.omg.CORBA.Any value)
```

This method constructs an Description, using the supplied parameters.

Parameter	Description
contained_object	The contained item.
kind	This item's kind. See “DefinitionKind” for more information.
value	An Any object that represents the value for this item.

DefinitionKind

```
public class DefinitionKind
```

The DefinitionKind class contains the constants that define the possible types of interface repository objects. There are a set of integer constants, prefixed with `dk_`, that correspond to all the possible definition kinds. For example, the DefinitionKind code for float is `dk_float`.

Helper and Holder versions of this class are also provided.

Variables

Constant	Represents
dk_none	Exclude all types (used in repository lookup methods).
dk_any	All possible types (used in repository lookup methods).
dk_Alias	An Alias.
dk_Array	An Array.
dk_Attribute	An Attribute.
dk_Constant	A Constant.
dk_Enum	An Enum.

Constant	Represents
dk_Estruct	An extended structure.
dk_Exception	An Exception
dk_Interface	An Interface.
dk_Module	A Module.
dk_Operation	An Interface Operation.
dk_Primitive	A primitive type (such as int or long)
dk_Typedef	A Typedef.
dk_Union	A Union.
dk_Repository	A Repository.
dk_Sequence	A Sequence.
dk_String	A String.
dk_Struct	A Struct.
dk_Wstring	A Unicode string.

Methods

- value

```
public int value()
```

This method returns an integral value representing the constant.

EnumDef

```
public interface EnumDef extends org.omg.CORBA.TypeDef
```

The interface is used to represent an enumeration that is stored in the Interface Repository. This interface provides methods for setting and retrieving the enumeration's list of members.

Helper and Holder versions of this class are also provided.

Methods

- members

```
public String[] members()
```

This method returns the enumeration's list of members.

```
public void members(String members[])
```

This method sets the enumeration's list of members.

Parameter	Description
members	The list of members.

EstructDef

```
public interface EstructDef extends org.omg.CORBA.TypeDef
```

The interface is used to represent an extensible structure that is stored in the Interface Repository. This interface provides methods for setting and retrieving the structure's list of members and base class definition. Extensible structures are used to represent classes that are passed as parameters for an operation request.

Helper and Holder versions of this class are also provided.

Methods

- base
- base_def
- members

```
public org.omg.CORBA.TypeCode base()
```

This method returns the TypeCode of the base class for this Estruct.

```
public void base_def(org.omg.CORBA.EstructDef base_def)
```

This method sets the base class for this EstructDef.

Parameter	Description
base_def	An EstructDef that represents the base class.

```
public org.omg.CORBA.EstructDef base_def()
```

This method returns the EstructDef that represents the base class for this Estruct.

```
public org.omg.CORBA.StructMember[] members()
```

This method returns the list of members for this EstructDef.

```
public void members(StructMember members[])
```

This method sets the list of members for this EstructDef.

Parameter	Description
members	The list of members for this EstructDef.

```
public void members(String members[])
```

This method sets the enumeration's list of members.

Parameter	Description
members	The list of members.

ExceptionDef

```
public interface ExceptionDef extends org.omg.CORBA.Contained
```

The interface is used to represent an exception that is stored in the Interface Repository. This interface provides methods for setting and retrieving the exception's list of members as well as a method for retrieving the exception's TypeCode.

Helper and Holder versions of this class are also provided.

Methods

- members
- type

```
public org.omg.CORBA.StructMember[] members()
```

This method returns this exception's list of members.

```
public void members(org.omg.CORBA.StructMember members[])
```

This method sets the exception's list of members.

Parameter	Description
members	The list of members.

```
public org.omg.CORBA.TypeCode type()
```

This method returns the TypeCode that represents this exception's type.

ExceptionDescription

```
public class ExceptionDescription
```

The ExceptionDescription class describes a exception that is stored in the interface repository.

Helper and Holder versions of this class are also provided.

Variables

public String **defined_in**	The name of the module or interface in which this exception is defined.
public String **id**	The repository id of the exception.

public String **name**	The name of the exception.
public org.omg.CORBA.TypeCode **type**	The exception's IDL type.
public org.omg.CORBA.Any **value**	The value of this exception.
public org.omg.CORBA.VersionSpec **version**	The exception's version.

Methods

- ExceptionDescription

```
public ExceptionDescription()
```

This method is the default constructor for a ExceptionDescription.

```
public ExceptionDescription(String name, String id, String defined_in,
                    org.omg.CORBA.VersionSpec version,
                    org.omg.CORBA.TypeCode type,
                    org.omg.CORBA.Any value)
```

This method constructs an ExceptionDescription, using the supplied parameters.

Parameter	Description
name	The name of this exception.
id	The repository id for this exception.
defined_in	The module or interface in which this exception is defined.
version	The object's version.
type	The exception's IDL type code.
value	The value of this exception.

IDLType

```
public interface IDLType extends org.omg.CORBA.IRObject
```

The IDLType interface is used to represent the TypeCode associated with an interface repository object. The TypeCode uniquely identifies the object's type.

Helper and Holder versions of this class are also provided.

IDL Definition

```
interface IDLType : CORBA::IRObject {
      readonly attribute TypeCode type;
    };
```

Methods

- type

```
public org.omg.CORBA.TypeCode type()
```

This method returns the TypeCode of the current IRObject.

InterfaceDef

```
public interface InterfaceDef extends org.omg.CORBA.Container,
                                      org.omg.CORBA.Contained,
                                      org.omg.CORBA.IDLType
```

The interface is used to represent an object implementation interface that is stored in the Interface Repository. This interface provides methods for setting and retrieving the base interface as well as creating attributes, operations and an interface description.

Helper and Holder versions of this class are also provided.

IDL Definition

```
interface InterfaceDef : CORBA::Container,
                         CORBA::Contained,
                         CORBA::IDLType {
      attribute CORBA::InterfaceDefSeq base_interfaces;
```

```
  struct FullInterfaceDescription {
    CORBA::Identifier name;
    CORBA::RepositoryId id;
    CORBA::RepositoryId defined_in;
    CORBA::VersionSpec version;
    CORBA::OpDescriptionSeq operations;
    CORBA::AttrDescriptionSeq attributes;
    CORBA::RepositoryIdSeq base_interfaces;
    TypeCode type;
  };
  CORBA::AttributeDef create_attribute(
    in CORBA::RepositoryId id,
    in CORBA::Identifier name,
    in CORBA::VersionSpec version,
    in CORBA::IDLType type,
    in CORBA::AttributeMode mode
  );
  CORBA::OperationDef create_operation(
    in CORBA::RepositoryId id,
    in CORBA::Identifier name,
    in CORBA::VersionSpec version,
    in CORBA::IDLType result,
    in CORBA::OperationMode mode,
    in CORBA::ParDescriptionSeq params,
    in CORBA::ExceptionDefSeq exceptions,
    in CORBA::ContextIdSeq contexts
  CORBA::InterfaceDef::FullInterfaceDescription describe_interface();
  boolean is_a(in CORBA::RepositoryId interface_id);
  );
};
```

Methods

```
public org.omg.CORBA.InterfaceDef[] base_interfaces()
```

This method returns the base interface list for this object.

```
public void base_interfaces(org.omg.CORBA.InterfaceDef
                            base_interfaces[])
```

This method sets the base interface list for this object.

Parameter	Description
base_interfaces	The list of base interfaces to be set.

```
public org.omg.CORBA.AttributeDef create_attribute(String id,
                    String name,
                    org.omg.CORBA.VersionSpec version,
                    org.omg.CORBA.IDLType type,
                    org.omg.CORBA.AttributeMode mode)
```

This method adds an attribute to an interface definition.

Parameter	Description
id	The attribute's identifier.
name	The attribute's name.
version	The attribute's version.
type	The attribute's IDL type.
mode	The attribute's mode. See AttributeMode for possible values.

```
public org.omg.CORBA.OperationDef create_operation(String id,
                    String name,
                    org.omg.CORBA.VersionSpec version,
                    org.omg.CORBA.IDLType result,
                    org.omg.CORBA.OperationMode mode,
                    org.omg.CORBA.ParameterDescription[] params,
                    org.omg.CORBA.ExceptionDef[] exceptions,
                    String[] contexts)
```

This method adds an operation to an interface definition.

Parameter	Description
id	The operation's identifier.
name	The operation's name.
version	The operation's version.
type	The operation's IDL type.
mode	The operation's mode.
params	The list of parameters for this operation.
exceptions	The list of exceptions that can be raised by this operation.
contexts	The list of contexts.

```
public org.omg.CORBA.InterfaceDef.FullInterfaceDescription
                    describe_interface()
```

This method returns an interface description for this object.

```
public boolean is_a(String interface_id)
```

This method returns true if this object has the specified interface identifier.

Parameter	Description
interface_id	The interface identifier to compare with this object.

InterfaceDefPackage.FullInterfaceDescription

```
public class FullInterfaceDescription
```

This class provides a description of an interface that is stored in the interface repository.

Variables

public org.omg.CORBA.AttributeDescription[] **attributes**	The interface's list of attributes.
public String **defined_in**	The name of the module in which this interface is defined.
public String **id**	The repository id of the interface.
public String **name**	The name of the interface.
public org.omg.CORBA.OperationDescription[] **operations**	The list of operations offered by the interface.
public org.omg.CORBA.TypeCode **type**	The interface's IDL type.
public org.omg.CORBA.VersionSpec **version**	The interface's version.

Methods

- FullIinterfaceDescription

```
public FullInterfaceDescription()
```

This method is the default constructor for a FullInterfaceDescription.

```
public FullInterfaceDescription(String name, String id, String defined_in,
                    org.omg.CORBA.VersionSpec version,
                    org.omg.CORBA.OperationDescription[] operations,
                    org.omg.CORBA.AttributeDescription[] attributes,
                    org.omg.CORBA.TypeCode type)
```

This method constructs an FullInterfaceDescription, using the supplied parameters.

Parameter	Description
name	The name of this interface.
id	The repository id for this interface.
defined_in	The module or interface in which this attribute is defined.
version	The object's version.
operations	The list of operation offered by this interface.

Parameter	Description
attributes	The list of this interface's attributes.
type	The interface's IDL type code.

InterfaceDescription

```
public class InterfaceDescription
```

The InterfaceDescription class describes a constant that is stored in the interface repository.

Helper and Holder versions of this class are also provided.

Variables

public String[] **base_interfaces**	Alist of base interfaces for this interface.
public String **defined_in**	The name of the module in which this interface is defined.
public String **id**	The repository id of the interface.
public String **name**	The name of the interface.
public org.omg.CORBA.VersionSpec **version**	The interface's version.

Methods

- InterfaceDescription

```
public InterfaceDescription()
```

This method is the default constructor for an InterfaceDescription.

```
public InterfaceDescription(String name, String id, String defined_in,
                    org.omg.CORBA.VersionSpec version,
                    String[] base_interfaces)
```

This method constructs an InterfaceDescription, using the supplied parameters.

Parameter	Description
name	The name of this interface.
id	The repository id for this interface.
defined_in	The module or interface in which this interface is defined.
version	The object's version.
base_interfaces	The interface's list of base interfaces.

IRObject

```
public interface IRObject extends org.omg.CORBA.Object
```

The IRObject interface offers a generic interface to any object stored in the Interface Repository.

Helper and Holder versions of this class are also provided.

IDL Definition

```
interface IRObject {
   readonly attribute CORBA::DefinitionKind def_kind;
   attribute string comment;
   void destroy();
};
```

Methods

```
public org.omg.CORBA.DefinitionKind def_kind()
```

This method returns the *type* of this IRObject. For a list of defined types, see "TCKind."

```
public void destroy()
```

This method deletes this IRObject from the Interface Repository.

ModuleDef

```
public interface ModuleDef extends org.omg.CORBA.Container,
                                   org.omg.CORBA.Contained
```

The interface is used to represent an IDL module in the interface repository.

Helper and Holder versions of this class are also provided.

ModuleDescription

```
public class ModuleDescription
```

The ModuleDescription class describes a constant that is stored in the interface repository.

Helper and Holder versions of this class are also provided.

Variables

public String **defined_in**	The name of the module in which this interface is defined.
public String **id**	The repository id of the interface.
public String **name**	The name of the interface.
public org.omg.CORBA.VersionSpec **version**	The interface's version.

Methods

- ModuleDescription

```
public ModuleDescription()
```

This method is the default constructor for an ModuleDescription.

```
public ModuleDescription(String name, String id, String defined_in,
                         org.omg.CORBA.VersionSpec version)
```

This method constructs an ModuleDescription, using the supplied parameters.

Parameter	Description
name	The name of this interface.
id	The repository id for this interface.
defined_in	The module or interface in which this interface is defined.
version	The object's version.

OperationDef

```
public interface OperationDef extends org.omg.CORBA.Contained
```

The interface is used to represent an interface operation that is stored in the Interface Repository. This interface provides methods for setting and retrieving the operation's contexts, mode, parameters, and result value. A method is also provided for retrieving a list of exceptions that may be raised by this operation.

Helper and Holder versions of this class are also provided.

Methods

- contexts
- exceptions
- mode
- params
- result
- result_def

```
public String[] contexts()
```

This method returns the contexts associated with this operation.

```
public void contexts(String[] contexts)
```

This method sets this operation's list of context.

Parameter	Description
contexts	The list of contexts.

```
public org.omg.CORBA.ExceptionDef[] exceptions()
```

This method returns a list of exceptions that may be raised by this operation.

```
public void exceptions(org.omg.CORBA.ExceptionDef[] exceptions)
```

This method sets the list of exceptions that may be raised by this operation.

Parameter	Description
exceptions	The list of exceptions.

```
public org.omg.CORBA.OperationMode mode()
```

This method returns the mode of this operation.

```
public void mode(org.omg.CORBA.OperationMode mode)
```

This method sets the mode of this operation.

Parameter	Description
mode	The mode to be set. See OperationMode for more details.

```
public org.omg.CORBA.ParameterDescription[] params()
```

This method returns a description of the parameters for this operation.

```
public void params(org.omg.CORBA.ParameterDescription[] params)
```

This method sets the parameter description for this operation.

Parameter	Description
params	The description of the parameters.

```
public org.omg.CORBA.TypeCode result()
```

This method returns the TypeCode of the result returned by this operation.

```
public org.omg.CORBA.IDLType result_def()
```

This method returns the IDL type of this operation's return value.

```
public void result_def(org.omg.CORBA.IDLType result_def)
```

This method sets the IDL type for this operation's return value.

Parameter	Description
result_def	The IDL type to set for the return value.

OperationDescription

```
public class OperationDescription
```

The OperationDescription class describes a constant that is stored in the interface repository.

Helper and Holder versions of this class are also provided.

Variables

public String[] **contexts**	The contexts associated with this operation.
public String **defined_in**	The name of the module in which this operation is defined.
public org.omg.CORBA.ExceptionDescription[] **exceptions**	The exceptions that this operation may raise.
public String **id**	The repository id of the operation.
public org.omg.CORBA.OperationMode **mode**	The operation's mode.
public String **name**	The name of the operation.
public org.omg.CORBA.ParameterDescription[] **parameters**	The operation's parameters.
public org.omg.CORBA.TypeCode **result**	The operation's result.
public org.omg.CORBA.VersionSpec **version**	The operation's version.

Methods

- OperationDescription

```
public OperationDescription()
```

This method is the default constructor for an OperationDescription.

```
public OperationDescription(String name, String id, String defined_in,
                    org.omg.CORBA.VersionSpec version,
                    org.omg.CORBA.TypeCode result,
                    org.omg.CORBA.OperationMode mode,
                    String[] contexts,
                    org.omg.CORBA.ParameterDescriptions parameters,
                    org.omg.CORBA.ExceptionDescription[] exceptions)
```

This method constructs an OperationDescription, using the supplied parameters.

Parameter	Description
name	The name of this interface.
id	The repository id for this interface.
defined_in	The operation or interface in which this interface is defined.
version	The object's version.
result	The IDL type of the result of the operation.
mode	The operation's mode.
contexts	A list of Context objects for this operation.
parameters	The list of parameters for this operation.
exceptions	The list of exceptions that this operation may raise.

OperationMode

```
public class OperationMode
```

The class is used to represent the *mode* of an operation; either *oneway* or normal. Oneway operations are those for which the client application does not expect a response. Normal requests involve a response being sent to the client by the object implementation that contains the results of the request.

Helper and Holder versions of this class are also provided.

Variables

Constant	Represents
_OP_NORMAL	A normal operation.
_OP_ONEWAY	A one-way operation.

Methods

- _value

```
public int _value()
```

This method returns the value of this object, which is either _OP_NORMAL or _OP_ONEWAY.

ParameterDescription

```
public class ParameterDescription
```

The ParameterDescription class describes a parameter for an operation that is stored in the interface repository.

Helper and Holder versions of this class are also provided.

Variables

`public String` **`name`**	The name of the parameter.
public org.omg.CORBA.TypeCode **`type`**	The parameter's type.
public org.omg.CORBA.IDLType **`type_def`**	The parameter's IDL type.
public org.omg.CORBA.ParameterMode **`mode`**	The parameter's mode.

Methods

- ParameterDescription

```
public ParameterDescription()
```

This method is the default constructor for an ParameterDescription.

```
public ParameterDescription(String name,
                    org.omg.CORBA.TypeCode type,
                    org.omg.CORBA.IDLType type_def,
                    org.omg.CORBA.ParameterMode mode)
```

This method constructs an ParameterDescription, using the supplied parameters.

Parameter	Description
name	The name of the parameter.
type	The type of the parameter.
type_def	The IDL type of the parameter.
mode	The mode of the parameter.

ParameterMode

```
public class ParameterMode
```

The class is used to represent the *mode* of a parameter. Parameters may be used in one of the following three ways:

- IN - Used for input from the client to the server.
- OUT - Used for output of results from the server to the client.
- INOUT - Used for both input from the client and output from the server.

Helper and Holder versions of this class are also provided.

Variables

Constant	Represents
_PARAM_IN	A IN parameter.
_OP_PARAM_OUT	An OUT parameter.
_OP_PARAM_INOUT	An INOUT parameter.

Methods

- _value

```
public int _value()
```

This method returns the value of this object, which is either _PARAM_IN, _PARAM_OUT, or _PARAM_INOUT.

PrimitiveDef

```
public interface PrimitiveDef extends org.omg.CORBA.IDLType
```

The interface is used to represent an primitive (such as an int or a long) that is stored in the Interface Repository. This interface provides a methods for retrieving the kind of primitive that is being represented.

Helper and Holder versions of this class are also provided.

Methods

- kind

```
public org.omg.CORBA.PrimitiveKind kind()
```

This method returns the kind of primitive represented by this object.

PrimitiveKind

```
public class PrimitiveKind
```

The PrimitiveKind class contains the constants that define the possible types of primitives interface repository objects. There are a set of integer constants, prefixed with pk_, that correspond to all the possible kinds of primitives.

Helper and Holder versions of this class are also provided.

Methods

- value

```
public int value()
```

This method returns an integral value representing the constant.

Constant	Represents
pk_null	A null.
pk_void	A void.
pk_short	A short.
pk_long	A long.
pk_ushort	A unsigned short.
pk_ulong	An unsigned long.
pk_float	A float.
pk_double	A double.
pk_boolean	A boolean.

Constant	Represents
pk_char	A character.
pk_octet	An octet string.
pk_any	An Any object.
pk_TypeCode	A TypeCode object.
pk_Principal	A Principal object.
pk_string	A string.
pk_objref	An object reference.
pk_longlong	A long long.
pk_ulonglong	An unsigned long.
pk_longdouble	A long double.
pk_wchar	A Unicode character.
pk_wstring	A Unicode string.

Repository

```
public interface Repository extends org.omg.CORBA.Container
```

The Repository provides an interface to the Interface Repository, which is used to contain the definitions of objects that are available to clients. The Repository interface provides methods for storing and retrieving definitions.

Helper and Holder versions of this class are also provided.

Methods

- create_array
- create_sequence
- create_string
- create_wstring
- get_primitive
- lookup_id

```
public org.omg.CORBA.ArrayDef create_array(int length,
                    org.omg.CORBA.IDLType element_type)
```

This method creates an array definition in the repository with the specified length and element type.

Parameter	Description
length	The number of elements in the array. This value must be greater than zero.
element_type	The IDL type of the elements contained in the array.

A reference to the ArrayDef that is created is returned.

```
public org.omg.CORBA.SequenceDef create_sequence(int bound,
                    org.omg.CORBA.IDLType element_type)
```

This method creates an array definition in the repository with the specified bound and element type. A reference to the SequenceDef that is created is returned.

Parameter	Description
bound	The maximum length of the sequence. This value must be greater than zero.
element_type	The IDL type of the elements contained in the sequence.

```
public org.omg.CORBA.StringDef create_string(int bound)
```

This method creates an string definition in the repository with the specified bound. A reference to the StringDef that is created is returned.

Parameter	Description
bound	The maximum bounds of the string. This value must be greater than zero.

```
public org.omg.CORBA.WstringDef create_wstring(int bound)
```

This method creates a Unicode string definition in the repository with the specified bound. A reference to the WstringDef that is created is returned.

Parameter	Description
bound	The maximum bounds of the string. This value must be greater than zero.

```
public org.omg.CORBA.PrimitiveDef get_primitive(org.omg.CORBA.PrimitiveKind kind)
```

This method returns a

Parameter	Description
kind	The primitive's kind.

PrimitiveDef object for the specified PrimitiveKind.

```
public Contained lookup_id(String search_id)
```

This method searches for an object in the interface repository that matches the specified search id. If a match is not found, a null value is returned.

Parameter	Description
search_id	The identifier to use for the search.

SequenceDef

```
public interface SequenceDef extends org.omg.CORBA.IDLType
```

The interface is used to represent an sequence that is stored in the Interface Repository. This interface provides methods for setting and retrieving the sequence's bound and element type.

Helper and Holder versions of this class are also provided.

Methods

- bound
- element_type
- element_type_def

```
public int bound()
```

This method returns the bound of the sequence.

```
public void bound(int bound)
```

This method sets the bound of the sequence.

Parameter	Description
members	The list of members.

```
public org.omg.CORBA.TypeCode element_type()
```

This method returns a TypeCode representing the type of elements in this sequence.

```
public org.omg.CORBA.IDLType element_type_def()
```

This method returns the IDL type of the elements stored in this sequence.

```
public void element_type_def(org.omg.CORBA.IDLType element_type_def)
```

This method sets the IDL type for the elements stored in this sequence.

Parameter	Description
element_type_def	The IDL type to set.

StringDef

```
public interface StringDef extends org.omg.CORBA.IDLType
```

The interface is used to represent a String that is stored in the Interface Repository. This interface provides methods for setting and retrieving the bounds of the string.

Helper and Holder versions of this class are also provided.

Methods

- bound
- members

```
public int bound()
```

This method returns the bounds of the String.

```
public void members(int bound)
```

This method sets the bounds of the String.

Parameter	Description
members	The list of members.

StructDef

```
public interface StructDef extends org.omg.CORBA.TypedefDef
```

The interface is used to represent a structure that is stored in the Interface Repository. This interface provides methods for setting and retrieving the structure's list of members.

Helper and Holder versions of this class are also provided.

Methods

- members

```
public org.omg.CORBA.StructMember[] members()
```

This method returns the structures's list of members.

```
public void members(org.omg.CORBA.StructMember[] members)
```

This method sets the structure's list of members.

Parameter	Description
members	The list of members.

TypedefDef

```
public interface TypedefDef extends org.omg.CORBA.Contained,
                                    org.omg.CORBA.IDLType
```

This abstract interface represents a user-defined structure that is stored in the Interface Repository. The following interfaces all inherit from this interface:

- AliasDef
- EnumDef
- EstructDef
- StructDef
- UnionDef
- WstringDef

Helper and Holder versions of this class are also provided.

TypeDescription

```
public class TypeDescription
```

The TypeDescription class describes a type for an operation that is stored in the interface repository.

Helper and Holder versions of this class are also provided.

Variables

public String **defined_in**	The name of the module in which this type is defined.
public String **id**	The repository id of the type.
public String **name**	The name of the type.
public org.omg.CORBA.TypeCode **type**	The type's IDL type.
public org.omg.CORBA.VersionSpec **version**	The type's version.

Methods

- TypeDescription

```
public TypeDescription()
```

This method is the default constructor for an TypeDescription.

```
public TypeDescription(String name, String id, String defined_in,
                       org.omg.CORBA.VersionSpec version,
                       org.omg.CORBA.TypeCode type)
```

This method constructs an TypeDescription, using the supplied parameters.

Parameter	Description
name	The name of this type.
id	The repository id for this type.
defined_in	The module or interface in which this type is defined.
version	The object's version.
type	The type's IDL type code.

UnionDef

```
public interface UnionDef extends org.omg.CORBA.TypedefDef
```

The interface is used to represent an Union that is stored in the Interface Repository. This interface provides methods for setting and retrieving the union's list of members and discriminator type.

Helper and Holder versions of this class are also provided.

Methods

- discriminator_type
- discriminator_type_def
- members

```
public org.omg.CORBA.TypeCode discriminator_type()
```

This method returns the TypeCode of the discriminator for the Union.

```
public org.omg.CORBAIDLType discriminator_type_def()
```

This method returns the IDL type of the union's discriminator.

```
public void discriminator_type_def(
                    org.omg.CORBA.IDLType discriminator_type_def)
```

This method sets the IDL type of the union's discriminator.

Parameter	Description
discriminator_type_def	The list of members.

```
public org.omg.CORBA.UnionMember[] members()
```

This method returns the union's list of members.

```
public void members(org.omg.CORBA.UnionMembers[] members)
```

This method sets the union's list of members.

Parameter	Description
members	The list of members.

UnionMember

```
public class UnionMember
```

The UnionMember class describes a union that is stored in the interface repository.

Helper and Holder versions of this class are also provided.

Variables

label	The label for this union.
name	The name of this union.
type	The union's type.
type_def	The union's IDL type code.

Methods

- UnionMember

```
public UnionMember()
```

This method is the default constructor for an UnionMember.

```
public UnionMember(String name, org.omg.CORBA.Any label,
                   org.omg.CORBA.TypeCode type,
                   org.omg.CORBA.IDLType type_def)
```

This method constructs an UnionMember, using the supplied parameters.

Parameter	Description
name	The name of this union.
label	The label for this union.
type	The union's type.
type_def	The union's IDL type code.

VersionSpec

```
public interface VersionSpec
```

The VersionSpec interface describes the version of an object that is stored in the interface repository.

Helper and Holder versions of this class are also provided.

Variables

major	The major version number.
minor	The minor version number.

Methods

- VersionSpec

```
public VersionSpec()
```

This method is the default constructor for an UnionMember.

```
public VersionSpec(short major, short minor)
```

This method constructs an VersionSpec, using the supplied parameters.

Parameter	Description
major	The major version number.
minor	The minor version number.

WstringDef

```
public interface WstringDef extends org.omg.CORBA.IDLType
```

The interface is used to represent a Unicode string that is stored in the Interface Repository.

Helper and Holder versions of this class are also provided.

Methods

- bound

```
public int bound()
```

This method returns the bounds of the WString.

```
public void members(int bound)
```

This method sets the bounds of the WString.

Parameter	Description
members	The list of members.

Chapter

7

Exceptions Classes

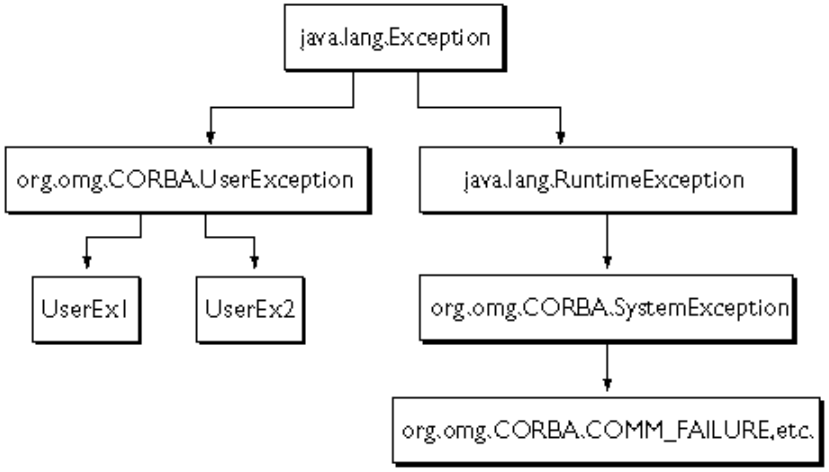
This chapter describes the exception classes used in ISB for Java. It includes the following major sections:

- Introduction
- SystemException
- UserException

Introduction

CORBA system exceptions have been made a subclass of `java.lang.RuntimeException`. This means that CORBA system exceptions do not need to be declared in all method signatures which might raise such exceptions.

The UserException is a subclass of `java.lang.Exception`. Due to the inheritance hierarchy, the class `org.omg.CORBA.Exception` no longer exists.



SystemException

```
abstract public class SystemException extends java.lang.RuntimeException
```

CORBA system exceptions are raised when the runtime encounters problems. They inherit from j`ava.lang.RuntimeException`. A table below summarizes all the SystemException classes that can be raised and their associated meanings.

The standard IDL system exceptions are mapped to final Java classes that extend `org.omg.CORBA.SystemException` and provide access to the IDL major and minor exception code, as well as a string describing the reason for the exception.

There are no public constructors for `org.omg.CORBA.SystemException`; only classes that extend it can be instantiated.

Currently, ISB for Java does not support the use of minor codes; consequently, there is no minor method.

Attributes

```
public CompletionStatus completed
```

This attribute indicates whether the operation was completed or not.

Methods

- minor

```
public void minor(int minor)
```

This method sets the minor code. The minor code is not used currently.

Parameter	Description
minor	The minor code to be set.

Exception Class Names

Exception Class Name	Description
BAD_CONTEXT	Error processing context object.
BAD_INV_ORDER	Routine invocations out of order.
BAD_OPERATION	Invalid operation.
BAD_PARAM	An invalid parameter was passed.
BAD_TYPECODE	Invalid typecode.
COMM_FAILURE	Communication failure.
DATA_CONVERSION	Data conversion error.
FREE_MEM	Unable to free memory.
IMP_LIMIT	Implementation limit violated.
INITIALIZE	ORB initialization failure.
INTERNAL	ORB internal error.
INTF_REPOS	Error accessing interface repository.

Exception Class Name	Description
INV_FLAG	Invalid flag was specified.
INV_INDENT	Invalid identifier syntax.
INV_OBJREF	Invalid object reference specified.
MARSHAL	Error marshalling parameter or result.
NO_IMPLEMENT	Operation implementation not available.
NO_MEMORY	Dynamic memory allocation failure.
NO_PERMISSION	No permission for attempted operation.
NO_RESOURCES	Insufficient resources to process request.
NO_RESPONSE	Response to request not yet available.
OBJ_ADAPTER	Failure detected by object adaptor.
OBJECT_NOT_EXIST	Object is not available.
PERSIST_STORE	Persistent storage failure.
TRANSIENT	Transient failure.
UNKNOWN	Unknown exception.

UserException

```
public class UserException extends java.lang.Exception
```

The UserException class is an abstract base class used to define exceptions that an object implementation may raise. There is no state information associated with these exception types, but derived classes may add their own state information. The primary use of this class is to simplify the use of catch blocks in a client's code, as shown in the following code example.

```
try {
  proxy.operation();
}
catch(org.omg.CORBA.SystemException se) {
  System.out.println("The runtime failed: " + se);
}
catch(org.omg.CORBA.UserException ue) {
  System.out.println("The implementation failed: " + ue);
}
```

Constructor

```
protected UserException()
```

This method creates a UserException exception.