

# Orbix 3.3.14

Programmer's Reference Java Edition

Micro Focus  
The Lawn  
22-30 Old Bath Road  
Newbury, Berkshire RG14 1QN  
UK

<http://www.microfocus.com>

Copyright © Micro Focus 2017. All rights reserved.

MICRO FOCUS, the Micro Focus logo, and Micro Focus product names are trademarks or registered trademarks of Micro Focus Development Limited or its subsidiaries or affiliated companies in the United States, United Kingdom, and other countries. All other marks are the property of their respective owners.

2017-04-24

# Contents

<b>Preface</b> .....	<b>xv</b>
Audience .....	xv
Organization of this Guide .....	xv
Document Conventions .....	xvi

## Part I Package org.omg.CORBA

Class org.omg.CORBA.ARG_IN .....	3
Class org.omg.CORBA.ARG_INOUT .....	4
Class org.omg.CORBA.ARG_OUT .....	5
Class org.omg.CORBA.Bounds .....	6
Bounds() .....	6
Class org.omg.CORBA.CompletionStatus .....	7
value() .....	7
from_int() .....	8
Class org.omg.CORBA.Context .....	9
Class org.omg.CORBA.ContextList .....	10
Class org.omg.CORBA.CTX_RESTRICT_SCOPE .....	11
Class org.omg.CORBA.Current .....	12
Class org.omg.CORBA.DynamicImplementation .....	13
Class org.omg.CORBA.Environment .....	14
Class org.omg.CORBA.ExceptionList .....	15
Class org.omg.CORBA.NamedValue .....	16
Class org.omg.CORBA.NVList .....	17
Interface org.omg.CORBA.Object .....	18
Class org.omg.CORBA.ORB .....	19
init() .....	22
init() .....	22
init() .....	23
Class org.omg.CORBA.ORBPackage.InvalidName .....	24
InvalidName() .....	24
Class org.omg.CORBA.portable.InputStream .....	25
Class org.omg.CORBA.portable.OutputStream .....	26
Class org.omg.CORBA.portable.Streamable .....	27
Class org.omg.CORBA.Principal .....	28
Class org.omg.CORBA.Request .....	29
Class org.omg.CORBA.ServerRequest .....	31
op_name() .....	32
ctx() .....	32
params() .....	32
result() .....	34
except() .....	34
Class org.omg.CORBA.SystemException .....	35
Class org.omg.CORBA.TCKind .....	36
value() .....	37
from_int() .....	37
Class org.omg.CORBA.TypeCode .....	38
Class org.omg.CORBA.TypeCodePackage.BadKind .....	40
BadKind() .....	40
Class org.omg.CORBA.TypeCodePackage.Bounds .....	41

Bounds()	41
Class org.omg.CORBA.UnknownUserException	42
UnknownUserException()	42
Class org.omg.CORBA.UserException	43

## Part II Package IE.Iona.OrbixWeb.CORBA

Class IE.Iona.OrbixWeb.CORBA.Any	47
Any()	49
Any()	50
Any()	50
Any()	51
clone()	51
containsType()	51
copy()	52
create_output_stream()	52
create_input_stream()	52
equal()	53
extract()	53
fromString()	54
insert()	55
read_value()	56
reset()	57
toString()	57
type()	57
write_value()	58
Interface IE.Iona.OrbixWeb.CORBA.BOA	59
anyClientsConnected()	63
change_implementation()	63
connect()	64
continueThreadDispatch()	65
create()	66
deactivate_impl()	66
deactivate_obj()	67
disconnect()	67
dispose()	68
enableLoaders()	68
enableProxyServer()	68
finalize()	69
get_current()	69
get_id()	70
get_principal()	70
get_principal_string()	70
impl_is_ready()	71
isEventPending()	73
myActivationMode()	73
myHost()	74
myHostIP()	74
myImplementationName()	74
myMarkerName()	75
myMarkerPattern()	75
myMethodName()	75
numClientsConnected()	75
obj_is_ready()	76
processEvents()	77
processNextEvent()	79

setNoHangup()	80
setProxyServer()	81
setServerName()	81
shutdown()	81
toString()	82
Class IE.Iona.OrbixWeb.CORBA.Context	83
Context()	85
Context()	85
Context()	85
Context()	85
_nil()	86
context_name()	86
create_child()	86
delete_values()	86
get_count()	87
get_count_all()	87
get_values()	87
IT_create()	88
parent()	88
set_one_value()	88
set_values()	88
Class IE.Iona.OrbixWeb.CORBA.ContextList	90
ContextList()	90
add()	91
count()	91
item()	91
remove()	91
Class IE.Iona.OrbixWeb.CORBA.ContextIterator	92
ContextIterator()	92
ContextIterator()	92
next()	92
setList()	93
Class IE.Iona.OrbixWeb.CORBA.Environment	94
exception()	94
exception()	94
clear()	94
Class IE.Iona.OrbixWeb.CORBA.ExceptionList	95
ExceptionList()	95
add()	96
count()	96
item()	96
remove()	96
Class IE.Iona.OrbixWeb.CORBA.InitService	97
InitService()	97
initialise()	97
Class IE.Iona.OrbixWeb.CORBA.NamedValue	98
NamedValue()	99
NamedValue()	99
NamedValue()	99
clone()	100
equals()	100
toString()	100
name()	100
value()	101
flags()	101
_nil()	101
Class IE.Iona.OrbixWeb.CORBA.NVList	102

NVList()	103
NVList()	103
NVList()	103
equals()	103
clone()	104
add()	104
add_item()	104
add_value()	105
count()	105
item()	105
_nil()	106
remove()	106
Class IE.Iona.OrbixWeb.CORBA.NVListIterator	107
NVListIterator()	107
NVListIterator()	107
next()	108
setList()	108
Interface IE.Iona.OrbixWeb.CORBA.ObjectRef	109
_create_request()	110
_create_request()	111
_deref()	112
_get_implementation();	112
_get_interface_def();	112
_hash()	113
_hasValidOpenChannel()	113
_host()	113
_id()	113
_implementation()	114
_interfaceHost()	114
_interfaceImplementation()	114
_interfaceMarker()	114
_is_a()	115
_is_equivalent()	115
_isRemote()	116
IT_PING()	116
_loader()	116
_marker()	116
_marker()	116
_name()	117
_non_existent()	117
_port()	117
_request()	117
_object_to_string()	118
_object_to_string()	118
_save()	118
Class IE.Iona.OrbixWeb.CORBA.ORB	120
baseInterfacesOf()	127
closeConnection()	127
collocated()	127
collocated()	128
config()	128
create_tc()	129
create_any()	130
create_context_list()	131
create_environment()	131
create_exception_list()	132
create_list()	132

create_named_value()	133
create_operation_list()	133
create_output_stream()	135
defaultTxTimeout()	135
finalize()	136
getConfigItem()	136
getConfiguration()	137
getDaemonConnections()	137
get_default_context()	138
getHostPort()	138
get_my_principal()	139
getMyReqTransformer()	139
get_next_response()	140
get_principal()	140
getTransformer()	140
get_principal_string()	141
hasValidOpenChannel()	141
init()	142
isBaseInterfaceOf()	144
list_initial_services()	144
locator()	145
makeIOR()	145
maxConnectRetries()	146
myHost()	147
_nil()	147
noReconnectOnFailure()	147
object_to_string()	148
pingDuringBind()	148
poll_next_response()	149
reSizeConnectionTable()	149
registerIOCallback()	150
resolve_initial_references()	150
send_multiple_requests_deferred()	151
send_multiple_requests_oneway()	152
setConfigItem()	152
setConfiguration()	153
setDiagnostics()	153
setHostPort()	154
setMyReqTransformer()	154
set_parameters()	155
set_principal()	155
setReqTransformer()	155
string_to_object()	157
string_to_object()	157
toString()	158
unregisterIOCallback()	158
Class IE.Iona.OrbixWeb.CORBA.OrbCurrent	159
get_object()	161
get_principal()	161
get_principal_string()	162
get_protocol()	162
get_request()	162
get_server()	163
get_socket()	163
Class IE.Iona.OrbixWeb.CORBA.Principal	164
Principal()	165
Principal()	165

Principal()	165
access_name	166
name()	166
name()	166
toString()	166
Class IE.Iona.OrbixWeb.CORBA.Request	167
Request()	170
Request()	170
Request()	170
add_arg()	171
add_in_arg()	171
add_inout_arg()	171
add_named_in_arg()	172
add_named_inout_arg()	172
add_named_out_arg()	172
add_out_arg()	173
arguments()	173
contexts()	173
contexts()	173
create_input_stream()	173
create_output_stream()	174
ctx()	174
ctx()	174
env()	174
env()	174
exceptions()	175
exceptions()	175
getClientConnection()	175
_getException()	175
getMessageLength()	175
get_response()	176
invoke()	176
isDynamic()	176
isException()	176
_nil()	177
operation()	177
poll_response()	177
reset()	177
result()	178
return_value()	178
send_deferred()	178
send_oneway()	179
setOperation()	179
set_return_type()	179
setTarget()	180
target()	180
Class IE.Iona.OrbixWeb.CORBA.singletonORB	181
create_any()	183
create_context_list()	184
create_environment()	184
create_exception_list()	185
create_list()	185
create_named_value()	186
create_output_stream()	186
get_default_context()	187
create_tc()	187
Class IE.Iona.OrbixWeb.CORBA.TypeCode	190



TypeCode() .....	194
TypeCode() .....	195
kind() .....	195
equals() .....	195
equal() .....	196
compare() .....	196
id() .....	197
name() .....	197
member_count() .....	197
member_type() .....	197
member_name() .....	198
member_label() .....	198
discriminator_type() .....	198
default_index() .....	199
length() .....	199
content_type() .....	199
orbixTypeCode() .....	200
toString() .....	200

## Part III Package IE.Iona.OrbixWeb.Feature

Class IE.Iona.OrbixWeb.Features.AuthenticationFilter .....	203
AuthenticationFilter() .....	204
AuthenticationFilter() .....	204
Class IE.Iona.OrbixWeb.Features.Config .....	205
Config() .....	205
getConfigItem() .....	205
setConfigItem() .....	206
Class IE.Iona.OrbixWeb.Features.Filter .....	207
Filter() .....	208
Filter() .....	208
Filter() .....	208
_delete() .....	209
inReplyFailure() .....	209
inReplyPostMarshal() .....	209
inReplyPreMarshal() .....	210
inRequestPostMarshal() .....	211
inRequestPreMarshal() .....	211
outReplyFailure() .....	212
outReplyPostMarshal() .....	213
outReplyPreMarshal() .....	214
outRequestPostMarshal() .....	215
outRequestPreMarshal() .....	216
Interface IE.Iona.OrbixWeb.Features.ioCallback .....	217
CloseCallBack() .....	218
OpenCallBack() .....	218
Class IE.Iona.OrbixWeb.Features.IT_reqTransformer .....	219
transform() .....	220
transform_error() .....	221
Class IE.Iona.OrbixWeb.Features.LoaderClass .....	222
LoaderClass() .....	222
LoaderClass() .....	223
LoaderClass() .....	223
load() .....	223
record() .....	225
rename() .....	225

save()	226
Class IE.Iona.OrbixWeb.Features.locatorClass	227
locatorClass()	227
lookUp()	227
Class IE.Iona.OrbixWeb.Features.OrbConfig	229
OrbConfig()	229
defaultConfigFile()	229
getConfigFile()	230
getConfigItem()	230
getConfiguration()	230
setConfiguration()	230
setConfiguration()	230
setConfigItem()	231
zeroConfiguration()	231
Class IE.Iona.OrbixWeb.Features.ProxyFactory	232
ProxyFactory()	232
ProxyFactory()	232
New()	233
Class IE.Iona.OrbixWeb.Features.ThreadFilter	234
ThreadFilter()	234
ThreadFilter()	235
inRequestPreMarshal()	235

## Part IV Package IE.Iona.OrbixWeb

Class IE.Iona.OrbixWeb._CORBA	239
explicit_call	240
IT_INFINITE_TIMEOUT	240
IT_INTEROPERABLE_OR_KIND	240
IT_ORBIX_OR_KIND	240
_ORBIX_VERSION	240
_MAX_LOCATOR_HOPS	240
objectDeletion	241
Orbix	241
processTermination	242
Class IE.Iona.OrbixWeb._OrbixWeb	243
Any()	244
Context()	244
ContextList()	244
Current()	245
NamedValue()	246
NVList()	246
Object()	246
Principal()	247
Request()	247
ServerRequest()	247
TypeCode()	248

## Part V IDL Interface to the Interface Repository

Common CORBA Data Types	251
CORBA::DefinitionKind	251
CORBA::Identifier	251
CORBA::RepositoryId	251
CORBA::ScopedName	251

CORBA::AliasDef .....	252
AliasDef::describe() .....	252
AliasDef::original_type_def.....	252
CORBA::ArrayDef .....	253
ArrayDef::element_type .....	253
ArrayDef::element_type_def .....	253
ArrayDef::length.....	253
CORBA::AttributeDef .....	254
AttributeDef::describe() .....	254
AttributeDef::mode .....	254
AttributeDef::type.....	255
AttributeDef::type_def.....	255
CORBA::ConstantDef .....	256
ConstantDef::describe().....	256
ConstantDef::type.....	256
ConstantDef::type_def .....	257
ConstantDef::value .....	257
CORBA::Contained .....	258
Contained::absolute_name() .....	258
Contained::containing_repository() .....	259
Contained::defined_in .....	259
Contained::describe() .....	259
Contained::id .....	260
Contained::move ().....	260
Contained::name .....	260
Contained::version.....	261
CORBA::Container .....	262
Container::contents() .....	263
Container::create_alias() .....	264
Container::create_constant().....	264
Container::create_enum().....	265
Container::create_exception() .....	265
Container::create_interface() .....	266
Container::create_module() .....	266
Container::create_struct() .....	267
Container::create_union().....	267
Container::describe_contents().....	268
Container::lookup().....	268
Container::lookup_name() .....	269
CORBA::EnumDef .....	270
EnumDef::describe() .....	270
EnumDef::members .....	270
CORBA::ExceptionDef .....	271
ExceptionDef::describe().....	271
ExceptionDef::members.....	272
ExceptionDef::type .....	272
CORBA::IDLType .....	273
IDLType::type .....	273
CORBA::InterfaceDef .....	274
InterfaceDef::base_interfaces.....	275
InterfaceDef::create_attribute().....	275
InterfaceDef::create_operation() .....	276
InterfaceDef::describe() .....	277
InterfaceDef::describe_interface ().....	277
InterfaceDef::is_a () .....	278
CORBA::IObject .....	279
IObject::def_kind.....	279

IObject::destroy()	279
CORBA::IT_InterfaceDef	280
IT_InterfaceDef::derived_interfaces()	280
CORBA::IT_Repository	281
IT_Repository::start()	281
IT_Repository::commit()	281
IT_Repository::rollBack()	282
IT_Repository::active_transactions()	282
CORBA::ModuleDef	283
ModuleDef::describe()	283
CORBA::OperationDef	284
OperationDef::contexts	284
OperationDef::exceptions	284
OperationDef::describe()	285
OperationDef::mode	285
OperationDef::params	286
OperationDef::result	286
OperationDef::result_def	286
CORBA::PrimitiveDef	287
PrimitiveDef::kind	287
CORBA::Repository	288
Repository::create_array()	288
Repository::create_sequence()	289
Repository::create_string()	289
Repository::get_primitive()	289
Repository::describe_contents()	290
Repository::lookup_id()	290
CORBA::SequenceDef	291
SequenceDef::bound	291
SequenceDef::element_type	291
SequenceDef::element_type_def	291
SequenceDef::type	292
CORBA::StructDef	293
StructDef::describe()	293
StructDef::members	294
CORBA::StringDef	295
StringDef::bound	295
CORBA::TypedefDef	296
TypedefDef::describe()	296
CORBA::UnionDef	297
UnionDef::describe()	297
UnionDef::discriminator_type	298
UnionDef::discriminator_type_def	298
UnionDef::members	298

## Part VI IDL Interface to the Orbix Java Daemon

IDL Interface to the Orbix Java Daemon	301
IT_daemon::addLaunchRightsDir()	304
IT_daemon::addInvokeRights()	304
IT_daemon::addInvokeRightsDir()	305
IT_daemon::addLaunchRights()	305
IT_daemon::addLaunchRightsDir()	305
IT_daemon::addMethod()	305
IT_daemon::addSharedMarker()	306
IT_daemon::addUnsharedMarker()	307

IT_daemon::changeOwnerServer()	307
IT_daemon::deleteDirectory()	308
IT_daemon::deleteServer()	308
IT_daemon::getServer()	308
IT_daemon::getServer2()	308
IT_daemon::getIOPDetails	309
IT_daemon::getImplementationDetails	309
IT_daemon::killServer()	311
IT_daemon::LaunchStatus	311
IT_daemon::listActiveServers()	311
IT_daemon::listHostsInServer()	312
IT_daemon::listServers()	312
IT_daemon::lookUp()	313
IT_daemon::newDirectory()	313
IT_daemon::newPerMethodServer()	313
IT_daemon::newSharedServer()	313
IT_daemon::newSharedServer2()	314
IT_daemon::newUnSharedServer()	315
IT_daemon::registerPersistentServer()	315
IT_daemon::removeDirRights()	316
IT_daemon::removeInvokeRights()	317
IT_daemon::removeInvokeRightsDir()	317
IT_daemon::removeLaunchRights()	317
IT_daemon::removeLaunchRightsDir()	318
IT_daemon::removeMethod()	318
IT_daemon::removeSharedMarker()	318
IT_daemon::removeUnsharedMarker()	318
IT_daemon::serverDetails	318
IT_daemon::serverExists()	319

## Part VII Appendices

<b>IDL Reference</b>	<b>323</b>
IDL Grammar	323
<b>Naming Service: IDL Definitions</b>	<b>329</b>
<b>System Exceptions</b>	<b>331</b>
System Exceptions Defined by CORBA	331
<b>Index</b>	<b>333</b>



# Preface

The *Orbix Programmer's Reference Java Edition* expands on the information presented in the *Orbix Programmer's Guide Java Edition* and provides a reference for the application programming interface (API) to Orbix Java.

## Audience

The *Orbix Programmer's Reference Java Edition* is designed as a reference for Orbix Java programmers. Before using this reference guide, read the *Orbix Programmer's Guide Java Edition* to learn about writing distributed applications using Orbix Java.

## Organization of this Guide

This guide is divided into the following parts:

### Parts I-IV API Reference

Parts I-IV provide a full reference listing for the following:

- The `org.omg.CORBA` classes and their methods.
- The `IE.Iona.OrbixWeb` classes and their methods.

### Part V "IDL Interface to the Interface Repository"

The Interface Repository is the component of Orbix Java that provides runtime access to IDL definitions. The API to this component is defined in IDL. Part V provides an exhaustive reference for the IDL interface to the Interface Repository.

### Part VI "IDL Interface to the Orbix Java Daemon"

The Orbix Java daemon, `orbixd`, manages several components of Orbix Java, including the Implementation Repository. Part VI provides a complete reference for the IDL interface to the Orbix Java daemon, which allows you to access the daemon functionality in your Orbix Java applications. The Orbix Java daemon acts as an Orbix Java server, with the server name `IT_daemon`.

The Orbix Java daemon `orbixdj` provides a subset of the functionality implemented for `orbixd`. Operations that are not supported by `orbixdj` are clearly indicated.

The IDL interfaces to both the Interface Repository and the Orbix Java daemon are compiled using the IDL to Java mapping defined in the *Orbix Programmer's Guide Java Edition*. The generated types for these interfaces are available in Orbix Java and are scoped by the package `IE.Iona.OrbixWeb.CORBA`.

### Part VII "Appendices"

This part contains the following appendices:

- "IDL Reference" lists the full syntax of the IDL language.
- "Naming Service: IDL Definitions" lists the `CosNaming` module.
- "System Exceptions" lists the Orbix Java system exceptions.

**Note:**

In this guide, abstract methods are listed as throwing an `org.omg.CORBA.SystemException`. This is no longer necessary because `SystemExceptionS` are now unchecked exceptions (extending `java.lang.RuntimeException`). These `SystemExceptionS` are still listed however, to encourage programmers to enclose all statements in `try{...}catch{}` blocks.

## Document Conventions

This guide uses the following typographical conventions:

**Constant width** Constant width (courier font) in normal text represents portions of code and literal names of items such as classes, functions, variables, and data structures. For example, text might refer to the `CORBA::Object` class.

Constant width paragraphs represent code examples or information a system displays on the screen. For example:

```
#include <stdio.h>
```

**Italic** Italic words in normal text represent *emphasis* and *new terms*.

Italic words or characters in code and commands represent variable values you must supply, such as arguments to commands or path names for your particular system. For example:

```
% cd /users/your_name
```

Note: some command examples may use angle brackets to represent variable values you must supply.

This guide may use the following keying conventions:

**No prompt** When a command's format is the same for multiple platforms, no prompt is used.

**%** A percent sign represents the UNIX command shell prompt for a command that does not require root privileges.

**#** A number sign represents the UNIX command shell prompt for a command that requires root privileges.

**>** The notation `>` represents the Windows command prompt.

**...** Horizontal or vertical ellipses in format and syntax descriptions indicate that material has been eliminated to simplify a discussion.



[ ]	Brackets enclose optional items in format and syntax descriptions.
{ }	Braces enclose a list from which you must choose an item in format and syntax descriptions.
	A vertical bar separates items in a list of choices enclosed in { } (braces) in format and syntax descriptions.

## Contacting Micro Focus

Our Web site gives up-to-date details of contact numbers and addresses.

## Further Information and Product Support

Additional technical information or advice is available from several sources.

The product support pages contain a considerable amount of additional information, such as:

- The WebSync service, where you can download fixes and documentation updates.
- The Knowledge Base, a large collection of product tips and workarounds.
- Examples and Utilities, including demos and additional product documentation.

To connect, enter <http://www.microfocus.com> in your browser to go to the Micro Focus home page.

### Note:

Some information may be available only to customers who have maintenance agreements.

If you obtained this product directly from Micro Focus, contact us as described on the Micro Focus Web site, <http://www.microfocus.com>. If you obtained the product from another source, such as an authorized distributor, contact them for help first. If they are unable to help, contact us.

## Information We Need

However you contact us, please try to include the information below, if you have it. The more information you can give, the better Micro Focus SupportLine can help you. But if you don't know all the answers, or you think some are irrelevant to your problem, please give whatever information you have.

- The name and version number of all products that you think might be causing a problem.
- Your computer make and model.
- Your operating system version number and details of any networking software you are using.

- The amount of memory in your computer.
- The relevant page reference or section in the documentation.
- Your serial number. To find out these numbers, look in the subject line and body of your Electronic Product Delivery Notice email that you received from Micro Focus.

## Contact information

Our Web site gives up-to-date details of contact numbers and addresses.

Additional technical information or advice is available from several sources.

The product support pages contain considerable additional information, including the WebSync service, where you can download fixes and documentation updates. To connect, enter <http://www.microfocus.com> in your browser to go to the Micro Focus home page.

If you are a Micro Focus SupportLine customer, please see your SupportLine Handbook for contact information. You can download it from our Web site or order it in printed form from your sales representative. Support from Micro Focus may be available only to customers who have maintenance agreements.

You may want to check these URLs in particular:

- <http://www.microfocus.com/products/corba/orbix/orbix-3.aspx> (trial software download and Micro Focus Community files)
- [https://supportline.microfocus.com/productdoc.aspx\\_](https://supportline.microfocus.com/productdoc.aspx_) (documentation updates and PDFs)

To subscribe to Micro Focus electronic newsletters, use the online form at:

<http://www.microfocus.com/Resources/Newsletters/infocus/newsletter-subscription.asp>

# Part I

## Package org.omg.CORBA

### In this part

This part contains the following:

Class org.omg.CORBA.ARG_IN	page 3
Class org.omg.CORBA.ARG_INOUT	page 4
Class org.omg.CORBA.ARG_OUT	page 5
Class org.omg.CORBA.Bounds	page 6
Class org.omg.CORBA.Bounds	page 6
Class org.omg.CORBA.CompletionStatus	page 7
Class org.omg.CORBA.Context	page 9
Class org.omg.CORBA.ContextList	page 10
Class org.omg.CORBA.CTX_RESTRICT_SCOPE	page 11
Class org.omg.CORBA.Current	page 12
Class org.omg.CORBA.DynamicImplementation	page 13
Class org.omg.CORBA.Environment	page 14
Class org.omg.CORBA.ExceptionList	page 15
Class org.omg.CORBA.NamedValue	page 16
Class org.omg.CORBA.NVList	page 17
Interface org.omg.CORBA.Object	page 18
Class org.omg.CORBA.ORB	page 19

Class org.omg.CORBA.ORBPackage.InvalidName	page 24
Class org.omg.CORBA.portable.InputStream	page 25
Class org.omg.CORBA.portable.OutputStream	page 26
Class org.omg.CORBA.portable.Streamable	page 27
Class org.omg.CORBA.Principal	page 28
Class org.omg.CORBA.Request	page 29
Class org.omg.CORBA.ServerRequest	page 31
Class org.omg.CORBA.SystemException	page 35
Class org.omg.CORBA.TCKind	page 36
Class org.omg.CORBA.TypeCode	page 38
Class org.omg.CORBA.TypeCodePackage.BadKind	page 40
Class org.omg.CORBA.TypeCodePackage.Bounds	page 41
Class org.omg.CORBA.UnknownUserException	page 42
Class org.omg.CORBA.UserException	page 43

# Class org.omg.CORBA.ARG\_IN

## Synopsis

ARG\_IN defines an integer constant for use with the DII when specifying an `in` parameter to a request.

Refer to the *Orbix Programmer's Guide Java Edition* for details on using the DII.

## Java

```
public interface ARG_IN
{
    public static final int value = 1;
}
```

## Notes

CORBA-defined.

## Class org.omg.CORBA.ARG\_INOUT

### Synopsis

ARG\_INOUT defines an integer constant for use with the DII when specifying an `inout` parameter to a request.

Refer to the *Orbix Programmer's Guide Java Edition* for details on using the DII.

### Java

```
public interface ARG_INOUT
{
    public static final int value = 3;
}
```

### Notes

CORBA-defined.

# Class org.omg.CORBA.ARG\_OUT

## Synopsis

ARG\_OUT defines an integer constant for use with the DII when specifying an out parameter to a request.

Refer to the *Orbix Programmer's Guide Java Edition* for details on using the DII.

## Java

```
public interface ARG_OUT
{
    public static final int value = 2;
}
```

## Notes

CORBA-defined.

# Class org.omg.CORBA.Bounds

**Synopsis** Bounds is a CORBA-defined `UserException` that can be thrown to flag an out-of-bounds access on a data structure.

For examples of its use, refer to “[Class org.omg.CORBA.ContextList](#)” and “[Class org.omg.CORBA.ExceptionList](#)”.

**Java**

```
final public class Bounds
extends org.omg.CORBA.UserException {
    public Bounds();
}
```

**Notes** CORBA-defined.

## Bounds()

**Synopsis** `public Bounds();`

**Description** Default constructor.

**Notes** CORBA-defined.



# Class org.omg.CORBA.CompletionStatus

## Synopsis

CompletionStatus contains an integer constant that enumerates the possible operation completion status at the time an Orbix Java system exception is raised.

## Java

```
public final class CompletionStatus {  
  
    // Completion Status constants  
    public static final int _COMPLETED_YES = 0,  
        _COMPLETED_NO = 1,  
        _COMPLETED_MAYBE = 2;  
    public static final CompletionStatus COMPLETED_YES =  
        new CompletionStatus(_COMPLETED_YES);  
    public static final CompletionStatus COMPLETED_NO =  
        new CompletionStatus(_COMPLETED_NO);  
    public static final CompletionStatus COMPLETED_MAYBE =  
        new CompletionStatus(_COMPLETED_MAYBE);  
  
    public int value()  
        throws SystemException;  
    public static final CompletionStatus from_int(int value)  
        throws SystemException;  
}
```

## value()

### Synopsis

```
public int value();
```

### Description

Returns an integer constant representing the operation's completion status.

### Notes

CORBA-defined.

## from\_int()

### Synopsis

```
public static final CompletionStatus from_int(int value);
```

### Description

Creates and returns a `CompletionStatus` object with its constant value set to that of the integer parameter.

### Parameters

value	An integer value of 0, 1 or 2 representing the operation completion status as defined above.
-------	----------------------------------------------------------------------------------------------

### Notes

CORBA-defined.

# Class org.omg.CORBA.Context

## Synopsis

For a description of Context and implementation details for abstract methods listed below refer to [“Class IE.Iona.OrbixWeb.CORBA.Context”](#).

## CORBA

```
pseudo interface Context{
    readonly attribute Identifier context_name;
    readonly attribute Context parent;
    Context create_child(in identifier child_ctx_name);
    void set_one_value(in Identifier propname,
                     in any propvalue);
    void set_values(in NVList values);
    void delete_values(in Identifier propname);
    NVList get_values(in Identifier start_scope,
                    in Flags op_flags,
                    in Identifier pattern);
};
```

## Java

```
package org.omg.CORBA;

public abstract class Context {

    public abstract String context_name()
        throws SystemException;
    public abstract Context parent()
        throws SystemException;
    public abstract Context create_child(String child_ctx_name)
        throws SystemException;
    public abstract void set_one_value(String propname,
                                      Any propvalue)
        throws SystemException;
    public abstract void set_values(NVList values)
        throws SystemException;
    public abstract void delete_values(String propname)
        throws SystemException;

    public abstract NVList get_values(String start_scpe,
                                     int op_flags,
                                     String pattern)
        throws SystemException;
};
```

## Notes

CORBA-defined.

# Class org.omg.CORBA.ContextList

## Synopsis

For a description of ContextList and implementation details for abstract methods listed here see [“Class IE.Iona.OrbixWeb.CORBA.Context”](#).

See also [“Class org.omg.CORBA.Context”](#) on page 9.

## CORBA

```
pseudo interface ContextList {
  readonly attribute unsigned long count;
  void add(in string ctx);
  string item(in unsigned long index) raises (CORBA::Bounds);
  void remove(in unsigned long index) raises (CORBA::Bounds);
};
```

## Java

```
package org.omg.CORBA;

public abstract class ContextList {
  public abstract int count()
    throws SystemException;
  public abstract void add(String ctx)
    throws SystemException;
  public abstract String item(int index)
    throws org.omg.CORBA.Bounds, SystemException;
  public abstract void remove(int index)
    throws org.omg.CORBA.Bounds, SystemException;
}
```

## Notes

CORBA-defined.

# Class org.omg.CORBA.CTX\_RESTRICT\_SCOPE

## Synopsis

Defines an integer constant used to specify that searching should be restricted to a specified scope when retrieving values from a Context object.

See also `get_values()` in ["Class org.omg.CORBA.Context" on page 9](#).

## Java

```
public interface CTX_RESTRICT_SCOPE
{
    public static final int value = 15;
}
```

## Notes

CORBA-defined.

# Class org.omg.CORBA.Current

## Synopsis

For a description of Current, see "[Class IE.Iona.OrbixWeb.CORBA.OrbCurrent](#)".

## CORBA

```
pseudo interface Current {  
}
```

## Java

```
public abstract class Current  
    extends org.omg.CORBA.portable.ObjectImpl {  
}
```

## Notes

CORBA-defined.

# Class org.omg.CORBA.DynamicImplementation

## Synopsis

The `DynamicImplementation` class defines the interface that a dynamic server is expected to implement.

For further information see the `grid_dsi` demonstration.

## Java

```
package org.omg.CORBA;

public abstract class DynamicImplementation
    extends org.omg.CORBA.portable.ObjectImpl {
    public abstract void invoke(org.omg.CORBA.ServerRequest
        request)
        throws SystemException;
}
```

## Notes

CORBA-defined.

# Class org.omg.CORBA.Environment

## Synopsis

For a description of `Environment` and the implementation of the methods listed below, see [“Class IE.Iona.OrbixWeb.CORBA.Environment”](#).

## Java

```
package org.omg.CORBA;

public abstract class Environment {
    void exception(java.lang.Exception except)
        throws SystemException;
    java.lang.Exception exception()
        throws SystemException;
    void clear()
        throws SystemException;
}
```

## Notes

CORBA-defined.



# Class org.omg.CORBA.ExceptionList

## Synopsis

For a description of `ExceptionList` and implementation of abstract methods listed below, see [“Class IE.Iona.OrbixWeb.CORBA.ExceptionList”](#).

See also [“Class org.omg.CORBA.TypeCode”](#).

## CORBA

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

## Java

```
package org.omg.CORBA;

public abstract class ExceptionList {
    public abstract int count()
        throws SystemException;
    public abstract void add(TypeCode exc)
        throws SystemException;
    public abstract TypeCode item(int index)
        throws org.omg.CORBA.Bounds, SystemException;
    public abstract void remove(int index)
        throws org.omg.CORBA.Bounds, SystemException;
}
```

## Notes

CORBA-defined.

# Class org.omg.CORBA.NamedValue

## Synopsis

For a description of NamedValue and implementation of abstract methods listed below, see [“Class IE.Iona.OrbixWeb.CORBA.NamedValue”](#).

## CORBA

```
pseudo interface NamedValue {  
    readonly attribute Identifier name;  
    readonly attribute Any value;  
    readonly attribute Flags flags;  
};
```

## Java

```
package org.omg.CORBA;  
  
public abstract class NamedValue  
{  
    public abstract String name()  
        throws SystemException;  
    public abstract Any value()  
        throws SystemException;  
    public abstract int flags()  
        throws SystemException;  
};
```

## Notes

CORBA-defined.

# Class org.omg.CORBA.NVList

## Synopsis

For a description of `NVList` and implementation of abstract methods listed below, see [“Class IE.Iona.OrbixWeb.CORBA.NVList”](#).

See also [“Class org.omg.CORBA.NamedValue”](#) on page 16.

## CORBA

```
pseudo interface NVList{
    readonly attribute unsigned long count;
    NamedValue add(in Flags flags);
    NamedValue add_item(in Identifier item_name, in Flags
        flags);
    NamedValue add_value(in Identifier item_name,
        in any val,
        in Flags flags);
    NamedValue item(in unsigned long index) raises
        (CORBA::Bounds);
    void remove(in unsigned long index) raises (CORBA::Bounds);
};
```

## Java

```
package org.omg.CORBA;

public abstract class NVList
{
    public abstract int count()
        throws SystemException;
    public abstract NamedValue add(int flags)
        throws SystemException;
    public abstract NamedValue add_item(String item_name,
        int flags)
        throws SystemException;
    public abstract NamedValue add_value(String item_name,
        Any value,int item_flags)
        throws SystemException;
    public abstract NamedValue item(int index)
        throws Bounds, SystemException;
    public abstract void remove(int index)
        throws Bounds, SystemException;
};
```

# Interface org.omg.CORBA.Object

## Synopsis

For the Orbix Java implementation of the Object interface, see [“Interface IE.Iona.OrbixWeb.CORBA.ObjectRef”](#) on page 109.

## Java

```
package org.omg.CORBA;

public interface Object {
    boolean _is_a(String Identifier)
        throws SystemException;
    boolean _is_equivalent(Object that)
        throws SystemException;;
    boolean _non_existent()
        throws SystemException;;
    int _hash(int maximum)
        throws SystemException;;
    org.omg.CORBA.Object _duplicate()
        throws SystemException;;
    void _release()
        throws SystemException;;
    ImplementationDef _get_implementation()
        throws SystemException;;
    InterfaceDef _get_interface()
        throws SystemException;;
    Request _request(String s)
        throws SystemException;;
    Request _create_request(Context ctx,
        String operation, NVList arg_list,
        NamedValue result)
        throws SystemException;;
    Request _create_request(Context ctx,
        String operation, NVList arg_list,
        NamedValue result, ExceptionList exclist,
        ContextList ctxlist)
        throws SystemException;;
}
```

## Notes

CORBA-defined.

# Class org.omg.CORBA.ORB

## Synopsis

For a description of ORB and implementation of abstract methods listed below, see "[Class IE.Iona.OrbixWeb.CORBA.ORB](#)".

## CORBA

```
pseudo interface ORB {
    exception InvalidName {};
    typedef string ObjectId;
    typedef sequence<ObjectId> ObjectIdList;
    ObjectIdList list_initial_services();
    Object resolve_initial_references(in ObjectId object_name)
    raises(InvalidName);
    string object_to_string(in Object obj);
    Object string_to_object(in string str);
    NVList create_list(in long count);
    NVList create_operation_list(in OperationDef oper);
    NamedValue create_named_value(in String name, in Any value,
    in Flags flags);
    ExceptionList create_exception_list();
    ContextList create_context_list();
    Context get_default_context();
    Environment create_environment();
    void send_multiple_requests_oneway(in RequestSeq req);
    void send_multiple_requests_deferred(in RequestSeq req);
    boolean poll_next_response();
    Request get_next_response();

    // typecode creation
    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);
    TypeCode create_enum_tc ( in RepositoryId id,
        in Identifier name,
        in EnumMemberSeq members);
    TypeCode create_alias_tc ( in RepositoryId id,
        in Identifier name,
        in TypeCode original_type);
    TypeCode create_exception_tc ( in RepositoryId id,
        in Identifier name,
        in StructMemberSeq members);
    TypeCode create_interface_tc ( in RepositoryId id,
        in Identifier name);
    TypeCode create_string_tc ( in unsigned long bound);
    TypeCode create_wstring_tc ( in unsigned long bound);
    TypeCode create_sequence_tc ( in unsigned long bound,
        in TypeCode element_type);
    TypeCode create_recursive_sequence_tc ( in unsigned long
        bound, in unsigned long offset);
    TypeCode create_array_tc ( in unsigned long length,
        in TypeCode element_type);
    Current get_current();

    // Additional operations for Java mapping
    TypeCode get_primitive_tc(in TCKind tcKind);
    Any create_any();
    OutputStream create_output_stream();
}
```

## Java

```
        void connect(Object obj);
        void disconnect(Object obj);
    }

package org.omg.CORBA;

public abstract class ORB {
public abstract String[] list_initial_services()
    throws SystemException;
public abstract org.omg.CORBA.Object resolve_initial_references(
    String object_name)
    throws org.omg.CORBA.ORBPackage.InvalidName, SystemException;
public abstract String object_to_string(org.omg.CORBA.Object
    obj)
    throws SystemException;
public abstract org.omg.CORBA.Object string_to_object(String
    str)
    throws SystemException;
public abstract NVList create_list(int count)
    throws SystemException;
public abstract NVList create_operation_list(OperationDef oper)
    throws SystemException;
public abstract NamedValue create_named_value(String name,
    Any value, int flags)
    throws SystemException;
public abstract ExceptionList create_exception_list()
    throws SystemException;
public abstract ContextList create_context_list()
    throws SystemException;
public abstract Context get_default_context()
    throws SystemException;
public abstract Environment create_environment()
    throws SystemException;
public abstract void send_multiple_requests_oneway(Request[]
    req)
    throws SystemException;
public abstract void sent_multiple_requests_deferred(Request[]
    req)
    throws SystemException;
public abstract boolean poll_next_response()
    throws SystemException;
public abstract Request get_next_response()
    throws SystemException;

    // typecode creation
public abstract TypeCode create_struct_tc(String id,
    String name, StructMember[] members)
    throws SystemException;
public abstract TypeCode create_union_tc(String id,
    String name, TypeCode discriminator_type,
    UnionMember[] members)
    throws SystemException;
public abstract TypeCode create_enum_tc(String id,
    String name, EnumMember[] members)
    throws SystemException;
public abstract TypeCode create_alias_tc(String id,
    String name, TypeCode original_type)
    throws SystemException;
public abstract TypeCode create_exception_tc(String id,
    String name, StructMember[] members)
```

```

        throws SystemException;
public abstract TypeCode create_interface_tc(String id,
String name)
        throws SystemException;
public abstract TypeCode create_string_tc(int bound)
        throws SystemException;
public abstract TypeCode create_wstring_tc(int bound)
        throws SystemException;
public abstract TypeCode create_sequence_tc(int bound,
TypeCode element_type)
        throws SystemException;
public abstract TypeCode create_recursive_sequence_tc(int bound,
int offset)
        throws SystemException;
public abstract TypeCode create_array_tc(int length,
TypeCode element_type)
        throws SystemException;
public abstract Current get_current()
        throws SystemException;

// additional methods for IDL/Java mapping
public abstract TypeCode get_primitive_tc(TCKind tcKind)
        throws SystemException;
public abstract Any create_any()
        throws SystemException;
public abstract org.omg.CORBA.portable.OutputStream
create_output_stream()
        throws SystemException;
public abstract void connect( org.omg.CORBA.Object obj)
        throws SystemException;
public abstract void disconnect( org.omg.CORBA.Object obj)
        throws SystemException;

// additional static methods for ORB initialization
public static ORB init(Strings[] args, Properties props)
        throws SystemException;
public static ORB init(Applet app, Properties props)
        throws SystemException;
public static ORB init()
        throws SystemException;
}

```

## Notes

CORBA-defined.

## init()

### Synopsis

```
public static ORB init ()
```

### Description

The parameterless `ORB.init()` method returns a singleton ORB. A singleton ORB has restricted functionality. It is used to create `TypeCode` objects and Pseudo IDL objects. The parameterless `ORB.init()` method should only be called by all clients and servers that need to use an underlying singleton ORB. If called multiple times, this method always returns the same Java object.

### Return Value

A parameterless `ORB.init()` method returns a singleton ORB. This gives access to only a subset of the methods defined in class `org.omg.CORBA.ORB`. Refer to [“Class IE.Iona.OrbixWeb.CORBA.singletonORB”](#) for more details.

### Notes

CORBA-defined.

### See Also

Other `init()` methods.  
[“Class IE.Iona.OrbixWeb.CORBA.ORB”](#)  
[“Class IE.Iona.OrbixWeb.CORBA.singletonORB”](#)

## init()

### Synopsis

```
public static ORB init(Strings[] args,  
                       java.util.Properties props);
```

### Description

All client and server applications in a CORBA system must call this version of `ORB.init()` before they can make use of the underlying fully-functional ORB.

Calling `ORB.init()` with parameters allows command-line arguments and properties to be passed to the ORB runtime. You can use properties supplied in the `props` parameter to customize the behaviour of the ORB. At initialization, the ORB reads whatever system properties it requires and then applies the `props` properties to its internal configuration settings. Both the argument array and properties can be null.

### Parameters

<code>args</code>	An array of <code>Strings</code> containing command-line arguments.
<code>props</code>	A <code>java.util.Properties</code> object containing property settings required for customizing of ORB behaviour.

### Return Value

This version of `ORB.init()` returns a fully-functional `org.omg.CORBA.ORB` object, giving access to the various methods defined in this class.

### Notes

CORBA-defined.

### See Also

Other `init()` methods  
[“Class IE.Iona.OrbixWeb.CORBA.ORB”](#)



## init()

### Synopsis

```
public static ORB init(Applet app, java.util.Properties props);
```

### Description

All client and server applets in a CORBA system must call this version of `ORB.init()` before they can make use of the underlying fully-functional ORB. Here, an applet client can pass a reference to itself, allowing configuration from applet parameters. Properties supplied in the `props` parameter can be used to customize the ORB behaviour. At initialization, the ORB reads whatever system properties it requires and applies the `props` properties to its internal configuration settings. Both the applet and the properties can be null. If you wish to pass a null applet, you must cast it as follows:

```
(Applet)null
```

### Parameters

<code>app</code>	The applet from which the call is made.
<code>props</code>	A <code>java.util.Properties</code> object, containing property settings.

### Return Value

This version of `ORB.init()` returns a fully-functional `org.omg.CORBA.ORB` object, giving access to the various methods defined in this class.

### Notes

CORBA-defined.

### See Also

Other `init()` methods.  
["Class IE.Iona.OrbixWeb.CORBA.ORB"](#)

# Class org.omg.CORBA.ORBPackage.InvalidName

**Synopsis** InvalidName is a CORBA-defined UserException. This can be thrown by the resolve\_initial\_references method of the ORB class to flag an invalid object\_name parameter.

**Java**

```
package org.omg.CORBA.ORBPackage;

public final class InvalidName extends
    org.omg.CORBA.UserException {
    public InvalidName();
}
```

**Notes** CORBA-defined.

## InvalidName()

**Synopsis** public InvalidName();

**Description** Default constructor.

**Notes** CORBA-defined.

# Class org.omg.CORBA.portable.InputStream

Java

```
package org.omg.CORBA.portable;
public abstract class InputStream extends java.io.InputStream
{
    public InputStream(){}
    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();
}
```

Notes

CORBA-defined.

# Class org.omg.CORBA.portable.OutputStream

## Synopsis

See also [“Class org.omg.CORBA.portable.InputStream” on page 25.](#)

## Java

```
package org.omg.CORBA.portable;
public abstract class OutputStream extends java.io.OutputStream
{
    public OutputStream(){}
    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);
}
```

## Notes

CORBA-defined.

# Class org.omg.CORBA.portable.Streamable

## Synopsis

The Streamable interface specifies methods allowing you to read and write complex data types. The Holder classes for handling complex out and inout parameters provide an example of its use.

See also “[Class org.omg.CORBA.portable.InputStream](#)” on [page 25](#), “[Class org.omg.CORBA.portable.OutputStream](#)” on [page 26](#), and the discussion of holders in the chapter “IDL to Java Mapping” in the *Orbix Programmer’s Guide Java Edition*.

## Java

```
package org.omg.CORBA.portable;

public interface Streamable {
    void _read(org.omg.CORBA.portable.InputStream istream)
        throws SystemException;
    void _write(org.omg.CORBA.portable.OutputStream ostream)
        throws SystemException;
    org.omg.CORBA.TypeCode _type()
        throws SystemException;
}
```

## Notes

CORBA-defined.

# Class org.omg.CORBA.Principal

<b>Synopsis</b>	For a description of <code>Principal</code> and implementation of abstract methods defined below, see " <a href="#">Class org.omg.CORBA.Principal</a> ".
<b>CORBA</b>	<pre>pseudo interface Principal {     attribute sequence&lt;octet&gt; name; }</pre>
<b>Java</b>	<pre>public abstract class Principal {     public abstract byte[] name()         throws SystemException;     public abstract void name(byte[] name)         throws SystemException;</pre>
<b>Notes</b>	CORBA-defined.

# Class org.omg.CORBA.Request

## Synopsis

For a description of `Request` and implementation of abstract methods listed below, see [“Class IE.Iona.OrbixWeb.CORBA.Request”](#).

See also the chapter “Dynamic Invocation Interface” in the *Orbix Programmer’s Guide Java Edition* and the demonstration in the `demos/dii_demo` directory of your Orbix Java installation.

## CORBA

```
pseudo interface Request {
    readonly attribute Object target;
    readonly attribute Identifier operation;
    readonly attribute NVList arguments;
    readonly attribute NamedValue result;
    readonly attribute Environment env;
    readonly attribute ExceptionList exceptions;
    readonly attribute ContextList contexts;
    attribute Context ctx;
    any add_in_arg();
    any add_named_in_arg(in string name);
    any add_inout_arg();
    any add_named_inout_arg(in string name);
    any add_out_arg();
    any add_named_out_arg(in string name);
    void set_return_type(in TypeCode tc);
    any return_value();
    void invoke();
    void send_oneway();
    void send_deferred();
    void get_response();
    boolean poll_response();
};
```

## Java

```
package org.omg.CORBA;

public abstract class Request {
    public abstract Object target()
        throws SystemException;
    public abstract String operation()
        throws SystemException;

    public abstract NVList arguments()
        throws SystemException;
    public abstract NamedValue result()
        throws SystemException;
    public abstract Environment env()
        throws SystemException;
    public abstract ExceptionList exceptions()
        throws SystemException;
    public abstract ContextList contexts()
        throws SystemException;
    public abstract Context ctx()
        throws SystemException;
    public abstract void ctx(Context c)
        throws SystemException;
    public abstract Any add_in_arg()
        throws SystemException;
    public abstract Any add_named_in_arg(String name)
        throws SystemException;
```

```
public abstract Any add_inout_arg()
    throws SystemException;
public abstract Any add_named_inout_arg(String name)
    throws SystemException;
public abstract Any add_out_arg()
    throws SystemException;
public abstract Any add_named_out_arg(String name)
    throws SystemException;
public abstract void set_return_type(TypeCode tc)
    throws SystemException;
public abstract Any return_value()
    throws SystemException;
public abstract void invoke()
    throws SystemException;
public abstract void send_oneway()
    throws SystemException;
public abstract void send_deferred()
    throws SystemException;
public abstract void get_response()
    throws SystemException;
public abstract boolean poll_response()
    throws SystemException;
}
```

## Notes

CORBA-defined.



# Class org.omg.CORBA.ServerRequest

## Synopsis

An object adapter (for example, the BOA) dispatches an invocation to a DSI-based object implementation by calling the `invoke()` method on a `org.omg.CORBA.DynamicImplementation` object. The parameter passed to this method is a `ServerRequest` object. This `ServerRequest` object contains the state of an incoming invocation for the DSI. This can be compared to how the `org.omg.CORBA.Request` object is used in the DII approach for clients.

Refer to the demonstration in the `demos/grid_dsi` directory of your Orbix Java installation for an example of how to use the DSI.

## CORBA

```
// PIDL
module CORBA {
    interface ServerRequest {
        Identifier op_name;
        Context ctx();
        void params(in NVList parms);
        void result(in Any res);
        void except(in Any ex);
    };
};
```

## Orbix Java

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

public abstract class ServerRequest {
    public ServerRequest() {}
    public abstract String op_name();
    public abstract Context ctx();
    public abstract void params(NVList parms);
    public abstract void result(Any a);
    public abstract void except(Any a);
};
```

## op\_name()

### Synopsis

```
public abstract String op_name();
```

### Description

Returns the name of the operation being invoked.

### Notes

CORBA-defined.

## ctx()

### Synopsis

```
public abstract Context ctx();
```

### Description

This method returns the `Context` object for this operation invocation. If no `Context` is sent, this method returns `null`.

### Notes

CORBA-defined.

### See Also

["Class IE.Iona.OrbixWeb.CORBA.Context" on page 83](#)

## params()

### Synopsis

```
public abstract void params(NVList parms);
```

### Description

This method marshals the parameters from the incoming `ServerRequest` into the supplied `parms NVList`. You should ensure that the `TypeCode` and flags (`ARG_IN`, `ARG_OUT` or `ARG_INOUT`) of each of the parameters are correct.

The Dynamic Implementation Routine (DIR) must call `params` with `parms` containing `TypeCodes` and flags describing the parameter types expected for the method.

After invoking `params()` you should use the unmarshalled `in` and `inout` values as parameters to the method invocation.

When the invocation completes, you should insert the values for any `out` and `inout` parameters into the `parms NVList` before returning.

If the operation has a return value you must also call `"result()"`.

For example:

```
// import org.omg.CORBA.*;
// Simulate the operations on the grid interface using the DSI.

public void invoke(org.omg.CORBA.ServerRequest _req) {
    String _opName = _req.op_name() ;
    org.omg.CORBA.Any _ret =
        org.omg.CORBA.ORB.init().create_any();
    org.omg.CORBA.NVList _nvl = null;

    if(_opName.equals("set")) {
        _nvl = org.omg.CORBA.ORB.init().create_list(3);

        // Create a new any.
        org.omg.CORBA.Any n = org.omg.CORBA.ORB.init().create_any();

        // Insert the TypeCode(tk_short) into the new Any.
        n.type(org.omg.CORBA.ORB.init().get_primitive_tc
            (org.omg.CORBA.TCKind.tk_short)) ;

        // Insert this Any into the NVList and set flag to IN.
        _nvl.add_value(null, n, org.omg.CORBA.ARG_IN.value);
    }
}
```

```

// Create new Any, set Typecode to short, insert into
NVList.
org.omg.CORBA.Any m =
org.omg.CORBA.ORB.init().create_any();
m.type(org.omg.CORBA.ORB.init().get_primitive_tc
      (org.omg.CORBA.TCKind.tk_short));
_nvl.add_value(null, m, org.omg.CORBA.ARG_IN.value);

// Create new Any, set Typecode to long, insert into
NVList.
org.omg.CORBA.Any value
      = org.omg.CORBA.ORB.init().create_any();
value.type(org.omg.CORBA.ORB.init().get_primitive_tc
      (org.omg.CORBA.TCKind.tk_long));
_nvl.add_value(null, value, org.omg.CORBA.ARG_IN.value);

// Use params() method to marshal data into _nvl.
_req.params(_nvl);

// Get the value of row, col from Any row, col
// and set this element in the array to the value.
m_a[n.extract_short()][m.extract_short()] =
      value.extract_long()
;
return;
}

if(_opName.equals("get")) {
    _ret = org.omg.CORBA.ORB.init().create_any();
    _nvl = org.omg.CORBA.ORB.init().create_list(2);

    org.omg.CORBA.Any n =
    org.omg.CORBA.ORB.init().create_any();
    ntype(org.omg.CORBA.ORB.init().get_primitive_tc
          (org.omg.CORBA.TCKind.tk_short));
    _nvl.add_value(null, n, org.omg.CORBA.ARG_IN.value);

    org.omg.CORBA.Any m =
    org.omg.CORBA.ORB.init().create_any();
    m.type(org.omg.CORBA.ORB.init().get_primitive_tc
          (org.omg.CORBA.TCKind.tk_short));
    _nvl.add_value(null, m, org.omg.CORBA.ARG_IN.value);
    _req.params(_nvl);
    int t = m_a[n.extract_short()][m.extract_short()] ;
    _ret.insert_long(t);
    _req.result(_ret);
    return;
}

if (_opName.equals("_get_height")) {
    _ret = org.omg.CORBA.ORB.init().create_any();
    _req.params(_nvl);
    _ret.insert_short(m_height);
    _req.result(_ret);
    return;
}

if (_opName.equals("_get_width")) {
    _ret = org.omg.CORBA.ORB.init().create_any();
    _req.params(_nvl);

```

```

        _ret.insert_short(m_width);
        _req.result(_ret);
        return;
    }
}

```

## Parameters

**parms** A `NVList` describing the parameter types for the operation in the order in which they appear in the IDL specification (left to right).

## Notes

CORBA-defined.

## See Also

["Class IE.Iona.OrbixWeb.CORBA.NVList"](#)  
[grid\\_dsi](#) demonstration

## result()

### Synopsis

```
void result(in Any res);
```

### Description

The `result()` method specifies the return value for the call. If the operation has a `void` result type, `result()` should be set to an `Any` whose type is `_tc_void`.

## Parameters

**res** An `Any` containing the return value and type for the operation.

## Notes

CORBA-defined.

## See Also

["Class IE.Iona.OrbixWeb.CORBA.Any"](#)

## except()

### Synopsis

```
void except(in Any ex);
```

### Description

The DIR can call `except()` at any time to return an exception to the client. The `Any` passed to `except()` must contain either a system exception or one of the user exceptions specified in the `raises` expression of the invoked operation's IDL definition.

## Parameters

**ex** An `Any` containing the exception to be returned to the client.

## Notes

CORBA-defined.

## See Also

["System Exceptions"](#)  
["Class IE.Iona.OrbixWeb.CORBA.Any"](#)  
["Class org.omg.CORBA.SystemException"](#)

# Class org.omg.CORBA.SystemException

## Synopsis

The Orbix Java system exceptions are organized into a class hierarchy: each system exception is a derived class of `SystemException` (which is in turn a derived class of `java.lang.RuntimeException`). This allows all system exceptions to be caught in one single Java catch clause.

`SystemException` provides access to the `minor` exception code and to the `CompletionStatus` of the associated operation.

`SystemException` itself is an abstract class; only derived exception classes may be instantiated.

See also "[Class org.omg.CORBA.CompletionStatus](#)" on page 7 and refer to the chapter "Exception Handling" in the *Orbix Programmer's Guide Java Edition*.

## Java

```
package org.omg.CORBA;

abstract public class SystemException
    extends java.lang.RuntimeException {
    public int minor;
    public CompletionStatus completed;
}
```

## Notes

CORBA-defined.

# Class org.omg.CORBA.TCKind

## Synopsis

Class TCKind defines TypeCode kind constant values required by class TypeCode.

Refer to the chapter “IDL to Java Mapping” in the *Orbix Programmer’s Guide Java Edition* for a discussion of the mapping from the IDL type `enum` to Java.

## Orbix Java

```
// Java
```

```
public final class TCKind {
    public static final int _tk_null = 0;
    public static final int _tk_void = 1;
    public static final int _tk_short = 2;
    public static final int _tk_long = 3;
    public static final int _tk_ushort = 4;
    public static final int _tk_ulong = 5;
    public static final int _tk_float = 6;
    public static final int _tk_double = 7;
    public static final int _tk_boolean = 8;
    public static final int _tk_char = 9;
    public static final int _tk_octet = 10;
    public static final int _tk_any = 11;
    public static final int _tk_TypeCode = 12;
    public static final int _tk_Principal = 13;
    public static final int _tk_objref = 14;
    public static final int _tk_struct = 15;
    public static final int _tk_union = 16;
    public static final int _tk_enum = 17;
    public static final int _tk_string = 18;
    public static final int _tk_sequence = 19;
    public static final int _tk_array = 20;
    public static final int _tk_alias = 21;
    public static final int _tk_except = 22;
    public static final int _tk_longlong = 23;
    public static final int _tk_ulonglong = 24;
    public static final int _tk_longdouble = 25;
    public static final int _tk_wchar = 26;
    public static final int _tk_wstring = 27;
    public static final int _tk_fixed = 28;
    public static final TCKind tk_null = new TCKind(0);
    public static final TCKind tk_void = new TCKind(1);
    public static final TCKind tk_short = new TCKind(2);
    public static final TCKind tk_long = new TCKind(3);
    public static final TCKind tk_ushort = new TCKind(4);
    public static final TCKind tk_ulong = new TCKind(5);
    public static final TCKind tk_float = new TCKind(6);
    public static final TCKind tk_double = new TCKind(7);
    public static final TCKind tk_boolean = new TCKind(8);
    public static final TCKind tk_char = new TCKind(9);
    public static final TCKind tk_octet = new TCKind(10);
    public static final TCKind tk_any = new TCKind(11);
    public static final TCKind tk_TypeCode = new TCKind(12);
    public static final TCKind tk_Principal = new TCKind(13);
    public static final TCKind tk_objref = new TCKind(14);
    public static final TCKind tk_struct = new TCKind(15);
    public static final TCKind tk_union = new TCKind(16);
    public static final TCKind tk_enum = new TCKind(17);
    public static final TCKind tk_string = new TCKind(18);
}
```

```

public static final TCKind tk_sequence = new TCKind(19);
public static final TCKind tk_array = new TCKind(20);
public static final TCKind tk_alias = new TCKind(21);
public static final TCKind tk_except = new TCKind(22);
public static final TCKind tk_longlong = new TCKind(23);
public static final TCKind tk_ulonglong = new TCKind(24);
public static final TCKind tk_longdouble = new TCKind(25);
public static final TCKind tk_wchar = new TCKind(26);
public static final TCKind tk_wstring = new TCKind(27);
public static final TCKind tk_fixed = new TCKind(28);

public int value() throws org.omg.CORBA.SystemException;
public static TCKind from_int(int i)
    throws org.omg.CORBA.SystemException;
}

```

**Notes** CORBA-defined.

**See Also** ["Class org.omg.CORBA.TypeCode"](#)  
["Class IE.Iona.OrbixWeb.CORBA.TypeCode"](#)

## value()

### Synopsis

```
public int value() throws org.omg.CORBA.SystemException;
```

### Description

Returns the integer value associated with a TCKind object.

### Notes

Orbix-specific.

## from\_int()

### Synopsis

```
public static TCKind from_int(int i)
    throws org.omg.CORBA.SystemException;
```

### Description

Returns a reference to a TCKind object given an integer value.

### Parameters

i	An integer value identifying a TCKind object.
---	-----------------------------------------------

### Return Value

TCKind	The TCKind object associated with the value specified by i.
--------	-------------------------------------------------------------

### Notes

Orbix-specific.

# Class org.omg.CORBA.TypeCode

## Synopsis

For a description of `TypeCode` and implementation of abstract methods listed for this class, see [“Class IE.Iona.OrbixWeb.CORBA.TypeCode”](#).

## CORBA

```
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 };
```

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

    // for all TypeCode kinds
    boolean equal(in TypeCode tc);
    TCKind kind();

    // for objref, struct, union, enum, alias, and except
    RepositoryID id() raises (BadKind);
    RepositoryID name() raises (BadKind);

    // for struct, union, enum, and except
    unsigned long member_count() raises (BadKind);
    Identifier member_name(in unsigned long index)
        raises (BadKind, Bounds);

    // for struct, union, and except
    TypeCode member_type(in unsigned long index)
        raises (BadKind, Bounds);

    // for union
    any member_label(in unsigned long index)
        raises (BadKind, Bounds);
    TypeCode discriminator_type() raises (BadKind);
    long default_index() raises (BadKind);

    // for string, sequence, and array
    unsigned long length() raises (BadKind);
    TypeCode content_type() raises (BadKind);
}
```

## Orbix Java

```
// Java

public abstract class TypeCode {
    public abstract boolean equal(TypeCode tc)
        throws org.omg.CORBA.SystemException;
    public abstract TCKind kind()
        throws org.omg.CORBA.SystemException;

    // for struct, union, enum, and except
    public abstract int member_count()
        throws org.omg.CORBA.SystemException, BadKind;
    public abstract String id()
        throws org.omg.CORBA.SystemException, BadKind;
    public abstract String name()
```



```

        throws org.omg.CORBA.SystemException, BadKind;
public abstract String member_name(int index)
        throws org.omg.CORBA.SystemException, BadKind, Bounds;

// for struct, union, and except
public abstract TypeCode member_type(int index)
        throws org.omg.CORBA.SystemException, BadKind, Bounds;

// for union
public abstract Any member_label(int index)
        throws org.omg.CORBA.SystemException, BadKind, Bounds;
public abstract TypeCode discriminator_type()
        throws org.omg.CORBA.SystemException, BadKind;
public abstract int default_index()
        throws org.omg.CORBA.SystemException, BadKind;

// for string, sequence, and array
public abstract int length()
        throws org.omg.CORBA.SystemException, BadKind;
public abstract TypeCode content_type()
        throws org.omg.CORBA.SystemException, BadKind;
public java.lang.Object clone()
        throws org.omg.CORBA.SystemException;
};

```

**Notes**

CORBA-defined.

**See Also**

["Class org.omg.CORBA.TypeCode"](#).

# Class org.omg.CORBA.TypeCodePackage.BadKind

## Synopsis

BadKind is a CORBA defined `UserException` that can be thrown by a number of the methods in `TypeCode`. It flags the use of a method that is not available on the `TypeCode` in question.

For more information, see "[Class org.omg.CORBA.TypeCode](#)".

## Java

```
package org.omg.CORBA.TypeCodePackage;  
  
public final class BadKind extends org.omg.CORBA.UserException {  
    public BadKind();  
}
```

## Notes

CORBA-defined.

## BadKind()

### Synopsis

```
public BadKind();
```

### Description

Default constructor.

### Notes

CORBA-defined.

# Class org.omg.CORBA.TypeCodePackage.Bounds

## Synopsis

Bounds is a CORBA defined `UserException` that can be thrown by a number of the methods in `TypeCode`. It flags an attempt to access a member with an invalid index.

For more information, see "[Class org.omg.CORBA.TypeCode](#)".

## Java

```
package org.omg.CORBA.TypeCodePackage;  
  
public final class Bounds extends org.omg.CORBA.UserException {  
    public Bounds();  
}
```

## Notes

CORBA-defined.

## Bounds()

### Synopsis

```
public Bounds();
```

### Description

Default constructor.

### Notes

CORBA-defined.

# Class org.omg.CORBA.UnknownUserException

**Synopsis** UnknownUserException is a derived class of UserException holding a member of type Any, allowing it to contain an unspecified, or unknown, UserException.

See also "[Class org.omg.CORBA.UserException](#)".

## Java

```
package org.omg.CORBA;

public class UnknownUserException extends UserException {
    public Any except;
    public UnknownUserException() {
        super();
    }
    public UnknownUserException(Any a) {
        super();
        except = a;
    }
}
```

**Notes** CORBA-defined.

## UnknownUserException()

**Synopsis**

```
public UnknownUserException();
public UnknownUserException(Any a);
```

**Description** Constructor

## Parameters

a An instance of Any.

**Notes** CORBA-defined.

# Class org.omg.CORBA.UserException

## Synopsis

Orbix Java implementations of user exceptions are organized into a class hierarchy. Each user exception is mapped to a derived class of `UserException`, which is a derived class of `java.lang.Exception`. This allows all user exceptions to be caught in one single Java catch clause.

See also "[Class org.omg.CORBA.SystemException](#)" and "[Class org.omg.CORBA.UnknownUserException](#)".

## Java

```
package org.omg.CORBA;
```

```
public abstract class UserException extends java.lang.Exception
{
    public UserException();
}
```

## Notes

CORBA-defined.



# Part II

## Package IE.Iona.OrbixWeb.CORBA

### In this part

This part contains the following:

Class IE.Iona.OrbixWeb.CORBA.Any	page 47
Interface IE.Iona.OrbixWeb.CORBA.BOA	page 59
Class IE.Iona.OrbixWeb.CORBA.Context	page 83
Class IE.Iona.OrbixWeb.CORBA.ContextList	page 90
Class IE.Iona.OrbixWeb.CORBA.ContextIterator	page 92
Class IE.Iona.OrbixWeb.CORBA.Environment	page 94
Class IE.Iona.OrbixWeb.CORBA.ExceptionList	page 95
Class IE.Iona.OrbixWeb.CORBA.InitService	page 97
Class IE.Iona.OrbixWeb.CORBA.NamedValue	page 98
Class IE.Iona.OrbixWeb.CORBA.NVList	page 102
Class IE.Iona.OrbixWeb.CORBA.NVListIterator	page 107
Interface IE.Iona.OrbixWeb.CORBA.ObjectRef	page 109
Class IE.Iona.OrbixWeb.CORBA.ORB	page 120
Class IE.Iona.OrbixWeb.CORBA.OrbCurrent	page 159
Class IE.Iona.OrbixWeb.CORBA.Principal	page 164
Class IE.Iona.OrbixWeb.CORBA.Request	page 167
Class IE.Iona.OrbixWeb.CORBA.singletonORB	page 181
Class IE.Iona.OrbixWeb.CORBA.TypeCode	page 190





# Class `org.omg.CORBA.Any`

## Synopsis

The class `Any` implements the IDL basic type `any`, which allows the specification of values that can express an arbitrary IDL type. This allows a program to handle values whose types are not known at compile time. The IDL type `any` is most often used in code that uses the Interface Repository or the Dynamic Invocation Interface (DII) or with CORBA services in general.

Consider the following interface:

```
// IDL
interface Example {
    void op(in any value);
};
```

A client can construct an `any` to contain an arbitrary type of value and then pass this in a call to operation `op()`. A process receiving an `any` must determine what type of value it stores and then extract the value (using the `TypeCode`). Refer to the chapter "Type `any`" in the *Orbix Programmer's Guide Java Edition*.

## Orbix Java

```
public class Any extends org.omg.CORBA.Any implements
    java.lang.Cloneable {
```

```
    //constructors
    public Any()
        throws org.omg.CORBA.SystemException;
    public Any(Any a)
        throws org.omg.CORBA.SystemException;
    public Any(int protocol)
        throws org.omg.CORBA.SystemException;
    public Any(String s)
        throws org.omg.CORBA.SystemException;
    public void copy(Any a)
        throws org.omg.CORBA.SystemException;
    public java.lang.Object clone()
        throws org.omg.CORBA.SystemException;
    public boolean equal(org.omg.CORBA.Any _obj)
        throws org.omg.CORBA.SystemException;
    public String toString()
        throws org.omg.CORBA.SystemException;
    public void fromString(String s)
        throws org.omg.CORBA.SystemException;
    public void reset()
        throws org.omg.CORBA.SystemException;
    public void type(org.omg.CORBA.TypeCode tc)
        throws org.omg.CORBA.SystemException;

    //insertion methods
    public void insert_short(int l)
        throws org.omg.CORBA.SystemException;
    public void insert_long(int l)
        throws org.omg.CORBA.SystemException;
    public void insert_longlong(long l)
        throws org.omg.CORBA.SystemException;
    public void insert_ulonglong(long l)
        throws org.omg.CORBA.SystemException;
    public void insert_ushort(short s)
        throws org.omg.CORBA.SystemException;
    public void insert_ulong(int l)
```

```

        throws org.omg.CORBA.SystemException;
public void insert_float(float f)
throws org.omg.CORBA.SystemException;
    public void insert_double(double d)
        throws org.omg.CORBA.SystemException;
public void insert_char(char c)
    throws org.omg.CORBA.SystemException;
public void insert_octet(byte b)
    throws org.omg.CORBA.SystemException;
public void insert_string(String s)
    throws org.omg.CORBA.SystemException;
public void insert_boolean(boolean b)
    throws org.omg.CORBA.SystemException;
public void insert_any(org.omg.CORBA.Any a)
    throws org.omg.CORBA.SystemException;
public void insert_TypeCode(org.omg.CORBA.TypeCode tc)
    throws org.omg.CORBA.SystemException;
public void insert_Object(org.omg.CORBA.Object oref)
    throws org.omg.CORBA.SystemException;
public void insert_Object(org.omg.CORBA.Object oref,
    org.omg.CORBA.TypeCode tc)
    throws org.omg.CORBA.SystemException;
public void insert_Streamable(org.omg.CORBA.portable.
    Streamable s)
    throws org.omg.CORBA.SystemException;
public void insert_Principal(org.omg.CORBA.Principal p)
    throws org.omg.CORBA.SystemException;
public void insert_wchar(char c)
    throws org.omg.CORBA.SystemException;
public void insert_wstring(String s)
    throws org.omg.CORBA.SystemException;

//input / output stream methods
public void read_value(org.omg.CORBA.portable.InputStream is,
    org.omg.CORBA.TypeCode t)
    throws MARSHAL,org.omg.CORBA.SystemException;
public void write_value(org.omg.CORBA.portable.OutputStream os)
    throws org.omg.CORBA.SystemException;
public org.omg.CORBA.portable.OutputStream
    create_output_stream()
    throws org.omg.CORBA.SystemException;
public org.omg.CORBA.portable.InputStream create_input_stream()
    throws org.omg.CORBA.SystemException;

//extraction methods
public short extract_short()
    throws org.omg.CORBA.SystemException;
public int extract_long()
    throws org.omg.CORBA.SystemException;
public long extract_ulonglong()
    throws org.omg.CORBA.SystemException;
public long extract_longlong()
    throws org.omg.CORBA.SystemException;
public char extract_wchar()
    throws org.omg.CORBA.SystemException;
public String extract_wstring()
    throws org.omg.CORBA.SystemException;
public short extract_ushort()
    throws org.omg.CORBA.SystemException;
public org.omg.CORBA.Principal extract_Principal()

```

```

        throws org.omg.CORBA.SystemException;
public int extract_ulong()
        throws org.omg.CORBA.SystemException;
public float extract_float()
        throws org.omg.CORBA.SystemException;
public double extract_double()
        throws org.omg.CORBA.SystemException;
public char extract_char()
        throws org.omg.CORBA.SystemException;
public byte extract_octet()
        throws org.omg.CORBA.SystemException;
public String extract_string()
        throws org.omg.CORBA.SystemException;
public boolean extract_boolean()
        throws org.omg.CORBA.SystemException;
public org.omg.CORBA.Any extract_any()
        throws org.omg.CORBA.SystemException;
public org.omg.CORBA.TypeCode extract_TypeCode()
        throws org.omg.CORBA.SystemException;
public org.omg.CORBA.Object extract_Object()
        throws org.omg.CORBA.SystemException;
public void extract_Streamable(
        org.omg.CORBA.portable.Streamable s)
        throws org.omg.CORBA.SystemException;

// Data accessor methods
public org.omg.CORBA.TypeCode type()
        throws org.omg.CORBA.SystemException;
public boolean containsType(TypeCode t)
        throws org.omg.CORBA.SystemException;
}

```

## Any()

### Synopsis

```

public Any()
        throws org.omg.CORBA.SystemException;

```

### Description

Create a new Any with Typecode set to `_tc_null` and value set to null. This allows the Any to be populated using the insertion and extraction methods, for example:

```

org.omg.CORBA.Any a = new IE.Iona.OrbixWeb.CORBA.Any()
// could also have used
// a = org.omg.CORBA.Orb.init().create_any();

a.insertLong(10);

```

For user-defined types the `insert()` method can be used. For example given the following IDL:

```

// IDL
struct details {
    string address;
    string name;
}

```

you can use the following java code to insert the struct into an any:

```

// Java code
details d = // get details struct from somewhere
org.omg.CORBA.Any a = detailsHelper.insert(d);

```

Similarly, objects can be extracted using the Any extraction method or the extraction methods. Refer to the chapter "IDL to Java Mapping" in the *Orbix Programmer's Guide Java Edition* for more details.

You can also use "Class [org.omg.CORBA.portable.InputStream](#)" and "Class [org.omg.CORBA.portable.OutputStream](#)" objects to insert and extract objects from type Any.

**Notes**

Orbix-specific

**See Also**

"Class [IE.Iona.OrbixWeb.CORBA.TypeCode](#)"  
"Class [org.omg.CORBA.portable.InputStream](#)"  
"Class [org.omg.CORBA.portable.OutputStream](#)"  
The chapter "IDL to Java Mapping" in the *Orbix Programmer's Guide Java Edition*.

## Any()

**Synopsis**

```
public Any(Any a)
    throws org.omg.CORBA.SystemException;
```

**Description**

The Any copy constructor, it creates a new Any with the same TypeCode and value as the Any passed in.

**Parameters**

a                    The Any to copy.

**Notes**

Orbix-specific

**See Also**

"Class [IE.Iona.OrbixWeb.CORBA.TypeCode](#)"

## Any()

**Synopsis**

```
public Any(int protocol)
    throws org.omg.CORBA.SystemException;
```

**Description**

Create a new Any for the given protocol type. This protocol can be one of the following:

- [IE.Iona.orbixWeb.\\_CORBA.IT\\_INTEROPERABLE\\_OR\\_KIND](#)
- [IE.Iona.orbixWeb.\\_CORBA.IT\\_ORBIX\\_OR\\_KIND](#)

This parameter indicates the marshalling protocol to be associated with the Any object.

**Parameters**

protocol            The protocol type (IIOP or Orbix Protocol) for this Any.

**Notes**

Orbix-specific

**See Also**

"Class [IE.Iona.OrbixWeb.CORBA.TypeCode](#)"  
"Class [IE.Iona.OrbixWeb.\\_CORBA](#)"

## Any()

### Synopsis

```
public Any(String s)
```

### Description

Create an `Any` from the `String` passed in.

### Parameters

`s`            `Any` data in persistent stringified form, this form of an `Any` can be generated by calling the `toString()` method.

### Notes

Orbix-specific

### See Also

["Class IE.Iona.OrbixWeb.CORBA.TypeCode"](#)  
["toString\(\)"](#)  
["fromString\(\)"](#)

## clone()

### Synopsis

```
public java.lang.Object clone()  
    throws org.omg.CORBA.SystemException;
```

### Description

This method creates a new `Any` containing a copy of the same data as the original.

### Return Value

`java.lang.Object`        A clone of the original `Any`.

### Exceptions

A `java.lang.Error` exception is thrown if this object does not support the `java.lang.Cloneable` interface.

### Notes

Orbix-specific

### See Also

["Class IE.Iona.OrbixWeb.CORBA.TypeCode"](#)

## containsType()

### Synopsis

```
public boolean containsType(TypeCode t)  
    throws org.omg.CORBA.SystemException;
```

### Description

This method returns `true` if the `Object` within the `Any` is of the same `Type` as the `TypeCode` passed in.

### Parameters

`t`            The `TypeCode` to check for.

### Return Value

`boolean`        This value is `true` if the `TypeCode` passed in and the `TypeCode` contained in the `Any` are the same

### Notes

Orbix-specific

### See Also

["Class IE.Iona.OrbixWeb.CORBA.TypeCode"](#)  
["Class org.omg.CORBA.portable.InputStream"](#)  
["insert\(\)"](#)

## copy()

### Synopsis

```
public void copy(Any a)
    throws org.omg.CORBA.SystemException;
```

### Description

Copy data from another *Any* into this one

### Parameters

a                    The *Any* to be copied.

### Notes

Orbix-specific

### See Also

["Class IE.Iona.OrbixWeb.CORBA.TypeCode"](#)

## create\_output\_stream()

### Synopsis

```
public org.omg.CORBA.portable.OutputStream
    create_output_stream() throws
    org.omg.CORBA.SystemException;
```

### Description

This method creates an `org.omg.CORBA.portable.OutputStream` object for this *Any*. This object allows the *Any* to be populated by calling the `write()` methods declared on `OutputStream` instead of using the `insert()` methods of the *Any*. It also allows us to access certain objects such as *Any*, *Request* and so on, in a uniform manner.

### Return Value

`OutputStream`            The `OutputStream` representing the *Any*

### Notes

CORBA-defined.

### See Also

["Class IE.Iona.OrbixWeb.CORBA.TypeCode"](#)  
["Class org.omg.CORBA.portable.OutputStream"](#)  
["insert\(\)"](#)

## create\_input\_stream()

### Synopsis

```
public org.omg.CORBA.portable.InputStream create_input_stream()
    throws org.omg.CORBA.SystemException;
```

### Description

This method creates an `org.omg.CORBA.portable.InputStream` object for this *Any*, so that the data contained within the *Any* can be accessed through the `read()` methods defined on `InputStream` rather than the `extract()` methods defined on *Any*.

### Return Value

`InputStream`            The `InputStream` representing the *Any*.

### Notes

CORBA-defined.

### See Also

["Class IE.Iona.OrbixWeb.CORBA.TypeCode"](#)  
["Class org.omg.CORBA.portable.InputStream"](#)  
["extract\(\)"](#)

## equal()

### Synopsis

```
public boolean equal(org.omg.CORBA.Any _obj)
```

### Description

This method compares the type and value of this `Any` with that of the `Any` passed in as a parameter.

### Parameters

`_obj`            The `Any` to compare against.

### Return Value

`boolean`            Set to `true` if the `Anys` are equal.

### Notes

CORBA-defined.

### See Also

["Class IE.Iona.OrbixWeb.CORBA.TypeCode"](#)

## extract()

### Synopsis

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

## Description

These methods are used to extract the indicated type from the `Any`. You can determine the type of the `Any` using the `org.omg.CORBA.Any` method. You can extract the value using the appropriate extraction method. To extract a user defined type, you can also use the Helper classes, for example:

```
org.omg.CORBA.Any a = // get the any from somewhere
                    // for example, through the DII,
                    // from one of the CORBA services.

Object val;

switch(a.type().kind()){
  case org.omg.CORBA.TCKind._tc_short:
    val = new Short(a.extract_short());
    break;

  //etc. for other basic types

  default :
    if(a.type().equal(AStructHelper.type()){
      val = AStructHelper.extract(a);
    }
    // else some other user defined types
    break;
};
```

You can also obtain the same kind of result by using class `org.omg.CORBA.portable.InputStream`.

## Notes

CORBA-defined.

## See Also

["Class IE.Iona.OrbixWeb.CORBA.TypeCode"](#)

["Class org.omg.CORBA.portable.InputStream"](#)

The chapter "IDL to Java Mapping" in the ***Orbix Programmer's Guide Java Edition***

## fromString()

### Synopsis

```
public void fromString(String s)
```

### Description

Using the `toString()` method, you can save the state of an `Any` to a `String`. This method allows the state of an `Any` to be regenerated from one of these strings.

### Parameters

`s`                    The `String` containing the representation of an `Any`.

### Exceptions

An `org.omg.CORBA.BAD_PARAM` exception is thrown if the string representation is incorrect.

### Notes

Orbix-specific

### See Also

["Class IE.Iona.OrbixWeb.CORBA.TypeCode"](#)

["toString\(\)"](#)



# insert()

## Synopsis

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

## Description

Insert a value of the indicated type into the `Any`.

Previous values held in the `Any` are discarded and each insertion method takes a copy of the value inserted.

You can use the `<name>Helper` class to insert a user-defined type. For example, given the following IDL:

```
//IDL
struct AStruct{
    string str;
    float number;
};
```

Use the `insert()` method generated on the `AStructHelper` class:

```
//Java
org.omg.CORBA.Any a = new IE.Iona.OrbixWeb.CORBA.Any();

Astruct s = new Astruct("String",1.0f);
try {
    AstructHelper.insert(a,s);
}
catch(org.omg.CORBA.SystemException){
    //do something here
}
```

The same result can be achieved using the ["Class org.omg.CORBA.portable.OutputStream"](#).

## Parameters

first parameter    The actual value to insert into the Any.  
tc                    The `TypeCode` of the value being inserted.

## Notes

CORBA-defined.

## See Also

["Class IE.Iona.OrbixWeb.CORBA.TypeCode"](#) on page 190  
["Class org.omg.CORBA.portable.OutputStream"](#) on page 26  
The chapter "IDL to Java Mapping" in the *Orbix Programmer's Guide Java Edition*.

## read\_value()

### Synopsis

```
public void read_value(org.omg.CORBA.portable.InputStream is,
                      org.omg.CORBA.TypeCode t)
    throws org.omg.CORBA.MARSHAL,
           org.omg.CORBA.SystemException;
```

### Description

This method is used to read an object from a ["Class org.omg.CORBA.portable.InputStream"](#) into the current Any.

### Parameters

is                    The `InputStream` to read the data from.  
t                     The `TypeCode` of the object to be read from the stream.

### Exceptions

An `org.omg.CORBA.MARSHAL` exception is thrown if the data in the `InputStream` is not an object of the type specified by the passed in `TypeCode`.

### Notes

CORBA-defined.

### See Also

["Class IE.Iona.OrbixWeb.CORBA.TypeCode"](#)

## reset()

### Synopsis

```
public void reset()
```

### Description

Sets the `TypeCode` to `tk_null` and the value to `null`. Use this method when you want to reuse an `Any`.

### Notes

Orbix-specific.

### See Also

["Class IE.Iona.OrbixWeb.CORBA.TypeCode"](#).

## toString()

### Synopsis

```
public String toString()
```

### Description

Convert the `Any` to a string. `Any`s can be freely converted to and from strings.

Format:

Orbix: "<typecode string> HDR:<marshalled value as hex string>"

IIOP: "<typecode string> CDR:<marshalled value as hex string>"

You can use the `fromString()` method to perform the reverse process and convert a string back to an `Any`.

### Return Value

`String`                      A string representation of the `Any`

### Notes

Orbix-specific

### See Also

["Class IE.Iona.OrbixWeb.CORBA.TypeCode"](#)  
["fromString\(\)"](#)

## type()

### Synopsis

```
public org.omg.CORBA.TypeCode type()  
    throws org.omg.CORBA.SystemException;
```

### Description

This method returns the `TypeCode` of the `Object` encapsulated within the `Any`.

### Return Value

`TypeCode`                      The `TypeCode` of the `Object` within the `Any`.

### Notes

CORBA-defined.

### See Also

["Class IE.Iona.OrbixWeb.CORBA.TypeCode"](#)

## write\_value()

### Synopsis

```
public void write_value(org.omg.CORBA.portable.OutputStream os)
    throws org.omg.CORBA.SystemException;
```

### Description

This method writes the object contained within the `Any` into the specified `OutputStream`.

### Parameters

`os`                    The `OutputStream` to write the data to.

### Notes

CORBA-defined.

### See Also

["Class IE.Iona.OrbixWeb.CORBA.TypeCode"](#)  
["Class org.omg.CORBA.portable.OutputStream"](#)

# Interface IE.Iona.OrbixWeb.CORBA.BOA

## Synopsis

The server-side element of the CORBA IDL to Java Language mapping specifies a minimal Java API based upon the OMG CORBA BOA pseudo-interface. BOA is an abbreviation of Basic Object Adapter.

In earlier versions of Orbix Java the BOA is implemented as a class extending the `IE.Iona.OrbixWeb.CORBA.ORB` class. By default, any ORB object created is a BOA and thus supported the full server functionality.

The BOA is now an interface which the ORB and the new `BOAImpl` class implement. The ORB delegates the BOA operations to an instance of `BOAImpl`. This class structure cleanly separates the BOA implementation from the ORB implementation while maintaining backward compatibility. This separation means that the BOA implementation class need only be instantiated dynamically when server-side operations are actually used. This prevents unnecessary resource consumption by clients. In addition, it dispenses with static configuration of BOA support to minimize download time of applets that do not require the BOA. All BOA operations should be called on an instance of the ORB; for example `_CORBA.Orbix`.

Interface BOA provides methods that control Orbix Java from the server. These include methods to:

- Activate and deactivate servers.
- Activate and deactivate objects.
- Create and interpret object references.

The methods of this class are invoked through the ORB on the server.

## CORBA

```
// Pseudo IDL

module CORBA {
    interface InterfaceDef; // from Interface Repository // PIDL
    interface ImplementationDef; // from Interface Repository
    interface Object; // an object reference
    interface Principal; // for the authentication service
    typedef sequence <octet, 1024> ReferenceData;

    interface BOA {
        void impl_is_ready (in ImplementationDef impl);
        void deactivate_impl (in ImplementationDef impl);
        void obj_is_ready (in Object obj, in ImplementationDef
            impl);
        void deactivate_obj (in Object obj);

        void change_implementation (
            in Object obj,
            in ImplementationDef impl
        );

        Principal get_principal (
            in Object obj,
            in Environment ev
        );
    };
};
```

```

void dispose (in Object obj);

Object create (
    in ReferenceData id,
    in InterfaceDef intf,
    in ImplementationDef impl
);
ReferenceData get_id (in Object obj);
};
};

public interface BOA {

    // General methods
    public java.lang.String toString();

    public void finalize();
    public void shutdown();

    // Event processing methods
    public int processNextEvent(int timeOut)
        throws org.omg.CORBA.SystemException;

    public int processNextEvent()
        throws org.omg.CORBA.SystemException;

    public int processEvents(int timeOut)
        throws org.omg.CORBA.SystemException;

    public int processEvents()
        throws org.omg.CORBA.SystemException;

    public boolean isEventPending()
        throws org.omg.CORBA.SystemException;

    public void impl_is_ready(java.lang.String serverName,
        int timeOut)
        throws org.omg.CORBA.SystemException;

    public void impl_is_ready(int timeOut)
        throws org.omg.CORBA.SystemException;

    public void impl_is_ready(java.lang.String serverName)
        throws org.omg.CORBA.SystemException;

    public void impl_is_ready()
        throws org.omg.CORBA.SystemException;

    public void deactivate_impl(java.lang.String impl)
        throws org.omg.CORBA.SystemException;

    public void connect(org.omg.CORBA.Object obj)
        throws org.omg.CORBA.SystemException;

    public void connect(org.omg.CORBA.Object obj,
        java.lang.String marker)
        throws org.omg.CORBA.SystemException;

    public void connect(org.omg.CORBA.Object obj,

```

## Orbix Java

```

        IE.Iona.OrbixWeb.Features.LoaderClass loader)
        throws org.omg.CORBA.SystemException;

public void connect(org.omg.CORBA.Object obj,
                   java.lang.String marker,
                   IE.Iona.OrbixWeb.Features.LoaderClass loader)
        throws org.omg.CORBA.SystemException;

public synchronized void disconnect
        (org.omg.CORBA.Object obj)
        throws org.omg.CORBA.SystemException;
public void dispose(org.omg.CORBA.Object obj)
        throws org.omg.CORBA.SystemException;

public void obj_is_ready( org.omg.CORBA.Object obj,
                          java.lang.String impl,
                          int timeOut)
        throws org.omg.CORBA.SystemException;

public void obj_is_ready( org.omg.CORBA.Object obj,
                          java.lang.String impl)
        throws org.omg.CORBA.SystemException;

public void deactivate_obj(org.omg.CORBA.Object obj)
        throws org.omg.CORBA.SystemException;

public void continueThreadDispatch(org.omg.CORBA.Request
r);

// No implementation methods
public org.omg.CORBA.Object create(byte[] id,
                                   java.lang.String intf,
                                   java.lang.String impl)
        throws org.omg.CORBA.SystemException;

public byte[] get_id (org.omg.CORBA.Object oref)
        throws org.omg.CORBA.SystemException;

// Configuration Methods
public boolean setNoHangup(boolean b)
        throws org.omg.CORBA.SystemException;

public static synchronized void
        setProxyServer(java.lang.String host,
                       int port)
        throws org.omg.CORBA.SystemException;

public static synchronized void
        enableProxyServer(boolean useProxy)
        throws org.omg.CORBA.SystemException;

public synchronized void
        setServerName(java.lang.String serverName)
        throws org.omg.CORBA.SystemException;
public void change_implementation(org.omg.CORBA.Object
obj, java.lang.String impl)
        throws org.omg.CORBA.SystemException;
public boolean enableLoaders(boolean b)
        throws org.omg.CORBA.SystemException;

```

```

// Accessor methods
public boolean anyClientsConnected()
    throws org.omg.CORBA.SystemException;

public int numClientsConnected()
    throws org.omg.CORBA.SystemException;

public org.omg.CORBA.Principal get_principal()
    throws org.omg.CORBA.SystemException;

public java.lang.String get_principal_string()
    throws org.omg.CORBA.SystemException;

public org.omg.CORBA.Current get_current()
    throws org.omg.CORBA.SystemException;

public java.lang.String myImplementationName()
    throws org.omg.CORBA.SystemException;

public java.lang.String myMarkerName()
    throws org.omg.CORBA.SystemException;

public java.lang.String myMarkerPattern()
    throws org.omg.CORBA.SystemException;

public java.lang.String myMethodName()
    throws org.omg.CORBA.SystemException;

public short myActivationMode()
    throws org.omg.CORBA.SystemException;

public java.lang.String myHost()
    throws org.omg.CORBA.SystemException;

public java.lang.String myHostIP()
    throws org.omg.CORBA.SystemException;

public static final short perMethodActivationMode;
public static final short persistentActivationMode;
public static final short sharedActivationMode;
public static final short unknownActivationMode;
public static final short unsharedActivationMode;
}

```

**Notes**

CORBA-defined.

**See Also**

["Class org.omg.CORBA.ORB"](#)

["Class IE.Iona.OrbixWeb.CORBA.ORB"](#)



## anyClientsConnected()

### Synopsis

```
public boolean anyClientsConnected()  
    throws org.omg.CORBA.SystemException;
```

### Description

Determines whether there are any connections from clients to this server.

This has a different behaviour if the server has been launched as a thread in the thread-per-server mode with the `orbixdj`. In this case the call determines whether or not there are any connections to the process at all, not just whether the connections are only used by the server thread that made the call.

### Return Value

Returns `true` if any clients are connected; returns `false` otherwise.

### Notes

Orbix-specific.

### See Also

["numClientsConnected\(\)"](#)

## change\_implementation()

### Synopsis

```
public void change_implementation  
    (    org.omg.CORBA.Object obj,  
        java.lang.String impl)  
    throws org.omg.CORBA.SystemException;
```

### Description

Changes the implementation (server name) associated with the object `oref`. You can use this method to overcome the problem of exporting an object reference from a persistent server before `impl_is_ready()` is called.

You can use the `BOA.setServerName()` method to change the implementation for all objects created by a server.

Note that if a server creates an object and clients then invoke on this object, subsequent invocations on the object may fail following a call to `BOA.change_implementation()` on that object.

### Parameters

`obj`      The object reference for which the implementation is to change.

`impl`     The name of the new implementation (server).

### Notes

CORBA-defined.

You are unlikely to need this method.

### See Also

["impl\\_is\\_ready\(\)"](#)  
["setServerName\(\)"](#)

## connect()

### Synopsis

```
public void connect ( org.omg.CORBA.Object obj)
                    throws org.omg.CORBA.SystemException;
public void connect ( org.omg.CORBA.Object obj,
                    IE.Iona.OrbixWeb.Features.LoaderClass loader)
                    throws org.omg.CORBA.SystemException;
public void connect ( org.omg.CORBA.Object obj,
                    java.lang.String marker)
                    throws org.omg.CORBA.SystemException;
public void connect ( org.omg.CORBA.Object obj
                    java.lang.String marker,
                    IE.Iona.OrbixWeb.Features.LoaderClass loader)
                    throws org.omg.CORBA.SystemException;
```

### Description

Used to explicitly connect object implementations to the ORB. The connect method does the following:

- Places the object into the object table.
- Associates it with a marker.
- Associates it with a loader.
- If there is no such thread running already, starts an event-processing thread in the background.

Instantiating an Orbix Java object automatically calls `connect()` for you. However, for strict CORBA compliance you should explicitly call it in your application code also.

To disconnect objects from the ORB and remove them from the object table call `BOA.disconnect()`. When the last object has been disconnected the event-processing thread is stopped.

If you call `BOA.impl_is_ready()` directly yourself its timeout functionality takes control of the event-processing thread. Therefore, when `BOA.impl_is_ready()` times out and returns, the event-processing thread has stopped.

Sometimes you may not want the instantiation of objects to have the side-effect of starting an event-processing thread. You may have a large number of objects that you wish to initialize fully before allowing events to be invoked upon them by the ORB. To prevent `connect()` from starting the event-processing thread, set the Orbix Java variable `IT_IMPL_READY_IF_CONNECTED` to false. If you are using persistent servers you must:

- ♦ call `setServerName(<serverName>)`
- ♦ or have the `java.lang.System` property `'orbixweb.server_name'` set to the implementation (server) name

before you instantiate any objects and you want to pass them out as object-references before calling `BOA.impl_is_ready()`.

When your persistent server is ready to process events you must call `BOA.impl_is_ready(<serverName>)`.

If `connect()` is called twice on an object that has already been connected it has no effect.

After instantiating an object it is possible to call `disconnect()` on that object to disconnect it from the ORB. You can then call `connect()` to reconnect it again.

For more information on naming objects and associating objects with loaders refer to the chapters “Making Objects Available with Orbix Java” and “Locating Servers at Runtime” in the *Orbix Programmer’s Guide Java Edition*.

For more information on `IT_IMPL_READY_IF_CONNECTED` refer to the configuration chapter in the *Orbix Administrator’s Guide Java Edition*.

## Parameters

<code>obj</code>	The object implementation that is to be connected to the ORB.
<code>marker</code>	The marker to be given to the object.
<code>loader</code>	The <code>LoaderClass</code> object to be associated with the object

**Notes** CORBA-defined.

**See Also** [“disconnect\(\)” on page 67](#)  
[“impl\\_is\\_ready\(\)” on page 71](#)

## continueThreadDispatch()

### Synopsis

```
public void continueThreadDispatch(org.omg.CORBA.Request r);
```

### Description

Instructs the Orbix Java runtime to continue dispatching the `Request` in the current thread.

You should use this method in conjunction with `ThreadFilters`. When an instance of `ThreadFilter` receives a `Request` object in its `inRequestPreMarshal` filter point, it can take responsibility from the Orbix Java runtime for scheduling when the `Request` is processed.

Normally the `ThreadFilter` passes the `Request` object into a user-defined queue. This queue is then serviced by one or more user-created threads. They decide when they want to continue dispatching the `Request` by calling the `continueThreadDispatch()` method.

When `continueThreadDispatch()` is called the `Request` object continues to be processed like a normal request. It is passed through filter points, processed by the implementation object and a reply is sent to the client.

The `Request`, however, bypasses any other `ThreadFilters` that are installed but not reached yet.

If an exception (`org.omg.CORBA.SystemException` or `java.lang.Exception`) is thrown while the `Request` is processed the exception is returned to the client.

A `continueThreadDispatch()` call returns when a reply or exception message is sent back to the client.

## Parameters

<code>r</code>	The <code>Request</code> object to be processed.
----------------	--------------------------------------------------

**Notes** Orbix-specific.

**See Also** [“Class org.omg.CORBA.Request” on page 29](#)  
[“Class IE.Iona.OrbixWeb.CORBA.Request” on page 167](#)  
[“Class IE.Iona.OrbixWeb.Features.ThreadFilter”](#)

## create()

### Synopsis

```
public org.omg.CORBA.Object create(byte[] id,
                                   java.lang.String intf,
                                   java.lang.String impl)
    throws org.omg.SystemException;
```

### Description

This method is currently not implemented and always raises an `org.omg.CORBA.NO_IMPLEMENT` exception when called. You should use `IE.Iona.OrbixWeb.ORB.makeIOR()` or `IE.Iona.OrbixWeb.ORB.string_to_object()` to create object references. The following description is for reader information only.

This method creates a new object reference. It does not create an implementation object. As such, it is only used when an implementation object exists in the server or an appropriate loader is installed.

### Parameters

`id` Opaque identification information, supplied by the caller, and stored in an object. This is an IDL sequence of octets that, in Orbix Java, maps to the object marker.

`intf` The Interface Repository object that specifies the set of interfaces implemented by the object.

`impl` The Implementation Repository entry (server name) that specifies the implementation to be used for the object.

### Exceptions

If the `id` does not match the marker of an object currently resident in (or loaded into) the server address space, `CORBA.BOA.create()` raises a `CORBA.INV_OBJREF` exception.

### Notes

CORBA-defined.

### See Also

["get\\_id\(\)" on page 70](#)  
["Interface IE.Iona.OrbixWeb.CORBA.ObjectRef"](#)  
["Class org.omg.CORBA.ORB" on page 19](#)

## deactivate\_impl()

### Synopsis

```
public void deactivate_impl(java.lang.String impl)
    throws org.omg.SystemException;
```

### Description

A server that has called `impl_is_ready()` to indicate that it has completed initialization and is ready to receive requests, may subsequently indicate to Orbix Java that it wishes to discontinue receiving requests. It does so by calling `deactivate_impl()`, and passing the server name in the parameter `impl`. Calling `deactivate_impl()` causes `impl_is_ready()` to return.

### Parameters

`impl` The server name, as passed to `impl_is_ready()`.

### Notes

CORBA-defined.

### See Also

["impl\\_is\\_ready\(\)" on page 71](#)

## deactivate\_obj()

### Synopsis

```
public void deactivate_obj(org.omg.CORBA.Object obj)
    throws org.omg.SystemException;
```

### Description

A server (running in unshared activation mode) that has called `obj_is_ready()` to indicate that it has completed initialization and is ready to receive requests, may subsequently indicate to Orbix Java that it wishes to discontinue receiving requests for this object. It does so by calling `deactivate_obj()`, and passing the object whose marker caused the server process to be launched, in the parameter `oref`.

### Parameters

`obj`            The object whose marker caused the server to be launched.

### Notes

CORBA-defined.

### See Also

["obj\\_is\\_ready\(\)" on page 76](#)

## disconnect()

### Synopsis

```
public synchronized void disconnect(org.omg.CORBA.Object obj)
    throws org.omg.CORBA.SystemException;
```

### Description

Removes the object reference `obj` from the runtime object table. Future invocations on the object fail with an `INV_OBJREF` exception.

When `disconnect()` is called, if `obj` has a `LoaderClass` object associated with it, the loader object's `save()` method is called. The reason passed to `save()` will be `_CORBA.explicitCall`. For more information refer to the ***Orbix Programmer's Guide Java Edition***.

`disconnect()` has the same behaviour as `BOA.dispose()`, `disconnect()`, and is the preferred method pending the introduction of the Portable Object Adapter to the IDL/Java Language Mapping.

To reconnect an object after it has been disposed call `BOA.connect()`.

### Parameters

`obj`            The object to be disconnected.

### Notes

CORBA-defined.

### See Also

["Interface org.omg.CORBA.Object" on page 18](#)  
["connect\(\)" on page 64](#)  
["Interface IE.Iona.OrbixWeb.CORBA.ObjectRef"](#)  
["save\(\)"](#)

## dispose()

### Synopsis

```
public void dispose(org.omg.CORBA.Object obj)
    throws org.omg.CORBA.SystemException;
```

### Description

`dispose()` has the same behaviour as `BOA.disconnect()`. However, `disconnect()` is the preferred method pending the introduction of the Portable Object Adapter to the IDL/Java Language Mapping. See `BOA.disconnect()` for a description of this method's behaviour.

### Notes

CORBA-defined.

### See Also

["Interface org.omg.CORBA.Object" on page 18](#)  
["connect\(\)" on page 64](#)  
["Interface IE.Iona.OrbixWeb.CORBA.ObjectRef"](#)  
["save\(\)"](#)

## enableLoaders()

### Synopsis

```
public boolean enableLoaders(boolean b)
    throws org.omg.CORBA.SystemException;
```

### Description

Globally enables or disables loaders. See the ***Orbix Programmer's Guide Java Edition*** for more details.

### Parameters

`b`        `true` enables the loaders, `false` disables the loaders.

### Return Value

Returns the previously set value.

### Notes

Orbix-specific.

### See Also

["Class IE.Iona.OrbixWeb.Features.LoaderClass"](#)

## enableProxyServer()

### Synopsis

```
public static synchronized void
    enableProxyServer(boolean useProxy)
    throws org.omg.CORBA.SystemException;
```

### Description

This method is provided for supporting the Wonderwall firewall product that was included in earlier versions of Orbix 3.

**Note:** From Orbix 3.3.13, Wonderwall is not supplied with Orbix 3, and its use is deprecated. However `enableProxyServer()` is still supported for customers who wish to use a Wonderwall instance from an earlier Orbix release.

If set to `true`, any CORBA objects created afterwards contain the host name and port number of the proxy server, that is, Wonderwall. If set to `false`, they contain the actual server's host and port.

Use `BOA.setProxyServer()` to set the port and host for the proxy server.

See the ***Wonderwall Administrator's Guide*** supplied with earlier versions of Orbix 3 for full details of how to use this method.

## Parameters

`useProxy` true means any new CORBA objects created contains the port and host of the proxy server.  
false means that they contain the actual server host and port.

## Notes

Orbix-specific.

## See Also

["Description" on page 81](#)

## finalize()

### Synopsis

```
public void finalize()
```

### Description

Cleans up the object table being used by the BOA object. As BOA inherits from ORB it also cleans up its connection table.

When the object table is finalizing, `save()` is called on the `LoaderClass` object associated with each object in the table. The reason given is `IE.Iona.OrbixWeb._CORBA.processTermination`. See the ***Orbix Programmer's Guide Java Edition*** for more details.

This method is called in two ways:

- By the garbage collector when the server shuts down. To be sure of this you must call the JDK1.1 method `java.lang.System.runFinalizersOnExit(true)`.
- By the user explicitly calling `finalize()` on the BOA object. You must do this just before the server exits. This is particularly important when using loaders, otherwise you cannot be sure that the currently active objects in the server get a chance to `save()` themselves.

## Notes

Orbix-specific.

## See Also

["finalize\(\)"](#)  
["Class IE.Iona.OrbixWeb.Features.LoaderClass"](#)

## get\_current()

### Synopsis

```
public org.omg.CORBA.Current get_current()  
    throws org.omg.CORBA.SystemException;
```

### Description

Orbix Java returns an instance of `IE.Iona.OrbixWeb.CORBA.Orbcurrent`. You can query this object for information.

The `IT_MULTI_THREADED_SERVER` configuration parameter must be set to true.

Refer to the configuration chapter in the ***Orbix Programmer's Guide Java Edition*** for more information on `IT_MULTI_THREADED_SERVER`.

## Notes

CORBA-defined.

## See Also

["Class org.omg.CORBA.ORB" on page 19](#)  
["Class IE.Iona.OrbixWeb.CORBA.OrbCurrent" on page 159](#)

## get\_id()

### Synopsis

```
public byte[] get_id(org.omg.CORBA.Object obj)
    throws org.omg.SystemException;
```

### Description

This method is currently not implemented and always raises an `org.omg.CORBA.NO_IMPLEMENT` exception when called. Orbix Java programmers should use `IE.Iona.OrbixWeb.ObjectRef._marker()` instead. The following description is for reader information.

Returns the identification information for the object `obj` as set in `BOA.create()`.

### Notes

CORBA-defined.

### See Also

["create\(\)" on page 66](#)  
["Interface IE.Iona.OrbixWeb.CORBA.ObjectRef"](#)

## get\_principal()

### Synopsis

```
public org.omg.CORBA.Principal get_principal()
    throws org.omg.CORBA.SystemException;
```

### Description

This method returns the `Principal` object for the client that made the operation call currently being processed.

Set `IT_MULTI_THREADED_SERVER` to `true` before initializing the ORB.

### Notes

CORBA-defined.

### See Also

["Class org.omg.CORBA.Principal" on page 28](#)  
["get\\_principal\\_string\(\)" on page 70](#)  
["Class IE.Iona.OrbixWeb.CORBA.Principal"](#)

## get\_principal\_string()

### Synopsis

```
public java.lang.String get_principal_string()
    throws org.omg.CORBA.SystemException;
```

### Description

This method returns a string version of the `Principal` object for the client that made the operation call currently being processed.

Set `IT_MULTI_THREADED_SERVER` to `true` before initializing the ORB.

### Notes

Orbix-specific.

### See Also

["Class org.omg.CORBA.Principal" on page 28](#)  
["get\\_principal\(\)" on page 70](#)  
["Class IE.Iona.OrbixWeb.CORBA.Principal"](#)



## impl\_is\_ready()

### Synopsis

```
public void impl_is_ready()
    throws org.omg.CORBA.SystemException;
public void impl_is_ready(java.lang.String serverName)
    throws org.omg.CORBA.SystemException;
public void impl_is_ready(int timeOut)
    throws org.omg.CORBA.SystemException;
public void impl_is_ready(java.lang.String serverName,
    int timeOut)
    throws org.omg.CORBA.SystemException;
```

### Description

Once you register a server with Orbix Java the daemon may automatically launch the server if an operation is invoked on one of its objects. You can also launch a server persistently (manually).

Once launched, the server initializes itself and creates any objects it requires. This automatically connects the objects to the ORB and optionally calls `impl_is_ready()` to use this call's timeout facility.

The `impl_is_ready()` method normally does not return immediately. It blocks the server until an event occurs, handles the event, and re-blocks the server to await another event. The methods `BOA.processEvents()` and `BOA.processNextEvent()` provide alternative ways of handling events.

The `impl_is_ready()` method returns only when:

- A timeout occurs.  
This occurs when the server has no clients for the timeout duration, or because none of its clients use it for that period.
- An exception occurs while waiting for an incoming event.
- The function `BOA.deactivate_impl()` is called.

If you are using persistent servers with the daemon and want to create objects and export them before processing events then you must call

```
BOA.impl_is_ready(<serverName>,0)
```

before creating any objects. This contacts the daemon, finds out what port the persistent server is expected to be listening on and makes sure that all the created objects contain the correct port number.

Persistent servers, once they have called `impl_is_ready()`, behave as shared activation mode servers. The `serverName` specified must match an unused registered server name or an exception is raised.

If a server is registered as unshared or per-method, `impl_is_ready()` fails if the server is launched manually.

Normally you must register a server in the Implementation Repository before it can call `impl_is_ready()`. However, if you specify the `-u` switch to the Orbix Java daemon, a persistent server can call `impl_is_ready()` without being registered in the Implementation Repository. Refer to the ***Orbix Programmer's Guide Java Edition*** for more information on unregistered servers.

## Parameters

<code>serverName</code>	<p>The <code>serverName</code> parameter is optional if the server is launched by Orbix Java. It is mandatory if the server is launched manually or externally to Orbix Java. You should always pass <code>serverName</code> explicitly.</p> <p>If the <code>serverName</code> parameter is specified, it must match exactly the server name in the Implementation Repository.</p>
<code>timeOut</code>	<p>Indicates the number of milliseconds to wait between events. A timeout occurs if Orbix Java waits longer than the specified timeout for the next event.</p> <p>A timeout of zero indicates that <code>impl_is_ready()</code> should process one event, presuming that there is one pending, and then return immediately.</p> <p>A timeout does not cause <code>impl_is_ready()</code> to raise an exception.</p> <p>The default timeout can be passed explicitly as <code>getConfigItem("IT_DEFAULT_TIMEOUT")</code>.</p> <p>An infinite timeout can be specified by using <code>getConfigItem("IT_INFINITE_TIMEOUT")</code>.</p> <p>The <code>timeOut</code> parameter is meaningless for the per-method call activation mode, since the server terminates once the operation call that caused it to be launched has completed.</p> <p>Calling <code>BOA.setNoHangup(true)</code> changes the time out behaviour of <code>impl_is_ready()</code>. Instead of timing out after an elapsed period between events it times out after an elapsed period between client connections. For more details see <code>BOA.setNoHangup()</code>.</p>

## Notes

CORBA-defined.

## See Also

["deactivate\\_impl\(\)" on page 66](#)  
["connect\(\)" on page 64](#)  
["obj\\_is\\_ready\(\)" on page 76](#)  
["processEvents\(\)" on page 77](#)  
["setNoHangup\(\)" on page 80](#)  
["setServerName\(\)" on page 81](#)  
["getConfigItem\(\)" on page 136](#)

## isEventPending()

### Synopsis

```
public boolean isEventPending()  
    throws org.omg.CORBA.SystemException;
```

### Description

Tests if there is an outstanding event waiting in the Orbix Java event queue. If it returns `true`, a subsequent call to `BOA.processNextEvent()`, `BOA.processEvents()` or `BOA.impl_is_ready()` removes the event from the event queue and processes it.

`isEventPending()` is normally used with `BOA.processNextEvent()` and `BOA.processEvents()` rather than with `impl_is_ready()`.

### Return Value

`true`     There is an event pending.  
`false`    There is no event pending.

### Notes

Orbix-specific.

### See Also

["impl\\_is\\_ready\(\)" on page 71](#)  
["processEvents\(\)" on page 77](#)  
["processNextEvent\(\)" on page 79](#)

## myActivationMode()

### Synopsis

```
public short myActivationMode()  
    throws org.omg.CORBA.SystemException;
```

### Description

Determines the primary activation mode with which the server was launched: `shared`, `unshared`, `persistent`, or `per-method`.

For more information on activation modes see the chapter "Registration and Activation of Servers" in the ***Orbix Programmer's Guide Java Edition***.

### Return Value

Returns one of the following values:

<code>persistentActivationMode</code>	Indicates that the server was launched manually.
<code>sharedActivationMode</code>	Indicates that the server was automatically launched in shared mode.
<code>unsharedActivationMode</code>	Indicates that the server was automatically launched in unshared mode.
<code>perMethodActivationMode</code>	Indicates that the server was automatically launched in per-method mode.
<code>unknownActivationMode</code>	Orbix Java cannot determine the mode in which the server was launched.

### Notes

Orbix-specific.

## myHost()

### Synopsis

```
public java.lang.String myHost()  
    throws org.omg.CORBA.SystemException;
```

### Description

If proxy servers are enabled, returns the proxy server host name. Otherwise, if the variable `IT_IORS_USE_DNS` is set to `false`, it returns the IP address of the local host.

If the variable `IT_IORS_USE_DNS` has been set to `true`, it returns the local host name. Refer to the `IT_IORS_USE_DNS` configuration parameter in the *Orbix Administrator's Guide Java Edition* for more details.

### Notes

Orbix-specific.

### See Also

["enableProxyServer\(\)" on page 68](#)  
["Description" on page 81](#)

## myHostIP()

### Synopsis

```
public java.lang.String myHostIP()  
    throws org.omg.CORBA.SystemException;
```

### Description

Returns the local host's IP address.

If the proxy server has been enabled, returns the IP address of the proxy server.

### Notes

Orbix-specific.

### See Also

["enableProxyServer\(\)" on page 68](#)  
["Description" on page 81](#)

## myImplementationName()

### Synopsis

```
public java.lang.String myImplementationName()  
    throws org.omg.CORBA.SystemException;
```

### Description

Returns the server name.

For automatically launched servers, this name is passed to the server by the daemon. It is same name as that with which the server is registered in the Implementation Repository.

For a persistent server, the contents of the string are undefined until `CORBA.BOA.impl_is_ready()`, `CORBA.BOA.obj_is_ready()` or `CORBA.ORB.setServerName()` is called.

### Notes

Orbix-specific.

### See Also

["impl\\_is\\_ready\(\)" on page 71](#)  
["setServerName\(\)" on page 81](#)

## myMarkerName()

### Synopsis

```
public java.lang.String myMarkerName()  
    throws org.omg.CORBA.SystemException;
```

### Description

Returns the marker name of the activation object that caused the server to be launched. For a persistent or a per-method server, this is null.

### Notes

Orbix-specific.

### See Also

["myMarkerPattern\(\)" on page 75](#)

## myMarkerPattern()

### Synopsis

```
public java.lang.String myMarkerPattern()  
    throws org.omg.CORBA.SystemException;
```

### Description

Returns the marker pattern which the activation object matched in the Implementation Repository and caused this server to be launched.

For a persistent or per-method server, this is null.

### Notes

Orbix-specific.

### See Also

["myMarkerName\(\)" on page 75](#)

## myMethodName()

### Synopsis

```
public java.lang.String myMethodName()  
    throws org.omg.CORBA.SystemException;
```

### Description

Returns the name of the method that caused the server to be launched. For a server not launched on a per-method basis this is null.

### Notes

Orbix-specific.

## numClientsConnected()

### Synopsis

```
public int numClientsConnected()  
    throws org.omg.CORBA.SystemException;
```

### Description

Returns the number of clients that are currently connected to the server.

This acts differently if the server is launched as a thread in the thread-per-server mode with the `orbixdj`. In this case the call determines the number of classes in the process, and not just whether those connections are only used by the server thread that made the call.

### Notes

Orbix-specific.

## obj\_is\_ready()

### Synopsis

```
public void obj_is_ready (org.omg.CORBA.Object obj,
                        java.lang.String impl,
                        int timeout)
    throws org.omg.CORBA.SystemException;
public void obj_is_ready (org.omg.CORBA.Object obj,
                        String impl)
    throws SystemException;
```

### Description

A server running in the unshared activation mode (with one registered object per process) can call the `BOA.obj_is_ready()` method on the `_CORBA.Orbix` object to indicate that it has completed its initialization. The server remains active and receives requests for its registered object until:

- It calls `BOA.deactivate_obj()`.
- The call to `obj_is_ready()` times out.
- An exception is raised while waiting for events.

### Parameters

<code>obj</code>	The registered object that the server manages.
<code>impl</code>	The server name.
<code>timeout</code>	The timeout period. A server can time out either because it has no clients for the timeout duration, or because none of its clients uses it for that period.  The default timeout can be passed explicitly as <code>_OrbixWeb.ORB(ORB.init()).getConfigItem("IT_DEFAULT_TIMEOUT")</code> .  An infinite timeout can be specified by passing <code>_OrbixWeb.ORB(ORB.init()).getConfigItem("IT_INFINITE_TIMEOUT")</code> .  Calling <code>BOA.setNoHangup(true)</code> changes the time out behaviour of <code>obj_is_ready()</code> . Instead of timing out after an elapsed period between events it times out after an elapsed period between client connections. For more details see <code>BOA.setNoHangup()</code> .

### Notes

CORBA-defined.

### See Also

["deactivate\\_obj\(\)" on page 67](#)  
["impl\\_is\\_ready\(\)" on page 71](#)  
["setNoHangup\(\)" on page 80](#)

## processEvents()

### Synopsis

```
public int processEvents()  
    throws org.omg.CORBA.SystemException;  
public int processEvents(int timeOut)  
    throws org.omg.CORBA.SystemException;
```

### Description

If a zero timeout period is given to `BOA.impl_is_ready()` or `BOA.obj_is_ready()`, the call returns immediately—allowing a program to subsequently state at what points it is willing to accept incoming Orbix Java events.

There are three kinds of event:

- Operation requests
- Connections from clients
- Disconnections of clients

The method `processEvents()` blocks the server until an event arrives, handles the event, and continues to process events, until none arrives within the timeout period. It has the same effect as calling `BOA.processNextEvent()` repeatedly until it times out.

The method `processEvents()` is similar in functionality to `impl_is_ready()` (or `obj_is_ready()`) because it processes any number of events until it times out. However, use of `processEvents()` does not initialize the server and therefore does not fulfill a server's requirement to call `impl_is_ready()` (or `obj_is_ready()`).

One example of using `processEvents()` is where a manually launched server wishes to interact with Orbix Java (for example, by calling a remote operation or by passing out or allowing clients to connect to it) before it is ready to handle incoming events from clients.

Before it interacts with Orbix Java, it must call `impl_is_ready()` (or `obj_is_ready()`) with a zero timeout thereby initializing the server and allowing clients to connect. The server can then do whatever other initialization work must be done and calls `processEvents()` when it is ready to handle incoming events from clients.

## Parameters

`timeOut` Indicates the number of milliseconds to wait between events; a timeout occurs if Orbix Java waits longer than the specified timeout for the next event.

A timeout of zero indicates that `processEvents()` should process one event (presuming that there is one pending) and then return immediately.

A timeout does not cause `processEvents()` to raise an exception.

The default timeout can be passed explicitly as `_OrbixWeb.ORB(ORB.init()).getConfigItem("IT_DEFAULT_TIMEOUT")`.

An infinite timeout can be specified by passing `_OrbixWeb.ORB(ORB.init()).getConfigItem("IT_INFINITE_TIMEOUT")`.

Calling `BOA.setNoHangup(true)` changes the time out behaviour of `processEvents()`. Instead of timing out after an elapsed period between events it times out after an elapsed period between client connections. For more details see ["setNoHangup\(\)" on page 80](#).

## Return Value

Always returns 1.

## Notes

Orbix-specific.

## See Also

["impl\\_is\\_ready\(\)" on page 71](#)  
["obj\\_is\\_ready\(\)" on page 76](#)  
["processNextEvent\(\)" on page 79](#)  
["setServerName\(\)" on page 81](#)  
["getConfigItem\(\)" on page 136](#)



## processNextEvent()

### Synopsis

```
public void processNextEvent()  
    throws org.omg.CORBA.SystemException;  
public void processNextEvent(int timeout)  
    throws org.omg.CORBA.SystemException;
```

### Description

A programmer may want more control over the handling of events in a server.

There are three kinds of events:

- Operation requests
- Connections from clients
- Disconnections of clients

This method blocks the server until an event arrives, handles that one event, and normally returns zero.

If a zero timeout period is given to `BOA.impl_is_ready()` or `BOA.obj_is_ready()`, the server initializes and returns immediately. This allows a program to subsequently state at what point it is willing to accept incoming Orbix Java events. This can be done by calling `_CORBA.Orbix.processNextEvent()`.

### Parameters

`timeout` Indicates the number of milliseconds to wait for an event; a timeout occurs if Orbix Java waits longer than the specified timeout for the next event.

A timeout of zero indicates that `processNextEvent()` should process one event (presuming that there is one pending) and then return immediately.

A timeout causes `processNextEvent()` to return 1.

The default timeout can be passed explicitly as `_OrbixWeb.ORB(ORB.init()).getConfigItem("IT_DEFAULT_TIMEOUT")`.

An infinite timeout can be specified by passing `_OrbixWeb.ORB(ORB.init()).getConfigItem("IT_INFINITE_TIMEOUT")`.

Calling `BOA.setNoHangup(true)` changes the time out behaviour of `processNextEvent()`. Instead of timing out after an elapsed period between events it times out after an elapsed period between client connections. For more details see "[setNoHangup\(\)](#)" on page 80.

### Return

Normally returns 0 (`false`). Returns 1 (`true`) if the server has been deactivated or if the call to `processNextEvent()` times out.

### Notes

Orbix-specific.

### See Also

["impl\\_is\\_ready\(\)" on page 71](#)  
["obj\\_is\\_ready\(\)" on page 76](#)  
["processNextEvent\(\)" on page 79](#)  
["setServerName\(\)" on page 81](#)  
["getConfigItem\(\)" on page 136](#)

## setNoHangup()

### Synopsis

```
public boolean setNoHangup(boolean b)
    throws org.omg.CORBA.SystemException;
```

### Description

By default, the `BOA.impl_is_ready()`, `BOA.obj_is_ready()`, `BOA.processEvents()` and `BOA.processNextEvent()` methods time out when a (user-defined or default) time period has elapsed between events. An event is an incoming operation call, or the connection or disconnection by a client.

This means that the methods mentioned above can timeout when clients are still connected but have been idle for a period of time. If you want a server to remain active while it has any clients connected, active or not, call the function `setNoHangup(true)` on the `_CORBA.Orbix` object.

### Parameters

**b** When `true` is passed as the parameter, the timeout period to `impl_is_ready()`, `obj_is_ready()`, `processEvents()`, and `processNextEvent()` specifies the amount of time a server waits while there are no client connections to it. The event handler does not time out until all clients are disconnected.

When `false` is passed as the parameter, the timeout period to `impl_is_ready()`, `obj_is_ready()`, `processEvents()`, and `processNextEvent()` specifies the amount of time a server waits while there are no events (operation calls, connections, disconnections). These calls time out if the specified period elapses between events.

The default setting is `false`.

### Return Value

Returns the previously set value.

### Notes

Orbix-specific.

### See Also

["impl\\_is\\_ready\(\)" on page 71](#)

["processEvents\(\)" on page 77](#)

["processNextEvent\(\)" on page 79](#)

## setProxyServer()

### Synopsis

```
public synchronized void setProxyServer(java.lang.String host,
                                         int port)
    throws org.omg.CORBA.SystemException;
```

### Description

This method is provided for supporting the Wonderwall firewall product that was included in earlier versions of Orbix 3.

**Note:** From Orbix 3.3.13, Wonderwall is not supplied with Orbix 3, and its use is deprecated. However `setProxyServer()` is still supported for customers who wish to use a Wonderwall instance from an earlier Orbix release.

The method sets the values for the host name and port number for the proxy server (Wonderwall).

If `BOA.enableProxyServer()` is set to `true`, any CORBA objects created contain the host name and port number as specified by this method.

See the *Wonderwall Administrator's Guide* supplied with earlier versions of Orbix 3 for full details of how to use this method.

### Parameters

`host` The name of the host on which your firewall proxy server (Wonderwall) is running.

`port` The port number on which your firewall proxy server (Wonderwall) is listening.

### Notes

Orbix-specific.

### See Also

["enableProxyServer\(\)" on page 68](#)

## setServerName()

### Synopsis

```
public synchronized void setServerName
    (java.lang.String serverName)
    throws org.omg.CORBA.SystemException;
```

### Description

Sets the server name for the current server.

### Notes

Orbix-specific.

### See Also

["change\\_implementation\(\)" on page 63](#)  
["impl\\_is\\_ready\(\)" on page 71](#)  
["myImplementationName\(\)" on page 74](#)

## shutdown()

### Synopsis

```
public void shutdown(wait_for_completion);
```

### Description

Shuts down the BOA and ORB when called on an ORB instance. The `wait_for_completion` flag is not currently supported by Orbix Java and can be ignored.

### Notes

Orbix-specific.

### See Also

["finalize\(\)" on page 136](#)  
["Class IE.Iona.OrbixWeb.Features.LoaderClass"](#)

## toString()

### Synopsis

```
public java.lang.String toString();
```

### Description

This method can be used to check if the `_CORBA.Orbix` object is an ORB or BOA object.

### Return Value

Returns the string `"IE.Iona.OrbixWeb.CORBA.BOA"`.

### Notes

Orbix-specific.

### See Also

["toString\(\)" on page 158](#)

# Class IE.Iona.OrbixWeb.CORBA.Context

## Synopsis

The Java class `Context` implements the OMG pseudo-interface `Context`. A context is intended to represent information about the client that is inconvenient to pass via parameters.

An IDL operation can specify that a `Context` is to be provided with the client's mapping for particular identifiers (properties)—it does this by listing these identifiers following the operation declaration in a `context` clause. An IDL operation that specifies a `context` clause is mapped to a Java method that takes an extra input parameter of type `org.omg.CORBA.Context`, at the end of the parameter list. A client can optionally maintain one or more `Context` objects, which provide a mapping from identifiers (string names) to string values. A `Context` object contains a list of properties; each property consists of a name and a string value associated with that name and can be passed to a method that takes a `Context` parameter.

You can arrange `Contexts` in a hierarchy by specifying parent-child relationships among them. In this case, a child passed to an operation also includes the identifiers of its parent(s). The called operation can decide whether to use just the context actually passed, or the hierarchy above it.

## Note:

The use of the `Context` constructor has been deprecated in Orbix Java. You should use the `IT_create` static methods instead.

## CORBA

```
// Pseudo IDL

pseudo interface Context {
    readonly attribute Identifier context_name;
    readonly attribute Context parent;

    Context create_child(
        in Identifier child_ctx_name);

    void set_one value(
        in Identifier propname, in any propvalue);
    void set_values(in NVList values);
    void delete_values(in Identifier propname);
    NVList get_values(in Identifier start_scope,
        in Flags op_flags,
        in Identifier pattern);
};
```

## Orbix Java

```
// Java

package IE.Iona.OrbixWeb.CORBA;

public class Context extends org.omg.CORBA.Context{
    // Constructors
    public Context();
    public Context(String name);
    public Context(Context parent);
    public Context(String name, Context parent);
}
```

```

// Methods
public org.omg.CORBA.Context create_child(String
    child_ctx_name)
    throws SystemException;
public void delete_values(String prop_name)
    throws SystemException;
public org.omg.CORBA.NVList get_values(
    String start_scope,
    Flags op_flags, String prop_name)
    throws SystemException;
public void set_one_value(String prop_name,
    org.omg.CORBA.Any val)
    throws SystemException;

public void set_values(org.omg.CORBA.NVList vals);
    throws SystemException;
public String toString();
    throws SystemException;
public java.lang.Object clone();
    throws SystemException;
public boolean equals(java.lang.Object _obj);
    throws SystemException;

// Object Methods
public static Context IT_create();
    throws SystemException;
public static Context IT_create(String name);
    throws SystemException;
public static Context IT_create(Context parent);
    throws SystemException;
public static Context IT_create(String name,
    Context parent);
    throws SystemException;
public static Context _nil();
    throws SystemException;

// Data Accessor Methods
public String context_name();
    throws SystemException;
public Context parent();
    throws SystemException;
public int get_count();
    throws SystemException;
public int get_count_all();
    throws SystemException;
}

```

**See Also**

["Class IE.Iona.OrbixWeb.CORBA.ContextIterator"](#)  
["Class IE.Iona.OrbixWeb.CORBA.NVList"](#)

## Context()

### Synopsis

```
public Context();
```

### Description

Creates a new context that is not a child context.

### Notes

Orbix-specific.

### See Also

["create\\_child\(\)" on page 86](#)

["IT\\_create\(\)" on page 88](#)

Other `Context` constructors.

## Context()

### Synopsis

```
public Context(String name);
```

### Description

Creates a new context (not a child context) with a specified name.

### Parameters

`name`                      The name of the context.

### Notes

Orbix-specific.

### See Also

["create\\_child\(\)" on page 86](#)

["IT\\_create\(\)" on page 88](#)

Other `Context` constructors.

## Context()

### Synopsis

```
public Context(Context parent);
```

### Description

Creates a new child context.

### Parameters

`parent`                      The parent context.

### Notes

Orbix-specific.

### See Also

["create\\_child\(\)" on page 86](#)

["IT\\_create\(\)" on page 88](#)

Other `Context` constructors.

## Context()

### Synopsis

```
public Context(String name, Context parent);
```

### Description

Creates a new child context with a specified name.

### Parameters

`name`                      The name of the context.

`parent`                      The parent context.

### Notes

Orbix-specific.

### See Also

["create\\_child\(\)" on page 86](#)

["IT\\_create\(\)" on page 88](#)

Other `Context` constructors.

## **\_nil()**

### **Synopsis**

```
public Static Context _nil();
```

### **Description**

Returns a nil object reference for a Context object.

### **Notes**

Orbix-specific.

## **context\_name()**

### **Synopsis**

```
public String context_name();
```

### **Description**

Returns the name of the Context object.

### **Notes**

CORBA-defined.

### **See Also**

["create\\_child\(\)" on page 86](#)

## **create\_child()**

### **Synopsis**

```
public org.omg.CORBA.Context create_child(String  
    child_ctx_name);
```

### **Description**

Creates a child context of the current context. When a child context is passed as a parameter to an operation, any searches (using `IE.Iona.OrbixWeb.CORBA.Context.get_values()`) look in parent contexts if necessary to find matching property names.

### **Parameters**

<code>ctx_name</code>	The name of the child context. Context object names follow the rules for IDL identifiers.
-----------------------	-------------------------------------------------------------------------------------------

### **Return Value**

Context	The newly-created Context.
---------	----------------------------

### **Notes**

CORBA-defined.

### **See Also**

["get\\_values\(\)" on page 87](#)

## **delete\_values()**

### **Synopsis**

```
public void delete_values(String prop_name)  
    throws SystemException;
```

### **Description**

Deletes the specified property value(s) from the context. The search scope is limited to the Context object on which the invocation is made.

### **Parameters**

<code>prop_name</code>	The property name to be deleted. If <code>prop_name</code> has a trailing '*', all matching properties are deleted.
------------------------	---------------------------------------------------------------------------------------------------------------------

### **Notes**

CORBA-defined.



## get\_count()

### Synopsis

```
public int get_count();
```

### Description

Finds the number of properties/value pairs in the context.

### Notes

Orbix-specific.

### See Also

["get\\_count\\_all\(\)" on page 87](#)

## get\_count\_all()

### Synopsis

```
public int get_count_all();
```

### Description

Finds the number of properties/value pairs in this context and all its parent contexts.

### Notes

Orbix-specific.

### See Also

["get\\_count\(\)" on page 87](#)

## get\_values()

### Synopsis

```
public org.omg.CORBA.NVList get_values(  
    String start_scope,  
    int op_flags, String prop_name);
```

### Description

Retrieves the specified context property values.

### Parameters

<code>start_scope</code>	The context in which the search for the values requested should be started. The name of a direct or indirect parent context may be specified to this parameter. If the null string is passed, the search begins in the context that is the target of the call.
<code>op_flags</code>	By default, searching propagates upwards to parent contexts; if <code>_CORBA.CTX_RESTRICT_SCOPE</code> is specified, searching is limited to the specified search scope or context object.
<code>prop_name</code>	If <code>prop_name</code> has a trailing <code>'*'</code> , all matching properties and their values are returned.

### Return Value

<code>NVList</code>	An <code>NVList</code> object to contain the returned property values.
---------------------	------------------------------------------------------------------------

### Notes

CORBA-defined.

### See Also

["Class IE.Iona.OrbixWeb.CORBA.NVList"](#)

## IT\_create()

### Synopsis

```
public static Context IT_create();  
public static Context IT_create(String name);  
public static Context IT_create(Context parent);  
public static Context IT_create(String name, Context parent);
```

### Description

In the absence of a CORBA specified way to create a (top level) Context pseudo object, Orbix Java provides the `IT_create()` method to initialize an object reference for a Context.

Use of this method is recommended in preference to Java `new` to ensure portability across future Orbix Java versions.

### Parameters

<code>name</code>	The name of the context.
<code>parent</code>	The parent context.

### Notes

Orbix-specific.

### See Also

["create\\_child\(\)" on page 86](#)  
Other Context constructors.

## parent()

### Synopsis

```
public Context parent();
```

### Description

Returns the parent of the Context object.

### Notes

CORBA-defined.

### See Also

["create\\_child\(\)" on page 86](#)

## set\_one\_value()

### Synopsis

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

### Description

Adds property name and value to context. Although the `value` member is of type `Any`, the type of the `Any` must be a string.

### Parameters

<code>prop_name</code>	The name of the property to add.
<code>val</code>	The value of the property to add.

### Notes

CORBA-defined.

### See Also

["set\\_values\(\)" on page 88](#)

## set\_values()

### Synopsis

```
public void set_values(org.omg.CORBA.NVList vals);
```

### Description

Sets one or more property values in the context. The previous value of the property, if any, is discarded.

## Parameters

values      An `NVList` containing the `property_name:values` to add or change. In the `NVList`, the `flags` field must be set to zero, and the `TypeCode` associated with an attribute value must be `_CORBA._tc_string`.

## Notes

CORBA-defined.

## See Also

["set\\_one\\_value\(\)" on page 88](#)

# Class IE.Iona.OrbixWeb.CORBA.ContextList

## Synopsis

The Java class `ContextList` implements the CORBA pseudo-object type `ContextList`. A `ContextList` is used in the DII to describe the context strings required for a particular operation. The list stores them as `Context` objects.

Refer to the chapter “Dynamic Invocation Interface” in the *Orbix Programmer’s Guide Java Edition* for further details.

## CORBA

```
// Pseudo IDL
```

```
pseudo interface ContextList {
    readonly attribute unsigned long count;
    void add(in string ctx);
    string item(in unsigned long index) raises (CORBA::Bounds);
    void remove(in unsigned long index) raises (CORBA::Bounds);
};
```

## Orbix Java

```
// Java
```

```
public class ContextList extends org.omg.CORBA.ContextList
{
    public ContextList()

    public int count();
    public void add(java.lang.String ctx)
        throws org.omg.CORBA.SystemException;
    public java.lang.String item(int index)
        throws org.omg.CORBA.Bounds,
               org.omg.CORBA.SystemException;
    public void remove(int index)
        throws org.omg.CORBA.Bounds,
               org.omg.CORBA.SystemException;
}
```

## Notes

CORBA-defined.

## See Also

[“contexts\(\)” on page 173](#)

[“Class org.omg.CORBA.Context” on page 9](#)

[“Class IE.Iona.OrbixWeb.CORBA.Context” on page 83](#)

## ContextList()

### Synopsis

```
public ContextList();
```

### Description

Default constructor.

### Notes

Orbix-specific. See `org.omg.CORBA.ORB.create_context_list()` for CORBA-defined ways to create a `ContextList`.

### See Also

[“Class org.omg.CORBA.ORB” on page 19](#)

## add()

### Synopsis

```
public void add(java.lang.String ctx)
    throws org.omg.CORBA.SystemException;
```

### Description

Uses `ctx` to create an `org.omg.CORBA.Context` object and adds that object to the end of the list of contexts.

### Parameters

`ctx`                      A string describing the variable or pattern the context should contain.

### Notes

CORBA-defined.

## count()

### Synopsis

```
public int count() throws org.omg.CORBA.SystemException;
```

### Description

Returns the number of elements in the `ContextList`.

### Return Value

`int`                      Number of elements in the list.

### Notes

CORBA-defined.

## item()

### Synopsis

```
public java.lang.String item(int index)
    throws            org.omg.CORBA.Bounds,
                   org.omg.CORBA.SystemException;
```

### Description

Returns the item at the given index. The first item is at index 0.

### Parameters

`index`                    Position within list to be returned.

### Return Value

`java.lang.String`        The string identifier for the context at `index`.

### Notes

CORBA-defined.

## remove()

### Synopsis

```
public void remove(int index)
    throws            org.omg.CORBA.Bounds,
                   org.omg.CORBA.SystemException;
```

### Description

Removes the item at the given index. The first item is at index 0.

### Parameters

`index`                    The position within the list of the item to be removed.

### Notes

CORBA-defined.

# Class IE.Iona.OrbixWeb.CORBA.ContextIterator

**Synopsis** Class ContextIterator defines a Java iterator class for Context.

**Orbix Java**

```
public class ContextIterator {
    public ContextIterator() throws
        org.omg.CORBA.SystemException;
    public ContextIterator(org.omg.CORBA.Context ctx)
        throws org.omg.CORBA.SystemException;

    public void setList org.omg.CORBA.Context ctx)
        throws org.omg.CORBA.SystemException;
    public String next() throws org.omg.CORBA.SystemException;
};
```

**Notes** Orbix-specific.

**See also**

["Class org.omg.CORBA.Context" on page 9](#)  
["Class org.omg.CORBA.ContextList" on page 10](#)  
["Class IE.Iona.OrbixWeb.CORBA.Context" on page 83](#)

## ContextIterator()

**Synopsis**

```
public ContextIterator() throws org.omg.CORBA.SystemException;
```

**Description**

Constructor. Creates an iterator without specifying which context object the iterator uses to traverse. You can use the method ["setList\(\)"](#) subsequently to set the context object.

**Notes**

Orbix-specific.

**See Also**

["setList\(\)" on page 93](#)

## ContextIterator()

**Synopsis**

```
public ContextIterator(org.omg.CORBA.Context ctx)
    throws org.omg.CORBA.SystemException;
```

**Description**

Constructor. Creates an iterator for context `ctx`.

**Parameters**

<code>ctx</code>	A context object.
------------------	-------------------

**Notes**

Orbix-specific.

## next()

**Synopsis**

```
public String next() throws org.omg.CORBA.SystemException;
```

**Description**

Returns the next item in the list or `null` if at the end. The  $i^{\text{th}}$  call returns the property name and the  $i+1^{\text{th}}$  call returns the property value within the Context.

**Return Value**

<code>String</code>	Property name or property value.
---------------------	----------------------------------

**Notes**

Orbix-specific.

## setList()

### Synopsis

```
public void setList(org.omg.CORBA.Context ctx)
    throws org.omg.CORBA.SystemException;
```

### Description

If a context object has not been specified for the iterator, you can use this method to set the context object. If a context object has been set, you can use this method to replace the context object (if the `ctx` parameter is not the current context) or to reset the iterator position for the current context (if the `ctx` parameter is the current context).

### Parameters

<code>ctx</code>	Either a new context, in which case this becomes the context to be traversed by the iterator, or the current context in which case the iterator is reset.
------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------

### Notes

Orbix-specific.

# Class IE.Iona.OrbixWeb.CORBA.Environment

## Synopsis

The `Environment` class provides a vehicle for dealing with exceptions in those cases where true exception mechanics are unavailable or undesirable.

For example, in the DII exceptions raised by remote invocation are stored in an `Environment` member variable in the `Request` object after the invocation returns. DII clients should test the value of this `Environment` variable by calling the `env()` method on the `Request` object. If the returned `java.lang.Exception` is null, no exception was raised. If it is not null, the returned exception should be examined and acted on in an appropriate manner.

Refer to the chapter "Dynamic Invocation Interface" in the *Orbix Programmer's Guide Java Edition* for more details.

## Orbix Java

```
// Java

package IE.Iona.OrbixWeb.CORBA;

public class Environment
    extends org.omg.CORBA.Environment {

    void exception(java.lang.Exception except);
    java.lang.Exception exception();
    void clear();
};
```

## Notes

CORBA-defined

## See Also

["Class org.omg.CORBA.Request" on page 29](#)

## exception()

### Synopsis

```
public void exception(java.lang.Exception except);
```

### Description

Sets the exception member variable in the `Environment` object to `except`.

### Notes

CORBA-defined.

## exception()

### Synopsis

```
public java.lang.Exception exception();
```

### Description

Extracts the exception contained in the `Environment` object.

### Return Value

`java.lang.Exception` The returned exception.

### Notes

CORBA-defined.

## clear()

### Synopsis

```
public void clear();
```

### Description

Sets the exception contained within the `Environment` object to null.

### Notes

CORBA-defined.



# Class `IE.Iona.OrbixWeb.CORBA.ExceptionList`

## Synopsis

The Java class `ExceptionList` implements the CORBA pseudo-object type `ExceptionList`. An `ExceptionList` is used in the DII to describe the exceptions that can be raised by IDL operations. It maintains a modifiable list of `Typecodes`.

See the "Dynamic Invocation Interface" chapter in the *Orbix Programmer's Guide Java Edition* for more details.

## CORBA

```
// Pseudo IDL

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

## Orbix Java

```
// Java
public class ExceptionList extends org.omg.CORBA.ExceptionList
{
    public ExceptionList();

    public int count();
    public void add(TypeCode exc)
        throws org.omg.CORBA.SystemException;
    public org.omg.CORBA.TypeCode item(int index)
        throws org.omg.CORBA.Bounds,
               org.omg.CORBA.SystemException;
    public void remove(int index)
        throws org.omg.CORBA.Bounds,
               org.omg.CORBA.SystemException;
}
```

## Notes

CORBA-defined.

## See Also

["exceptions\(\)"](#)  
["\\_create\\_request\(\)"](#)  
["Class IE.Iona.OrbixWeb.CORBA.TypeCode"](#)

## ExceptionList()

### Synopsis

```
public ExceptionList();
```

### Description

Default constructor.

### Notes

Orbix-specific. See `create_exception_list()` in "Class [org.omg.CORBA.ORB](#)" for CORBA-defined ways to create an `ExceptionList`.

### See Also

["Class org.omg.CORBA.ORB" on page 19](#)

## add()

### Synopsis

```
public void add(org.omg.CORBA.TypeCode exc)
    throws org.omg.CORBA.SystemException;
```

### Description

Adds `exc` to the end of the list of exception `TypeCodes`.

### Parameters

`exc`                      The `TypeCode` to be added to the list. Should be a `TypeCode` for an exception.

### Notes

CORBA-defined.

## count()

### Synopsis

```
public int count() throws org.omg.CORBA.SystemException;
```

### Description

Returns the number of elements in the `ExceptionList`.

### Return Value

`int`                      Number of elements in the list.

### Notes

CORBA-defined.

## item()

### Synopsis

```
public org.omg.CORBA.TypeCode item(int index)
    throws org.omg.CORBA.Bounds,
           org.omg.CORBA.SystemException;
```

### Description

Returns the item at the given index. The first item is at `index 0`.

### Parameters

`index`                      Position within list to be returned.

### Return Value

`TypeCode`                      The item at the position specified by `index`.

### Notes

CORBA-defined.

## remove()

### Synopsis

```
public void remove(int index)
    throws org.omg.CORBA.Bounds,
           org.omg.CORBA.SystemException;
```

### Description

Removes the item at the given index. The first item is at `index 0`.

### Parameters

`index`                      The position within the list of the item to be removed.

### Notes

CORBA-defined.

# Class `IE.Iona.OrbixWeb.CORBA.InitService`

## Synopsis

The `InitService` class is responsible for maintaining a set of IORs (Interoperable Object References) that you can retrieve by calling `ORB.resolve_initial_references()`. The `InitService` class can be useful for example, if you need to access more than one Naming Service host from the same client, and you do not want to use a federated Naming Service.

## Orbix Java

```
// Java

public class InitService {

    // Constructor
    public InitService(ORB);

    // Methods
    public static void initialise();;
    ...
}
```

## Notes

Orbix-specific.

## See Also

[“Class `IE.Iona.OrbixWeb.CORBA.ORB`” on page 120](#)

## `InitService()`

### Synopsis

```
public InitService(ORB orb);
```

### Description

The default constructor. Takes the specified ORB as a parameter, enabling support for multiple ORBs.

### Notes

Orbix-specific.

## `initialise()`

### Synopsis

```
public synchronized void initialise(ORB orb);
```

### Description

Initializes the IORs for each service (for example, the Naming Service or the Trading Object Service) for the nominated ORB. It uses configuration information such as `IT_NS_HOSTNAME` and `IT_NS_PORT` when creating these IORs.

This method can be useful for example, if you wish to run two mirrored (not federated) NS hosts. While the client is running it can substitute “backup” configuration into the `IT_NS_HOSTNAME` and `IT_NS_PORT` configuration parameters using `setConfigItem()`. The client can then call `resolve_initial_references()` again to update the IOR to point to the backup NS.

Having to create a new ORB in this situation is not always ideal. The existing client ORB may already have other session information (for example, connections to servers, loaders, filters) that the client does not want to lose in the event that the NS fails.

### Notes

Orbix-specific.

### See Also

***Orbix Administrator’s Guide Java Edition***  
[“Class `IE.Iona.OrbixWeb.Features.OrbConfig`”](#)

# Class IE.Iona.OrbixWeb.CORBA.NamedValue

## Synopsis

The Java class `NamedValue` implements the IDL pseudo-object type `NamedValue` that is used only as an element of an `NVList`, chiefly in the DII. A `NamedValue` describes a parameter to a `Request`. It contains an optional name, an any value and labelling flags.

## CORBA

```
// Pseudo IDL
```

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

## Orbix Java

```
// Java
```

```
public class NamedValue extends org.omg.CORBA.NamedValue  
    implements java.lang.Cloneable  
{  
    // Constructors  
    public NamedValue() throws org.omg.CORBA.SystemException;  
    public NamedValue(String name, org.omg.CORBA.Any value,  
        int arg_modes)  
        throws org.omg.CORBA.SystemException;  
    public NamedValue(NamedValue nv)  
        throws org.omg.CORBA.SystemException;  
  
    // Methods  
    public java.lang.Object clone()  
        throws org.omg.CORBA.SystemException;  
    public boolean equals(java.lang.Object _obj)  
        throws org.omg.CORBA.SystemException;  
    public String toString() throws  
        org.omg.CORBA.SystemException;  
  
    // Data Accessor Methods  
    public String name() throws org.omg.CORBA.SystemException;  
    public org.omg.CORBA.Any value()  
        throws org.omg.CORBA.SystemException;  
    public int flags() throws org.omg.CORBA.SystemException;  
    public static NamedValue _nil()  
        throws org.omg.CORBA.SystemException;  
};
```

## Notes

CORBA-defined.

## See Also

["Class IE.Iona.OrbixWeb.CORBA.NVList"](#)  
["Class IE.Iona.OrbixWeb.CORBA.Request"](#)

## NamedValue()

### Synopsis

```
public NamedValue() throws org.omg.CORBA.SystemException;
```

### Description

Default constructor.

### Notes

Orbix-specific.

### See Also

["add\(\)" on page 104](#)  
["add\\_item\(\)" on page 104](#)  
["add\\_value\(\)" on page 105](#)

## NamedValue()

### Synopsis

```
public NamedValue(String name, org.omg.CORBA.Any value,  
                  int arg_modes) throws org.omg.CORBA.SystemException;
```

### Description

Constructor that supports the initialization of NamedValue member data with specified values.

### Parameters

name	Name associated with the NamedValue.
value	The object contained within the NamedValue.
modes	Parameter passing modes org.omg.CORBA.ARG_IN, org.omg.CORBA.ARG_OUT, org.omg.CORBA.ARG_INOUT.

### Notes

Orbix-specific.

### See Also

["add\(\)" on page 104](#)  
["add\\_item\(\)" on page 104](#)  
["add\\_value\(\)" on page 105](#)

## NamedValue()

### Synopsis

```
public NamedValue(NamedValue nv)  
    throws org.omg.CORBA.SystemException;
```

### Description

Copy constructor.

### Parameters

nv	The NamedValue to copy.
----	-------------------------

### Notes

Orbix-specific.

### See Also

["add\(\)" on page 104](#)  
["add\\_item\(\)" on page 104](#)  
["add\\_value\(\)" on page 105](#)

## clone()

### Synopsis

```
public java.lang.Object clone()  
    throws org.omg.CORBA.SystemException;
```

### Description

Copies the current `NamedValue` to a new `NamedValue` object and returns the new object.

### Notes

Orbix-specific.

## equals()

### Synopsis

```
public boolean equals(java.lang.Object _obj)  
    throws org.omg.CORBA.SystemException;
```

### Description

Compares the current `NamedValue` to parameter `_obj`. If `_obj` is not a `NamedValue` object or the name, flags, and value members of `_obj` are not equal to those of the current object, this method returns false; otherwise, it returns true.

### Parameters

<code>_obj</code>	<code>NamedValue</code> object against which the current <code>NamedValue</code> is compared.
-------------------	-----------------------------------------------------------------------------------------------

### Return Value

boolean	true if objects are equal; false otherwise.
---------	---------------------------------------------

### Notes

Orbix-specific.

## toString()

### Synopsis

```
public String toString() throws org.omg.CORBA.SystemException;
```

### Description

Converts the name and value of the `NamedValue` object to a human-readable string of the form:

```
<name>, <value>
```

### Return Value

String	Human readable form of <code>NamedValue</code> object.
--------	--------------------------------------------------------

### Notes

Orbix-specific.

## name()

### Synopsis

```
public String name() throws org.omg.CORBA.SystemException;
```

### Description

Returns the (optional) name associated with the `NamedValue`. This is the name of a parameter or argument to a request.

### Notes

CORBA-defined.

## value()

### Synopsis

```
public org.omg.CORBA.Any value()  
    throws org.omg.CORBA.SystemException;
```

### Description

Returns a reference to the `org.omg.CORBA.Any` object contained in the `NamedValue`.

### Return Value

Any                      Reference to object contained in the `NamedValue`.

### Notes

CORBA-defined.

## flags()

### Synopsis

```
public int flags() throws org.omg.CORBA.SystemException;
```

### Description

Returns the parameter mode associated with the `NamedValue`.

### Return Value

int                      Flags associated with the `NamedValue`.

### Notes

CORBA-defined.

### See Also

`org.omg.CORBA.Flags`

## \_nil()

### Synopsis

```
public static NamedValue _nil()  
    throws org.omg.CORBA.SystemException;
```

### Description

Returns a nil object reference for a `NamedValue`.

### Notes

CORBA-defined.

# Class `IE.Iona.OrbixWeb.CORBA.NVList`

## Synopsis

The Java class `NVList` implements the CORBA pseudo-object type `NVList`. An `NVList` is a list of `NamedValue` elements. A `NamedValue` describes an argument to a `Request`.

## CORBA

```
// Pseudo IDL

pseudo interface NVList {
    readonly attribute unsigned long count;
    NamedValue add(in Flags flags);
    NamedValue add_item(in Identifier item_name, in Flags
        flags);
    NamedValue add_value(in Identifier item_name, in any val,
        in Flags flags);
    NamedValue item(in unsigned long index) raises
        (CORBA::Bounds);
    void remove(in unsigned long index) raises (CORBA::Bounds);
};
```

## Orbix Java

```
// Java
public class NVList extends org.omg.CORBA.NVList
    implements java.lang.Cloneable {

    public NVList() throws org.omg.CORBA.SystemException;
    public NVList(int size)
        throws org.omg.CORBA.SystemException;
    public NVList(NVList nvl)
        throws org.omg.CORBA.SystemException;

    public boolean equals(java.lang.Object _obj)
        throws org.omg.CORBA.SystemException;
    public Object clone()
        throws org.omg.CORBA.SystemException;

    public org.omg.CORBA.NamedValue add(int item_flags)
        throws org.omg.CORBA.SystemException;
    public org.omg.CORBA.NamedValue add_item(String
        item_name, int item_flags)
        throws org.omg.CORBA.SystemException;

    public org.omg.CORBA.NamedValue add_value(
        String item_name,
        org.omg.CORBA.Any value,
        int item_flags)
        throws org.omg.CORBA.SystemException;

    public int count() throws org.omg.CORBA.SystemException;
    public org.omg.CORBA.NamedValue item(int index)
        throws Bounds, org.omg.CORBA.SystemException;
    public void remove(int index)
        throws Bounds, org.omg.CORBA.SystemException;

    public static NVList _nil()
        throws org.omg.CORBA.SystemException;
};
```

## Notes

CORBA-defined.

## See Also

["Class org.omg.CORBA.NamedValue" on page 16](#)  
["Class org.omg.CORBA.Request" on page 29](#)



## NVList()

### Synopsis

```
public NVList() throws org.omg.CORBA.SystemException;
```

### Description

Default constructor.

### Notes

Orbix-specific.

### See Also

[create\\_list\(\)](#) and [create\\_operation\\_list\(\)](#) in ["Class org.omg.CORBA.ORB" on page 19](#)

## NVList()

### Synopsis

```
public NVList(int size) throws org.omg.CORBA.SystemException;
```

### Description

Constructs an `NVList` of length `size`.

### Parameters

`size`                      Length of `NVList`.

### Notes

Orbix-specific. See the methods [create\\_list\(\)](#) and [create\\_operation\\_list\(\)](#) in ["Class org.omg.CORBA.ORB" on page 19](#) for CORBA-defined ways to create an `NVList`.

## NVList()

### Synopsis

```
public NVList(NVList nvl) throws org.omg.CORBA.SystemException;
```

### Description

Copy constructor.

### Parameters

`nvl`                      `NVList` to copy.

### Notes

Orbix-specific.

## equals()

### Synopsis

```
public boolean equals(java.lang.Object _obj)  
    throws org.omg.CORBA.SystemException;
```

### Description

Compare two `NVList` objects for equality. `NVList` objects are equal if they have the same contents.

### Parameters

`_obj`                      An `NVList` object to compare against.

### Return Value

`boolean`                      `true` if the objects have the same contents, `false` otherwise.

### Notes

Orbix-specific.

## clone()

### Synopsis

```
public Object clone() throws org.omg.CORBA.SystemException;
```

### Description

Copies the current `NVList` to a new `NVList` object and returns the new object.

### Return Value

Object                      A copy of the current `NVList` object.

### Notes

Orbix-specific.

## add()

### Synopsis

```
public org.omg.CORBA.NamedValue add(int item_flags)
    throws org.omg.CORBA.SystemException;
```

### Description

Creates an unnamed `NamedValue`, initializing only the `Flags`, and adds it to the list.

### Parameters

`item_flags`                      `org.omg.CORBA.ARG_IN`,  
                                  `org.omg.CORBA.ARG_OUT`,  
                                  `org.omg.CORBA.ARG_INOUT`

### Return Value

`NamedValue`                      New `NamedValue` instance.

### Notes

CORBA-defined.

## add\_item()

### Synopsis

```
public org.omg.CORBA.NamedValue add_item(String item_name,
                                           int item_flags)
    throws org.omg.CORBA.SystemException;
```

### Description

Creates a `NamedValue` with name and flags initialized, and adds it to the list.

### Parameters

`item_name`                      Name of item.  
`item_flags`                      `org.omg.CORBA.ARG_IN`,  
                                  `org.omg.CORBA.ARG_OUT`,  
                                  `org.omg.CORBA.ARG_INOUT`

### Return Value

`NamedValue`                      New `NamedValue` instance added to list.

### Notes

CORBA-defined.

## add\_value()

### Synopsis

```
public org.omg.CORBA.NamedValue add_value(String item_name,
                                           org.omg.CORBA.Any value,
                                           int item_flags)
    throws org.omg.CORBA.SystemException;
```

### Description

Creates a `NamedValue` with name, value and flags initialized and adds it to the list.

### Parameters

<code>item_name</code>	Name of item.
<code>value</code>	Value of item.
<code>item_flags</code>	<code>CORBA.ARG_IN</code> , <code>CORBA.ARG_OUT</code> , <code>CORBA.ARG_INOUT</code> .

### Return Value

<code>NamedValue</code>	New <code>NamedValue</code> instance.
-------------------------	---------------------------------------

### Notes

CORBA-defined.

## count()

### Synopsis

```
public int count() throws org.omg.CORBA.SystemException;
```

### Description

Returns the number of elements in the `NVList`.

`add_item()` is the only way to add an item to an `NVList`.

`get_length()` returns number of times `add_item()` was called. The `NVList` always starts with element number 0.

### Return Value

<code>int</code>	Number of elements in the list.
------------------	---------------------------------

### Notes

CORBA-defined.

## item()

### Synopsis

```
public org.omg.CORBA.NamedValue item(int index)
    throws Bounds, org.omg.CORBA.SystemException;
```

### Description

Returns the list item at the given index. The first item is at index 0.

### Parameters

<code>index</code>	Position within list to be returned.
--------------------	--------------------------------------

### Return Value

<code>NamedValue</code>	The list item at the position specified by <code>index</code> .
-------------------------	-----------------------------------------------------------------

### Notes

CORBA-defined.

## **\_nil()**

### **Synopsis**

```
public static NVList _nil() throws  
    org.omg.CORBA.SystemException;
```

### **Description**

Returns a nil object reference for an NVList object.

### **Notes**

CORBA-defined.

## **remove()**

### **Synopsis**

```
public void remove(int index)  
    throws Bounds, org.omg.CORBA.SystemException;
```

### **Description**

Removes the item at the given index. The first item is at index 0.

### **Parameters**

<code>index</code>	The position within the list of the item to be removed.
--------------------	---------------------------------------------------------

### **Return Value**

<code>int</code>	Returns 1 on success, 0 on failure.
------------------	-------------------------------------

### **Notes**

CORBA-defined.

# Class `E.Iona.OrbixWeb.CORBA.NVListIterator`

**Synopsis** Class `NVListIterator` defines a Java iterator class for `NVList` which returns the next `NamedValue` in the list.

**Orbix Java**

```
// Java

public class NVListIterator {
    // Constructors
    public NVListIterator() throws
        org.omg.CORBA.SystemException;
    public NVListIterator(NVList nvl)
        throws org.omg.CORBA.SystemException;

    // Methods
    public void setList(NVList nvl)
        throws org.omg.CORBA.SystemException;
    public org.omg.CORBA.NamedValue next()
        throws org.omg.CORBA.SystemException;
};
```

**Notes** Orbix-specific.

**See also** ["Class org.omg.CORBA.NVList" on page 17](#)  
["Class org.omg.CORBA.NamedValue" on page 16](#)

## `NVListIterator()`

**Synopsis** `public NVListIterator() throws org.omg.CORBA.SystemException;`

**Description** Constructor. Creates an iterator without specifying which `NVList` object the iterator uses to traverse. You can subsequently use the method `CORBA.NVList.setList()` to set the `NVList` object.

**Notes** Orbix-specific.

**See Also** ["setList\(\)" on page 108](#)

## `NVListIterator(NVList)`

**Synopsis** `public NVListIterator(NVList nvl)
 throws org.omg.CORBA.SystemException;`

**Description** Constructor. Creates an iterator for `NVList nvl`.

**Parameters**

<code>nvl</code>	The <code>NVList</code> object for which an iterator is to be created.
------------------	------------------------------------------------------------------------

**Return Value**

<code>NVListIterator</code>	The iterator for <code>nvl</code> .
-----------------------------	-------------------------------------

**Notes** Orbix-specific.

## next()

### Synopsis

```
public org.omg.CORBA.NamedValue next()  
    throws org.omg.CORBA.SystemException;
```

### Description

The  $i^{\text{th}}$  call returns the  $i^{\text{th}}$  `NamedValue` object in the `NVList`.

### Return Value

`NamedValue`                      An element in the `NVList`.

### Notes

Orbix-specific.

## setList()

### Synopsis

```
public void setList(NVList nvl)  
    throws org.omg.CORBA.SystemException;
```

### Description

If an `NVList` object has not been specified for the iterator, you can use this method to set the `NVList` object. If an `NVList` object has been set, you can use this method to replace the `NVList` object (if the `nvl` parameter is not the current `NVList`) or to reset the iterator position for the current `NVList` (if `nvl` is the current `NVList`).

### Parameters

`nvl`                                      Either a new `NVList` object to associate with the iterator or the current `NVList` object in which case the iterator position is reset.

### Notes

Orbix-specific.

# Interface IE.Iona.OrbixWeb.CORBA.ObjectRef

## Synopsis

ObjectRef is a Java interface that extends the implementation methods for the IDL `Object` interface.

This interface is implemented by proxy objects generated by the Orbix Java runtime.

In Orbix Java, all objects that implement `ObjectRef` hold their own full object reference in their member data. `ObjectRef` defines extra methods to retrieve object reference fields in string format and to convert between object references and strings.

## CORBA

```
// PIDL

interface ObjectRef {
    boolean is_a(in String Identifier);
    boolean is_equivalent(Object that);
    boolean non_existent();
    unsigned long hash(in unsigned long maximum);
    boolean is_nil();
    Object _duplicate();
    void release();
    InterfaceDef get_interface();
    Status create_request(
        in Context ctx,
        in Identifier operation,
        in NVList arg_list,
        in NamedValue result,
        out Request request,
        in Flags req_flags);
    Status create_request(
        in Context ctx,
        in Identifier operation,
        in NVList arg_list,
        in NamedValue result,
        int ExceptionList exclist,
        out Request request,
        in Flags req_flags);
};
```

## Orbix Java

See ["Interface org.omg.CORBA.Object"](#) on page 18

```
// Java
package IE.Iona.OrbixWeb.CORBA;

public interface ObjectRef extends
    org.omg.CORBA.Object {
    // Methods
    public String _host();
    public int _port();
    public String _id();
    public String _name();
    public String _implementation();
    public String _marker();
    public boolean _marker(String marker);
    public String _interfaceHost();
    public String _interfaceImplementation();
    public String _interfaceMarker();
    public String _object_to_string();
    public String _object_to_string(int kind);
}
```

```

    public String toString();
    public boolean _isRemote();
    public void _IT_PING();
    public LoaderClass _loader ();
    public java.lang.Object _deref ();
    public void _save ();
    public boolean _hasValidOpenChannel();
}

```

**See Also** ["Interface org.omg.CORBA.Object" on page 18](#)

## **\_create\_request()**

### **Synopsis**

```

Request _create_request(Context ctx,
                        String operation,
                        NVList arg_list,
                        NamedValue result);

```

### **Description**

Constructs an `org.omg.CORBA.Request` object. See "[\\_request\(\)](#)" for an alternative way to create a `Request`. See the chapter "Dynamic Invocation Interface" in the *Orbix Programmer's Guide Java Edition* for examples of the use of this function.

### **Parameters**

<code>ctx</code>	Context object, if any, to be sent in the <code>Request</code> . If the <code>ctx</code> argument to <code>_create_request()</code> is null, the Context may be added by calling the <code>ctx()</code> function on the <code>Request</code> object.
<code>operation</code>	The name of the operation.
<code>arg_list</code>	The parameter (each parameter in the list is of type <code>NamedValue</code> ). If the <code>arg_list</code> argument of the constructor is null, the arguments must be added by calling the <code>arguments()</code> method on the <code>Request</code> object — one call to <code>arguments()</code> for each argument that is to be passed.
<code>result</code>	Contains the return value of the operation.

### **Return Value**

The constructed `Request` object.

### **Notes**

CORBA-defined. This function is part of the Dynamic Invocation Interface.

### **See Also**

["\\_request\(\)" on page 117](#)  
["Class IE.Iona.OrbixWeb.CORBA.Request" on page 167](#)  
["Class IE.Iona.OrbixWeb.CORBA.Context" on page 83](#)  
["arguments\(\)" on page 173](#)  
["ctx\(\)" on page 174](#)  
["Class IE.Iona.OrbixWeb.CORBA.NVList" on page 102](#)



## `_create_request()`

### Synopsis

```
Request _create_request(Context ctx,  
                        String operation,  
                        NVList arg_list,  
                        NamedValue result,  
                        ExceptionList exclist,  
                        ContextList ctxlist);
```

### Description

Constructs an `org.omg.CORBA.Request` object. See [“\\_request\(\)” on page 117](#) for an alternative way to create a `Request`. See the chapter “Dynamic Invocation Interface” in the ***Orbix Programmer’s Guide Java Edition*** for examples of the use of this function.

### Parameters

<code>ctx</code>	Context object, if any, to be sent in the <code>Request</code> . If the <code>ctx</code> argument to <code>_create_request()</code> is null, the <code>Context</code> may be added by calling the <code>ctx()</code> function on the <code>Request</code> object.
<code>operation</code>	The name of the operation.
<code>arg_list</code>	The parameter (each parameter in the list is of type <code>NamedValue</code> ). If the <code>arg_list</code> argument of the constructor is null, the arguments must be added by calling the <code>arguments()</code> method on the <code>Request</code> object—one call to <code>arguments()</code> for each argument to be passed.
<code>result</code>	Contains the return value of the operation.
<code>exclist</code>	Allows an application to provide a list of <code>TypeCodes</code> for all user-defined exceptions that may result when the <code>Request</code> is invoked.
<code>ctxlist</code>	Allows an application to provide a list of <code>Context</code> strings that must be supplied with the <code>Request</code> invocation.

### Return Value

The constructed `Request` object.

### Notes

CORBA-defined. This function is part of the Dynamic Invocation Interface.

### See Also

[“\\_request\(\)” on page 117](#)  
[“Class IE.Iona.OrbixWeb.CORBA.Request” on page 167](#)  
[“Class IE.Iona.OrbixWeb.CORBA.Context” on page 83](#)  
[“Class IE.Iona.OrbixWeb.CORBA.NVList” on page 102](#)  
[“Class IE.Iona.OrbixWeb.CORBA.NamedValue” on page 98](#)  
[“Class IE.Iona.OrbixWeb.CORBA.ExceptionList” on page 95](#)  
[“Class IE.Iona.OrbixWeb.CORBA.ContextList” on page 90](#)

## **`_deref()`**

### **Synopsis**

```
public java.lang.Object _deref();
```

### **Description**

In the TIE approach, two objects exist: the TIE object and the true object. The reference returned by `_deref()` is that of the true implementation object. This function is defined to allow 'casting' from an interface class to an implementation class. In the TIE approach, a direct cast down is not permitted because an implementation class does not inherit from its IDL Java class. For example:

```
//Java
//TIE Implementation.
//Account is the CORBA object,
//Account_i is the implementation class.
Account acc;
Account_i acc_i = (Account_i)acc; //Illegal
```

You may wish to use a function defined on an implementation class but not in the IDL interface. To use this function requires a reference to the implementation class. The IDL compiler generates a `_deref()` method defined for all TIE interface classes that returns a reference to the implementation class, if it exists.

Also when using the `ImplBase` approach, the IDL compiler generates a `_deref()` method that returns a reference to the `ImplBase` object. However, `_deref()` could be implemented in the implementation class as follows:

```
//Java
public class Account_i extends Account {
    java.lang.Object _deref() { return this; }
};
```

### **Notes**

Orbix-specific.

## **`_get_implementation()`**

### **Synopsis**

```
public ImplementationDef _get_implementation();
```

### **Description**

Find the name of the target object's server as registered in the Implementation Repository. For a local object in a server, this is the server's name if it is known, otherwise it is the process identifier of the server process. For an object created in a client, it is the process identifier of the client process. Note that the server name will be known if the server is launched by the Orbix daemon; or if it is launched manually and the server name is passed to `IE.Iona.OrbixWeb.CORBA.BOA.impl_is_ready()` or set by `IE.Iona.OrbixWeb.CORBA.BOA.setServerName()`.

### **Notes**

CORBA-defined.

## **`_get_interface_def()`**

### **Synopsis**

```
public InterfaceDef _get_interface_def();
```

### **Description**

Returns a reference to an object in the Interface Repository that describes the interface to this object.

The Interface Repository must have been previously populated with this information for this command to work.

### **Notes**

CORBA-defined.

## **\_hash()**

### **Synopsis**

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

### **Description**

Every object reference has an internal identifier associated with it—a value that remains constant throughout the lifetime of the object reference.

The `_hash()` function returns a hashed value determined via a hashing function from the internal identifier. Two different object references may yield the same hashed value. However, if two object references return different hash values, these object references are known to be for different objects.

### **Parameters**

<code>maximum</code>	The maximum value to be returned from the hash function. For example, setting <code>maximum</code> to 7 partitions the object reference space into a maximum of 8 sub-spaces (because the lower bound of the function is 0).
----------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### **Return Value**

A hashed value for the object reference in the range 0...maximum.

### **Notes**

CORBA-defined.

## **\_hasValidOpenChannel()**

### **Synopsis**

```
public boolean _hasValidOpenChannel();
```

### **Description**

This method returns `true` if there is a valid open connection between this process and the process in which this object resides.

### **Return Value**

<code>boolean</code>	Returns <code>true</code> if there is a valid, open connection between this process and the process in which this object resides.
----------------------	-----------------------------------------------------------------------------------------------------------------------------------

### **Notes**

Orbix-specific.

### **See Also**

`IE.Iona.OrbixWeb.CORBA.ORB.hasValidOpenChannel()`

## **\_host()**

### **Synopsis**

```
public String _host();
```

### **Description**

Returns the name of the host on which the target object is located.

### **Notes**

Orbix-specific.

## **\_id()**

### **Synopsis**

```
public String _id();
```

### **Description**

Returns the object's Interface Repository ID.

### **Notes**

Orbix-specific.

## **\_implementation()**

**Synopsis**

```
public String _implementation();
```

**Description**

Find the name of the target object's server as registered in the Implementation Repository. For a local object in a server, this is the server's name if known, otherwise it is the process identifier of the server process. For an object created in a client, it is the process identifier of the client process. Note that the server name is known if the server is launched by the Orbix daemon; or if it is launched manually and the server name is passed to `IE.Iona.OrbixWeb.CORBA.BOA.impl_is_ready()` or set by `IE.Iona.OrbixWeb.CORBA.BOA.setServerName()`.

**Notes**

Orbix-specific. The CORBA-defined version of this function is ["\\_get\\_interface\\_def\(\);" on page 112](#)

**See Also**

["\\_get\\_implementation\(\);" on page 112](#)

## **\_interfaceHost()**

**Synopsis**

```
public String _interfaceHost();
```

**Description**

Returns the name of a host running the Interface Repository server that stores the target object's IDL definition.

**Notes**

Orbix-specific.

## **\_interfaceImplementation()**

**Synopsis**

```
public String _interfaceImplementation();
```

**Description**

Returns the object's Interface Repository server name. The default is "IR".

**Notes**

Orbix-specific.

## **\_interfaceMarker()**

**Synopsis**

```
public String _interfaceMarker();
```

**Description**

Returns the name of the target object's interface.

**Notes**

Orbix-specific.

## `_is_a()`

### Synopsis

```
public boolean _is_a(String logical_type_id);
```

### Description

Determines if the target object is an instance of the type specified in `logical_type_id` or is an instance of a derived type of the type in `logical_type_id`.

### Parameters

<code>logical_type_id</code>	The fully scoped name of the IDL interface. You should use a forward slash ('/') rather than a scope operator ('.') to delimit scope in a name.
------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------

### Return Value

<code>true</code>	The object is an instance of the type specified by <code>logical_type_id</code> or is an instance of a type derived from that type.
<code>false</code>	The object is not an instance of that type or any of its derived types.

### Notes

CORBA-defined.

### See Also

["Interface org.omg.CORBA.Object" on page 18](#)  
["\\_non\\_existent\(\)" on page 117](#)

## `_is_equivalent()`

### Synopsis

```
public boolean _is_equivalent(Object that);
```

### Description

Tests if two object references are equivalent. Two object references are said to be equivalent if they have the same object reference, or they both refer to the same object.

### Parameters

<code>obj</code>	The object that is to be compared for equivalence with the target object.
------------------	---------------------------------------------------------------------------

### Return Value

<code>true</code>	The object reference refers to the same object.
<code>false</code>	This return value does not mean that the object references are not equivalent—only that the ORB cannot confirm that they reference the same object.

### Notes

CORBA-defined.

### See Also

["Interface org.omg.CORBA.Object" on page 18](#)  
["\\_is\\_a\(\)" on page 115](#)

## **\_isRemote()**

### **Synopsis**

```
public boolean _isRemote();
```

### **Description**

Returns `true` if the reference is to a remote object, `false` otherwise.

### **Notes**

Orbix-specific.

## **IT\_PING()**

### **Synopsis**

```
public void _IT_PING();
```

### **Description**

This method verifies that a remote object exists. If no connection exists to the target object's server, Orbix Java attempts to create a connection, relaunching the server if necessary. If this fails, or the object is not located in the server specified in the object reference, an `INVALID_OBJREF` system exception is thrown.

The target object is guaranteed to exist if this operation completes normally.

### **Notes**

Orbix-specific.

### **See Also**

["noReconnectOnFailure\(\)" on page 147](#)

## **\_loader()**

### **Synopsis**

```
public LoaderClass _loader();
```

### **Description**

Returns the `loaderClass` object associated with this object. This call is only available within the objects' server process space.

### **Notes**

Orbix-specific.

## **\_marker()**

### **Synopsis**

```
public String _marker();
```

### **Description**

Returns the object's marker.

### **Notes**

Orbix-specific.

## **\_marker()**

### **Synopsis**

```
public boolean _marker(String marker);
```

### **Description**

This method attempts to reset an object's marker to the string specified.

### **Return Value**

<code>true</code>	The object's marker was successfully changed.
<code>false</code>	There was a problem assigning the new marker to the object. This is typically because an object already exists that has the specified marker.

### **Notes**

Orbix-specific.

## **\_name()**

### **Synopsis**

```
public String _name();
```

### **Description**

Returns the object's interface name.

### **Notes**

Orbix-specific.

## **\_non\_existent()**

### **Synopsis**

```
public boolean _non_existent();
```

### **Description**

Tests if the target exists. Normally this function is invoked on a proxy and determines whether the real object exists.

### **Return Value**

true	The ORB knows that the object does not exist (it does not raise an exception if the object does not exist).
false	Either the object exists, or the ORB cannot determine whether it exists or not.

### **Notes**

CORBA-defined.

## **\_port()**

### **Synopsis**

```
public int _port()
```

### **Description**

Returns the object's server listening port.

### **Notes**

Orbix-specific.

## **\_request()**

### **Synopsis**

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

### **Description**

Constructs an `org.omg.CORBA.Request` on the target object. This is the shorter form of `IE.Iona.OrbixWeb.CORBA.ObjectRef._create_request()`. You may add arguments and contexts after construction using `IE.Iona.OrbixWeb.CORBA.Request.arguments()` and `IE.Iona.OrbixWeb.CORBA.Request.ctx()`. See the chapter "Dynamic Invocation Interface" in the *Orbix Programmer's Guide Java Edition* for examples of the use of this function.

### **Parameters**

operation	The name of the operation.
-----------	----------------------------

### **Return Value**

Request	The Request object constructed.
---------	---------------------------------

### **Notes**

CORBA-defined.

### **See Also**

["\\_create\\_request\(\)" on page 110](#)  
["arguments\(\)" on page 173](#)  
["ctx\(\)" on page 174](#)

## **\_object\_to\_string()**

### **Synopsis**

```
public String _object_to_string();
```

### **Description**

Converts the target object's reference to a string. The format of that string depends on the communication protocol used to create that object.

If you are using the IIOP protocol, this function creates a stringified Interoperable Object Reference, which is a string of hexadecimal numbers, beginning with the string 'IOR:'.

If you are using the Orbix protocol, a stringified object reference has the form:

```
:\host:serverName:marker:IR_host:IR_Server:interfaceMarker
```

See ["object\\_to\\_string\(\)"](#) for an explanation of these fields.

### **Return Value**

Returns a stringified object reference.

### **Notes**

Orbix-specific.

### **See Also**

["object\\_to\\_string\(\)"](#) on page 148

["\\_object\\_to\\_string\(\)"](#) on page 118

## **\_object\_to\_string()**

### **Synopsis**

```
public String _object_to_string(int kind);
```

### **Description**

Converts the target object's reference to a string. The format of the string returned is determined by the value of parameter `kind`.

If the value of `kind` is `IT_ORBIX_OR_KIND`, the string returned is the standard Orbix Protocol object reference string format:

```
:\host:serverName:marker:IR_host:IR_Server:interfaceMarker
```

See ["object\\_to\\_string\(\)"](#) on page 148 for an explanation of these fields.

If the value of `kind` is `IT_INTEROPERABLE_OR_KIND`, then the string returned is a CORBA Interoperable Object Reference (IOR) for use with the Internet Inter-ORB Protocol (IIOP) as described in the chapter "ORB Interoperability" in the *Orbix Programmer's Guide Java Edition*.

### **Return Value**

Returns a stringified object reference.

### **Notes**

Orbix-specific.

### **See Also**

["object\\_to\\_string\(\)"](#) on page 148

["\\_object\\_to\\_string\(\)"](#) on page 118

## **\_save()**

### **Synopsis**

```
public void _save();
```

### **Description**

Calls the `save()` method on the `LoaderClass` associated with this object. The reason provided is `_CORBA.explicitCall`.

Call `_save()` when you want to make the state of the object persistent but:

- do not wish to wait until the server exits (see ["finalize\(\)"](#)),
- and want to keep the object registered with the ORB (see ["disconnect\(\)"](#)).



The implementation of `save()` in the default `LoaderClass` object does nothing.

You must call `_save()` in the same server space as the target object. If you call it in a client process (on a proxy), there is no effect.

**Notes**

Orbix-specific.

**See Also**

["explicit\\_call"](#)

["disconnect\(\)" on page 67](#)

["dispose\(\)" on page 68](#)

["finalize\(\)" on page 69](#)

["Class IE.Iona.OrbixWeb.Features.LoaderClass"](#)

["save\(\)"](#)

# Class `IE.Iona.OrbixWeb.CORBA.ORB`

## Synopsis

Class `ORB` implements the OMG CORBA `ORB` pseudo-interface and adds a number of methods specific to Orbix Java.

`ORB` provides a set of methods that control Orbix Java from the client. These include operations to convert between strings and object references, and operations for use with the Dynamic Invocation Interface (DII).

To maintain portability of your client-side CORBA code, you must use only those methods defined in pseudo IDL and defined on `org.omg.CORBA.ORB`. To use these methods, invoke on the `ORB` object as follows:

```
ORB.init(args,null).<method name>
```

To access the Orbix Java value added methods, use the `_OrbixWeb` conversion function as follows:

```
_OrbixWeb.ORB(ORB.init(args,null)).<Orbix Java specific method name>
```

or equivalently

```
_CORBA.Orbix.<Orbix Java specific method name>
```

## CORBA

```
// Pseudo IDL
```

```
pseudo interface ORB {
```

```
    exception InvalidName {};
```

```
    // Typedefs
```

```
    typedef string ObjectId;
```

```
    typedef sequence<ObjectId> ObjectIdList;
```

```
    // Service methods
```

```
    ObjectIdList list_initial_services();
```

```
    Object resolve_initial_references  
        (in ObjectId object_name)  
        raises(InvalidName);
```

```
    // Object Reference methods
```

```
    string object_to_string(in Object obj);
```

```
    Object string_to_object(in string str);
```

```
    // Creation methods
```

```
    NVList create_list(in long count);
```

```
    NVList create_operation_list  
        (in OperationDef oper);
```

```
    NamedValue create_named_value(in String name,  
                                   in Any value,  
                                   in Flags flags);
```

```
    ExceptionList create_exception_list();
```

```
    ContextList create_context_list();
```

```
    Context get_default_context();
```

```
    Environment create_environment();
```

```
    Any create_any();
```

```
    OutputStream create_output_stream();
```

```

// Typecode creation
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);
TypeCode create_enum_tc
    (    in RepositoryId id,
        in Identifier name,
        in EnumMemberSeq members);
TypeCode create_alias_tc
    (    in RepositoryId id,
        in Identifier name,
        in TypeCode original_type);
TypeCode create_exception_tc
    (    in RepositoryId id,
        in Identifier name,
        in StructMemberSeq members);
TypeCode create_interface_tc
    (    in RepositoryId id,
        in Identifier name);
TypeCode create_string_tc
    (    in unsigned long bound);
TypeCode create_wstring_tc
    (    in unsigned long bound);
TypeCode create_sequence_tc
    (    in unsigned long bound,
        in TypeCode element_type);
TypeCode create_recursive_sequence_tc
    (    in unsigned long bound,
        in unsigned long offset);
TypeCode create_array_tc
    (    in unsigned long length,
        in TypeCode element_type);
TypeCode get_primitive_tc(in TCKind tcKind);

// Invocation methods
void send_multiple_requests_oneway
    (in RequestSeq
req);
void send_multiple_requests_deferred
    (in RequestSeq
req);
boolean poll_next_response();
Request get_next_response();

// Server-side methods
void connect(Object obj);
void disconnect(Object obj);
Current get_current();

// Additional operations for Java mapping

```

```

// additional methods for ORB initialization go
// here, but only appear in the mapped Java
// Java signatures
// public static ORB init
//             (   Strings[] args,
//               Properties props);
// public static ORB init
//             (   Applet app,
//               Properties props);
// public static ORB init();
// abstract protected void set_parameters
//             (   String[] args,
//               java.util.Properties props);

// abstract protected void set_parameters
//             (   java.applet.Applet app,
//               java.util.Properties props);
};

```

## Orbix Java

```

public class ORB extends IE.Iona.OrbixWeb.CORBA.singletonORB {
    public String[] baseInterfacesOf(String derivedStubClass)
        throws org.omg.create_opSystemException;
    public boolean closeConnection(org.omg.CORBA.Object oref)
        throws org.omg.CORBA.SystemException;
    public boolean collocated()
        throws org.omg.CORBA.SystemException;
    public boolean collocated(boolean b)
        throws org.omg.CORBA.SystemException;
    public final OrbConfig config()
        throws org.omg.CORBA.SystemException;
    public org.omg.CORBA.TypeCode create_alias_tc(
        String id,
        String name,
        org.omg.CORBA.TypeCode original_type)
        throws org.omg.CORBA.SystemException;
    public org.omg.CORBA.Any create_any()
        throws org.omg.CORBA.SystemException;
    public org.omg.CORBA.TypeCode create_array_tc(int length,
        org.omg.CORBA.TypeCode element_type)
        throws org.omg.CORBA.SystemException;
    public org.omg.CORBA.ContextList create_context_list()
        throws org.omg.CORBA.SystemException;
    public org.omg.CORBA.TypeCode create_enum_tc(String id,
        String name,
        String[] members)
        throws org.omg.CORBA.SystemException;
    public org.omg.CORBA.Environment create_environment()
        throws org.omg.CORBA.SystemException;
    public org.omg.CORBA.ExceptionList create_exception_list()
        throws org.omg.CORBA.SystemException;
    public org.omg.CORBA.TypeCode create_exception_tc(
        String id,
        String name,
        org.omg.CORBA.StructMember[] members)
        throws org.omg.CORBA.SystemException;
    public org.omg.CORBA.TypeCode create_interface_tc(String id,
        String name)
        throws org.omg.CORBA.SystemException;
    public org.omg.CORBA.NVLlist create_list(int count)
        throws org.omg.CORBA.SystemException;

```

```

public org.omg.CORBA.NamedValue create_named_value(
    String name,
    org.omg.CORBA.Any value,
    int flags)
    throws org.omg.CORBA.SystemException;
public org.omg.CORBA.NVList
    create_operation_list(org.omg.CORBA.OperationDef oper)
    throws org.omg.CORBA.SystemException;
public org.omg.CORBA.portable.OutputStream
    create_output_stream()
    throws org.omg.CORBA.SystemException;
public org.omg.CORBA.TypeCode create_recursive_sequence_tc(
    int bound,
    int offset)
    throws org.omg.CORBA.SystemException;
public org.omg.CORBA.TypeCode create_sequence_tc(
    int bound,
    TypeCode element_type)
    throws org.omg.CORBA.SystemException;
public org.omg.CORBA.TypeCode create_string_tc(int bound)
    throws org.omg.CORBA.SystemException;
public org.omg.CORBA.TypeCode create_struct_tc(
    String id,
    String name,
    org.omg.CORBA.StructMember[] members)
    throws org.omg.CORBA.SystemException;
public org.omg.CORBA.TypeCode create_union_tc(
    String id,
    String name,
    org.omg.CORBA.TypeCode disc_type,
    org.omg.CORBA.UnionMember[] members)
    throws org.omg.CORBA.SystemException;
public org.omg.CORBA.TypeCode create_wstring_tc(int bound)
    throws org.omg.CORBA.SystemException;
public int defaultTxTimeout(int val)
    throws org.omg.CORBA.SystemException;
public void finalize()
    throws org.omg.CORBA.SystemException;
public static String getConfigItem (String name)
    throws org.omg.CORBA.SystemException;
public static Properties getConfiguration ()
    throws org.omg.CORBA.SystemException;
public DaemonMgr[] getDaemonConnections()
    throws org.omg.CORBA.SystemException;
public org.omg.CORBA.Context get_default_context()
    throws org.omg.CORBA.SystemException;
public int getHostPort(String hostname)
    throws org.omg.CORBA.SystemException;
public org.omg.CORBA.Principal get_my_principal()
    throws org.omg.CORBA.SystemException;
public IT_reqTransformer getMyReqTransformer()
    throws org.omg.CORBA.SystemException;
public org.omg.CORBA.Request get_next_response()
    throws org.omg.CORBA.SystemException;
public org.omg.CORBA.TypeCode get_primitive_tc(
    org.omg.CORBA.TCKind tcKind)
    throws org.omg.CORBA.SystemException;
public org.omg.CORBA.Principal get_principal()
    throws org.omg.CORBA.SystemException;

```

```

public static IT_reqTransformer getTransformer(String server,
                                             String host)
    throws org.omg.CORBA.SystemException;
public String get_principal_string()
    throws org.omg.CORBA.SystemException;
public boolean isValidOpenChannel(ObjectRef oref)
    throws org.omg.CORBA.SystemException;
public boolean isValidOpenChannel(org.omg.CORBA.Object oref)
    throws org.omg.CORBA.SystemException;
static public synchronized org.omg.CORBA.ORB init ()
    throws org.omg.CORBA.SystemException;
static public org.omg.CORBA.ORB init (String[] args,
                                       java.util.Properties props)
    throws org.omg.CORBA.SystemException,
           org.omg.CORBA.INITIALIZE;
static public org.omg.CORBA.ORB init(java.util.Properties props)
    throws org.omg.CORBA.SystemException,
           org.omg.CORBA.INITIALIZE;
static public org.omg.CORBA.ORB init (java.applet.Applet app,
                                       java.util.Properties props)
    throws org.omg.CORBA.SystemException,
           org.omg.CORBA.INITIALIZE;
public boolean isBaseInterfaceOf(String derivedStubClass,
                                 String maybeABase)
    throws org.omg.CORBA.SystemException;
public boolean isBaseInterfaceOf(org.omg.CORBA.Object obj,
                                 String maybeABase)
    throws org.omg.CORBA.SystemException;
public String[] list_initial_services()
    throws org.omg.CORBA.SystemException;
public final void locator(locatorClass l)
    throws org.omg.CORBA.SystemException;
public String makeIOR(String ip_addr,
                     int port,
                     String orbixHost,
                     String server,
                     String marker,
                     String typeID)
    throws org.omg.CORBA.SystemException;
public String makeIOR(String ip_addr,
                     int port,
                     byte[] objKey,
                     String typeID)
    throws org.omg.CORBA.SystemException;
public int maxConnectRetries(int retries)
    throws org.omg.CORBA.SystemException;
public String myHost() {
    throws org.omg.CORBA.SystemException;
public static ORB _nil()
    throws org.omg.CORBA.SystemException;
public boolean noReconnectOnFailure(boolean b)
    throws org.omg.CORBA.SystemException;
public String object_to_string(org.omg.CORBA.Object oref)
    throws org.omg.CORBA.SystemException;
public boolean pingDuringBind(boolean pingOn)
    throws org.omg.CORBA.SystemException;
public boolean poll_next_response()
    throws org.omg.CORBA.SystemException;
public void reSizeConnectionTable(int size)
    throws org.omg.CORBA.SystemException;

```

```

public void registerIOCallback(ioCallback cb)
    throws org.omg.CORBA.SystemException;

public org.omg.CORBA.Object resolve_initial_references(
    String serviceName)
    throws org.omg.CORBA.SystemException;
public void send_multiple_requests_deferred(
    org.omg.CORBA.Request[] requests)
    throws org.omg.CORBA.SystemException;
public void send_multiple_requests_oneway(
    org.omg.CORBA.Request[] requests)
    throws org.omg.CORBA.SystemException;
public static void setConfigItem (String name, String value)
    throws org.omg.CORBA.SystemException;
public static void setConfiguration (Properties props)
    throws org.omg.CORBA.SystemException;
public static int setDiagnostics(int level)
    throws org.omg.CORBA.SystemException;
public void setHostPort(String hostname, int port)
    throws org.omg.CORBA.SystemException;
public IT_reqTransformer setMyReqTransformer(
    IT_reqTransformer new_transform)
    throws org.omg.CORBA.SystemException;
public void set_parameters(String[] args,
    java.util.Properties props)
    throws org.omg.CORBA.SystemException;
public void set_parameters(java.applet.Applet app,
    java.util.Properties props)
    throws org.omg.CORBA.SystemException;
public void set_principal(String new_user)
    throws org.omg.CORBA.SystemException;
public void set_principal(org.omg.CORBA.Principal new_user)
    throws org.omg.CORBA.SystemException;
public void setReqTransformer(IT_reqTransformer new_transform,
    String serverName,
    String host)
    throws org.omg.CORBA.SystemException;
public org.omg.CORBA.Object string_to_object(String host,
    String IR_host,
    String ServerName,
    String marker,
    String IR_server,
    String interfaceName)
    throws org.omg.CORBA.SystemException;
public org.omg.CORBA.Object string_to_object(String s)
    throws org.omg.CORBA.SystemException;
public String toString()
    throws org.omg.CORBA.SystemException;

public void unregisterIOCallback()
    throws org.omg.CORBA.SystemException;
}

```

## Notes

## See Also

CORBA-defined.

["Class org.omg.CORBA.ORB" on page 19](#)

["Class org.omg.CORBA.ORBPackage.InvalidName" on page 24](#)

["Class IE.Iona.OrbixWeb.\\_OrbixWeb"](#)

["Interface IE.Iona.OrbixWeb.CORBA.BOA" on page 59](#)

**Server-Side:**

See ["Interface IE.Iona.OrbixWeb.CORBA.BOA"](#) on page 59 for details.

```
public boolean anyClientsConnected()
public void change_implementation(org.omg.CORBA.Object obj,
                                 java.lang.String impl)
public void connect(org.omg.CORBA.Object obj)
public void connect( org.omg.CORBA.Object obj,
                    LoaderClass loader)
public void connect( org.omg.CORBA.Object obj,
                    String marker)
public void connect( org.omg.CORBA.Object obj,
                    String marker,
                    LoaderClass loader)
public void continueThreadDispatch (org.omg.CORBA.Request r)
public org.omg.CORBA.Object create( byte[] id,
                                   java.lang.String intf,
                                   java.lang.String impl)
public void deactivate_impl (String impl)
public void deactivate_obj(org.omg.CORBA.Object obj)
public void disconnect(org.omg.CORBA.Object obj)
public void dispose (org.omg.CORBA.Object obj)
public boolean enableLoaders (boolean b)
public boolean filterBadConnectAttempts (boolean b)
public org.omg.CORBA.Current get_current()
public byte[] get_id(org.omg.CORBA.Object obj)
public org.omg.CORBA.Principal get_principal
                               (org.omg.CORBA.Object obj)
public void impl_is_ready ()
public void impl_is_ready (String serverName)
public void impl_is_ready (String serverName, int timeOut)
public void impl_is_ready (int timeOut)
public boolean isEventPending ()
public short myActivationMode ()
public String myImplementationName ()
public String myMarkerName ()
public String myMarkerPattern ()
public String myMethodName ()
public int numClientsConnected ()
public void obj_is_ready (org.omg.CORBA.Object obj,
                         java.lang.String impl,
                         int timeOut)
public void obj_is_ready (org.omg.CORBA.Object obj,
                         java.lang.String impl)
public int processEvents ()
public int processEvents (int timeOut)
public int processNextEvent (int timeOut)
public int processNextEvent ()
public org.omg.CORBA.Current set_current()
public boolean setNoHangup (boolean b)
public synchronized void setServerName (String serverName)
```



## baseInterfacesOf()

### Synopsis

```
public String[] baseInterfacesOf(String derivedStubClass)
    throws org.omg.CORBA.SystemException;
```

### Description

For a given Java stub, return a list of supported IDL interfaces. The interface `derivedStubClass` is also returned in the sequence, because it is considered a base interface of itself.

### Return Value

`String []` An array of strings containing all the base interfaces for this type of object.

### Notes

Orbix-specific.

### See Also

The method `isBaseInterfaceOf()` in ["Class org.omg.CORBA.ORB" on page 19](#).

## closeConnection()

### Synopsis

```
public boolean closeConnection(org.omg.CORBA.Object oref)
    throws org.omg.CORBA.SystemException;
```

### Description

Used to explicitly close a connection associated with an object. Returns true if connection was successfully closed.

### Parameters

`oref` The object whose connection is to be closed.

### Return Value

`boolean` Returns true if the call was successful.

### Notes

Orbix-specific.

### See Also

["Class org.omg.CORBA.ORB" on page 19](#)

## collocated()

### Synopsis

```
public boolean collocated()
    throws org.omg.CORBA.SystemException;
```

### Description

This method returns true if collocation is set. If set to true, `bind()` and `string_to_object()`, invocations to objects outside this address space are not allowed.

### Return Value

`boolean` Returns true if collocation is set, otherwise it returns false (the default is false).

### Notes

Orbix-specific.

### See Also

`string_to_object()` in ["Class org.omg.CORBA.ORB" on page 19](#).

## collocated()

### Synopsis

```
public boolean collocated(boolean b)
    throws org.omg.CORBA.SystemException;
```

### Description

This methods sets collocation flag to boolean value passed in and returns the previous value. If collocation is set to `true`, `bind()` and `string_to_object()`, invocations to objects outside this address space are not allowed. By default, collocation is set to `false` (off).

### Parameters

`b`                                    The new setting for collocation flag.

### Return Value

`boolean`                            The old value of the collocated flag.

### Notes

Orbix-specific.

### See Also

`collocated()` in ["Class org.omg.CORBA.ORB" on page 19](#)

## config()

### Synopsis

```
public final OrbConfig config();
```

### Description

This method initializes per-ORB configuration. It enables you to get and set Orbix Java configuration parameters for the ORB, using the methods provided in class `IE.Iona.OrbixWeb.Features.OrbConfig`.

`OrbConfig` calls should be made on the object returned by calling `config()` on the selected ORB. For example, `myOrb.config().getConfigItem("IT_BIND_USING_IIOP")`.

### Notes

Orbix-specific.

### See Also

["Class IE.Iona.OrbixWeb.Features.OrbConfig"](#)

## create\_tc()

### Synopsis

```
public org.omg.CORBA.TypeCode create_alias_tc(
    String id,
    String name,
    org.omg.CORBA.TypeCode original_type)
    throws org.omg.CORBA.SystemException;
public org.omg.CORBA.TypeCode create_array_tc(
    int length,
    org.omg.CORBA.TypeCode element_type)
    throws org.omg.CORBA.SystemException;
public org.omg.CORBA.TypeCode create_enum_tc(String id,
    String name,
    String[] members)
    throws org.omg.CORBA.SystemException;
public org.omg.CORBA.TypeCode create_exception_tc(
    String id,
    String name,
    org.omg.CORBA.StructMember[] members)
    throws org.omg.CORBA.SystemException;
public org.omg.CORBA.TypeCode create_interface_tc(String id,
    String name)
    throws org.omg.CORBA.SystemException;
public org.omg.CORBA.TypeCode create_recursive_sequence_tc(
    int bound,
    int offset)
    throws org.omg.CORBA.SystemException;
public org.omg.CORBA.TypeCode create_sequence_tc(public
    int bound,
    TypeCode element_type)
    throws org.omg.CORBA.SystemException;
public org.omg.CORBA.TypeCode create_string_tc(int bound)
    throws org.omg.CORBA.SystemException;
public org.omg.CORBA.TypeCode create_struct_tc(
    String id,
    String name,
    org.omg.CORBA.StructMember[] members)
    throws org.omg.CORBA.SystemException;
public org.omg.CORBA.TypeCode create_union_tc(
    String id,
    String name,
    org.omg.CORBA.TypeCode disc_type,
    org.omg.CORBA.UnionMember[] members)
    throws org.omg.CORBA.SystemException;
public org.omg.CORBA.TypeCode create_wstring_tc(int bound)
    throws org.omg.CORBA.SystemException;
```

### Description

Creates a new `TypeCode` for a specified IDL type. See the Interface Repository section of the CORBA specification.

Normally the `TypeCodes` describing IDL types are generated automatically in the Helper classes or are accessible from the Interface Repository.

In some situations, such as bridges between ORBs, `TypeCodes` need to be constructed outside of any Interface Repository. You can do this using the `create_<type>_tc()` methods on the ORB pseudo-object.

Refer to [“Class org.omg.CORBA.TypeCode” on page 38](#) for details of parameters.

For example:

For an array `TypeCode` the methods available are `length()`, which returns the size of this dimension of the array, and `content_type()` which returns the `TypeCode` for the contents of the array.

**Note:**

For a multi-dimensional array, this is also an array `TypeCode`.

The parameters for `create_array_tc()` correspond to the values that represent the length of the array and the `TypeCode` of the elements contained in the array. So to create a `TypeCode` for a 2-dimensional array of `longs` you can use the following code:

```
import IE.Iona.OrbixWeb._CORBA;
import org.omg.CORBA.TypeCode;
import org.omg.CORBA.ORB;
// create a TypeCode for an array defined as follows in IDL
// typedef long LongArray[4][5];

TypeCode tempTC=
    ORB.init().create_array_tc(5,_CORBA._tc_long);
TypeCode tc = ORB.init().create_array_tc(4, tempTC);
```

**Parameters**

<code>id</code>	The <code>RepositoryId</code> String for the object described in the CORBA specification. This is a string of the form “IDL:/<ident1>/<ident2>/.../<identn>:<version number>”. For example, for the grid interface the <code>RepositoryId</code> is “IDL:grid:1.0”. If the Helper classes are available, the <code>id()</code> method returns this string.
<code>name</code>	The name of the IDL type.

**Return Value**

<code>TypeCode</code>	The <code>TypeCode</code> generated.
-----------------------	--------------------------------------

**Notes**

CORBA-defined.

**See Also**

[“Class org.omg.CORBA.ORB” on page 19](#)  
[“Class IE.Iona.OrbixWeb.CORBA.ORB” on page 120](#)

**create\_any()**

**Synopsis**

```
public org.omg.CORBA.Any create_any()
    throws org.omg.CORBA.SystemException;
```

**Description**

Creates a new empty “Class `IE.Iona.OrbixWeb.CORBA.Any`”.

**Return Value**

<code>org.omg.CORBA.Any</code>	The new <code>Any</code> .
--------------------------------	----------------------------

**Notes**

CORBA-defined.

**See Also**

[“Class IE.Iona.OrbixWeb.CORBA.Any” on page 47”](#)

## create\_context\_list()

### Synopsis

```
public org.omg.CORBA.ContextList create_context_list()  
    throws org.omg.CORBA.SystemException;
```

### Description

When making an invocation using the DII on an IDL operation that has a Context specified, you must pass a ContextList parameter to the `_create_request()` method in ["Interface org.omg.CORBA.Object"](#) on page 18. This method allows you to create an empty ["Class org.omg.CORBA.ContextList"](#) on page 10.

### Return Value

contextList                      An empty ContextList object.

### Notes

CORBA-defined.

### See Also

["Class org.omg.CORBA.ORB"](#) on page 19  
["Class org.omg.CORBA.ContextList"](#) on page 10  
["Class IE.Iona.OrbixWeb.CORBA.ContextList"](#)  
["Class IE.Iona.OrbixWeb.CORBA.Request"](#)  
["Class org.omg.CORBA.Context"](#) on page 9  
`_create_request()` method in ["Interface org.omg.CORBA.Object"](#) on page 18

## create\_environment()

### Synopsis

```
public org.omg.CORBA.Environment create_environment()  
    throws org.omg.CORBA.SystemException;
```

### Description

The ["Class org.omg.CORBA.Environment"](#) object is used with the DII to allow exception information to be returned from an operation invocation. This methods creates a new empty Environment object that you can use as a parameter to the `_create_request()` method in ["Interface org.omg.CORBA.Object"](#) on page 18.

### Return Value

Environment                      A new empty Environment object.

### Notes

CORBA-defined.

### See Also

["Class org.omg.CORBA.ORB"](#) on page 19  
["Class org.omg.CORBA.ContextList"](#) on page 10  
["Class IE.Iona.OrbixWeb.CORBA.Request"](#) on page 167  
["Class IE.Iona.OrbixWeb.CORBA.Environment"](#) on page 94  
["Class org.omg.CORBA.Environment"](#) on page 14  
`_create_request()` method in ["Interface org.omg.CORBA.Object"](#) on page 18

## create\_exception\_list()

### Synopsis

```
public org.omg.CORBA.ExceptionList create_exception_list()  
    throws org.omg.CORBA.SystemException;
```

### Description

An ["Class org.omg.CORBA.ExceptionList"](#) is used by the DII to describe the exceptions that can be raised by IDL operations. This method creates an empty `ExceptionList` to be inserted into the Request.

### Return Value

`ExceptionList`            An empty `ExceptionList`.

### Notes

CORBA-defined.

### See Also

["Class org.omg.CORBA.ORB"](#) on page 19  
["Class org.omg.CORBA.ExceptionList"](#) on page 15  
["Class IE.Iona.OrbixWeb.CORBA.Request"](#) on page 167  
["Class IE.Iona.OrbixWeb.CORBA.Environment"](#) on page 94  
["Class org.omg.CORBA.Environment"](#) on page 14  
`_create_request()` method in ["Interface org.omg.CORBA.Object"](#)  
on page 18

## create\_list()

### Synopsis

```
public org.omg.CORBA.NVList create_list(int count)  
    throws org.omg.CORBA.SystemException;
```

### Description

Creates an empty ["Class org.omg.CORBA.NVList"](#) for use as a parameter type or for use as a way to describe the arguments to an operation invocation when using the `_create_request()` method in ["Interface org.omg.CORBA.Object"](#)

`add_item()` is the only way to add an item to an `NVList`.  
`get_length()` returns number of times `add_item()` was called. The `NVList` always starts with element number 0.

### Parameters

`count`                    The size of the `NVList` to create.

### Return Value

`NVList`                    The empty `NVList` returned by this method.

### Notes

CORBA-defined.

### See Also

["Class org.omg.CORBA.ORB"](#) on page 19  
["Class org.omg.CORBA.NVList"](#) on page 17  
["Class IE.Iona.OrbixWeb.CORBA.NVList"](#) on page 102  
["Class IE.Iona.OrbixWeb.CORBA.Request"](#) on page 167  
["Class IE.Iona.OrbixWeb.CORBA.NamedValue"](#) on page 98  
["Class org.omg.CORBA.NamedValue"](#) on page 16  
`_create_request()` method in ["Interface org.omg.CORBA.Object"](#)  
on page 18

## create\_named\_value()

### Synopsis

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

### Description

Creates an empty "Class [org.omg.CORBA.NamedValue](#)". You can insert this into a "Class [IE.Iona.OrbixWeb.CORBA.NVList](#)" when using the DII as one of the parameters to the operation invocation.

### Parameters

name	The name of the NamedValue.
value	An Any to be inserted into the value of the NamedValue.
flags	Indicates whether this is an IN, OUT or INOUT parameter.

### Return Value

NamedValue	The NamedValue constructed by this method call.
------------	-------------------------------------------------

### Notes

CORBA-defined.

### See Also

["Class org.omg.CORBA.ORB" on page 19](#)  
["Class org.omg.CORBA.NVList" on page 17](#)  
["Class IE.Iona.OrbixWeb.CORBA.NVList" on page 102](#)  
["Class IE.Iona.OrbixWeb.CORBA.Request" on page 167](#)  
["Class IE.Iona.OrbixWeb.CORBA.NamedValue" on page 98](#)  
["Class org.omg.CORBA.NamedValue" on page 16](#)  
[\\_create\\_request\(\) method in "Interface org.omg.CORBA.Object" on page 18](#)

## create\_operation\_list()

### Synopsis

```
public org.omg.CORBA.NVList  
    create_operation_list(org.omg.CORBA.OperationDef oper)  
    throws org.omg.CORBA.SystemException;
```

### Description

Create a new NVList for use with operation invocations when using the DII.

The returned NVList is the correct length, each of the NamedValues contained within has a valid name and the correct flags (for the parameter passing mode, either ARG\_IN, ARG\_OUT or ARG\_INOUT).

The value part of the NamedValue is empty—you must populate it with correct values. For a NamedValue with flag set to ARG\_IN or ARG\_INOUT the value must be initialized.

The IFR must be populated for this call to work.

For example, consider the following used to call the first operation on an Object. You can use the following code to set up the `NVList` for the parameters:

```
org.omg.CORBA.Object ref = // get the reference from somewhere :
                          // For example :
                          //     - from a file and then call
                          //     string_to_object(),
                          //     - from the naming service
                          //     - using the bind call
org.omg.CORBA.InterfaceDef iDefRef = null;
org.omg.CORBA.OperationDef[] oDef = null;
org.omg.CORBA.NVList nvl = null;
try {
    iDefRef = ref._get_interface();
    oDef =
        iDefRef.contents(
            org.omg.CORBA.DefinitionKind.dk_Operation,
            true);
    if( oDef != null){
        nvl = ORB.init().create_operation_list(oDef[0]);
    }
}
catch(org.omg.CORBA.SystemException e){
    // do something
}

// continue on with DII invocation
```

## Parameters

<code>oper</code>	A reference to an <code>org.omg.CORBA.OperationDef</code> object in the Interface Repository.
-------------------	-----------------------------------------------------------------------------------------------

## Return Value

<code>NVList</code>	<code>NVList</code> set-up for this operation invocation; the correct number of the <code>NamedValue</code> and the <code>NamedValues</code> have a valid name and the correct flag.
---------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## Notes

CORBA-defined.

## See Also

["Class org.omg.CORBA.ORB" on page 19](#)  
["Class org.omg.CORBA.NVList" on page 17](#)  
["Class IE.Iona.OrbixWeb.CORBA.NVList" on page 102](#)  
["Class IE.Iona.OrbixWeb.CORBA.Request" on page 167](#)  
["Class IE.Iona.OrbixWeb.CORBA.NamedValue" on page 98](#)  
[`\_create\_request\(\)` method in "Interface org.omg.CORBA.Object" on page 18](#)



## create\_output\_stream()

### Synopsis

```
public org.omg.CORBA.portable.OutputStream
    create_output_stream()
    throws org.omg.CORBA.SystemException;
```

### Description

The `Input/OutputStream` classes provide methods for the reading and writing of all of the mapped IDL types to and from streams. Their implementations are used inside the ORB to marshal parameters and to insert and extract complex data types into and from `AnyS` and `RequestS`.

The streaming APIs are found in the [“Class `org.omg.CORBA.portable.Streamable`”](#) package.

The ORB object is used as a factory to create an output stream. You can create an input stream from an output stream.

### Return Value

`OutputStream`                      A new `OutputStream` object.

### Notes

CORBA-defined.

### See Also

[“Class `org.omg.CORBA.portable.Streamable`”](#) on page 27  
[“Class `org.omg.CORBA.portable.InputStream`”](#) on page 25  
[“Class `org.omg.CORBA.portable.OutputStream`”](#) on page 26  
[“Class `IE.Iona.OrbixWeb.CORBA.Request`”](#) on page 167

## defaultTxTimeout()

### Synopsis

```
public int defaultTxTimeout(int val)
    throws org.omg.CORBA.SystemException;
```

### Description

Sets the default timeout (in milliseconds) for all `RequestS`, and returns the previous setting. The default is an infinite timeout, which has a value of zero.

This value is ignored when establishing the initial connection to an object. It only comes into affect when the connection has been established.

### Parameters

`val`                                  The new connection timeout value in milliseconds.

### Return Value

`int`                                      The previous `Tx` timeout value.

### Notes

Orbix-specific.

### See Also

`IT_INFINITE_TIMEOUT` in class [“Class `IE.Iona.OrbixWeb.\_CORBA`”](#) on page 239

## finalize()

### Synopsis

```
public void finalize()  
    throws org.omg.CORBA.SystemException;
```

### Description

You can use this method in a client to clean up the Connections Table. To ensure that all connections are correctly cleaned up, you must call this method before the client exits.

In a server this method is used to call the loaders for all the objects in a process.

If you want to ensure that your objects are correctly saved you *must* call this method before you exit your program. This is because Java does not have destructors and the `finalize()` method for an object is not guaranteed to be called. For each of your objects in the server the `save()` method in class `IE.Iona.OrbixWeb.Features.LoaderClass` is called with the reason set to 'processTermination' in class `IE.Iona.OrbixWeb._CORBA`.

This method has no effect for in-process servers.

The `finalize()` method is now deprecated. Calls to `finalize()` now call `shutdown()` internally.

### Notes

Orbix-specific.

### See Also

["Class IE.Iona.OrbixWeb.Features.LoaderClass" on page 222](#)  
["shutdown\(\)" on page 81](#)

## getConfigItem()

### Synopsis

```
public String getConfigItem (String name)  
    throws org.omg.CORBA.SystemException;
```

### Description

This method gets the value of the configuration property item specified by name.

For example, the following code:

```
String iiopString = ORB.init().getConfigItem (  
    "IT_BIND_USING_IIOP");  
boolean iiop = iiopString.equals("true");
```

sets the value of the `iiop` boolean flag to `true` if IIOP is the default protocol.

### Parameters

<code>name</code>	The name of the property to get the value of.
-------------------	-----------------------------------------------

### Return Value

<code>String</code>	A <code>String</code> representing the value of the specified property.
---------------------	-------------------------------------------------------------------------

### Notes

Orbix-specific.

### See Also

["setConfiguration\(\)" on page 153](#)

## getConfiguration()

### Synopsis

```
public Properties getConfiguration ()
    throws org.omg.CORBA.SystemException;
```

### Description

Gets the `java.util.Properties` object containing the configuration for this process. See the configuration chapter in the ***Orbix Administrator's Guide Java Edition*** for further information. This object then allows you to query process configuration.

```
java.util.Properties props = ORB.init().getConfiguration();

String iiopString = props.getProperty (
    "IT_BIND_USING_IIOP");
boolean iiop = iiopString.equals("true");
```

### Return Value

`Properties`                      The `Properties` object for this process.

### Notes

Orbix-specific

### See Also

["getConfigItem\(\)" on page 136](#)  
["setConfigItem\(\)" on page 152](#)

## getDaemonConnections()

### Synopsis

```
public DaemonMgr[] getDaemonConnections()
    throws org.omg.CORBA.SystemException;
```

### Description

This method returns an array of Orbix Java daemon client proxies. These daemon manager objects allow processes to query or set (like calling the `putit` and `catit` utilities) information in the Implementation Repository.

It also allows explicit management of Orbix Java daemon connections. For example, most systems have a physical limit on the number of connections you can have open at any given time. It may be necessary to close connections to the daemon, once connections to your servers have been established.

```
import IE.Iona.OrbixWeb.CORBA.DaemonMgr;
// call this method if you have run out of resources
// and want to free up some connections that are not being used
public static void freeResources(){
    DaemonMgr[] dList =
        _OrbixWeb.ORB(ORB.init()).getDaemonConnections();
    try {
        System.out.println( "Calling freeResources : len == " +
            dList.length);

        for(int i = 0;i < dList.length; i++){
            System.out.println(
                "Closing Connection to daemon : host " +
                dList[i]._host());
            _OrbixWeb.ORB(ORB.init()).closeConnection(dList[i]);
        }
    }
    catch(java.lang.NullPointerException e){
    }
}
```

## Return Value

DaemonMgr[]                      An array of Orbix daemon client proxy objects.

**Notes**                              Orbix-specific.

**See Also**                            ["closeConnection\(\)" on page 127](#)

## get\_default\_context()

### Synopsis

```
public org.omg.CORBA.Context get_default_context()  
    throws org.omg.CORBA.SystemException;
```

### Description

This operation returns a reference to the default process ["Class org.omg.CORBA.Context"](#) object. You can then use this for the Context parameter to an invocation.

## Return Value

context                              The default Context object for this process.

**Notes**                              CORBA-defined.

**See Also**                            ["Class org.omg.CORBA.ORB" on page 19](#)  
["Class org.omg.CORBA.Context" on page 9](#)  
["Class IE.Iona.OrbixWeb.CORBA.Context" on page 83](#)

## getHostPort()

### Synopsis

```
public int getHostPort(String hostname)  
    throws org.omg.CORBA.SystemException;
```

### Description

Get the TCP/IP port on which this process tries to contact the Orbix Java daemon process located at hostname. This method is only relevant when binding to an object using the proprietary Orbix protocol. This method is deprecated.

### Parameters

hostname                            The hostname for the daemon process.

## Return Value

int                                    The port number for the daemon process located at hostname.

**Notes**                              Orbix-specific.

**See Also**                            *Orbix Administrator's Guide Java Edition*

## get\_my\_principal()

### Synopsis

```
public org.omg.CORBA.Principal get_my_principal()  
    throws org.omg.CORBA.SystemException;
```

### Description

This method returns the "Class org.omg.CORBA.Principal" user name associated with this process. Orbix Java adds this `Principal` to outgoing requests by for identification purposes.

### Return Value

`Principal`                      The username for this process.

### Notes

Orbix-specific.

### See Also

["Class org.omg.CORBA.Principal" on page 28](#)  
["Class IE.Iona.OrbixWeb.CORBA.Principal" on page 164](#)  
["Class IE.Iona.OrbixWeb.CORBA.OrbCurrent" on page 159](#)

## getMyReqTransformer()

### Synopsis

```
public IT_reqTransformer getMyReqTransformer()  
    throws org.omg.CORBA.SystemException;
```

### Description

The method returns the "Class IE.Iona.OrbixWeb.Features.IT\_reqTransformer" object associated with this process. This `IT_reqTransformer` is applied to all invocations leaving this process.

The `IT_reqTransformer` is a class that receives all invocations just before they leave or immediately after they arrive at an Orbix Java process. This class allows you to modify this incoming or outgoing data.

### Return Value

`IT_reqTransformer`              The `IT_reqTransformer` object associated with the current process.

### Notes

Orbix-specific.

### See Also

["Class IE.Iona.OrbixWeb.Features.IT\\_reqTransformer" on page 219](#)  
["setReqTransformer\(\)" on page 155](#)

## get\_next\_response()

### Synopsis

```
public org.omg.CORBA.Request get_next_response()  
    throws org.omg.CORBA.SystemException;
```

### Description

You can call this method to determine the outcome of the individual requests specified in a `send_multiple_requests_oneway()` call. This method blocks until a response is available.

The order in which the replies are returned is not necessarily the same as the order in which they were sent. In a multi-threaded or multi-process environment some requests may be processed faster than others.

### Return Value

Request                      The next available response.

### Notes

CORBA-defined.

### See Also

["Class org.omg.CORBA.ORB" on page 19](#)

## get\_principal()

### Synopsis

```
public org.omg.CORBA.Principal get_principal()  
    throws org.omg.CORBA.SystemException;
```

### Description

In a client process this method returns the class `org.omg.CORBA.Principal` username associated with this process. However, in a server process it returns the principal of the client whose `Request` is currently being processed if `IT_MULTI_THREADED_SERVER` configuration parameter is `true`.

### Return Value

Principal                      In a client this is the user name of this process. In a server this returns the user name of the client.

### Notes

CORBA-defined.

### See Also

["Class org.omg.CORBA.ORB" on page 19](#)  
["Class org.omg.CORBA.Principal" on page 28](#)  
["Class IE.Iona.OrbixWeb.CORBA.OrbCurrent" on page 159](#)

## getTransformer()

### Synopsis

```
public static IT_reqTransformer getTransformer(String server,  
                                              String host)  
    throws org.omg.CORBA.SystemException;
```

### Description

You can call this method to determine the `IT_reqTransformer` for a specified server and (optionally) a host.

If a process level transformer has been set using the `setMyReqTransformer()` method, this `IT_reqTransformer` object is returned immediately.

Otherwise, if an `IT_reqTransformer` is set for a particular server and host (using the `setReqTransformer()` method), this `IT_reqTransformer` object is returned.

Finally, if a transformer was set for the server only, this `IT_reqTransformer` object is returned.

#### Return Value

`IT_reqTransformer`      The requested transformer object for the specified server and (optionally) host.

#### Notes

Orbix-specific.

#### See Also

["Class IE.Iona.OrbixWeb.Features.IT\\_reqTransformer"](#)

## get\_principal\_string()

#### Synopsis

```
public String get_principal_string()  
    throws org.omg.CORBA.SystemException;
```

#### Description

In a client, this method returns the user name (`Principal`) associated with this process in string form. This `Principal` is added to outgoing requests by Orbix Java for identification processes.

In a server, this method returns the user name of the client whose invocation is currently being processed.

#### Return Value

`String`      In a client, the user name associated with this process.  
In a server, the user name of the client whose invocation is currently being processed if `IT_MULTI_THREADED_SERVER` is `true` (otherwise, not thread safe).

#### Notes

Orbix-specific.

#### See Also

["Class org.omg.CORBA.ORB" on page 19](#)

["Class IE.Iona.OrbixWeb.CORBA.OrbCurrent" on page 159](#)

## hasValidOpenChannel()

#### Synopsis

```
public boolean hasValidOpenChannel(ObjectRef oref)  
    throws org.omg.CORBA.SystemException;  
public boolean hasValidOpenChannel(org.omg.CORBA.Object oref)  
    throws org.omg.CORBA.SystemException;
```

#### Description

This method returns `true` if there is a valid open connection between this process and the process in which the object `Oref` resides.

#### Parameters

`oref`      The object whose connection is checked.

#### Return Value

`boolean`      This is `true` if there is a valid, open connection between this process and the process in which `oref` resides.

#### Notes

Orbix-specific.

## init()

### Synopsis

```
static public org.omg.CORBA.ORB init ()
    throws org.omg.CORBA.SystemException,
           org.omg.CORBA.INITIALIZE;

static public org.omg.CORBA.ORB init (String[] args,
                                       java.util.Properties props)
    throws org.omg.CORBA.SystemException,
           org.omg.CORBA.INITIALIZE;

static public org.omg.CORBA.ORB init(java.util.Properties props)
    throws org.omg.CORBA.SystemException,
           org.omg.CORBA.INITIALIZE;
static public org.omg.CORBA.ORB init ( java.applet.Applet app,
                                       java.util.Properties props)
    throws org.omg.CORBA.SystemException,
           org.omg.CORBA.INITIALIZE;
```

### Description

The `init()` method configures the ORB object and returns a reference to the current ORB object.

Order of configurations (in order of increasing priority):

- Defaults
- External configuration file
- Properties
- Applet tags
- Command-line parameters

The `init (String[] args, java.util.Properties props)` method configures using command-line parameters.

The call to initialize Orbix Java from an application's `main()` method is shown in the following code sample. This code also illustrates how an application that wishes to use other command-line parameters can ignore the ORB parameters, because the Orbix Java parameters all start with the string `"-OrbixWeb."`.

```
// Initialize the ORB.
try {
    IE.Iona.OrbixWeb.CORBA.ORB.init (args, null);
}
catch (Exception ex) {
    System.err.println ("argument error: "+ex);
    System.exit(1);
}

// Read in the command-line parameters, and ignore any of
// the OrbixWeb parameters.

for (int i = 0; i < args.length; i++) {
    String ignore = "-OrbixWeb.";
    if (args[i].length() < ignore.length() ||

        !(args[i].substring(0,ignore.length()).equalsIgnoreCase
            (ignore))
```



```

    {
        // this is a non-OrbixWeb command-line parameter,
        // take appropriate action.
    }
}
// Your application initialization code can continue below...

```

This mechanism allows Orbix Java to search for configuration parameters both in the application command-line arguments, and in the system properties.

The `init(java.util.Properties props)` method configures Orbix Java using system properties.

If you use either of these mechanisms, the added benefit is that you can also use the Java system properties for parameters. However, there is no standard way you can set Java system properties. The JDK, for example, uses a file containing a list of the property names and values, and most browsers do not allow any properties to be set. The most useful way to use this functionality is to pass parameters using the JDK Java interpreter's `-D` command-line argument. This supplements the command-line argument support already shown.

The `init (java.applet.Applet app, java.util.Properties props)` method configures Orbix Java using applet tags.

The call to initialize Orbix Java from inside an applet's `init()` method is as follows:

```

public void init () {
    // Initialize the ORB.
    try {
        IE.Iona.OrbixWeb.CORBA.ORB.init (this, null);
    }
    catch (Exception ex) {
        System.err.println ("bad applet tag: "+ex);
    }
    // Your applet initialization code can continue below...
}

```

The initialization method is passed `this`; the applet object itself.

This mechanism allows Orbix Java to search for configuration parameters both in the applet tags, and in the system properties.

### Singleton ORB

An `ORB.init()` call with no parameters returns an instance of `IE.Iona.OrbixWeb.CORBA.singletonORB`. There is only one instance of the singleton ORB in a virtual machine. The singleton ORB restricted functionality is mainly for applet security reasons. You can call the following operations on the singleton ORB:

```

create_list()
create_named_value()
create_exception_list()
create_context_list()
get_default_context()
create_environment()
create_xxx_tc()
(where xxx is a defined Typecode type)
get_primitive_tc()
create_any()
create_output_stream()

```

An attempt to call any other ORB operations on the singleton ORB results in a system exception.

### Fully Functional ORB

Any of the forms of `ORB.init()` with parameters returns a new, fully-functional ORB. In earlier versions of Orbix Java, each call in the same VM returns the same ORB (`_CORBA.Orbix`). In this version, each call returns a different new ORB. This adds considerable flexibility to some applications, because each new ORB is completely independent from any other. For example, ORB configuration, connections, listener ports and server object tables are all per-ORB. Multiple ORBs also facilitate applet separation.

#### Notes

CORBA-defined.

#### See Also

[set\\_parameters\(\)](#) in ["Class org.omg.CORBA.ORB" on page 19](#)  
["getConfigItem\(\)" on page 136](#)  
["setConfigItem\(\)" on page 152](#)  
["setConfiguration\(\)" on page 153](#)  
["getConfigItem\(\)" on page 136](#)

## isBaseInterfaceOf()

### Synopsis

```
public boolean isBaseInterfaceOf(String derivedStubClass,  
                                String maybeABase)  
    throws org.omg.CORBA.SystemException;  
public boolean isBaseInterfaceOf(org.omg.CORBA.Object obj,  
                                String maybeABase)
```

### Description

This method determines whether the interface of the stub class `derivedStubClass` or the object reference `obj` is derived from the interface `maybeABase`.

### Parameters

<code>derivedStubClass</code>	Name of the class to check.
<code>obj</code>	An object reference to the class to check.
<code>maybeABase</code>	Name of the interface to check.

### Return Value

<code>boolean</code>	Return true if <code>derivedStubClass</code> or <code>obj</code> is derived from <code>maybeABase</code> .
----------------------	------------------------------------------------------------------------------------------------------------

#### Notes

Orbix-specific.

#### See Also

["Class org.omg.CORBA.ORB" on page 19](#)

## list\_initial\_services()

### Synopsis

```
public String[] list_initial_services()  
    throws org.omg.CORBA.SystemException;
```

### Description

This method returns a list of the available Services. At present only three are available by default:

- The Naming Service (NS).
- The Interface Repository (IFR).
- The Trader Service.

You can add other services using the `IT_INITIAL_REFERENCES` configuration parameter. Refer to the *Orbix Administrator's Guide Java Edition* for details.

## Return Value

`String[]` A list of the currently available services

## Notes

CORBA-defined.

## See Also

`resolve_initial_references` in ["Class org.omg.CORBA.ORB" on page 19](#)

## locator()

### Synopsis

```
public final void locator(locatorClass l);
```

### Description

This method enables you to create a user-defined locator. If you do not create a user-defined locator, the Orbix Java default locator mechanism is used when required. If you wish to override the default locator with a new class inheriting from `CORBA.locatorClass`, you should assign an object of your new class to the `locator()` method.

This new locator class is created on a per-ORB basis, enabling support for multiple ORBs.

## Notes

Orbix-specific.

## See Also

["Class IE.Iona.OrbixWeb.Features.locatorClass" on page 227](#)

## makeIOR()

### Synopsis

```
public String makeIOR(String ip_addr,
                      int port,
                      String orbixHost,
                      String server,
                      String marker,
                      String typeID)
    throws org.omg.CORBA.SystemException;
public String makeIOR(String ip_addr,
                      int port,
                      byte[] objKey,
                      String typeID)
    throws org.omg.CORBA.SystemException;
```

### Description

You can use this method to create a reference to a remote object.

The `ip_addr`, `port` and `typeID` are the respective standard elements of the IOR (Interoperable Object Reference, for use with IIOP). `server`, `marker` and `orbixHost` are fields of the Orbix object reference from which the IOR object key is created. If `'ip_addr'` is null, `'orbixhost'` is also used as the host field of the IOR.

The `objKey` is the opaque object key part of the IOR as required by the CORBA specification. This can be a non-Orbix object Key. The Orbix object key is made up of the `orbixHost`, `server` and `marker`.

### Parameters

<code>ip_addr</code>	IP address or hostname of the IOR
<code>port</code>	TCP/IP port number of the IOR

orbixHost	Orbix object reference hostname or IP address.
server	The name of the target object's server as registered in the Implementation Repository and also as specified to <code>CORBA.BOA.impl_is_ready()</code> , <code>CORBA.BOA.object_is_ready()</code> or set by <code>setServerName()</code> .
marker	The object marker name. This can be chosen by the application, or can be a string of digits chosen by Orbix Java. If <code>null</code> , it matches any marker.
typeID	Orbix object reference object <code>typeID</code> , described in the CORBA Specification section 7.6, as a string of the form "IDL:/<ident1>/<ident2>/.../<identn>:<version number>". For example, for the grid interface the <code>RepositoryId</code> is "IDL:grid:1.0". If the Helper classes are available, the <code>id()</code> method return this string.
objKey	Opaque object key part of the IOR.

#### Return Value

String	The IOR constructed from the method parameters.
--------	-------------------------------------------------

#### Notes

Orbix-specific.

#### See Also

`object_to_string()` in ["Class org.omg.CORBA.ORB" on page 19](#)  
["object\\_to\\_string\(\)" on page 148](#)

## maxConnectRetries()

#### Synopsis

```
public int maxConnectRetries(int retries)
    throws org.omg.CORBA.SystemException;
```

#### Description

This method sets the maximum connection retries and returns the old value. The default value is 5.

If an operation call cannot be made on the first attempt because the transport (for example, TCP/IP) connection cannot be established, Orbix Java retries the attempt every two seconds until either the call can be made or until there are too many retries.

#### Note:

The value set by `maxConnectRetries()` is ignored when connecting to the Orbix Java daemon. It is only used when connecting to servers.

#### Return Value

int	The previous maximum connection retries value.
-----	------------------------------------------------

#### Notes

Orbix-specific.

#### See Also

***Orbix Administrator's Guide Java Edition***

## myHost()

### Synopsis

```
public java.lang.String myHost()  
    throws org.omg.CORBA.SystemException;
```

### Description

If the variable `IT_IORS_USE_DNS` is set to `false`, returns the local host's IP address.

If the variable `IT_IORS_USE_DNS` is set to `true`, returns the local host's name.

See the *Orbix Administrator's Guide Java Edition* for more details on `IT_IORS_USE_DNS`.

### Notes

Orbix-specific.

## \_nil()

### Synopsis

```
public static IE.Iona.OrbixWeb.CORBA.ORB _nil();
```

### Description

Returns a nil ORB object.

Defined to always be a Java `null` in the IDL-to-Java mapping.

### Notes

Orbix-specific.

## noReconnectOnFailure()

### Synopsis

```
public boolean noReconnectOnFailure(boolean b)  
    throws org.omg.CORBA.SystemException;
```

### Description

When an Orbix Java client first contacts a server, a single communications channel is established between the client-server pair. This connection is used for all subsequent communications between the client and the server (presuming that bidirectional IIOP or the Orbix protocol is used). If you are using pure IIOP and you have callback object in your client, a separate connection is created from the server to the client. The connection is closed only when the client or the server exits, or the client calls `ORB.closeConnection()`.

If a connection from the client to the server is closed, the next invocation by the client (presuming the client is still up) causes the Orbix Java runtime to transparently try to automatically re-establish the connection. This may involve relaunching the server.

You can change this default behavior by passing `true` to the function `ORB.noReconnectOnFailure()`. Then, all client attempts to contact a server subsequent to closure of the communications channel raise an `org.omg.CORBA.COMM_FAILURE` system exception. This behavior is not guaranteed for `oneway` calls.

### Parameters

- `b`      True means always return `org.omg.CORBA.COMM_FAILURE` exceptions for invocations after a connection has been closed.
- False means always try to re-establish the connection.

### Return Value

Returns the previously-set value.

### Notes

Orbix-specific.

## object\_to\_string()

### Synopsis

```
public java.lang.String object_to_string  
    (org.omg.CORBA.Object obj);
```

### Description

Converts an object reference to a string. If `obj` is an Interoperable Object Reference (IOR), the return value is a stringified IOR. If `obj` is an Orbix Java object reference, the resulting `String` conforms to the Orbix communications protocol object reference format.

You do not need to know the structure of an object reference.

### Parameters

`obj` The object whose string representation you wish to obtain.

### Notes

CORBA-defined.

### See Also

[object\\_to\\_string\(\)](#) in ["Class org.omg.CORBA.ORB" on page 19](#)  
["string\\_to\\_object\(\)" on page 157](#)

## pingDuringBind()

### Synopsis

```
public boolean pingDuringBind(boolean pingOn)  
    throws org.omg.CORBA.SystemException;
```

### Description

By default, the `bind()` method (defined in the Helper class for an interface) raises an exception if the object on which the `bind()` is attempted is unknown to Orbix Java. Doing so requires Orbix Java to ping the desired object (the `ping` operation is defined by Orbix Java and it has no effect on the target object). The pinging causes the target server process to be activated if necessary, and confirms that this server recognizes the target object.

You may wish to improve efficiency by disabling pinging, which reduces the overall number of remote invocations. You can disable pinging by using `ORB.pingDuringBind()` and passing `false` to the parameter `pingOn`.

If `pingDuringBind(false)` is called:

- A `bind()` to an unavailable object does not immediately raise an exception. Subsequent requests using the object reference returned from `bind()` fail by raising the system exception `org.omg.CORBA.INV_OBJREF`.
- If a hostname is specified to `bind()`, the `bind()` does not itself make any remote calls; it simply sets up a proxy with the required fields.

### Parameters

`b` Returns the previous setting. The default is `true`.

### Notes

Orbix-specific.

### See Also

["IT\\_PING\(\)" on page 116](#)

## poll\_next\_response()

### Synopsis

```
public boolean poll_next_response()  
    throws org.omg.CORBA.SystemException;
```

### Description

Determines whether or not a response is in the response queue. The method returns immediately. It does not affect the response queue.

Normally used after invoking

`ORB.send_multiple_requests_deferred()`. If a response has arrived you can get it by calling `ORB.get_next_response()`.

You can also use `poll_next_response()` when using `Request.send_deferred()`. However, you still have to use `Request.get_response()` to determine whether a response has arrived for a particular request.

If you wish to find out whether a response has been received for a particular request, call `org.omg.CORBA.Request.get_response()` on that `Request` object. Unlike `poll_next_response()`, `Request.get_response()` takes the response off the response queue and unmarshals it.

### Return Value

<code>true</code>	A response has been received.
<code>false</code>	No response has been received.

### Notes

CORBA-defined.

### See Also

["get\\_next\\_response\(\)" on page 140](#)  
["send\\_multiple\\_requests\\_deferred\(\)" on page 151](#)  
["get\\_response\(\)" on page 176](#)  
["send\\_deferred\(\)" on page 178](#)

## reSizeConnectionTable()

### Synopsis

```
public void reSizeConnectionTable(int size)  
    throws org.omg.CORBA.SystemException;
```

### Description

Sets the maximum size for the hash table of connection objects to `size`. There is one connection object for every connection between the client and a server process.

If the number of currently-open connections is greater than the maximum size, Orbix Java automatically shuts down the oldest connections until the number of open connections is less than 80% of `size`.

The default `size` is 100 and is set by `IT_CONNECT_TABLE_SIZE_DEFAULT`.

### Parameters

<code>size</code>	The maximum capacity to which you want to resize the connection table.
-------------------	------------------------------------------------------------------------

### Notes

Orbix-specific.

## registerIOCallback()

### Synopsis

```
public void registerIOCallback
    (IE.Iona.OrbixWeb.CORBA.Features.ioCallback cb)
    throws org.omg.CORBA.SystemException;
```

### Description

Registers an object that implements "Interface IE.Iona.OrbixWeb.Features.ioCallback". This object is informed of connection establishment and connection termination events.

A connection is opened when a client first communicates with the server; and it is closed when the client terminates or the communication's level reports a break in service between the server and client.

A client or server application may register a callback object which is informed when either of the two events occur. This callback object must implement the Interface IE.Iona.OrbixWeb.Features.ioCallback.

This interface contains the following operations:

```
public void OpenCallBack
    (String serverName, Object conn)
public void CloseCallBack
    (String serverName, Object conn)
```

Only one ioCallback object can be registered at a time. If you register a second ioCallback object, it replaces the previous ioCallback object.

To unregister the iocallback use `unregisterIOCallback()`.

### Parameters

cb      The ioCallback object to be registered. It is informed of any connections being established or broken.

### Notes

Orbix-specific.

### See Also

["unregisterIOCallback\(\)" on page 158](#)  
["Interface IE.Iona.OrbixWeb.Features.ioCallback" on page 217](#)

## resolve\_initial\_references()

### Synopsis

```
public org.omg.CORBA.Object
    resolve_initial_references(java.lang.String serviceName)
    throws org.omg.CORBA.SystemException;
```

### Description

Returns an object reference through which a service (for example, Interface Repository or a CORBA service such as the Naming Service) can be used.

The list of available services is found calling `list_initial_services()`.

The default list of available services is:

- The Naming Service (NS): "NameService"
- The Interface Repository (IFR): "InterfaceRepository"
- The Trader Service: "TradingService"

You can also add other services using the `IT_INITIAL_REFERENCES` configuration parameter.



This is the CORBA-defined way for obtaining your first object reference. You can also use the Orbix Java value-added method `bind()`.

#### Parameters

`serviceName` The name of the desired service. You can obtain a list of services supported by Orbix Java using `ORB.list_initial_services()`.

#### Return Value

Returns an object reference for the desired service. The object reference returned must be narrowed to the correct object type. For example, the object reference returned from resolving the name 'NameService' must be narrowed to the IDL type `NamingContext`.

#### Notes

CORBA-defined.

#### See Also

["list\\_initial\\_services\(\)" on page 144](#)

## send\_multiple\_requests\_deferred()

#### Synopsis

```
public void send_multiple_requests_deferred
    (org.omg.CORBA.Request[] req)
    throws org.omg.CORBA.SystemException;
```

#### Description

Sends a number of requests in parallel. It does not wait for the requests to finish before returning to the caller.

The caller can use `ORB.get_next_response()`, `ORB.poll_next_response()` and `CORBA.Request.get_response()` to determine the outcome of the requests.

#### Parameters

`req` A sequence of `Request` objects.

#### Notes

CORBA-defined.

#### See Also

["get\\_next\\_response\(\)" on page 140](#)

["poll\\_next\\_response\(\)" on page 149](#)

["send\\_multiple\\_requests\\_oneway\(\)" on page 152](#)

["get\\_response\(\)" on page 176](#)

["send\\_deferred\(\)" on page 178](#)

## send\_multiple\_requests\_oneway()

### Synopsis

```
public void send_multiple_requests_oneway
    (org.omg.CORBA.Request[] req)
    throws org.omg.CORBA.SystemException;
```

### Description

Sends a number of requests in parallel. It does not wait for the requests to finish before returning to the caller.

There is no response to the requests even if the operations being invoked are defined in IDL as two-way operations.

### Parameters

`req` A sequence of request objects. The operations in this sequence do not have to be IDL *oneway* operations. The caller does not expect a response, nor does it expect `out` or `inout` parameters to be updated.

### Notes

CORBA-defined.

### See Also

["send\\_multiple\\_requests\\_deferred\(\)" on page 151](#)  
["send\\_oneway\(\)" on page 179](#)

## setConfigItem()

### Synopsis

```
public void setConfigItem(java.lang.String name,
    java.lang.String value)
    throws org.omg.CORBA.SystemException;
```

### Description

Sets the configuration parameter `name` to `value`.

Orbix Java applications read in their configuration settings from the Orbix Java configuration files, as specified in `iona.cfg`. Sometimes, however, you may want to change configuration settings dynamically at runtime. Orbix Java provides this method to allow you to do this.

If you want to change multiple settings at the same time you can use `ORB.setConfiguration()`.

See the ***Orbix Administrator's Guide Java Edition*** for a list of the available configuration settings.

### Notes

Orbix-specific.

### See Also

["getConfigItem\(\)" on page 136](#)  
["setConfiguration\(\)" on page 153](#)

## setConfiguration()

### Synopsis

```
public void setConfiguration  
    (java.util.Properties configuration)  
    throws org.omg.CORBA.SystemException;
```

### Description

Sets the variables passed in by `configuration`. Any variables not listed in `configuration` remain at their previous (usually the default) value.

Orbix Java applications read in their configuration settings from the Orbix Java configuration files, as specified in `iona.cfg`. Sometimes, however, you may want to change configuration settings dynamically at runtime. Orbix Java provides this method to allow you to do this.

If you want to change individual settings you can use `ORB.setConfigItem()`.

See the *Orbix Administrator's Guide Java Edition* for a list of configuration settings.

### Notes

Orbix-specific.

### See Also

["setConfigItem\(\)" on page 152](#)  
["getConfigItem\(\)" on page 136](#)

## setDiagnostics()

### Synopsis

```
public int setDiagnostics(int level)  
    throws org.omg.CORBA.SystemException;
```

### Description

Controls the level of diagnostic messages output by Orbix Java. Returns the previous setting. Orbix Java provides diagnostics for various components, each associated with a particular `level`.

### Parameters

The `level` parameter must be in the range of 0-255.

level	Diagnostics Component
0	No diagnostics
1	LO
2	HI
4	ORB
8	BOA
16	PROXY
32	REQUEST
64	CONNECTION
128	DETAILED

To obtain diagnostics output from particular components, add the values associated with the components together. The values `LO` and `HI` correspond to the diagnostic levels 1 and 2 from earlier versions of Orbix Java, and are provided for backwards compatibility.

The `DETAILED` level is of special significance, as this controls the amount of diagnostics produced by the components. Setting this means that all diagnostics from the selected components are output.

For example, obtaining detailed diagnostics associated with the BOA and Requests. You can do this by adding  $8 + 32 + 128 = 168$ , and passing this total to `setDiagnostics()`.

You can obtain full diagnostic output by setting the value to 255 (the result of adding all levels together). This produces very comprehensive output including full buffer dumps of messages.

**Return Value** Returns the previous setting.

**Notes** Orbix-specific.

## setHostPort()

**Synopsis**

```
public void setHostPort(java.lang.String hostname,
                        int port)
    throws org.omg.CORBA.SystemException;
```

**Description** Sets the daemon port for a given host, and allows the default port selection to be over-ridden dynamically. This only has effect when binding to an object using the Orbix protocol.

**Note:** This method is deprecated.

**Notes** Orbix-specific.

**See Also** ["getHostPort\(\)" on page 138](#)

## setMyReqTransformer()

**Synopsis**

```
public IE.Iona.OrbixWeb.CORBA.IT_reqTransformer
    setMyReqTransformer
    (IE.Iona.OrbixWeb.CORBA.IT_reqTransformer new_transformer)
    throws org.omg.CORBA.SystemException;
```

**Description** Sets a global request transformer object. It overrides any other transformers that may have been set. Request transformers are an Orbix-specific mechanism for encrypting and decrypting requests.

To set a transformer for a particular server on a particular host you can use `ORB.setReqTransformer()`.

For full details on how to use request transformers see the ***Orbix Programmer's Guide Java Edition***.

**Return Value** Returns the previous transformer.

**Notes** Orbix-specific.

**See Also** ["Class IE.Iona.OrbixWeb.Features.IT\\_reqTransformer" on page 219](#)  
["getMyReqTransformer\(\)" on page 139](#)  
["setMyReqTransformer\(\)" on page 154](#)  
["setReqTransformer\(\)" on page 155](#)

## set\_parameters()

### Synopsis

```
public void set_parameters(java.lang.String[] args,
                           java.util.Properties props)
    throws org.omg.CORBA.SystemException;
public void set_parametersS(java.applet.Applet app,
                           java.util.Properties props)
    throws org.omg.CORBA.SystemException;
```

### Description

Sets the configuration parameters for the current ORB object and can only be called after `ORB.init()` is called at least once.

This method has the same behavior as the `ORB.init()` methods with the same parameters. However, unlike `ORB.init()`, `set_parameters()` does not initialize the ORB object nor does it return an ORB object.

See `init()` for more details.

### Notes

Orbix-specific.

### See Also

["init\(\)" on page 142](#)

## set\_principal()

### Synopsis

```
public void set_principal(java.lang.String new_user)
    throws org.omg.CORBA.SystemException;
public void set_principal(org.omg.CORBA.Principal new_user)
    throws org.omg.CORBA.SystemException;
```

### Description

This method sets the `Principal` for the client.

This is a CORBA concept which identifies a client.

By default it is the client's user name.

### Notes

CORBA-defined.

### See Also

["Class org.omg.CORBA.Principal" on page 28](#)

[get\\_principal\\_string\(\) in](#)

["Interface IE.Iona.OrbixWeb.CORBA.BO" on page 59](#)

["get\\_principal\(\)" on page 140](#)

["Class IE.Iona.OrbixWeb.CORBA.Principal" on page 164](#)

## setReqTransformer()

### Synopsis

```
public void setReqTransformer
    IE.Iona.OrbixWeb.CORBA.IT_reqTransformer new_transformer,
    java.lang.String serverName,
    java.lang.String host)
    throws org.omg.CORBA.SystemException;
```

### Description

Registers a request transformer object for a particular server name and host.

All communication between the client that invokes `setReqTransformer()` and the server on the particular host pass through the registered request transformer.

If `host` is `null`, communications between the client and all servers with the name `serverName`, regardless of host, are subject to the request transformer.

If `serverName` is `null`, the method has no effect.

Request transformers are an Orbix-specific mechanism for encrypting and decrypting requests.

To set a global transformer for all communications, you can use `ORB.setMyReqTransformer()`. This overrides any transformers that you may have set using `setReqTransformer()`.

For full details on how to use request transformers refer to the ***Orbix Programmer's Reference Java Edition***.

## Parameters

<code>new_transformer</code>	The transformer object to be registered for the given <code>host</code> and <code>serverName</code> .
<code>serverName</code>	The server with all communication encrypted and decrypted using <code>new_transformer</code> .
<code>host</code>	The <code>host</code> on which the server is registered. If this is null, communication with all servers registered as <code>serverName</code> is decrypted and encrypted by <code>new_transformer</code> .

## Notes

Orbix-specific.

## See Also

["Class IE.Iona.OrbixWeb.Features.IT\\_reqTransformer" on page 219](#)  
["getMyReqTransformer\(\)" on page 139](#)  
["setMyReqTransformer\(\)" on page 154](#)  
["setReqTransformer\(\)" on page 155](#)

## string\_to\_object()

### Synopsis

```
public org.omg.CORBA.Object string_to_object (java.lang.String s)
    throws org.omg.CORBA.SystemException;
```

### Description

Converts the stringified object reference *s* to an object reference. No network communication occurs and so you do not know whether or not the target object exists until you invoke upon the object. When the object is invoked upon for the first time, Orbix Java attempts to establish a connection to the target server.

### Parameters

*s*                    The object reference in string form to be converted into an object reference.

### Return Value

Returns an object reference. This is a proxy object if the stringified object references refers to a remote object.

### Notes

CORBA-defined.

### See Also

["object\\_to\\_string\(\)" on page 148](#)  
[\\_object\\_to\\_string\(\) in](#)  
["Interface IE.Iona.OrbixWeb.CORBA.ObjectRef" on page 109](#)

## string\_to\_object()

### Synopsis

```
public org.omg.CORBA.Object string_to_object (
    java.lang.String host,
    java.lang.String IR_host,
    java.lang.String ServerName,
    java.lang.String marker,
    java.lang.String IR_server,
    java.lang.String interfaceName)
    throws org.omg.CORBA.SystemException;
```

### Description

Creates an object reference from the parameter strings. The parameter strings are the Orbix-specific way of specifying an object.

### Parameters

*host*                    The host name of the target object.  
*IR\_host*                The name of a host running an Interface Repository that stores the target object's IDL definition.  
*serverName*            The name of the target object's server.  
*marker*                The object's marker name.  
*IR\_server*             The string "IR" or "IFR".  
*interfaceName*        The target object's interface.

### Return Value

Returns an (Orbix communications protocol format) object reference constructed from the parameters passed to the method.

### Notes

Orbix-specific.

**See Also** ["object\\_to\\_string\(\)" on page 148](#)  
[\\_object\\_to\\_string\(\) in](#)  
["Interface IE.Iona.OrbixWeb.CORBA.ObjectRef" on page 109](#)

## toString()

**Synopsis** `public java.lang.String toString();`

**Description** You can use this method to test if the `_CORBA.Orbix` object is an ORB or BOA object.

**Return Value** Returns the string `"IE.Iona.OrbixWeb.CORBA.ORB"`.

**Notes** Orbix-specific.

**See Also** ["object\\_to\\_string\(\)" on page 148](#)  
[\\_object\\_to\\_string\(\) in](#)  
["Interface IE.Iona.OrbixWeb.CORBA.ObjectRef" on page 109](#)

## unregisterIOCallback()

**Synopsis** `public void unregisterIOCallback()  
throws org.omg.CORBA.SystemException;`

**Description** Unregisters the `ioCallback` object, if there is one. Has no effect otherwise.

For more information on `ioCallbacks` SEE "Interface `IE.Iona.OrbixWeb.Features.ioCallback`".

**Notes** Orbix-specific.

**See Also** ["Interface IE.Iona.OrbixWeb.Features.ioCallback" on page 217](#)



# Class `IE.Iona.OrbixWeb.CORBA.OrbCurrent`

## Synopsis

The `OrbCurrent` object allows a thread processing a `Request` to get context information about the invocation (such as the name of the person making the invocation and the object being invoked). So, for example, in the processing of a invocation you can do the following:

### Using the Current Object

Use of the Orbix Java `Current` object is not restricted to in-process servers but is used internally by the Activator and may have particular value for in-process servers.

If the operations of the `Current` object, or equivalent operations, are used in out-of-process servers, you should set the Orbix Java configurable property `IT_MULTI_THREADED_SERVER` to `true` before calling `ORB.init()` to ensure thread-safe results. You can call operations for the `Current` object within the server to get information about the current server or server operation.

As noted above, it should normally be called from the operation dispatching thread. It can be called in server-side filters and loaders as well as in the operation itself. A reference to the object is returned by `_OrbixWeb.Current()`, and the following operations are relevant:

- `public static org.omg.CORBA.Principal get_principal():` returns the principal of the client that invoked the current request.  
Equivalent to the ORB operation of the same name.
- `public static String get_principal_string():` as `get_principal()`, but in string form.
- `public static org.omg.CORBA.ServerRequest get_request():` get the current DSI request.
- `public static org.omg.CORBA.Object get_object():` return the target object of the current request.
- `public static int get_protocol():` return the protocol in which the current request was transmitted; for example, `IIOP = _CORBA.IT_INTEROPERABLE_OR_KIND`.
- `public static java.lang.Object get_socket():` returns an Orbix Java `SocketConnection` class that has access to the connection the current request arrived on.
- `public static String get_server():` returns the server name (equivalent to `_CORBA.Orbix.myServer()`).

## Synopsis

The `OrbCurrent` object allows a thread processing a `Request` to get context information about the invocation (such as the name of the person making the invocation and the object being invoked upon). So, for example, in the processing of a invocation you can do the following:

```
// get the OrbCurrent Object
IE.Iona.OrbixWeb.CORBA.OrbCurrent curr = _OrbixWeb.Current();

// show both ways of getting the Principal
//
// 1) get the Principal Object
org.omg.CORBA.Principal p = curr.get_principal();
String name = new String (p.name());
System.out.println("Principal = " + name );

// 2) get the principal string from the CurrentObject
System.out.println("Principal = " +
    curr.get_principal_string());

// 3) get the current ServerRequest object
org.omg.CORBA.ServerRequest req =
    curr.get_request();
String op_name = req.op_name();
System.out.println(
    "target method (from request)    = " + op_name);

// 4) get the target object for this invocation
System.out.println(
    "target object (from OrbCurrent) = " +
    curr.get_object());

// 5) get the current protocol, either IIOP or
// Orbix Protocol
System.out.println("protocol = " + curr.get_protocol());
// 6) get the socket object from which this
// Request came and get the port number from
// that sock object
Object sock = curr.get_socket();
long port = 0;

// if the sock object is an instance of Socket
// then either using IIOP or Orbix Protocol
if( sock instanceof java.net.Socket){
    java.net.Socket conn = (java.net.Socket)sock;
    port = conn.getLocalPort();
}
// otherwise we are using HTTP tunnelling
else if (sock instanceof java.net.URLConnection){
    System.out.println(
        "protocol ==Http tunnelling ");
    java.net.URLConnection conn =
        (java.net.URLConnection)sock;
    port = conn.getLocalPort().getURL().getPort();
}
System.out.println("Port num = " + port);

// 7) get the servername
System.out.println("Server = " + curr.get_server());
```

To use any of these methods the `IT_MULTI_THREADED_SERVER`, described in the Orbix Java configuration must be set to `true`.

## CORBA

### Orbix Java

```
pseudo interface Current { };

public class OrbCurrent extends org.omg.CORBA.Current
{
    public static org.omg.CORBA.Principal get_principal()
        throws org.omg.CORBA.SystemException;
    public static String get_principal_string()
        throws org.omg.CORBA.SystemException;
    public static org.omg.CORBA.ServerRequest get_request()
        throws org.omg.CORBA.SystemException;
    public static org.omg.CORBA.Object get_object()
        throws org.omg.CORBA.SystemException;
    public static int get_protocol()
        throws org.omg.CORBA.SystemException;
    public static java.lang.Object get_socket()
        throws org.omg.CORBA.SystemException;
    public static String get_server()
        throws org.omg.CORBA.SystemException;
}
```

## get\_object()

### Synopsis

```
public static org.omg.CORBA.Object get_object()
    throws org.omg.CORBA.SystemException;
```

### Description

Get the target invocation "Class `org.omg.CORBA.Object`" associated with this invocation.

### Return Value

`org.omg.CORBA.Object` The target object for this CORBA method invocation.

### Notes

Orbix-specific

### See Also

["Class `org.omg.CORBA.Current`" on page 12](#)  
["Interface `org.omg.CORBA.Object`" on page 18](#)

## get\_principal()

### Synopsis

```
public static org.omg.CORBA.Principal get_principal()
    throws org.omg.CORBA.SystemException;
```

### Description

This method returns the "Class `org.omg.CORBA.Principal`" object associated with this invocation. This allows the receiving object to establish who invoked the method.

### Return Value

`org.omg.CORBA.Principal` An object representing information about the invoker of a method

### Notes

Orbix-specific

### See Also

["Class `org.omg.CORBA.Principal`" on page 28](#)  
["Interface `org.omg.CORBA.Object`" on page 18](#)

## get\_principal\_string()

### Synopsis

```
public static String get_principal_string()
    throws org.omg.CORBA.SystemException;
```

### Description

This method returns the principal string of the invoker of a remote invocation.

### Return Value

String            The invoker of the remote method.

### Notes

Orbix-specific

### See Also

["Class org.omg.CORBA.Principal" on page 28](#)  
["Interface org.omg.CORBA.Object" on page 18](#)

## get\_protocol()

### Synopsis

```
public static int get_protocol()
    throws org.omg.CORBA.SystemException;
```

### Description

This method returns the protocol type associated with this invocation, it will be one of the following:

```
_CORBA.IT_INTEROPERABLE_OR_KIND  
_CORBA.IT_ORBIX_OR_KIND
```

### Return Value

int                The protocol type for this invocation, it is either IIOP or the Orbix protocol.

### Notes

Orbix-specific

### See Also

["Class org.omg.CORBA.Current" on page 12](#)  
["IT\\_INTEROPERABLE\\_OR\\_KIND" on page 240](#)  
["IT\\_ORBIX\\_OR\\_KIND" on page 240](#)

## get\_request()

### Synopsis

```
public static org.omg.CORBA.ServerRequest get_request()
    throws org.omg.CORBA.SystemException;
```

### Description

This method returns the "Class org.omg.CORBA.ServerRequest" object associated with this invocation.

### Return Value

org.omg.CORBA.ServerRequest    The ServerRequest object associated with this invocation.

### Notes

Orbix-specific

### See Also

["Class org.omg.CORBA.Principal" on page 28](#)  
["Class org.omg.CORBA.Current" on page 12](#)

## get\_server()

### Synopsis

```
public static String get_server()  
    throws org.omg.CORBA.SystemException;
```

### Description

Get the server name of this server.

### Return Value

String            The server name.

### Notes

Orbix-specific.

### See Also

["Class org.omg.CORBA.Current" on page 12](#)  
["impl\\_is\\_ready\(\)" on page 71](#)

## get\_socket()

### Synopsis

```
public static java.lang.Object get_socket()  
    throws org.omg.CORBA.SystemException;
```

### Description

This method returns the connection object associated with this invocation, it is either a connection of type `java.net.URLConnection` (for HTTP tunneling) or `java.lang.Socket`.

### Return Value

java.lang.Object    The connection object for this invocation.

### Notes

Orbix-specific.

### See Also

["Class org.omg.CORBA.Current" on page 12](#)

# Class `IE.Iona.OrbixWeb.CORBA.Principal`

## Synopsis

Class `Principal` implements the IDL pseudo-interface `Principal`. This represents information about principals (users). This information may be used to provide authentication and access control.

For more details on principals refer to the *Orbix Programmer's Guide Java Edition*.

## CORBA

```
// Pseudo IDL
```

```
pseudo interface Principal {  
    attribute sequence<octet> name;  
}
```

## Orbix Java

```
// Java
```

```
public class Principal extends org.omg.CORBA.Principal {  
    // Constructors  
    public Principal() throws org.omg.CORBA.SystemException;  
    public Principal(byte[] name,boolean doCopy)  
        throws org.omg.CORBA.SystemException;  
    public Principal(String name)  
        throws org.omg.CORBA.SystemException;  
  
    // Data accessor/modifier methods  
    public byte[] access_name()  
        throws org.omg.CORBA.SystemException;  
    public byte[] name() throws org.omg.CORBA.SystemException;  
    public void name(byte[] name)  
        throws org.omg.CORBA.SystemException;  
    public String toString()  
        throws org.omg.CORBA.SystemException;  
};
```

## Notes

CORBA-defined.

## Principal()

### Synopsis

```
public Principal() throws org.omg.CORBA.SystemException;
```

### Description

Default constructor.

### Notes

Orbix-specific.

## Principal()

### Synopsis

```
public Principal(byte[] name,boolean doCopy)
    throws org.omg.CORBA.SystemException;
```

### Description

Create a new `Principal` object identified by `name`. The second parameter indicates whether the supplied `name` should be used by the new object or whether a copy of the `name` should be taken.

### Parameters

<code>name</code>	Create a new <code>Principal</code> using this name.
<code>doCopy</code>	If <code>true</code> take a copy of the <code>name</code> parameter, otherwise, let the new <code>Principal</code> object refer to the <code>name</code> parameter.

### Notes

Orbix-specific.

## Principal()

### Synopsis

```
public Principal(String name)
    throws org.omg.CORBA.SystemException;
```

### Description

Create a new `Principal` object identified by `name`.

### Parameters

<code>name</code>	Use this to identify the <code>Principal</code> .
-------------------	---------------------------------------------------

### Notes

Orbix-specific.

### See Also

Other `Principal` constructors

## **access\_name**

### **Synopsis**

```
public byte[] access_name() throws  
    org.omg.CORBA.SystemException;
```

### **Description**

Accessor method that returns the Principal name.

### **Return Value**

byte[]                      A reference to the actual Principal name.

### **Notes**

Orbix-specific.

## **name()**

### **Synopsis**

```
public byte[] name() throws org.omg.CORBA.SystemException;
```

### **Description**

Accessor method used to retrieve the name of a Principal.

### **Return Value**

byte[]                      A copy of the Principal name.

### **Notes**

CORBA-defined.

## **name()**

### **Synopsis**

```
public void name(byte[] name)  
    throws org.omg.CORBA.SystemException;
```

### **Description**

Modifier method to change the name of a Principal.

### **Parameters**

name                      New name for Principal object.

### **Notes**

CORBA-defined.

## **toString()**

### **Synopsis**

```
public java.lang.String toString()  
    throws org.omg.CORBA.SystemException;
```

### **Description**

Accessor method that returns the Principal name.

### **Return Value**

String                      The actual Principal name.

### **Notes**

Orbix-specific.



# Class IE.Iona.OrbixWeb.CORBA.Request

## Synopsis

Class `Request` supports the Dynamic Invocation Interface (DII), whereby an application may issue a request for any interface, even if that interface was unknown at the time the application was compiled.

Orbix Java allows invocations, which are instances of class `Request`, to be constructed by specifying at runtime the target object reference, the operation name and the parameters. Such calls are termed *dynamic* because the IDL interfaces used by a program do not have to be *statically* determined at the time the program is designed and implemented.

Individual `Request` objects can be intercepted by `Filter` objects and `IT_reqTransformer` objects. For information on how to use these see the *Orbix Programmer's Guide Java Edition* and "[Class IE.Iona.OrbixWeb.Features.IT\\_reqTransformer](#)" on [page 219](#).

## Note:

The Orbix-specific `extract()` and `insert()` methods are no longer supported.

## CORBA

```
// Pseudo IDL
interface Request {
    readonly attribute Object target;
    readonly attribute Identifier operation;
    readonly attribute NVList arguments;
    readonly attribute NamedValue result;
    readonly attribute Environment env;
    readonly attribute ExceptionList exceptions;
    readonly attribute ContextList contexts;
    attribute Context ctx;
    any add_in_arg();
    any add_named_in_arg(in string name);
    any add_inout_arg();
    any add_named_inout_arg(in string name);
    any add_out_arg();
    any add_named_out_arg(in string name);
    void set_return_type(in TypeCode tc);
    any return_value();
    void invoke();
    void send_oneway();
    void send_deferred();
    void get_response();
    boolean poll_response();
};
```

## Orbix Java

```
public class Request extends org.omg.CORBA.Request {

    // Constructors
    public Request();
        throws org.omg.CORBA.SystemException;
    public Request(IE.Iona.OrbixWeb.CORBA.ObjectRef target);
        throws org.omg.CORBA.SystemException;
    public Request(IE.Iona.OrbixWeb.CORBA.ObjectRef target,
        java.lang.String operationName);
        throws org.omg.CORBA.SystemException;

    public static Request _nil();
        throws org.omg.CORBA.SystemException;
```

```

// Accessor Methods
    public java.lang.String operation();
        throws org.omg.CORBA.SystemException;
    public void setOperation(java.lang.String operationName);
        throws org.omg.CORBA.SystemException;
    public org.omg.CORBA.Object target();
        throws org.omg.CORBA.SystemException;
    public void setTarget(org.omg.CORBA.Object target);
        throws org.omg.CORBA.SystemException;
    public org.omg.CORBA.NVList arguments();
        throws org.omg.CORBA.SystemException;
    public org.omg.CORBA.NamedValue result();
        throws org.omg.CORBA.SystemException;
    public org.omg.CORBA.Environment env();
        throws org.omg.CORBA.SystemException;
    public void env(org.omg.CORBA.Environment env);
        throws org.omg.CORBA.SystemException;
    public org.omg.CORBA.ExceptionList exceptions();
        throws org.omg.CORBA.SystemException;
    public void exceptions(org.omg.CORBA.ExceptionList
        exceptions);
        throws org.omg.CORBA.SystemException;
    public org.omg.CORBA.ContextList contexts();
        throws org.omg.CORBA.SystemException;
    public void contexts(org.omg.CORBA.ContextList contexts);
        throws org.omg.CORBA.SystemException;
    public org.omg.CORBA.Context ctx();
        throws org.omg.CORBA.SystemException;
    public void ctx(org.omg.CORBA.Context c);
        throws org.omg.CORBA.SystemException;

// Object reuse methods
    public void reset();
        throws org.omg.CORBA.SystemException;
    public void reset(org.omg.CORBA.Object target);
        throws org.omg.CORBA.SystemException;
    public void reset(java.lang.String operationName);
        throws org.omg.CORBA.SystemException;
    public void reset(org.omg.CORBA.Object target,
        java.lang.String operationName);
        throws org.omg.CORBA.SystemException;

// Argument manipulation methods
    public void add_arg(java.lang.String name,
        org.omg.CORBA.Any a, int flag);
        throws org.omg.CORBA.SystemException;
    public org.omg.CORBA.Any add_in_arg();
        throws org.omg.CORBA.SystemException;
    public org.omg.CORBA.Any add_inout_arg();
        throws org.omg.CORBA.SystemException;
    public org.omg.CORBA.Any add_out_arg();
        throws org.omg.CORBA.SystemException;
    public org.omg.CORBA.Any add_named_in_arg
        (java.lang.String name);
        throws org.omg.CORBA.SystemException;
    public org.omg.CORBA.Any add_named_out_arg
        (java.lang.String name);
        throws org.omg.CORBA.SystemException;
    public org.omg.CORBA.Any add_named_inout_arg

```

```

        (java.lang.String name);
        throws org.omg.CORBA.SystemException;
public void set_return_type(org.omg.CORBA.TypeCode tc);
        throws org.omg.CORBA.SystemException;
public org.omg.CORBA.Any return_value();
        throws org.omg.CORBA.SystemException;

// Invocation methods
public void invoke();
        throws org.omg.CORBA.SystemException;
public void send_deferred();
        throws org.omg.CORBA.SystemException;
public void send_oneway();
        throws org.omg.CORBA.SystemException;
public boolean isDynamic();
        throws org.omg.CORBA.SystemException;
public void get_response();
        throws org.omg.CORBA.SystemException;
public boolean poll_response();
        throws org.omg.CORBA.SystemException;
public boolean isException();
        throws org.omg.CORBA.SystemException;
public java.lang.Exception _getException();
        throws org.omg.CORBA.SystemException;
public boolean isOneWay();
        throws org.omg.CORBA.SystemException;
public org.omg.CORBA.portable.InputStream
create_input_stream();
        throws org.omg.CORBA.SystemException;
public org.omg.CORBA.portable.OutputStream
create_output_stream();
        throws org.omg.CORBA.SystemException;
public java.lang.Object getClientConnection();
        throws org.omg.CORBA.SystemException;
public int getMessageLength();
        throws org.omg.CORBA.SystemException;

// ServiceContext methods
public ServiceContext getServiceContext( int id );
        throws org.omg.CORBA.SystemException;
public void addServiceContext( ServiceContext ctx );
        throws org.omg.CORBA.SystemException;
public ServiceContext[] getSCL();
        throws org.omg.CORBA.SystemException;
public setSCL( ServiceContext[] scl );
        throws org.omg.CORBA.SystemException;
public void deleteSCL();
        throws org.omg.CORBA.SystemException;
}

```

## Notes

CORBA-defined.

## See Also

["Class IE.Iona.OrbixWeb.Features.Filter" on page 207](#)  
["Class IE.Iona.OrbixWeb.Features.AuthenticationFilter" on page 203](#)  
["Class IE.Iona.OrbixWeb.Features.ThreadFilter" on page 234](#)  
["Class IE.Iona.OrbixWeb.Features.IT\\_reqTransformer" on page 219](#)

## Request()

### Synopsis

```
public Request();
```

### Description

Default constructor. The target object and the operation name for the request should be filled in.

### Notes

Orbix-specific.

### See Also

`create_request()` and `_request()` in  
["Interface org.omg.CORBA.Object" on page 18](#)

## Request()

### Synopsis

```
public Request(IE.Iona.OrbixWeb.CORBA.ObjectRef target);
```

### Description

Constructs a `Request` by specifying its target object's reference. You can set the operation name for the request using `Request.setOperation()`.

### Parameters

`target`            The object which is the target of the request.

### Notes

Orbix-specific.

### See Also

`create_request()` and `_request()` in  
["Interface org.omg.CORBA.Object" on page 18](#)  
["setOperation\(\)" on page 179](#)

## Request()

### Synopsis

```
public Request(IE.Iona.OrbixWeb.CORBA.ObjectRef target  
              java.lang.String operationName);
```

### Description

Constructs a `Request` by specifying its target object's reference and the required operation name.

### Parameters

`target`            The object that is the target of the request.  
`operationName`    The operation name for the request.

### Notes

Orbix-specific.

### See Also

`create_request()` and `_request()` in  
["Interface org.omg.CORBA.Object" on page 18](#)

## add\_arg()

### Synopsis

```
public void add_arg(java.lang.String name,  
                   org.omg.CORBA.Any a,  
                   int flag);
```

### Description

Adds the contents of the *Any* (not the *Any* itself) as a parameter to the *Request*. For out and inout parameters the *Any* itself is updated with the returned value.

### Parameters

**name**        A programmer description of the argument being passed into the *Request*.

**a**            The value to be added to the *Request* object.

**flag**        Specifies whether the argument is in, out, or inout.

### Notes

Orbix-specific.

### See Also

"Class org.omg.CORBA.ARG\_IN" on page 99  
"Class org.omg.CORBA.ARG\_INOUT" on page 4  
"Class org.omg.CORBA.ARG\_OUT" on page 5  
Other `add_arg` methods

## add\_in\_arg()

### Synopsis

```
public org.omg.CORBA.Any add_in_arg();
```

### Description

Creates an instance of `org.omg.CORBA.Any` as a holder for an in parameter for the *Request* object. You insert the in parameter value into the returned *Any* object.

### Return Value

Returns the constructed *Any*.

### Notes

CORBA-defined.

### See Also

Other `add_arg` methods

## add\_inout\_arg()

### Synopsis

```
public org.omg.CORBA.Any add_inout_arg();
```

### Description

Creates an instance of `org.omg.CORBA.Any` as a holder for an inout parameter for the *Request* object. The client code must insert the inout parameter value into the *Any* object. When the request is invoked the server may change the contents of the *Any*. The *Any* is updated when the reply is received. The client can then examine the *Any* for the updated value.

### Return Value

Returns the constructed *Any*.

### Notes

CORBA-defined.

### See Also

Other `add_arg` methods

## add\_named\_in\_arg()

**Synopsis** `public org.omg.CORBA.Any add_named_in_arg(java.lang.String name);`

**Description** Creates an instance of `org.omg.CORBA.Any` as a holder for an `in` parameter for the `Request` object. You insert the `in` parameter value into the returned `Any` object.

**Parameters**

<code>name</code>	A programmer description of the argument passed into the <code>Request</code> .
-------------------	---------------------------------------------------------------------------------

**Return Value** Returns the constructed `Any`.

**Notes** CORBA-defined.

**See Also** Other `add_arg` methods

## add\_named\_inout\_arg()

**Synopsis** `public org.omg.CORBA.Any add_named_inout_arg(java.lang.String name);`

**Description** Creates an instance of `org.omg.CORBA.Any` as a holder for an `inout` parameter for the `Request` object. The client code must insert the `inout` parameter value into the `Any` object. When the request is invoked the server may change the contents of the `Any`. The `Any` is updated when the reply is received. The client can then examine the `Any` for the updated value.

**Parameters**

<code>name</code>	A programmer description of the argument passed into the <code>Request</code> .
-------------------	---------------------------------------------------------------------------------

**Return Value** Returns the constructed `Any`.

**Notes** CORBA-defined

**See Also** Other `add_arg` methods

## add\_named\_out\_arg()

**Synopsis** `public org.omg.CORBA.Any add_named_out_arg(java.lang.String name);`

**Description** Creates an instance of `org.omg.CORBA.Any` as a holder for an `out` parameter for the `Request` object. When the request is invoked the server may change the contents of the `Any`. The `Any` is updated when the reply is received. The client can then examine the `Any` for the returned value.

**Parameters**

<code>name</code>	A programmer description of the argument passed into the <code>Request</code>
-------------------	-------------------------------------------------------------------------------

**Return Value** Returns the constructed `Any`.

**Notes** CORBA-defined.

**See Also** Other `add_arg` methods

## add\_out\_arg()

### Synopsis

```
public org.omg.CORBA.Any add_out_arg();
```

### Description

Creates an instance of `org.omg.CORBA.Any` as a holder for an `out` parameter for the `Request` object. When the request is invoked the server may change the contents of the `Any`. The `Any` will be updated when the reply is received. The client can then examine the `Any` for the returned value.

### Return Value

Returns the constructed `Any`.

### Notes

CORBA-defined.

### See Also

Other `add_arg` methods

## arguments()

### Synopsis

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

### Description

Returns the arguments to the `Request`'s operation in an `NVList`.

### Notes

CORBA-defined.

### See Also

["Class org.omg.CORBA.NVList" on page 17](#)

## contexts()

### Synopsis

```
public org.omg.CORBA.ContextList contexts();
```

### Description

Returns the contexts for the `Request`'s operation in a `ContextList`.

### Notes

CORBA-defined.

### See Also

["Class org.omg.CORBA.ContextList" on page 10](#)

["Class IE.Iona.OrbixWeb.CORBA.Request" on page 167](#)

## contexts()

### Synopsis

```
public void contexts(org.omg.CORBA.ContextList contexts);
```

### Description

Sets the list of contexts for the `Request`'s operation.

### Notes

Orbix-specific.

### See Also

["Class org.omg.CORBA.ContextList" on page 10](#)

["Class org.omg.CORBA.Request" on page 29](#)

["Class IE.Iona.OrbixWeb.CORBA.Request" on page 167](#)

## create\_input\_stream()

### Synopsis

```
public org.omg.CORBA.portable.InputStream create_input_stream();
```

### Description

Returns a new `InputStream` object for accessing data contained within the `Request` object.

This method is useful when piggy-backing with filters.

### Notes

Orbix-specific.

### See Also

["Class org.omg.CORBA.portable.InputStream" on page 25](#)

["Class IE.Iona.OrbixWeb.Features.Filter" on page 207](#)

["create\\_output\\_stream\(\)" on page 174](#)

## create\_output\_stream()

### Synopsis

```
public org.omg.CORBA.portable.OutputStream  
create_output_stream();
```

### Description

Returns a new `OutputStream` object for inputting data contained into the `Request` object.

This method is useful when piggy-backing with filters.

### Notes

Orbix-specific.

### See Also

["Class org.omg.CORBA.portable.OutputStream" on page 26](#)  
["Class IE.Iona.OrbixWeb.Features.Filter" on page 207](#)  
["create\\_input\\_stream\(\)" on page 173](#)

## ctx()

### Synopsis

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

### Description

Returns the `Context` object for the `Request`.

### Notes

CORBA-defined.

### See Also

["Class org.omg.CORBA.Context" on page 9](#)

## ctx()

### Synopsis

```
public void ctx(org.omg.CORBA.Context c);
```

### Description

Inserts the `Context` object `c` into the `Request`.

### Notes

CORBA-defined.

### See Also

["Class org.omg.CORBA.Context" on page 9](#)  
["Class org.omg.CORBA.Request" on page 29](#)

## env()

### Synopsis

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

### Description

Returns the `Environment` object for the `Request`. After the `Request` is invoked upon the `Environment` object, contains any `Exception` object thrown during the invocation.

### Notes

CORBA-defined.

### See Also

["Class org.omg.CORBA.Environment" on page 14](#)  
["Class IE.Iona.OrbixWeb.CORBA.Environment" on page 94](#)

## env()

### Synopsis

```
public void env(org.omg.CORBA.Environment env);
```

### Description

Inserts the `Environment` object `env` into the `Request`.

### Notes

Orbix-specific.

### See Also

["Class org.omg.CORBA.Environment" on page 14](#)  
["Class IE.Iona.OrbixWeb.CORBA.Environment" on page 94](#)  
["Class org.omg.CORBA.Request" on page 29](#)



## exceptions()

### Synopsis

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

### Description

Returns the list of user-defined exceptions that the `Request` object understands. After the `Request` is invoked the `Environment` object, contains any `Exception` object thrown during the invocation.

### Notes

CORBA-defined.

### See Also

["Class org.omg.CORBA.Environment" on page 14](#)  
["Class org.omg.CORBA.ExceptionList" on page 15](#)  
["Class IE.Iona.OrbixWeb.CORBA.Environment" on page 94](#)  
["Class IE.Iona.OrbixWeb.CORBA.Request" on page 167](#)

## exceptions()

### Synopsis

```
public void exceptions(org.omg.CORBA.ExceptionList exceptions);
```

### Description

Sets the list of `Exception` classes that the `Request` object understands for the current invocation.

### Notes

Orbix-specific.

### See Also

["Class org.omg.CORBA.ExceptionList" on page 15](#)  
["Class org.omg.CORBA.Request" on page 29](#)

## getClientConnection()

### Synopsis

```
public java.lang.Object getClientConnection();
```

### Description

Returns the socket object for the `Request` object's connection. This is either a `java.net.Socket` or `java.net.URLConnection` object depending on the type of connection established.

### Notes

Orbix-specific.

### See Also

["Interface IE.Iona.OrbixWeb.Features.ioCallback" on page 217](#)

## \_getException()

### Synopsis

```
public java.lang.Exception _getException();
```

### Description

If an exception has been raised when invoking the `Request` object `_getException` returns the `Exception` object thrown.

This method is deprecated in favour of using the CORBA-defined `Request.env().exception()`

### Notes

Orbix-specific.

### See Also

["Class org.omg.CORBA.Environment" on page 14](#)  
["Class org.omg.CORBA.Request" on page 29](#)

## getMessageLength()

### Synopsis

```
public int getMessageLength();
```

### Description

Returns the size of the buffer in the `Request` object.

You can use this function in conjunction with filters to monitor the volume of traffic into your server.

### Notes

Orbix-specific.

### See Also

["Class IE.Iona.OrbixWeb.CORBA.Request" on page 167](#)

## get\_response()

### Synopsis

```
public void get_response();
```

### Description

Determines whether a request has completed successfully. The call blocks until the `Request` (invoked using `send_deferred()`) has completed.

### Notes

CORBA-defined.

### See Also

["Class org.omg.CORBA.Request" on page 29](#)  
["env\(\)" on page 174](#)  
["result\(\)" on page 178](#)  
["send\\_deferred\(\)" on page 178](#)

## invoke()

### Synopsis

```
public void invoke();
```

### Description

Instructs Orbix Java to invoke the request. The parameters to the request must already be set up. The caller is blocked until the request has been processed by the target object, or until an exception occurs.

To make a non-blocking request, see `IE.Iona.OrbixWeb.CORBA.Request.send_deferred()` and `IE.Iona.OrbixWeb.CORBA.Request.send_oneway()`.

### Notes

CORBA-defined.

### See Also

["Class org.omg.CORBA.Request" on page 29](#)  
["env\(\)" on page 174](#)  
["result\(\)" on page 178](#)  
["send\\_deferred\(\)" on page 178](#)  
["send\\_oneway\(\)" on page 179](#)

## isDynamic()

### Synopsis

```
public boolean isDynamic();
```

### Description

Returns `true` or `false` depending on whether or not the `Request` object has been invoked by the DII.

It only makes sense to use this in conjunction with filter points.

### Notes

Orbix-specific.

### See Also

["Class IE.Iona.OrbixWeb.Features.Filter" on page 207](#)

## isException()

### Synopsis

```
public boolean isException();
```

### Description

Returns `true` or `false` depending on whether or not the `Request` object contains an exception object.

This method is deprecated in favour of using the CORBA-defined `org.omg.CORBA.Request.env()`.

### Notes

Orbix-specific.

### See Also

`env()` in ["Class org.omg.CORBA.Request" on page 29](#)  
["env\(\)" on page 174](#)

## **\_nil()**

**Synopsis** `public org.omg.CORBA.Request _nil();`  
**Description** Returns a null.  
**Notes** CORBA-defined.

## **operation()**

**Synopsis** `public java.lang.String operation();`  
**Description** Returns the request's operation.  
**Notes** CORBA-defined.  
**See Also** ["setOperation\(\)" on page 179](#)

## **poll\_response()**

**Synopsis** `public boolean poll_response();`  
**Description** Checks to see if a reply has been received for a deferred invocation upon a `Request` object. If the reply has arrived, `poll_response` automatically updates the `Request` object.  
**Notes** CORBA-defined.  
**See Also** [`getResponse\(\)`, `result\(\)`, and `send\_deferred\(\)` in "Class org.omg.CORBA.Request" on page 29](#)  
["`get\_response\(\)`" on page 176](#)  
["`result\(\)`" on page 178](#)  
["`send\_deferred\(\)`" on page 178](#)

## **reset()**

**Synopsis** `public void reset();`  
`public void reset(org.omg.CORBA.object target);`  
`public void reset(java.lang.String operationName);`  
`public void reset(org.omg.CORBA.object target, java.lang.String operationName);`

**Description** Allows the `Request` object to be reused. This can be particularly useful if you have a whole batch of requests to be invoked upon the same object or if you have a batch of objects on which you wish to invoke the same operation.  
The previous `target` or `operationName` is reused if you do not use a version of `reset()` that specifies a new `target`, or `operation`, or both.  
Although you can reuse the `targets` and `operationNames`, you must always refresh the arguments being passed to the `Request`. Arguments cannot be reused.

### **Parameters**

<code>target</code>	The target object on which the <code>Request</code> is invoked. If this is not specified, the previous target object is reused.
<code>operationName</code>	The operation to be invoked upon the target object. If this is not specified, the previous operation is reused.

**Notes** Orbix-specific.

**See Also** ["Request\(\)" on page 170](#)  
["setTarget\(\)" on page 180](#)  
["setOperation\(\)" on page 179](#)

## result()

**Synopsis**

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

**Description** Returns the Request's result as a NamedValue object. It only makes sense to call this method after a Request has been invoked and has returned.

**Notes** CORBA-defined.

**See Also** ["Class IE.Iona.OrbixWeb.CORBA.Request" on page 167](#)

## return\_value()

**Synopsis**

```
public org.omg.CORBA.Any return_value();
```

**Description** Returns the Request's result as an Any. It only makes sense to call this method after a Request has been invoked and has returned.

This method is preferred over `IE.Iona.OrbixWeb.CORBA.result()`.

**Notes** CORBA-defined.

## send\_deferred()

**Synopsis**

```
public void send_deferred();
```

**Description** Instructs Orbix Java to invoke the request in a non-blocking fashion. The parameters to the request must already be set up. As you, the caller, are not blocked you may continue to do work in parallel with the target object's processing of the call.

The caller can use the method `org.omg.CORBA.Request.poll_response()` to determine whether the operation completed.

You should use the method `org.omg.CORBA.Request.get_response()` to determine the outcome of the request.

To make a blocking request, refer to the `invoke()` method in ["Class org.omg.CORBA.Request" on page 29](#).

To send multiple deferred requests, refer to the `multiple_requests_deferred` method in ["Class org.omg.CORBA.ORB" on page 19](#).

**Notes** CORBA-defined.

**See Also** `getResponse()`, `invoke()`, `poll_response()`, and `send_oneway()` in ["Class org.omg.CORBA.Request" on page 29](#)  
["get\\_response\(\)" on page 176](#)  
["invoke\(\)" on page 176](#)  
["poll\\_response\(\)" on page 177](#)  
["send\\_oneway\(\)" on page 179](#)

## send\_oneway()

### Synopsis

```
public void send_oneway();
```

### Description

Instructs Orbix Java to invoke a `oneway` request.

You can use this method even if the operation has not been defined to be `oneway` in its IDL definition. The caller should not expect any `in` or `inout` parameters to be updated.

The parameters to the request must already be set up before making the call. The caller is not blocked, and can continue to work in parallel with the target object's processing of the call.

To make a blocking request, see `CORBA.Request.invoke()`.

### Notes

Orbix-specific.

### See Also

`getResponse()`, `invoke()`, `poll_response()`, and `send_deferred()` in  
["Class org.omg.CORBA.Request" on page 29](#)  
["get\\_response\(\)" on page 176](#)  
["invoke\(\)" on page 176](#)  
["poll\\_response\(\)" on page 177](#)  
["send\\_deferred\(\)" on page 178](#)

## setOperation()

### Synopsis

```
public void setOperation(java.lang.String operationName);
```

### Description

Sets the operation for the `Request` object.

### Notes

Orbix-specific.

### See Also

`create_request()`, `request()`, and `operation()` in  
["Class org.omg.CORBA.Request" on page 29](#)  
["operation\(\)" on page 177](#)

## set\_return\_type()

### Synopsis

```
public void set_return_type(org.omg.CORBA.TypeCode tc);
```

### Description

Sets the `TypeCode` for the return type of the `Request` object.

### Notes

CORBA-defined.

### See Also

`result()` and `return_value()` in  
["Class org.omg.CORBA.Request" on page 29](#)  
["result\(\)" on page 178](#)  
["return\\_value\(\)" on page 178](#)

## setTarget()

### Synopsis

```
public void setTarget(org.omg.CORBA.object target);
```

### Description

Sets the target CORBA object for the Request.

### Notes

Orbix-specific.

### See Also

[create\\_request\(\)](#) and [target\(\)](#) in  
["Class org.omg.CORBA.Request" on page 29](#)  
[request\(\)](#) in ["Interface org.omg.CORBA.Object" on page 18](#).  
["reset\(\)" on page 177](#)

## target()

### Synopsis

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

### Description

Returns the target CORBA object for the Request.

### Notes

CORBA-defined

### See Also

[\\_create\\_request\(\)](#) in ["Interface org.omg.CORBA.Object" on page 18](#)  
[\\_request\(\)](#) in ["Interface org.omg.CORBA.Object" on page 18](#)  
[target\(\)](#) in ["Class org.omg.CORBA.Request" on page 29](#)  
["reset\(\)" on page 177](#)

# Class `com.Iona.OrbixWeb.CORBA.singletonORB`

## Synopsis

The IDL to Java mapping distinguishes between a fully-functional ORB and a singleton ORB with restricted functionality. Earlier versions of Orbix Java do not make this distinction. In this version however, there are important differences between these ORB types, compliant with the revised mapping. The type of ORB that `ORB.init()` returns depends on whether the call has parameters.

### Singleton ORB

An `ORB.init()` call with no parameters returns an instance of a singleton ORB. There is only one instance of the singleton ORB in a virtual machine. The singleton ORB's restricted functionality is mainly for security reasons in applets. You can call the following operations on the singleton ORB:

- `create_list()`
- `create_named_value()`
- `create_exception_list()`
- `create_context_list()`
- `get_default_context()`
- `create_environment()`
- `create_xxx_tc()`  
(where xxx is a defined Typecode type)
- `get_primitive_tc()`
- `create_any()`
- `create_output_stream()`

An attempt to call any other ORB operations on the singleton ORB results in a system exception.

### Fully Functional ORB

Any of the forms of `ORB.init()` with parameters returns a new, fully functional ORB. In earlier versions of Orbix Java, each call in the same VM returns the same ORB (`_CORBA.Orbix`). In this version each call returns a different new ORB. This adds considerable flexibility to some applications, as each new ORB is completely independent from any other. For example, in its configuration, connections, listener ports and server object tables. Multiple ORBs also facilitate applet separation.

## CORBA

```
// Pseudo IDL

pseudo interface ORB {

// Creation methods
NVList create_list(in long count);
NamedValue create_named_value(in String name,
                               in Any value,
                               in Flags flags);
ExceptionList create_exception_list();
ContextList create_context_list();
Context get_default_context();
Environment create_environment();
Any create_any();
OutputStream create_output_stream();
// Typecode creation
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);
TypeCode create_enum_tc
    (    in RepositoryId id,
      in Identifier name,
      in EnumMemberSeq members);
TypeCode create_alias_tc
    (    in RepositoryId id,
      in Identifier name,
      in TypeCode original_type);
TypeCode create_exception_tc
    (    in RepositoryId id,
      in Identifier name,
      in StructMemberSeq members);
TypeCode create_interface_tc
    (    in RepositoryId id,
      in Identifier name);
TypeCode create_string_tc
    (    in unsigned long bound);
TypeCode create_wstring_tc
    (    in unsigned long bound);
TypeCode create_sequence_tc
    (    in unsigned long bound,
      in TypeCode element_type);
TypeCode create_recursive_sequence_tc
    (    in unsigned long bound,
      in unsigned long offset);
TypeCode create_array_tc
    (    in unsigned long length,
      in TypeCode element_type);
TypeCode get_primitive_tc(in TCKind tcKind);
```



## Orbix Java

```
public class ORB extends org.omg.CORBA.ORB {
    public org.omg.CORBA.Any create_any()
        throws org.omg.CORBA.SystemException;
    public org.omg.CORBA.ContextList create_context_list()
        throws org.omg.CORBA.SystemException;
    public org.omg.CORBA.Environment create_environment()
        throws org.omg.CORBA.SystemException;
    public org.omg.CORBA.ExceptionList create_exception_list()
        throws org.omg.CORBA.SystemException;
    public org.omg.CORBA.NVList create_list(int count)
        throws org.omg.CORBA.SystemException;
    public org.omg.CORBA.NamedValue create_named_value(
        String name,
        org.omg.CORBA.Any value,
        int flags)
        throws org.omg.CORBA.SystemException;
    public org.omg.CORBA.portable.OutputStream
        create_output_stream()
        throws org.omg.CORBA.SystemException;
    public org.omg.CORBA.Context get_default_context()
        throws org.omg.CORBA.SystemException;
    public org.omg.CORBA.TypeCode get_primitive_tc(
        org.omg.CORBA.TCKind tcKind)
        throws org.omg.CORBA.SystemException;
}
```

### Notes

CORBA-defined.

### See Also

["Class IE.Iona.OrbixWeb.CORBA.ORB" on page 120](#)

## create\_any()

### Synopsis

```
public org.omg.CORBA.Any create_any()
    throws org.omg.CORBA.SystemException;
```

### Description

Creates a new empty "Class IE.Iona.OrbixWeb.CORBA.Any".

### Return Value

org.omg.CORBA.Any     The new Any.

### Notes

CORBA-defined

### See Also

["Class IE.Iona.OrbixWeb.CORBA.ORB" on page 120](#)  
["Class IE.Iona.OrbixWeb.CORBA.Any" on page 47](#)

## create\_context\_list()

### Synopsis

```
public org.omg.CORBA.ContextList create_context_list()  
    throws org.omg.CORBA.SystemException;
```

### Description

When making an invocation using the DII on an IDL operation that has a Context specified, you must pass a ContextList parameter to the `_create_request()` method in ["Interface org.omg.CORBA.Object" on page 18](#). This method allows you to create an empty ["Class org.omg.CORBA.ContextList"](#).

### Return Value

contextList            An empty contextList object.

### Notes

CORBA-defined.

### See Also

["Class IE.Iona.OrbixWeb.CORBA.ORB" on page 120](#)  
["Class org.omg.CORBA.ORB" on page 19](#)  
["Class org.omg.CORBA.ContextList" on page 10](#)  
["Class IE.Iona.OrbixWeb.CORBA.ContextList" on page 90](#)  
["Class IE.Iona.OrbixWeb.CORBA.Request" on page 167](#)  
["Class org.omg.CORBA.Context" on page 9](#)  
`_create_request()` method in ["Interface org.omg.CORBA.Object" on page 18](#)

## create\_environment()

### Synopsis

```
public org.omg.CORBA.Environment create_environment()  
    throws org.omg.CORBA.SystemException;
```

### Description

The ["Class org.omg.CORBA.Environment"](#) object is used with the DII to allow exception information to be returned from an operation invocation. This methods creates a new empty Environment object that you can use as a parameter to the `_create_request()` method in ["Interface org.omg.CORBA.Object" on page 18](#).

### Return Value

Environment            A new empty Environment object.

### Notes

CORBA-defined.

### See Also

["Class IE.Iona.OrbixWeb.CORBA.ORB" on page 120](#)  
["Class org.omg.CORBA.ORB" on page 19](#)  
["Class org.omg.CORBA.ContextList" on page 10](#)  
["Class IE.Iona.OrbixWeb.CORBA.Request" on page 167](#)  
["Class IE.Iona.OrbixWeb.CORBA.Environment" on page 94](#)  
["Class org.omg.CORBA.Environment" on page 14](#)  
`_create_request()` method in ["Interface org.omg.CORBA.Object" on page 18](#)

## create\_exception\_list()

### Synopsis

```
public org.omg.CORBA.ExceptionList create_exception_list()
    throws org.omg.CORBA.SystemException;
```

### Description

A "Class [org.omg.CORBA.ExceptionList](#)" is used by the DII to describe the exceptions that can be raised by IDL operations. This method creates an empty `ExceptionList` to be inserted into the `Request`.

### Return Value

`ExceptionList`            An empty `ExceptionList`.

### Notes

CORBA-defined.

### See Also

["Class org.omg.CORBA.ExceptionList" on page 15](#)  
["Class IE.Iona.OrbixWeb.CORBA.ORB" on page 120](#)  
["Class org.omg.CORBA.ORB" on page 19](#)  
["Class IE.Iona.OrbixWeb.CORBA.Request" on page 167](#)  
["Class IE.Iona.OrbixWeb.CORBA.Environment" on page 94](#)  
["Class org.omg.CORBA.Environment" on page 14](#)  
[\\_create\\_request\(\) method in "Interface org.omg.CORBA.Object" on page 18](#)

## create\_list()

### Synopsis

```
public org.omg.CORBA.NVList create_list(int count)
    throws org.omg.CORBA.SystemException;
```

### Description

Creates an empty "Class [org.omg.CORBA.NVList](#)" for use as a parameter type or for use as a way to describe the parameters to an operation invocation when using the `_create_request()` method in "[Interface org.omg.CORBA.Object](#)" on page 18.

`add_item()` is the only way to add an item to an `NVList`.  
`get_length()` returns number of times `add_item()` was called. The `NVList` always starts with element number 0.

### Parameters

`count`                    The size of the `NVList` to create.

### Return Value

`NVList`                    The empty `NVList` returned by this method.

### Notes

CORBA-defined.

### See Also

["Class IE.Iona.OrbixWeb.CORBA.ORB" on page 120](#)  
["Class org.omg.CORBA.ORB" on page 19](#)  
["Class org.omg.CORBA.NVList" on page 17](#)  
["Class IE.Iona.OrbixWeb.CORBA.NVList" page 102](#)  
["Class IE.Iona.OrbixWeb.CORBA.Request" on page 167](#)  
[\\_create\\_request\(\) method in "Interface org.omg.CORBA.Object" on page 18](#)

## create\_named\_value()

### Synopsis

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

### Description

Create an empty "Class [org.omg.CORBA.NamedValue](#)". This can be inserted into a "Class [IE.Iona.OrbixWeb.CORBA.NVList](#)" when using the DII as one of the parameters to the operation invocation.

### Parameters

name	The name of the NamedValue.
value	An Any to be inserted into the value of the NamedValue.
flags	Indicates whether this is an IN, OUT or INOUT parameter.

### Return Value

NameValue	The NamedValue constructed by this method call.
-----------	-------------------------------------------------

### Notes

CORBA-defined.

### See Also

["Class IE.Iona.OrbixWeb.CORBA.ORB" on page 120](#)  
["Class org.omg.CORBA.ORB" on page 19](#)  
["Class org.omg.CORBA.NVList" on page 17](#)  
["Class IE.Iona.OrbixWeb.CORBA.NVList" page 102](#)  
["Class IE.Iona.OrbixWeb.CORBA.NamedValue" on page 98](#)  
["Class org.omg.CORBA.NamedValue" on page 16](#)  
[\\_create\\_request\(\) method in "Interface org.omg.CORBA.Object" on page 18](#)

## create\_output\_stream()

### Synopsis

```
public org.omg.CORBA.portable.OutputStream  
    create_output_stream()  
    throws org.omg.CORBA.SystemException;
```

### Description

The Input/OutputStream classes provide methods for the reading and writing of all of the mapped IDL types to and from streams. Their implementations are used inside the ORB to marshal parameters and to insert and extract complex data types into and from Anys and Requests.

The streaming APIs are found in the "Class [org.omg.CORBA.portable.Streamable](#)" package.

The ORB object is used as a factory to create an output stream. An input stream may be created from an output stream.

### Return Value

OutputStream	A new OutputStream object.
--------------	----------------------------

### Notes

CORBA-defined.

## See Also

["Class IE.Iona.OrbixWeb.CORBA.ORB" on page 120](#)  
["Class org.omg.CORBA.portable.Streamable" on page 27](#)  
["Class org.omg.CORBA.portable.InputStream" on page 25](#)  
["Class org.omg.CORBA.portable.OutputStream" on page 26](#)  
["Class IE.Iona.OrbixWeb.CORBA.Request" on page 167](#)

## get\_default\_context()

### Synopsis

```
public org.omg.CORBA.Context get_default_context()  
    throws org.omg.CORBA.SystemException;
```

### Description

This operation returns a reference to the default process ["Class org.omg.CORBA.Context"](#) object. You can then use this for the Context parameter to an invocation.

### Return Value

context                      The default Context object for this process.

### Notes

CORBA-defined.

### See Also

["Class IE.Iona.OrbixWeb.CORBA.ORB" on page 120](#)  
["Class org.omg.CORBA.ORB" on page 19](#)  
["Class org.omg.CORBA.Context" on page 9](#)  
["Class IE.Iona.OrbixWeb.CORBA.Context" on page 83](#)

## create\_tc()

### Synopsis

```
public org.omg.CORBA.TypeCode create_alias_tc(  
    String id,  
    String name,  
    org.omg.CORBA.TypeCode original_type)  
    throws org.omg.CORBA.SystemException;  
public org.omg.CORBA.TypeCode create_array_tc(  
    int length,  
    org.omg.CORBA.TypeCode element_type)  
    throws org.omg.CORBA.SystemException;  
public org.omg.CORBA.TypeCode create_enum_tc(String id,  
    String name,  
    String[] members)  
    throws org.omg.CORBA.SystemException;  
public org.omg.CORBA.TypeCode create_exception_tc(  
    String id,  
    String name,  
    org.omg.CORBA.StructMember[] members)  
    throws org.omg.CORBA.SystemException;  
public org.omg.CORBA.TypeCode create_interface_tc(String id,  
    String name)  
    throws org.omg.CORBA.SystemException;  
public org.omg.CORBA.TypeCode create_recursive_sequence_tc(  
    int bound,  
    int offset)  
    throws org.omg.CORBA.SystemException;  
public org.omg.CORBA.TypeCode create_sequence_tc(int bound,  
    int offset)  
    throws org.omg.CORBA.SystemException;  
public org.omg.CORBA.TypeCode create_string_tc(int bound)  
    throws org.omg.CORBA.SystemException;
```

```

public org.omg.CORBA.TypeCode create_struct_tc(
    String id,
    String name,
    org.omg.CORBA.StructMember[] members)
    throws org.omg.CORBA.SystemException;
public org.omg.CORBA.TypeCode create_union_tc(
    String id,
    String name,
    org.omg.CORBA.TypeCode disc_type,
    org.omg.CORBA.UnionMember[] members)
    throws org.omg.CORBA.SystemException;
public org.omg.CORBA.TypeCode create_wstring_tc(int bound)
    throws org.omg.CORBA.SystemException;

```

## Description

Creates a new `TypeCode` for a specified IDL type. Refer to the Interface Repository section of the CORBA specification.

Normally the `TypeCodes` describing IDL types are generated automatically in the Helper classes or are accessible from the Interface Repository.

In some situations, such as bridges between ORBs, `TypeCodes` need to be constructed outside of any Interface Repository. You can do this using the `create_<type>_tc()` methods on the ORB pseudo-object.

Refer to [“Class org.omg.CORBA.TypeCode” on page 38](#) for details of parameters.

For example:

For an array `TypeCode` the methods available are `length()`, which returns the size of this dimension of the array, and `content_type()`, which returns the `TypeCode` for the contents of the array.

## Note:

For a multi-dimensional array, it is also possible to be an array `TypeCode`.

The parameters for `create_array_tc()` correspond to the values that represent the length of the array and the `TypeCode` of the elements contained in the array. For example, to create a `TypeCode` for a 2 dimensional array of `longs`, use the following code:

```

import IE.Iona.OrbixWeb._CORBA;
import org.omg.CORBA.TypeCode;
import org.omg.CORBA.ORB;

// Create a TypeCode for an array defined as follows in IDL.
// typedef long LongArray[4][5];

TypeCode tempTC= ORB.init().create_array_tc(5,
    _CORBA._tc_long);
TypeCode tc = ORB.init().create_array_tc(4, tempTC);

```

## Parameters

id	The <code>RepositoryId</code> String for the object described in the CORBA specification is a string of the form "IDL:<ident1>/<ident2>/.../<identn>:<version number>".  For example for the grid interface the <code>RepositoryId</code> is "IDL:grid:1.0". If the Helper classes are available, the <code>id()</code> method returns this string.
name	The name of the IDL type.

**Return Value**

TypeCode                      The TypeCode generated.

**Notes**

CORBA-defined.

**See Also**

["Class IE.Iona.OrbixWeb.CORBA.ORB" on page 120](#)  
["Class org.omg.CORBA.ORB" on page 19](#)

# Class `IE.Iona.OrbixWeb.CORBA.TypeCode`

## Synopsis

The Java class `TypeCode` implements the IDL pseudo interface `TypeCode`. `TypeCode` is used to describe arbitrary complex IDL type structures at runtime. A `TypeCode` consists of a *kind* and a sequence of *parameters* (a parameter is of type `CORBA.Any`). The kind classifies the `TypeCode`: for example, whether it is an IDL basic type, a struct, a sequence, and so on. `TypeCode` constant values are defined in the Orbix Java class `TCKind`. The parameters give the details of the type definition. For example, the IDL type `sequence<long, 20>` has the kind `TCKind.tk_sequence` and has parameters `long` and `20`.

The parameters of each `TypeCode` are:

KIND	PARAMETER LIST
<code>TCKind.tk_null</code>	NONE
<code>TCKind.tk_void</code>	NONE
<code>TCKind.tk_short</code>	NONE
<code>TCKind.tk_long</code>	NONE
<code>TCKind.tk_ushort</code>	NONE
<code>TCKind.tk_ulong</code>	NONE
<code>TCKind.tk_float</code>	NONE
<code>TCKind.tk_double</code>	NONE
<code>TCKind.tk_boolean</code>	NONE
<code>TCKind.tk_char</code>	NONE
<code>TCKind.tk_octet</code>	NONE
<code>TCKind.tk_any</code>	NONE
<code>TCKind.tk_TypeCode</code>	NONE
<code>TCKind.tk_Principal</code>	NONE
<code>TCKind.tk_objref</code>	{ interface-id }
<code>TCKind.tk_struct</code>	{ struct-name, member-name, <code>TypeCode</code> , ...<repeat pairs>... }
<code>TCKind.tk_union</code>	{ union-name, switch- <code>TypeCode</code> , label-value, member-name, <code>TypeCode</code> , ...<repeat triples>... }
<code>TCKind.tk_enum</code>	{ enum-name, enum-id, ...<repeat enum-id>... }
<code>TCKind.tk_string</code>	{ maxlen-integer }
<code>TCKind.tk_sequence</code>	{ <code>TypeCode</code> , maxlen-integer }
<code>TCKind.tk_array</code>	{ <code>TypeCode</code> , length-integer }



KIND	PARAMETER LIST
TCKind.tk_alias	{ alias-name, TypeCode }
TCKind.tk_except	{ except-name, member_name, TypeCode, ...<repeat pairs>... }
TCKind.tk_longlong	NONE
TCKind.tk_ulonglong	NONE
TCKind.tk_longdouble	NONE
TCKind.tk_wchar	NONE
TCKind.tk_wstring	{ maxlen_integer }
TCKind.tk_fixed	{ digits, scale }

A TypeCode of kind TCKind.tk\_objref has a single parameter giving the interface name.

A TypeCode of kind TCKind.tk\_struct has one parameter giving the struct name, and has two parameters for each member of the struct: the first giving the member's name and the second giving its TypeCode. A struct with N members has 2N+1 parameters.

A TypeCode of kind TCKind.tk\_union has parameters giving the union name, the TypeCode of the switch (discriminator) of the union, and three parameters for each member of the union: the first giving the label value, the second giving the member name, and the third giving the member's TypeCode. If the union has a default member, the triple for this has a label-value of 0 and the TypeCode of the corresponding any returned by member\_label() is the TypeCode for an octet, which is not a valid switch type for a union. Thus this 0 can be distinguished from a normal 0 switch value.

A TypeCode of kind TCKind.tk\_enum has one parameter giving the enum name, and one parameter for each enumerate constant. Enumerate constants are represented as strings.

A TypeCode of kind TCKind.tk\_string has one parameter—an integer giving the maximum length of the string. A 0 length indicates an unbounded string.

A TypeCode of kind TCKind.tk\_sequence has two parameters: a TypeCode for the element types, and a long for the length. A 0 length indicates an unbounded sequence.

A TypeCode of kind TCKind.tk\_array has two parameters. For single-dimensional arrays the first parameter is the TypeCode for the array and the second parameter is the length of the array. For multi-dimensional arrays a recursive approach is taken. The TypeCode is of kind TCKind.tk\_array and the second parameter is the length of the first dimension of the array. The first TypeCode parameter can then be queried for its contents (either an array of basic or constructed types or another array of arrays). This process is repeated until the TypeCode parameter is a different type than TCKind.tk\_array.

A TypeCode of kind TCKind.tk\_alias has two parameters, the first specifying the name of the alias and the second giving the TypeCode of the type being aliased.

A `TypeCode` of kind `TCKind.tk_except` has one parameter giving the exception name, and has two parameters for each member of the exception: the first giving the member's name and the second giving its `TypeCode`. Exceptions with no members are allowed.

A `TypeCode` of kind `TCKind.tk_fixed` has two parameters, the first giving the precision of the fixed-point number in decimal digits and the second giving the position of the decimal point (scale).

An IDL operation with a parameter of type `TypeCode` translates into a Java method with a parameter of type `TypeCode`.

A `TypeCode` object reference constant declaration can be generated by the IDL compiler from named type definitions that appear in an IDL file—that is, from the following types:

```
interface
typedef
struct
union
enum
alias
except
```

A number of `TypeCode` object reference constants are always available to allow the user to access `TypeCodes` for standard types. These are defined in the `CORBA` class.

```
CORBA._tc_null          CORBA._tc_void
CORBA._tc_short        CORBA._tc_long
CORBA._tc_ushort      CORBA._tc_ulong
CORBA._tc_float        CORBA._tc_double
CORBA._tc_boolean     CORBA._tc_char
CORBA._tc_wchar       CORBA._tc_wstring
CORBA._tc_longlong    CORBA._tc_ulonglong
CORBA._tc_octet       CORBA._tc_any
CORBA._tc_TypeCode    CORBA._tc_Principal
CORBA._tc_Object      CORBA._tc_string
CORBA._tc_NamedValue
```

Using `ORB.get_primitive_tc` gives similar results.

## CORBA

```
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 };
```

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

    // for all TypeCode kinds
    boolean equal(in TypeCode tc);
    TCKind kind();
```

```

// for objref, struct, union, enum, alias, and except
RepositoryID id() raises (BadKind);
RepositoryID name() raises (BadKind);

// for struct, union, enum, and except
unsigned long member_count() raises (BadKind);
Identifier member_name(in unsigned long index) raises
    (BadKind, Bounds);

// for struct, union, and except
TypeCode member_type(in unsigned long index) raises
    (BadKind, Bounds);

// for union
any member_label(in unsigned long index) raises (BadKind,
    Bounds);
TypeCode discriminator_type() raises (BadKind);
long default_index() raises (BadKind);

// for string, sequence, and array
unsigned long length() raises (BadKind);
TypeCode content_type() raises (BadKind);
}

```

## Orbix Java

```

public class TypeCode extends org.omg.CORBA.TypeCode {

    // Constructors
    public TypeCode(TCKind tcKind, String id, String name,
        java.lang.Object[] members, int length,
        org.omg.CORBA.TypeCode content_type)
        throws org.omg.CORBA.SystemException;
    public TypeCode(TCKind tcKind)
        throws org.omg.CORBA.SystemException;

    // Get the TCKind code
    public TCKind kind() throws org.omg.CORBA.SystemException;

    public boolean equals(java.lang.Object _obj)
        throws org.omg.CORBA.SystemException;

    // OMG equality operation
    public boolean equal(org.omg.CORBA.TypeCode _obj)
        throws org.omg.CORBA.SystemException;

    // Compare two TypeCodes
    public static boolean compare(org.omg.CORBA.TypeCode tc1,
        org.omg.CORBA.TypeCode tc2)
        throws org.omg.CORBA.SystemException;

    // Repository id for TypeCode
    public String id()
        throws BadKind, org.omg.CORBA.SystemException;
    public String name()
        throws BadKind, org.omg.CORBA.SystemException;

    // Number of members in struct, union etc.
    public int member_count()
        throws BadKind, org.omg.CORBA.SystemException;
}

```

```

// struct, union etc. member type
public org.omg.CORBA.TypeCode member_type(int index)
    throws BadKind, Bounds, org.omg.CORBA.SystemException;

// struct, union etc. member name
public String member_name(int index)
    throws BadKind, Bounds, org.omg.CORBA.SystemException;
public org.omg.CORBA.Any member_label(int index)
    throws BadKind, Bounds, org.omg.CORBA.SystemException;

public org.omg.CORBA.TypeCode discriminator_type()
    throws BadKind, org.omg.CORBA.SystemException;

public int default_index()
    throws BadKind, org.omg.CORBA.SystemException;
public int length()
    throws BadKind, org.omg.CORBA.SystemException;

// Get contents typecode
public org.omg.CORBA.TypeCode content_type()
    throws BadKind, org.omg.CORBA.SystemException;

public OrbixTypeCode orbixTypeCode()
    throws org.omg.CORBA.SystemException;

public String toString()
    throws org.omg.CORBA.SystemException;
};

```

## Notes

CORBA-defined.

## See Also

["Class org.omg.CORBA.TCKind" on page 36](#)  
["Class org.omg.CORBA.TypeCodePackage.BadKind" on page 40](#)  
["Class org.omg.CORBA.TypeCodePackage.Bounds" on page 41](#)

## TypeCode()

### Synopsis

```

public TypeCode(TCKind tcKind, String id, String name,
    java.lang.Object[] members, int length,
    org.omg.CORBA.TypeCode content_type)
    throws org.omg.CORBA.SystemException;

```

### Description

Construct a `TypeCode` from structural type information. Typecodes are created using ORB operations by the user.

### Parameters

<code>tcKind</code>	The "kind" of the <code>TypeCode</code> . Can use the object reference constants to represent the standard types; for example, <code>CORBA._tc_char</code>
<code>id</code>	Repository ID for a type in the Interface Repository.
<code>name</code>	IDL name.
<code>members</code>	Members of complex types; for example, union, struct, except, and so on.
<code>length</code>	Number of elements for arrays and sequences, length of a string
<code>content_type</code>	Base type for elements of sequences and arrays

**Notes** Orbix-specific.

**See Also** Other `TypeCode` constructors  
["Class IE.Iona.OrbixWeb.CORBA.ORB" on page 120](#)

## TypeCode()

**Synopsis**

```
public TypeCode(TCKind tcKind)
    throws org.omg.CORBA.SystemException;
```

**Description** Construct a `TypeCode` from a given typecode kind. Typecodes are created by the user, using ORB operations.

### Parameters

`tcKind` Create a `TypeCode` of type `tcKind`; for example, `CORBA._tc_Object` for a `TypeCode` which represents an `Object`.

**Notes** Orbix-specific.

**See Also** Other `TypeCode` constructors  
["Class IE.Iona.OrbixWeb.CORBA.ORB" on page 120](#)

## kind()

**Synopsis**

```
public TCKind kind() throws org.omg.CORBA.SystemException;
```

**Description** Return the kind of `TypeCode` as defined in ["Class org.omg.CORBA.TCKind" on page 36](#). Typecodes are created by the user, using ORB operations.

**Notes** CORBA-defined.

**See Also** Other `TypeCode` constructors  
["Class IE.Iona.OrbixWeb.CORBA.ORB" on page 120](#)

## equals()

**Synopsis**

```
public boolean equals(java.lang.Object _obj)
    throws org.omg.CORBA.SystemException;
```

**Description** Compares the current `TypeCode` object with `_obj`. Two `TypeCodes` are equal when the IDL definitions from which they are compiled denote equal types.

### Parameters

`obj` Must be an instance of `org.omg.CORBA.TypeCode`

### Return Value

`boolean` Returns `true` if the `TypeCodes` are equal; returns `false` if they are unequal, or if `_obj` is not a `TypeCode` object.

**Notes** Orbix-specific.

**See Also** ["equal\(\)" on page 196](#)  
["compare\(\)" on page 196](#)

## equal()

### Synopsis

```
public boolean equal(org.omg.CORBA.TypeCode _obj)
throws org.omg.CORBA.SystemException;
```

### Description

OMG-defined method for checking equality of `TypeCodes`.

### Parameters

`obj`            Check equality of current `TypeCode` against `obj`.

### Return Value

`boolean`       Returns `true` if the `TypeCodes` are equal; returns `false` if they are unequal.

### Notes

CORBA-defined.

### See Also

["equal\(\)" on page 196](#)  
["compare\(\)" on page 196](#)

## compare()

### Synopsis

```
public static boolean compare(org.omg.CORBA.TypeCode tc1,
                             org.omg.CORBA.TypeCode tc2)
throws org.omg.CORBA.SystemException;
```

### Description

Returns `true` if the `TypeCode` specified in parameter `tc1` holds the same value as the `TypeCode` specified in `tc2`.

### Parameters

`tc1`            First `TypeCode` instance to be compared.  
`tc2`            Second `TypeCode` to be compared.

### Return Value

`boolean`       Returns `true` if the `TypeCodes` are equal; returns `false` if they are unequal.

### Notes

Orbix-specific.

### See Also

["equal\(\)" on page 196](#)  
["compare\(\)" on page 196](#)

## id()

### Synopsis

```
public String id()
    throws BadKind, org.omg.CORBA.SystemException;
```

### Description

Returns the Interface Repository identifier for the `TypeCode`.

### Return Value

String      Interface Repository identifier.

### Notes

CORBA-defined.

## name()

### Synopsis

```
public String name()
    throws BadKind, org.omg.CORBA.SystemException;
```

### Description

Returns an identifying name for object references, structs, enums, unions, aliases and exceptions.

### Return Value

String      Name of `TypeCode` as specified in IDL.

### Notes

CORBA-defined.

## member\_count()

### Synopsis

```
public int member_count()
    throws BadKind, org.omg.CORBA.SystemException;
```

### Description

Returns the number of constituent members for `struct`, `union`, `enum` and `except` `TypeCodes`. Throws `BadKind` exception if called on a `TypeCode` other than these.

### Return Value

int          Number of members in `struct`, `union`, `except` or `enum`.

### Notes

CORBA-defined.

## member\_type()

### Synopsis

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

### Description

Returns the `TypeCode` describing the member identified by `index`. Valid only for `struct` and `union` and `except` `TypeCodes`. Calling this on any other type results in a `BadKind` exception.

### Parameters

index      Identifies the member within the `struct` or `union` for which the `TypeCode` is to be retrieved.

### Return Value

TypeCode    `TypeCode` that describes the member at the specified position within the `struct/union`.

### Notes

CORBA-defined.

## member\_name()

### Synopsis

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

### Description

Returns the name of the member identified by `index` in a struct, union, except or enum.

### Parameters

`index` Identifies the member within struct/union or enum.

### Return Value

String Name of indexed member.

### Notes

CORBA-defined.

## member\_label()

### Synopsis

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

### Description

Returns the label of the union member identified by `index`.

### Parameters

`index` Identifies member in the union. A `Bounds` exception is thrown if `index` does not correspond to a member.

### Return Value

Any The type of the label value is designated by the union discriminator `TypeCode`. The default member is identified with the 0 octet. The Any can be used to distinguish between the default member label and labels for the legal switch types.

### Notes

CORBA-defined.

## discriminator\_type()

### Synopsis

```
public org.omg.CORBA.TypeCode discriminator_type()
    throws BadKind, org.omg.CORBA.SystemException;
```

### Description

Returns the type of all non-default member labels in a union.

### Return Value

TypeCode The type of the discriminator in a union.

### Notes

CORBA-defined.



## default\_index()

### Synopsis

```
public int default_index()  
    throws BadKind, org.omg.CORBA.SystemException;
```

### Description

Returns the index of the default member of a union or -1 if no default member has been defined. Raises a `BadKind` exception if called on anything other than a union.

### Return Value

int            Index of the default union member or -1 if not defined.

### Notes

CORBA-defined.

## length()

### Synopsis

```
public int length()  
    throws BadKind, org.omg.CORBA.SystemException;
```

### Description

Can be invoked on `string`, `wstring`, `array` and `sequence` `TypeCodes`. For `string`, `wstring` and `sequence`, it returns the bound or 0 for an unbounded `string`, `wstring` or `sequence`. For arrays it returns the number of elements in the array. Invoking on all other types causes the `BadKind` exception to be thrown.

### Return Value

int            For strings, wstrings and sequences returns the bounds or 0 if unbounded. For arrays returns number of elements in the array.

### Notes

CORBA-defined.

## content\_type()

### Synopsis

```
public org.omg.CORBA.TypeCode content_type()  
    throws BadKind, org.omg.CORBA.SystemException;
```

### Description

Can be invoked on `sequence`, `array` and `alias` `TypeCodes`. All others result in `BadKind` being thrown. For `sequence` and `array` it returns the type of the elements, for `alias` it returns the original type being aliased.

### Return Value

TypeCode            Type of elements for `sequence` and `array` `TypeCodes`, original type for `alias` `TypeCodes`.

### Notes

CORBA-defined.

## orbixTypeCode()

### Synopsis

```
public OrbixTypeCode orbixTypeCode()  
    throws org.omg.CORBA.SystemException;
```

### Description

Return Orbix-specific type for `TypeCode` identification.

### Return Value

`OrbixTypeCode`                      Orbix-specific identification for `TypeCode`.

### Notes

Orbix-specific.

## toString()

### Synopsis

```
public java.lang.String toString()  
    throws org.omg.CORBA.SystemException;
```

### Description

Return a stringified version of the `TypeCode` in the following format:  
"TypeCode : kind = <kind> id = <id> name = <name>".

### Return Value

`String`                                      A stringified version of the `TypeCode`

### Notes

Orbix-specific.

# Part III

## Package IE.Iona.OrbixWeb.Feature

### In this part

This part contains the following:

Class IE.Iona.OrbixWeb.Features.AuthenticationFilter	page 203
Class IE.Iona.OrbixWeb.Features.Config	page 205
Class IE.Iona.OrbixWeb.Features.Filter	page 207
Interface IE.Iona.OrbixWeb.Features.ioCallback	page 217
Class IE.Iona.OrbixWeb.Features.IT_reqTransformer	page 219
Class IE.Iona.OrbixWeb.Features.LoaderClass	page 222
Class IE.Iona.OrbixWeb.Features.locatorClass	page 227
Class IE.Iona.OrbixWeb.Features.OrbConfig	page 229
Class IE.Iona.OrbixWeb.Features.ProxyFactory	page 232
Class IE.Iona.OrbixWeb.Features.ThreadFilter	page 234



# Class

## IE.Iona.OrbixWeb.Features.AuthenticationFilter

### Synopsis

`AuthenticationFilter` is a derived class of class `Filter`. It is used to pass authentication information between processes.

There are two important differences between an `AuthenticationFilter` and a normal `Filter`.

First, there can only ever be one instance of the `AuthenticationFilter`.

Second, this instance of the `AuthenticationFilter` is placed in the `Filter` chain before normal filters, but after `ThreadFilter` instances. This allows you to authenticate the request before passing the request to any application code.

The usual way to use an `AuthenticationFilter` is to declare a derived class of `AuthenticationFilter` and then provide implementations of the filter points `outRequestPreMarshal()` and `inRequestPreMarshal()`. These implementations add or remove authentication information respectively. You then create an instance of the derived class in both the client and the server.

### Orbix Java

```
// Java

package IE.Iona.OrbixWeb.Features;

public class AuthenticationFilter extends
    IE.Iona.OrbixWeb.Features.Filter {
    protected AuthenticationFilter()
        throws SystemException;
    protected AuthenticationFilter(org.omg.CORBA.ORB orb)
        throws SystemException;
}
```

### Notes

Orbix-specific.

### See Also

["Class IE.Iona.OrbixWeb.Features.Filter" on page 207](#)

## AuthenticationFilter()

**Synopsis** `protected AuthenticationFilter();`

**Description** You cannot create direct instances of `AuthenticationFilter`. The constructor is protected to enforce this.

**Notes** Orbix-specific.

## AuthenticationFilter()

**Synopsis** `protected AuthenticationFilter(org.omg.CORBA.ORB orb);`

**Description** Direct instances of `AuthenticationFilter` cannot be created. The constructor is protected to enforce this.

The `orb` parameter provides support for multiple ORBs. This allows the newly- created `AuthenticationFilter` to be associated with a specific ORB instance.

**Notes** Orbix-specific.

# Class IE.Iona.OrbixWeb.Features.Config

## Synopsis

Class `Config` provides methods for accessing and setting Orbix Java configuration variables using Java code.

## Note:

The `Config` class is deprecated because configuration is now on a per-ORB basis. To get and set configuration variables using Orbix Java APIs, you should use the methods provided by class `IE.Iona.OrbixWeb.Features.OrbConfig`.

## Orbix Java

```
// Java

package IE.Iona.OrbixWeb.Features;

public class Config {

    // Constructor
    public Config();

    // Methods
    public static String getConfigItem(String);

    public static synchronized void setConfigItem(String,
        String);
}
```

## Notes

Orbix-specific.

## See Also

["Class IE.Iona.OrbixWeb.Features.OrbConfig" on page 229](#)

## Config()

### Synopsis

```
public Config();
```

### Description

The default constructor.

### Notes

Orbix-specific.

## getConfigItem()

### Synopsis

```
public String getConfigItem(String name)
```

### Description

Get a configuration item in `String` form. Refer to the ***Orbix Administrator's Guide Java Edition*** for a complete list of Orbix Java configuration items.

### Parameters

name	The configuration item name in <code>String</code> form; for example: <code>getConfigItem("IT_DEFAULT_TIMEOUT")</code>
------	---------------------------------------------------------------------------------------------------------------------------

## Notes

Orbix-specific.

## setConfigItem()

### Synopsis

```
public synchronized void setConfigItem(String name, String  
value)
```

### Description

Sets a configuration item in `String` form. Refer to the ***Orbix Administrator's Guide Java Edition*** for a complete list of Orbix Java configuration items.

### Parameters

<code>name</code>	The configuration item name in <code>String</code> form.
<code>value</code>	The configuration item value in <code>String</code> form; for example: <pre>setConfigItem("IT_DEFAULT_TIMEOUT", "80000")</pre>

### Notes

Orbix-specific.



# Class IE.Iona.OrbixWeb.Features.Filter

## Synopsis

Class `Filter` is a conceptually abstract class describing the interface to a per-process filter.

If you wish to implement a per-process filter you may define a derived class of `Filter` and redefine some or all of the eight monitoring method and two failure points as described in the *Orbix Programmer's Guide Java Edition*.

For a list of the available `SystemExceptions` that can be raised see the ["System Exceptions" on page 331](#).

## Orbix Java

```
// Java

package IE.Iona.OrbixWeb.Features;

public class Filter {
    // Constructor
    protected Filter();
    protected Filter(boolean installme);
    protected Filter(org.omg.CORBA.ORB orb, boolean installme)

    // Methods
    public void _delete();

    // Client-side filter points
    public boolean outRequestPreMarshal(Request r)
        throws org.omg.CORBA.SystemException;
    public boolean outRequestPostMarshal(Request r)
        throws org.omg.CORBA.SystemException;
    public boolean inReplyPreMarshal(Request r)
        throws org.omg.CORBA.SystemException;
    public boolean inReplyPostMarshal(Request r)
        throws org.omg.CORBA.SystemException;

    // Server-side filter points
    public boolean inRequestPreMarshal(Request r)
        throws org.omg.CORBA.SystemException;
    public boolean inRequestPostMarshal(Request r)
        throws org.omg.CORBA.SystemException;
    public boolean outReplyPreMarshal(Request r)
        throws org.omg.CORBA.SystemException;
    public boolean outReplyPostMarshal(Request r)
        throws org.omg.CORBA.SystemException;

    // Failure filter points
    public void outReplyFailure(Request r, Exception ex)
        throws org.omg.CORBA.SystemException;
    public void inReplyFailure(Request r, Exception ex)
        throws org.omg.CORBA.SystemException;
}
```

## Note

Orbix-specific.

## See Also

["Class org.omg.CORBA.Request" on page 29](#)  
["Class org.omg.CORBA.SystemException" on page 35](#)  
["Class org.omg.CORBA.UserException" on page 43](#)  
["Class IE.Iona.OrbixWeb.Features.AuthenticationFilter" on page 203](#)  
["Class IE.Iona.OrbixWeb.Features.Filter" on page 207](#)

## Filter()

### Synopsis

```
protected Filter();
```

### Description

The default constructor adds the newly-created filter object into the per-process filter chain. Direct instances of `Filter` cannot be created, and the constructor is protected to enforce this. The derived classes of `Filter` normally have public constructors.

### Notes

Orbix-specific.

## Filter()

### Synopsis

```
protected Filter(boolean installMe);
```

### Description

If the value of `installMe` is `true`, this constructor adds the newly-created filter object into the per-process filter chain.

If the value of `installMe` is `false`, the newly created filter object is not added to the per-process filter chain. This flexibility is useful for implementing master-slave architectures.

Direct instances of `Filter` cannot be created; the constructor is protected to enforce this. The derived classes of `Filter` may have public constructors overriding this constructor.

### Notes

Orbix-specific.

## Filter()

### Synopsis

```
protected Filter(org.omg.CORBA.ORB orb, boolean installMe);
```

### Description

Direct instances of `Filter` cannot be create; the constructor is protected to enforce this. The derived classes of `Filter` may have public constructors that overriding this constructor.

### Parameters

<code>orb</code>	The <code>orb</code> parameter provides support for multiple ORBs. This allows the newly-created filter to be associated with a specific ORB instance.
<code>installMe</code>	If the value of <code>installMe</code> is <code>true</code> , this constructor adds the newly-created filter object into the per-process filter chain. If the value of <code>installMe</code> is <code>false</code> , the newly created filter object is not added to the per-process filter chain. This flexibility is useful for implementing master-slave architectures.

### Notes

Orbix-specific.

## **\_delete()**

### **Synopsis**

```
public void _delete();
```

### **Description**

Removes any references to the `Filter` object from the Orbix Java runtime by removing the object from the process filter chain. You must call this method when your client or server is about to exit or when you have finished using the `Filter` object in question so that it can be garbage collected.

### **Notes**

Orbix-specific.

### **See Also**

["finalize\(\)" on page 69](#)

["finalize\(\)" on page 136](#)

["unregisterIOCallback\(\)" on page 158](#)

## **inReplyFailure()**

### **Synopsis**

```
public void inReplyFailure(Request r, java.lang.Exception ex);
```

### **Description**

Defines the action to perform if the target object raises an exception or if any preceding marshalling filter point ('out request', 'in request', 'out reply' or 'in reply') raises an exception or uses its return value to indicate that the call should not be processed any further.

If not redefined in a derived class, this filter point carries out no actions.

### **Parameters**

<code>r</code>	This is the <code>Request</code> object for the current invocation.
<code>ex</code>	This is the exception object that has just been thrown by the implementation object, by a preceding filter point, or by the ORB runtime.

### **Notes**

Orbix-specific.

## **inReplyPostMarshal()**

### **Synopsis**

```
public boolean inReplyPostMarshal(Request r);
```

### **Description**

Defines the action to perform after any operation on any object in another address space. In particular, after the operation response has arrived at the caller's address space, and after the operation's return parameters and return value have been removed from the reply packet.

If not redefined in a derived class, the following implementation is inherited:

```
// Java
{ return true; } // Continue the call.
```

### **Parameters**

<code>r</code>	This is the <code>Request</code> for the current invocation.
----------------	--------------------------------------------------------------

## Return Value

<code>true</code>	Continue with the request as normal. The reply is sent to the next filter on the chain, or, if this is the last filter, it is sent to the calling object.
<code>false</code>	Do not continue with the call. Raise a <code>FILTER_SUPPRESS</code> exception and do not run the remaining filters.

## Exceptions

If you redefine this method in a derived class, you can raise a `SystemException` to indicate that the call is not to be continued. You can also return `false`. If you do this, Orbix Java raises a `FILTER_SUPPRESS` exception and the call is not continued.

## Notes

Orbix-specific.

## `inReplyPreMarshal()`

### Synopsis

```
public boolean inReplyPreMarshal(Request r)
```

### Description

Defines the action to perform after any operation on any object in another address space. Specifically, after the operation response has arrived at the caller's address space, and before the operation's return parameters and return value have are removed from the reply packet.

If not redefined in a derived class, the following implementation is inherited:

```
// Java
{ return true; } // Continue the call.
```

### Parameters

`r` This is the `Request` object for the current invocation.

## Return Value

<code>true</code>	Continue with the request as normal. The reply is sent to the next filter on the chain, or, if this is the last filter, it is sent to the calling object.
<code>false</code>	Do not continue with the call. Raise a <code>FILTER_SUPPRESS</code> exception and do not run the remaining filters.

## Exceptions

If you redefine this method in a derived class, you can raise a `SystemException` to indicate that the call is not to be continued. You can also return `false`. If you do this, Orbix Java raises a `FILTER_SUPPRESS` exception and the call is not continued.

## Notes

Orbix-specific.

## inRequestPostMarshal()

### Synopsis

```
public boolean inRequestPostMarshal(Request r);
```

### Description

Defines the action to perform before any incoming operation on any object in the address space. Specifically, before the operation has been sent to the target object, and after the operation's parameters have been removed from the request packet.

If not redefined in a derived class, the following implementation is inherited:

```
// Java
{ return true; } // Continue the call.
```

### Parameters

**r** This is the `Request` object for the current invocation.

### Return Value

**true** Continue with the request as normal. The operation call is sent to the next filter on the chain, or, if this is the last filter, it is sent to the per-object filters, if any, and then to the target object.

**false** Do not continue with the call. Raise a `FILTER_SUPPRESS` exception; do not run the remaining filters and do not pass the call onto the target object.

### Exceptions

If you redefine this method in a derived class, you may raise a `SystemException` to indicate that the call is not to be continued.

You can also return `false`. If you do this, Orbix Java raises a `FILTER_SUPPRESS` exception and the call is not continued.

### Notes

Orbix-specific.

## inRequestPreMarshal()

### Synopsis

```
public boolean inRequestPreMarshal(Request r);
```

### Description

Defines the action to perform before incoming requests: before any incoming operation on any object in the address space. Specifically, before the operation has been sent to the target object, and before the operation's parameters have been removed from the request packet.

If not redefined in a derived class, the following implementation is inherited:

```
// Java
{ return true; } // Continue the call.
```

### Parameters

**r** This is the `Request` object for the current invocation.

## Return Value

<code>true</code>	Continue with the request as normal. The operation call is sent to the next filter on the chain, or, if this is the last filter, it is sent to the per-object filters, if any, and then to the target object.
<code>false</code>	Do not continue with the call. Raise a <code>FILTER_SUPPRESS</code> exception; do not run the remaining filters and do not pass the call onto the target object.

## Exceptions

If you redefine this method in a derived class, you can raise a `SystemException` to indicate that the call is not to be continued. You can also return `false`. If you do this, Orbix Java raises a `FILTER_SUPPRESS` exception and the call is not continued.

## Notes

Orbix-specific.

## See Also

["Class IE.Iona.OrbixWeb.Features.ThreadFilter" on page 234](#)  
["inRequestPreMarshal\(\)" on page 211](#)

## outReplyFailure()

### Synopsis

```
public void outReplyFailure(Request r, java.lang.Exception ex);
```

### Description

Defines the action to perform if the target object raises an exception, or if any preceding marshalling filter point on the server's side ('in request' or 'out reply') raises an exception or uses its return value to indicate that the call should not be processed any further.

If not redefined in a derived class, this filter point performs no actions.

### Parameters

<code>r</code>	This is the <code>Request</code> object for the current invocation.
<code>ex</code>	This is the exception object that has just been thrown by the implementation object, by a preceding filter point or by the ORB runtime.

## Notes

Orbix-specific.

## outReplyPostMarshal()

### Synopsis

```
public boolean outReplyPostMarshal(Request r);
```

### Description

Defines the action to perform before outgoing replies: after any incoming operation on any CORBA object in the address space. Specifically, after the operation call is processed, and after the operation's return parameters and return value are added to the reply packet.

If not redefined in a derived class, the following implementation is inherited:

```
// Java
{ return true; } // Continue the call.
```

### Parameters

**r** This is the `Request` object for the current invocation.

### Return Value

**true** Continue with the request as normal. The reply is sent to the next filter on the chain, or, if this is the last filter, it is sent to the calling object's address space (where it is handled by 'in reply' filters).

**false** Do not continue with the call. Raise a `FILTER_SUPPRESS` exception and do not run the remaining filters.

### Exceptions

If you redefine this method in a derived class, you can raise a `SystemException` to indicate that the call is not to be continued.

You can also return `false`. If you do this, Orbix Java raises a `FILTER_SUPPRESS` exception and the call is not continued.

### Notes

Orbix-specific.

## outReplyPreMarshal()

### Synopsis

```
public boolean outReplyPreMarshal(Request r);
```

### Description

Defines the action to perform before outgoing replies: after any incoming operation on any CORBA object in the address space. Specifically, after the operation call has been processed, and before the operation's return parameters and return value have been added to the reply packet.

If not redefined in a derived class, the following implementation is inherited:

```
// Java
{ return true; } // Continue the call.
```

### Parameters

`r` This is the `Request` object for the current invocation.

### Return Value

`true` Continue with the request as normal. The reply is sent to the next filter on the chain, or, if this is the last filter, it is sent to the calling object's address space (where it is handled by 'in reply' filters).

`false` Do not continue with the call. Raise a `FILTER_SUPPRESS` exception and do not run the remaining filters.

### Exceptions

If you redefine this method in a derived class, you may raise a `SystemException` to indicate that the call is not to be continued.

You can also return `false`. If you do this, Orbix Java raises a `FILTER_SUPPRESS` exception and the call is not continued.

### Notes

Orbix-specific.



## outRequestPostMarshal()

### Synopsis

```
public boolean outRequestPostMarshal(Request r);
```

### Description

Defines the action to perform before outgoing requests; before any operation from this address space to any object in another address space. Specifically, before the invocation has been transmitted and after the operation's parameters have been added to the request packet.

If not redefined in a derived class, the following implementation is inherited:

```
// Java
{ return true; } // Continue the call.
```

### Parameters

**r** This is the `Request` object for the current invocation.

### Return Value

**true** Continue with the request as normal. The operation call is sent to the next filter on the chain, or, if this is the last filter it is transmitted to the address space of the target object (where it is first handled by any per-process filters and then per-object filters).

**false** Do not continue with the call. Raise a `FILTER_SUPPRESS` exception. Do not run the remaining filters and do not transmit the operation call out of this address space.

### Exceptions

If you redefine this method in a derived class, you can raise a `SystemException` to indicate that the call is not to be continued.

You can also return `false`. If you do this, Orbix Java raises a `FILTER_SUPPRESS` exception and the call is not continued.

### Notes

Orbix-specific.

## outRequestPreMarshal()

### Synopsis

```
public boolean outRequestPreMarshal(Request r);
```

### Description

Defines the action to perform before outgoing requests; before any operation from this address space to any object in another address space. In particular, before the invocation has been transmitted and after the operation's parameters have been added to the request packet.

If not redefined in a derived class, the following implementation is inherited:

```
// Java
{ return true; } // Continue the call.
```

### Parameters

`r` This is the `Request` object for the current invocation.

### Return Value

`true` Continue with the request as normal. The operation call is sent to the next filter on the chain, or, if this is the last filter, it is transmitted to the address space of the target object (where it is first handled by any per-process filters and then per-object filters).

`false` Do not continue with the call. Raise a `FILTER_SUPPRESS` exception. Do not run the remaining filters and do not transmit the operation call out of this address space.

### Exceptions

If you redefine this method in a derived class, you can raise a `SystemException` to indicate that the call is not to be continued.

You can also return `false`. If you do this, Orbix Java raises a `FILTER_SUPPRESS` exception and the call is not continued.

### Notes

Orbix-specific.

# Interface IE.Iona.OrbixWeb.Features.ioCallback

## Synopsis

An application may wish to be informed when a new connection is established, or when an existing connection is closed. A connection is opened when a client first communicates with the server; it is closed when the client terminates or the communications layer reports a break in service between the server and the client.

To implement an input/output callback you must write an application `ioCallback` class implementing the `IE.Iona.OrbixWeb.Features.ioCallback` Java interface.

You then pass an instance of the application callback object to the ORB object using:

```
IE.Iona.OrbixWeb.CORBA.ORB.registerIOCallback  
(IE.Iona.OrbixWeb.CORBA.ioCallback)
```

for example:

```
_CORBA.Orbix.registerIOCallback(new myioCallbackObj());
```

To unregister a callback object use

```
IE.Iona.OrbixWeb.CORBA.ORB.unregisterIOCallback();
```

`OpenCallBack()` is called whenever a connection is created, and `CloseCallBack()` is called when a connection is closed.

Your code should not depend on the calls being made in a particular thread because the thread in which these calls are made is variable. For example, if you are binding to a server, the `OpenCallBack()` is made in the thread which is making the bind. If a connection is abruptly closed, the `CloseCallBack()` is made in the reader thread. If a client thread calls `closeConnection()`, `CloseCallBack()` is called in that thread.

You can register only one `ioCallback` object at a time. When an object is registered it replaces the object that is currently registered there.

You should unregister the `ioCallback` object when you are finished with it to ensure that your objects are swiftly garbage collected.

By default there is no `ioCallback` object registered.

## Orbix Java

```
public interface ioCallback {  
    public void OpenCallBack(String serverName, Object conn)  
        throws SystemException;  
    public void CloseCallBack(String serverName, Object conn)  
        throws SystemException;  
}
```

## Notes

Orbix-specific.

## See Also

["registerIOCallback\(\)" on page 150](#)

["unregisterIOCallback\(\)" on page 158](#)

## CloseCallback()

### Synopsis

```
public void CloseCallback(String serverName, Object conn);
```

### Description

This method will get called when any connection to the process in which your `ioCallback` is registered is closed or broken.

### Parameters

<code>serverName</code>	The name of the server, to which your connection has just been broken. If you were connected to a client process from the server, this contains a unique number (a port or process ID) to identify the client that you were communicating with.
<code>conn</code>	This is the socket connection object for the connection that has just been broken. It is either of type <code>java.net.Socket</code> or <code>java.net.URLConnection</code> depending on the kind of connection that existed.

### Notes

Orbix-specific.

### See Also

["OpenCallback\(\)" on page 218](#)

## OpenCallback()

### Synopsis

```
public void OpenCallback(String serverName, Object conn);
```

### Description

This method is called when a connection is established between the process in which your `ioCallback` is registered and some other process.

### Parameters

<code>serverName</code>	The name of the server, with which a connection has just been established. If you are connecting with a client process from the server, this contains a unique number (a port or process ID) to identify the client that was communicating with you.
<code>conn</code>	This is the socket connection object for the connection that has just been established.  It is either of type <code>java.net.Socket</code> or <code>java.net.URLConnection</code> depending on the kind of connection that existed.

### Notes

Orbix-specific.

### See Also

["CloseCallback\(\)" on page 218](#)

# Class

## IE.Iona.OrbixWeb.Features.IT\_reqTransformer

### Synopsis

Class `IE.Iona.OrbixWeb.Features.IT_reqTransformer` is a conceptually abstract class describing the interface for transformer objects that allow an `org.omg.CORBA.Request` data buffer to be modified before an operation invocation is transmitted to a server, and before a reply is returned to a client.

If you wish to implement specific `Request` transformation behaviour you can define a derived class of `IE.Iona.OrbixWeb.Features.IT_reqTransformer` and redefine the `transform` and `transform_error` methods as described in the *Orbix Programmer's Guide Java Edition*.

Request transformers are registered with the Orbix Java runtime using ["setMyReqTransformer\(\)" on page 154](#) or ["setReqTransformer\(\)" on page 155](#).

### Orbix Java

```
// Java

package IE.Iona.OrbixWeb.Features;

import IE.Iona.OrbixWeb.CORBA.octetSeqHolder;

public class IT_reqTransformer {
    public boolean transform(octetSeqHolder data,
                           String host,
                           boolean is_send,
                           org.omg.CORBA.Request req)

        public String transform_error()
}

```

### See Also

["setMyReqTransformer\(\)" on page 154](#)  
["setReqTransformer\(\)" on page 155](#)  
["getMyReqTransformer\(\)" on page 139](#)

## transform()

### Synopsis

```
public boolean transform(octetSeqHolder data,  
                        String host,  
                        boolean is_send,  
                        org.omg.CORBA.Request req)
```

### Description

Defines the transformation to be performed on `org.omg.CORBA.Request`. This function is automatically invoked on the registered transformer object immediately prior to transmitting data in an `org.omg.CORBA.Request` (and after filtering) and directly subsequent (before any filtering) to receiving data in an `org.omg.CORBA.Request`.

A derived class of `IE.Iona.OrbixWeb.Features.IT_reqTransformer` should override this function to implement a transformation.

An implementation of this function may raise an `org.omg.CORBA.COMM_FAILURE` system exception to indicate that an error has occurred during the transformation. The `reason` member of the exception contains the text returned by `IE.Iona.OrbixWeb.Features.IT_reqTransformer.transform_error()`.

### Parameters

<code>data</code>	The raw binary data buffer to be transformed.
<code>host</code>	If sending a request, this is the host on which the server process resides.  If receiving a request, this is an empty string.
<code>is_send</code>	A flag identifying whether a request is being sent from an address space or received into an address space.  A value of <code>true</code> indicates that the request is being sent.  A value of <code>false</code> indicates that the request is being received.
<code>req</code>	If sending a request, this is the request object that is normally sent. You can use this <code>Request</code> object, for example, to obtain information such as what host, server, and interface the outgoing data is being sent to.  If receiving a request, this is <code>null</code> .

### Return Value

<code>true</code>	Transformation was successful.
<code>false</code>	Transformation failed. This automatically raises an <code>org.omg.CORBA.COMM_FAILURE</code> system exception, with the <code>reason</code> string set to the text provided by <code>transform_error()</code>

### Notes

Orbix-specific.

### See Also

["transform\\_error\(\)" on page 221](#)  
["Class IE.Iona.OrbixWeb.Features.Filter" on page 207](#)

## transform\_error()

### Synopsis

```
public String transform_error()
```

### Description

Returns a string describing every error that has occurred in the `transform()` function. A derived class may override this function to return a text string specific to the transformation implemented by the class. The Orbix Java runtime automatically calls this method to get the error string whenever the `transform()` method returns `false` (that is, raises a `COMM_FAILURE` exception).

### Return Value

The string returned by this function is used in the `reason` element of an `org.omg.CORBA.COMM_FAILURE` exception thrown by `IE.Iona.OrbixWeb.Features.IT_reqTransformer.transform()`.

### Notes

Orbix-specific.

### See Also

["transform\(\)" on page 220](#)

# Class IE.Iona.OrbixWeb.Features.LoaderClass

## Synopsis

When an operation invocation arrives at a process, Orbix Java searches for the target object in the object table for the process. By default, if the object is not found, Orbix Java returns an `INV_OBJREF` exception to the caller. However, by installing one or more loader objects in a process, you can choose to intervene and be informed about the failure to find the object.

A `LoaderClass` object can handle such an object fault by reconstructing the required object from a persistent store, such as a flat file, an RDBMS, or an ODBMS.

Orbix Java can then retry the invocation on the newly-created object. This occurs transparently to the caller.

Loaders are defined by defining a derived class of `IE.Iona.OrbixWeb.Features.LoaderClass` and are installed by creating an instance of the new class.

## Orbix Java

```
// Java

package IE.Iona.OrbixWeb.Features;

public class LoaderClass {
    // Constructors.
    public LoaderClass();
    public LoaderClass(boolean registerMe);
    public LoaderClass(org.omg.CORBA.ORB orb, boolean
        registerMe);

    // Methods.
    public org.omg.CORBA.Object load (String interfaceName,
        String marker, boolean local)
        throws SystemException;
    public void save (org.omg.CORBA.Object obj, int reason)
        throws SystemException;
    public void record
        (org.omg.CORBA.Object obj, StringHolder marker)
        throws SystemException;
    public boolean rename (org.omg.CORBA.Object obj,
        StringHolder marker)
        throws SystemException;
}

```

## Notes

Orbix-specific.

## LoaderClass()

### Synopsis

```
public LoaderClass();
```

### Description

The `LoaderClass` constructor has protected access. You cannot create a direct instance of class `LoaderClass`.

### Notes

Orbix-specific.

### See Also

Other class constructors.



## LoaderClass()

### Synopsis

```
public LoaderClass(boolean registerMe);
```

### Description

The `LoaderClass` constructor has protected access. You cannot create a direct instance of class `LoaderClass`.

### Parameters

`registerMe` The value of this parameter must be `true` if the `load()` method of the new loader is to be called by Orbix Java, rather than explicitly by the programmer.

### Notes

Orbix-specific.

### See Also

Other class constructors.

## LoaderClass()

### Synopsis

```
public LoaderClass(org.omg.CORBA.ORB orb, boolean registerMe);
```

### Description

The `LoaderClass` constructor has protected access. You cannot create a direct instance of class `LoaderClass`.

### Parameters

`orb` The `orb` parameter provides support for multiple ORBs. This allows the newly-created loader to be associated with a specific ORB instance.

`registerMe` The value of this parameter must be `true` if the `load()` method of the new loader is to be called by Orbix Java, rather than explicitly by the programmer.

### Notes

Orbix-specific.

### See Also

Other class constructors.

## load()

### Synopsis

```
public org.omg.CORBA.Object  
    load (String interfaceName, String marker, boolean local)  
        throws SystemException;
```

### Description

When an object fault occurs, the `load()` method is called on each loader in turn until one of them successfully returns the address of the object, or until they have all returned `null`.

The responsibility of the `load()` method is to determine if the required object is to be loaded by the current loader, and, if so, create the object and assign the correct marker to it.

## Parameters

<code>interfaceName</code>	<p>The interface name of the missing object is determined as follows:</p> <ul style="list-style-type: none"><li>• If an object fault occurs during the call: <pre>// Java g = GridHelper._bind( &lt;parameters&gt; );</pre>the interface name in <code>load()</code> is "Grid".</li><li>• If the first parameter to <code>bind()</code> is a full object reference string, Orbix Java returns an exception if the reference's interface field is not <code>Grid</code> or a derived interface of <code>Grid</code>.</li><li>• If an object fault occurs during the call: <pre>// Java g = org.omg.CORBA.ORB.string_to_object ( &lt;full object reference string&gt; );</pre>the interface name in <code>load()</code> is that extracted from the full object reference string.</li><li>• If a loader is called because of a reference entering an address space (as an <code>in</code>, <code>out</code> or <code>inout</code> parameter, a return value, or as the target object of an operation call), then the interface name in <code>load()</code> is the interface name extracted from the object reference.</li></ul> <p>The switches passed to the IDL compiler affect how the interface name is seen by <code>load()</code>. See the method "<a href="#">_interfaceMarker()</a>" on page 114.</p>
<code>marker</code>	<p>The marker of the required object.</p>
<code>local</code>	<p>Set to <code>true</code> if the object fault occurred because of a collocated call to <code>bind()</code> or <code>org.omg.CORBA.ORB.string_to_object()</code> by the process itself.</p> <p>Set to <code>false</code> if the object fault occurred because of an object fault on the target object of an incoming operation invocation, or on an <code>in</code>, <code>out</code> or <code>inout</code> parameter or return value</p>

**Return Value** Returns the object reference, if the object is made available, or `null` otherwise.

**Notes** Orbix-specific.

**See Also** ["Class org.omg.CORBA.ORB" on page 19](#)  
["save\(\)" on page 226](#)  
["\\_interfaceMarker\(\)" on page 114](#)

## record()

### Synopsis

```
public void record (org.omg.CORBA.Object obj,  
                  StringHolder marker);
```

### Description

When you name an object by passing a marker name to the TIE or ImplBase constructor, Orbix Java calls `record()` on the object's loader.

A derived class may redefine `record()` to override your choice of name.

The default loader inherits its implementation of `record()` from `LoaderClass`. This implementation does not change the chosen name. It may, however, raise an exception if the name is in use (that is, an object with the same interface name and marker name already exists in the server process) and the object consequently cannot be registered.

If no marker name is suggested, the default `record()` method chooses a name that is a string of decimal digits, different to any generated before in the current process.

You can also name an object using the method [“\\_marker\(\)” on page 116](#). In this case, Orbix Java calls `rename()` rather than `record()` on the object's loader.

### Parameters

<code>obj</code>	The object currently being registered.
<code>marker</code>	The name of the object marker.

### Notes

Orbix-specific.

### See Also

[“rename\(\)” on page 225](#)  
[“\\_marker\(\)” on page 116](#)

## rename()

### Synopsis

```
public boolean rename (org.omg.CORBA.Object obj,  
                     StringHolder marker);
```

### Description

When you name an object by calling `_marker()` in class `IE.Iona.OrbixWeb.CORBA.BaseObject`, Orbix Java calls `rename()` on the object's loader.

A derived class may redefine `rename()` to override the programmer's choice of name.

The default loader inherits its implementation of `rename()` from `LoaderClass`. This implementation does not change the chosen name. It may, however, raise an exception if the name is in use (that is, an object with the same interface name and marker name already exists in the server process).

Note that an object may also be named by passing a marker name to the TIE or ImplBase constructor. In this case, Orbix Java calls `record()` on the object's loader.

### Parameters

<code>obj</code>	The object that is currently being renamed.
<code>marker</code>	The new name to be used for the objects marker.

**Return Value** Returns `true` if the object is successfully renamed. Returns `null` otherwise.

**Notes** Orbix-specific.

**See Also** ["record\(\)" on page 225](#)  
["\\_marker\(\)" on page 116](#)

## save()

**Synopsis**

```
public void save (org.omg.CORBA.Object obj, int reason)
```

**Description** Immediately before process termination, you may invoke the method `IE.Iona.OrbixWeb.CORBA.ORB.finalize()` on the `_CORBA.Orbix` object. Orbix Java then iterates through all of the objects in its object table and calls `save()` on the loader associated with each object. A loader may save the object to persistent storage (either by calling a method on the object, or by accessing the object's data).

The associated loader's `save()` method is also called when an object is destroyed using `org.omg.CORBA.ORB.disconnect()`.

You can call `save()` explicitly for an object by calling its `IE.Iona.OrbixWeb.CORBA.ObjectRef._save()` method. `IE.Iona.OrbixWeb.CORBA.ObjectRef._save()` calls the `save()` method on the object's loader. The `_save()` method must be called in the same address space as the target object: calling it in a client process (that is, on a proxy) has effect.

## Parameters

`obj` The object on who's loader `save()` is asked to store.

`reason` The reason that `save()` has been called. This may be:

- `_CORBA.processTermination`: The process is about to exit.
- `_CORBA.explicitCall`: The object's `_save()` method has been called.
- `_CORBA.objectDeletion`: A call to `org.omg.CORBA.ORB.disconnect()` has been made with `obj`.

**Notes** Orbix-specific.

**See Also** ["\\_save\(\)" on page 118](#)  
["load\(\)" on page 223](#)  
["Class org.omg.CORBA.ORB" on page 19](#)  
["processTermination" on page 242](#)  
["explicit\\_call" on page 240](#)  
["objectDeletion" on page 241](#)

# Class IE.Iona.OrbixWeb.Features.locatorClass

## Synopsis

Class `IE.Iona.OrbixWeb.Features.locatorClass` is a base class defining the interface to a location server. Orbix Java uses the installed locator to find a target object in the distributed system when `bind()` is called with a null host name.

The default locator, provided by Orbix Java, searches for the target object of a `bind()` call. For details on the `bind()` method, refer to the *Orbix Programmer's Guide Java Edition*.

You can override the default locator by defining and implementing a derived class of `locatorClass` as described in the *Orbix Programmer's Guide Java Edition*.

## Orbix Java

```
// Java

package IE.Iona.OrbixWeb.Features;

public class locatorClass {

    protected locatorClass();

    public String[] lookUp (String serviceName, int MaxHops,
                           Context ctx);
}
```

## See Also

*Orbix Programmer's Guide Java Edition*

## locatorClass()

### Synopsis

```
protected locatorClass()
```

### Description

Default constructor.

### Notes

Orbix-specific. This class is `protected`, so you cannot create an instance of this class directly.

## lookUp()

### Synopsis

```
public String[] lookUp (String serviceName, int MaxHops,
                        Context ctx);
```

### Description

Searches for a server. It is called on the locator pointed to by `ORB.locator()`, when `bind()` is called with a null host name. Any parameters to `bind()` are passed to `lookUp()`.

### Parameters

<code>serviceName</code>	The name of the server being sought
<code>MaxHops</code>	In the default locator, this is interpreted as the maximum number of machines to search for the required server.  An interpretation similar to this one should be retained in a user-defined locator if it is to be used without changing client code that explicitly calls <code>lookUp()</code>

ctx

This allows a client to pass extra information to the locator; for example, constraints on how to search for the server. You can use the context parameter to define properties to be used when deciding between a set of servers with the same name.

The Context passed to `lookup()` originates in the Context value passed to a `bind()` call.

The default locator ignores this parameter.

## Return Value

String[]

The default locator returns a list of names of hosts on which the server is registered in the Implementation Repository. The default implementation of the locator randomizes the sequence before returning it. It also adds the `localhost` to the end of the array. This is a basic technique in load balancing to avoid swamping any one server. A user-defined `locatorClass` should keep a similar interpretation. An empty array is returned if no host names can be found for the specified server.

There is a different rule for applets. Refer to the chapter "Locating Servers at Runtime" in the ***Orbix Programmer's Guide Java Edition*** for details.

## Notes

Orbix-specific.

## See Also

"Locating Servers at Runtime" in the ***Orbix Programmer's Guide Java Edition***.

# Class IE.Iona.OrbixWeb.Features.OrbConfig

## Synopsis

Class `OrbConfig` provides methods for accessing and setting Orbix Java configuration variables. The `OrbConfig` class replaces class `IE.Iona.OrbixWeb.Features.Config`. The `Config` class is deprecated because of Orbix Java's support for multiple ORBs. Configuration is now on a per-ORB basis.

Refer to the ***Orbix Programmer's Guide Java Edition*** for full details of Orbix Java configuration variables.

## Orbix Java

```
// Java

public class OrbConfig {

    // Constructor
    public OrbConfig(ORB);

    // Methods
    public String defaultConfigFile();
    public Hashtable getConfigFile();
    public String getConfigItem(String);
    public synchronized Properties getConfiguration();
    public synchronized void setConfigItem(String, String);
    public synchronized void setConfiguration(Properties);
    public synchronized void setConfiguration (String name);
    public synchronized void zeroConfiguration();
}
```

## Notes

Orbix-specific.

## See Also

["Class IE.Iona.OrbixWeb.Features.Config" on page 205](#)  
["config\(\)" on page 128](#)

## OrbConfig()

### Synopsis

```
public OrbConfig(Orb orb);
```

### Description

The default `OrbConfig` constructor.

### Parameters

<code>orb</code>	Enables configuration to be associated with a specific ORB instance. This provides support for multiple ORBs.
------------------	---------------------------------------------------------------------------------------------------------------

## Notes

Orbix-specific.

## defaultConfigFile()

### Synopsis

```
public String defaultConfigFile();
```

### Description

Returns a `String` containing the default `orbixweb3.cfg` configuration file. This method is provided for emergency use—if you accidentally delete your configuration file. Calling this API creates a new configuration file containing the default values.

## Notes

Orbix-specific.

## getConfigFile()

### Synopsis

```
public Hashtable getConfigFile();
```

### Description

Gets the current settings in the Orbix Java configuration files. This includes the contents of `iona.cfg`, `common.cfg`, `Orbixweb3.cfg`, and `orbixnames3.cfg`.

This method also supports `orbix.cfg` and `orbixweb.properties` files, for backwards compatibility.

### Notes

Orbix-specific.

## getConfigItem()

### Synopsis

```
public String getConfigItem(String name);
```

### Description

Gets the value of the specified configuration parameter in `String` form. Refer to the *Orbix Administrator's Guide Java Edition* for a complete list of Orbix Java configuration parameters.

### Parameters

<code>name</code>	The configuration item name in <code>String</code> form; for example: <pre>getConfigItem("IT_DEFAULT_TIMEOUT");</pre>
-------------------	--------------------------------------------------------------------------------------------------------------------------

### Notes

Orbix-specific.

## getConfiguration()

### Synopsis

```
public synchronized Properties getConfiguration();
```

### Description

Returns the configuration parameters programmatically set by the user.

### Notes

Orbix-specific.

### See Also

["setConfiguration\(\)" on page 230](#)

## setConfiguration()

### Synopsis

```
public synchronized void setConfiguration(Properties props);
```

### Description

Sets configuration parameters from the specified `java.util.Properties` object.

### Notes

Orbix-specific.

### See Also

["setConfiguration\(\)" on page 230](#)

## setConfiguration()

### Synopsis

```
public synchronized void setConfiguration (String name);
```

### Description

Sets configuration from a specified configuration file. The file you specify must be included on the classpath.

### Notes

Orbix-specific.

### See Also

["setConfiguration\(\)" on page 230](#)



## setConfigItem()

### Synopsis

```
public synchronized void setConfigItem(String name, String  
value)
```

### Description

Sets a configuration item in `String` form. Refer to the ***Orbix Administrator's Guide Java Edition*** for a complete list of Orbix Java configuration items.

### Parameters

name	The configuration item name in <code>String</code> form.
value	The configuration item value in <code>String</code> form; for example: <pre>setConfigItem("IT_DEFAULT_TIMEOUT", "80000");</pre>

### Notes

Orbix-specific.

## zeroConfiguration()

### Synopsis

```
public synchronized void zeroConfiguration();
```

### Description

Resets any configuration settings, returning to compiled-in defaults. This method also resets any configuration that has been loaded from configuration files or URLs.

### Notes

Orbix-specific.

### See also

["setConfiguration\(\)" on page 230](#)  
["setConfigItem\(\)" on page 231](#)

# Class IE.Iona.OrbixWeb.Features.ProxyFactory

## Synopsis

`ProxyFactory` is the base class for Orbix Java proxy factory classes. Orbix Java allows smart proxies to be implemented for IDL interfaces. When implementing a smart proxy, you must declare a derived class of the default proxy factory class for an IDL type and override the proxy factory `New()` method. To state which IDL interface type your `ProxyFactory` is to be used for, pass the interface ID to the constructor for the base class.

## Orbix Java

```
// Java

package IE.Iona.OrbixWeb.Features;

public class ProxyFactory {
    // Ctor
    protected ProxyFactory(String id);
    protected ProxyFactory(org.omg.CORBA.ORB orb, String name);
    // Method
    public org.omg.CORBA.Object
        New(org.omg.CORBA.portable.Delegate d);
}
```

## ProxyFactory()

### Synopsis

```
protected ProxyFactory(String id);
```

### Description

This constructor registers this object with the system as providing the specified interface ID. The name can be retrieved at runtime from the static method `<interfaceName>Helper.id()`. This constructor should be called from the constructor of the smart proxy being created using `super(<InterfaceName>Helper.id())`.

### Parameters

`id`            The ID of the IDL interface implemented by this object.

### Notes

Orbix-specific.

## ProxyFactory()

### Synopsis

```
protected ProxyFactory(org.omg.CORBA.ORB orb, String name);
```

### Description

This constructor registers this object with the system as providing the specified interface name. The name can be retrieved at runtime from the static method `<interfaceName>Helper.id()`. This constructor should be called from the constructor of the Smart Proxy being created using `super(<InterfaceName>Helper.id())`.

### Parameters

`orb`            The `orb` parameter provides support for multiple ORBs. This allows the smart proxy to be associated with a specific ORB instance.

`name`          The name of the IDL interface implemented by this object

### Notes

Orbix-specific.

## New()

### Synopsis

```
public org.omg.CORBA.Object New(org.omg.CORBA.portable.Delegate d)
```

### Description

This method should be overridden in a derived smart proxy class. The derived method is invoked by Orbix Java each time a new smart proxy is created. Orbix Java passes it the delegate of the real object for which the proxy is required in the `d` parameter.

### Parameters

`d`            The delegate of the real object for which the proxy is required

### Return Value

Returns the new proxy object reference.

### Notes

Orbix-specific.

### See Also

[“Interface org.omg.CORBA.Object” on page 18](#)

# Class IE.Iona.OrbixWeb.Features.ThreadFilter

## Synopsis

The conceptually abstract class `ThreadFilter` is special kind of filter that you can use to implement custom threading and queuing policies.

To take advantage of a `ThreadFilter`'s special functionality you should define a derived class of `ThreadFilter` and redefine the `inRequestPreMarshal()` method. When a request enters this filter point, you can take over responsibility for dispatching the request. You can then pass the request into a custom event queue serviced by one or more threads, or you can create a thread directly and pass it the `Request` object to be dispatched.

The class `ThreadFilter` inherits from the class `Filter`. Although the `ThreadFilter` does not redefine any of the `Filter` class' method signatures, it does change the behaviour of `inRequestPreMarshal` (see below) and the default constructor. Redefining any of the other `Filter` operations has no special effect. A separate chain of `ThreadFilter`s is maintained by the Orbix Java run-time.

To take advantage of the special abilities of the `ThreadFilter` you must either use its default constructor, or pass an ORB instance to the constructor to add the filter to that ORB's `ThreadFilter` chain.

See the "thread1", "thread2" and "thread3" demos for sample code.

## Orbix Java

```
// Java

package IE.Iona.OrbixWeb.Features;

public class ThreadFilter extends Filter {
    protected ThreadFilter();
    protected ThreadFilter(org.omg.CORBA.ORB orb)
}

```

## Notes

Orbix-specific.

## See Also

["Class org.omg.CORBA.Request" on page 29](#)  
["Class org.omg.CORBA.SystemException" on page 35](#)  
["continueThreadDispatch\(\)" on page 65](#)  
["Class IE.Iona.OrbixWeb.Features.Filter" on page 207](#)  
["Class IE.Iona.OrbixWeb.Features.AuthenticationFilter" on page 203](#)

## ThreadFilter()

### Synopsis

```
protected ThreadFilter();
```

### Description

The default constructor adds the newly-created `ThreadFilter` object onto the `ThreadFilter` chain. Direct instances of `ThreadFilter` cannot be created, and the constructor is protected to enforce this. The derived classes of `ThreadFilter` normally have public constructors.

### Notes

Orbix-specific.

## ThreadFilter()

### Synopsis

```
protected ThreadFilter(org.omg.CORBA.ORB orb);
```

### Description

The `orb` parameter provides support for multiple ORBs. This allows the newly created `ThreadFilter` to be associated with a specific ORB instance. This constructor adds the newly-created `ThreadFilter` object onto the ORB's `ThreadFilter` chain.

Direct instances of `ThreadFilter` cannot be created, and the constructor is protected to enforce this. The derived classes of `ThreadFilter` normally have public constructors.

### Notes

Orbix-specific.

## inRequestPreMarshal()

### Synopsis

```
public boolean inRequestPreMarshal(Request r);
```

### Description

Defines the action to perform for incoming requests: before any incoming operation on any object in the address space. In particular, before the operation has been sent to the target object and before the operation's parameters have been removed from the request packet.

The special behaviour for `ThreadFilters` is tied up with the return value.

If `true` is returned, the responsibility for processing the request object stays with the Orbix Java runtime and the request gets immediately dispatched.

If `false` is returned, (unlike a normal filter where a `FILTER_SUPPRESS` exception is thrown) it is your responsibility to decide when to process the request.

You can pass the current request object to a different thread or put the request into a custom request queue.

Once the application comes to a stage where it wishes to process the request, it should call the method

```
IE.Iona.OrbixWeb.CORBA.BOA.continueThreadDispatch().
```

The thread that calls `continueThreadDispatch` blocks on the call until the request has been fully processed and then the call returns.

If not redefined the default implementation is as follows:

```
// Java
{ return true; } // Continue the call.
```

### Parameters

`r` This is the `Request` object for the current invocation.

### Return Value

`true` Continue with the request as normal. The operation call is sent to the next filter on the chain, or, if this is the last filter, it is sent to the per-object filters, if any, and then to the target object.

false

Remove responsibility for dispatching this `Request` from the runtime. The runtime goes back to the internal queue and starts processing the next request to arrive. It is up to you to ensure that the request gets dispatched at some later time, whether in the current or in a different thread.

### Exceptions

If you redefine this method in a derived class, you can raise a `SystemException` to indicate that the call is not to be continued.

If you wish to raise a `FILTER_SUPPRESS` exception, you should create and throw it as a normal `SystemException`.

### Notes

Orbix-specific.

### See Also

["Class org.omg.CORBA.SystemException" on page 35](#)

["continueThreadDispatch\(\)" on page 65](#)

# Part IV

## Package IE.Iona.OrbixWeb

### In this part

This part contains the following:

<a href="#">Class IE.Iona.OrbixWeb._CORBA</a>	<a href="#">page 239</a>
<a href="#">Class IE.Iona.OrbixWeb._OrbixWeb</a>	<a href="#">page 243</a>





# Class IE.Iona.OrbixWeb.\_CORBA

## Synopsis

The `_CORBA` class implements constants and operations defined at the highest level of scope within the IDL `CORBA` module, and includes a number of other definitions specific to Orbix Java. This class includes a set of `TypeCode` constant definitions (including `_tc_null`, `_tc_short`, and `_tc_long`) that are described in detail in [“Class IE.Iona.OrbixWeb.CORBA.TypeCode” on page 190](#).

## Orbix Java

```
// Java
package IE.Iona.OrbixWeb;
public class _CORBA {

    public static final TypeCode _tc_null;
    public static final TypeCode _tc_void;
    public static final TypeCode _tc_short;
    public static final TypeCode _tc_long;
    public static final TypeCode _tc_ushort;
    public static final TypeCode _tc_ulong;
    public static final TypeCode _tc_float;
    public static final TypeCode _tc_double;
    public static final TypeCode _tc_boolean;
    public static final TypeCode _tc_char;
    public static final TypeCode _tc_octet;
    public static final TypeCode _tc_any;
    public static final TypeCode _tc_TypeCode;
    public static final TypeCode _tc_Principal;
    public static final TypeCode _tc_Object;
    public static final TypeCode _tc_struct;
    public static final TypeCode _tc_union;
    public static final TypeCode _tc_enum;
    public static final TypeCode _tc_string;
    public static final TypeCode _tc_NamedValue;
    public static final TypeCode _tc_wchar;
    public static final TypeCode _tc_wstring;
    public static final TypeCode _tc_longlong;
    public static final TypeCode _tc_ulonglong;
    public static final TypeCode _tc_fixed;

    //Locator and maximum number of hops
    public static IE.Iona.OrbixWeb.Features.locatorClass locator;

    //Loader constants
    public static final int processTermination;
    public static final int objectDeletion;
    public static final int explicitCall;

    //Object type constants
    public static final int IT_INTEROPERABLE_OR_KIND;
    public static final int IT_ORBIX_OR_KIND;

    //Timeout constant
    public static final int IT_INFINITE_TIMEOUT;

    public static IE.Iona.OrbixWeb.CORBA.ORB Orbix;

    public static final String _ORBIX_VERSION;
    public static final int _MAX_LOCATOR_HOPS;
};
```

## explicit\_call

<b>Synopsis</b>	<pre>public static final int explicit_call;</pre>
<b>Description</b>	You can pass this value to the method <code>IE.Iona.OrbixWeb.CORBA.Features.LoaderClass.save()</code> to indicate that <code>save()</code> has been called directly from an application using the <code>save()</code> method.
<b>Notes</b>	Orbix-specific.
<b>See Also</b>	<a href="#">"objectDeletion" on page 241</a> <a href="#">"processTermination" on page 242</a> <a href="#">"save()" on page 226</a>

## IT\_INFINITE\_TIMEOUT

<b>Synopsis</b>	<pre>public static final int IT_INFINITE_TIMEOUT</pre>
<b>Description</b>	A value you can pass as a timeout parameter to an Orbix Java event processing call to indicate that the call should never return.
<b>Notes</b>	Orbix-specific.
<b>See Also</b>	<a href="#">"Interface IE.Iona.OrbixWeb.CORBA.BOA" on page 59</a>

## IT\_INTEROPERABLE\_OR\_KIND

<b>Synopsis</b>	<pre>public static final int IT_INTEROPERABLE_OR_KIND;</pre>
<b>Description</b>	This value should be passed as the <code>kind</code> parameter to <code>IE.Iona.OrbixWeb.CORBA.BaseObject._object_to_string()</code> to convert an object reference to a stringified Interoperable Object Reference (IOR).
<b>Notes</b>	Orbix-specific.

## IT\_ORBIX\_OR\_KIND

<b>Synopsis</b>	<pre>public static final int IT_ORBIX_OR_KIND;</pre>
<b>Description</b>	You should pass this value as the <code>kind</code> parameter to <code>IE.Iona.OrbixWeb.CORBA.BaseObject._object_to_string()</code> to convert an object reference to a stringified Orbix object reference.
<b>Notes</b>	Orbix-specific.

## \_ORBIX\_VERSION

<b>Synopsis</b>	<pre>public static final String _ORBIX_VERSION;</pre>
<b>Description</b>	A string containing a description of this version of Orbix, for example, "OrbixWeb 3.2".
<b>Notes</b>	Orbix-specific.

## \_MAX\_LOCATOR\_HOPS

<b>Synopsis</b>	<pre>public static final int _MAX_LOCATOR_HOPS;</pre>
<b>Description</b>	This specifies the maximum value that <code>_LOCATOR_HOPS</code> can be set to. The value of <code>MAX_LOCATOR_HOPS</code> is 10.
<b>Notes</b>	Orbix-specific.

**See Also** ["Class IE.Iona.OrbixWeb.Features.locatorClass" on page 227](#)

## objectDeletion

### Synopsis

```
public static final int objectDeletion;
```

### Description

You can pass this value to the method `IE.Iona.OrbixWeb.Features.LoaderClass.save()` to indicate that `save()` is called as the result of `IE.Iona.OrbixWeb.CORBA.BOA.dispose()` invocation on an object reference.

### Notes

Orbix-specific.

### See Also

["Class IE.Iona.OrbixWeb.Features.locatorClass" on page 227](#)

## Orbix

### Synopsis

```
public static IE.Iona.OrbixWeb.CORBA.ORB Orbix
```

### Description

This variable is a convenience for Orbix Java programmers. Its use is deprecated, you should use the `ORB.init()` method instead.

It is an instance of an ORB object of type `IE.Iona.OrbixWeb.CORBA.ORB` (in a pure client) or of type `IE.Iona.OrbixWeb.CORBA.BOA` (in a server). This is the same object as that returned by `ORB.init()` except that its type has already been narrowed to the Orbix Java correct implementation class.

Before you use `_CORBA.Orbix` you must call `ORB.init()` at least once in order to create the ORB object in the first place.

### Multiple ORB Support

`_CORBA.Orbix` refers to the *default ORB*, the first full ORB created. If that ORB is destroyed, `_CORBA.Orbix` becomes `null` until another ORB is created.

Only server objects connected to the default ORB can be persistent and contacted via `orbxid`; for example, by clients calling `bind()`. Server objects connected to other ORBs in the virtual machine listen on other ports unknown to `orbxid` and should be regarded as transient objects.

Features such as filters smart proxies and loaders are associated with particular ORB instances. By default, this is the default ORB. The Orbix Java API allows any of these to be associated with a selected ORB instance.

### Notes

Orbix-specific.

### See Also

`init()` in ["Class IE.Iona.OrbixWeb.CORBA.ORB" on page 120](#)  
["Class IE.Iona.OrbixWeb.Features.Filter" on page 207](#)  
["Class IE.Iona.OrbixWeb.Features.AuthenticationFilter" on page 203](#)  
["Class IE.Iona.OrbixWeb.Features.LoaderClass" on page 222](#)  
["Class IE.Iona.OrbixWeb.Features.ProxyFactory" on page 232](#)

## processTermination

### Synopsis

```
public static final int processTermination;
```

### Description

You can pass this value to the method `IE.Iona.OrbixWeb.Features.LoaderClass.save()` to indicate that `save()` has been called due to the termination of the current process. This can be, for example, due to a server timing out.

### Notes

Orbix-specific.

### See Also

["Class IE.Iona.OrbixWeb.Features.locatorClass" on page 227](#)

# Class IE.Iona.OrbixWeb.\_OrbixWeb

## Synopsis

The methods in the `_OrbixWeb` class return an Orbix Java implementation class for a supplied OMG abstract base class. The `_OrbixWeb` class provides a consistent interface to allow you to access the value-added features and methods of the Orbix Java implementations of the CORBA abstract base classes. Typically, this involves casting from an abstract class in the `org.omg.CORBA` package to an implementation class in the `IE.Iona.OrbixWeb` package. The system exception `BAD_PARAM` is thrown if the supplied object cannot be converted to the desired return type, for example, if there are multiple ORBs in the same virtual machine.

## Orbix Java

```
// Java

package IE.Iona.OrbixWeb;
public class _OrbixWeb {
    public static IE.Iona.OrbixWeb.CORBA.InputCoder
        MarshalBuffer(org.omg.CORBA.portable.InputStream stream)
            throws org.omg.CORBA.BAD_PARAM, SystemException;
    public static IE.Iona.OrbixWeb.CORBA.OutputCoder
        MarshalBuffer(org.omg.CORBA.portable.OutputStream
            stream)
            throws org.omg.CORBA.BAD_PARAM, SystemException;
    public static IE.Iona.OrbixWeb.CORBA.ORB
        ORB(org.omg.CORBA.ORB orb)
            throws org.omg.CORBA.BAD_PARAM, SystemException;
    public static IE.Iona.OrbixWeb.CORBA.BOA
        BOA(org.omg.CORBA.ORB orb)
            throws org.omg.CORBA.BAD_PARAM, SystemException;
    public static IE.Iona.OrbixWeb.CORBA.Any
        Any(org.omg.CORBA.Any a)
            throws org.omg.CORBA.BAD_PARAM, SystemException;
    public static final IE.Iona.OrbixWeb.OrbixProt.OrbixTypeCode
        TypeCode(org.omg.CORBA.TypeCode t)
            throws org.omg.CORBA.BAD_PARAM, SystemException;
    public static final IE.Iona.OrbixWeb.CORBA.NamedValue
        NamedValue(org.omg.CORBA.NamedValue nv)
            throws org.omg.CORBA.BAD_PARAM, SystemException;
    public static final IE.Iona.OrbixWeb.CORBA.NVList
        NVList(org.omg.CORBA.NVList nvl)
            throws org.omg.CORBA.BAD_PARAM, SystemException;
    public static final IE.Iona.OrbixWeb.CORBA.Context
        Context(org.omg.CORBA.Context c)
            throws org.omg.CORBA.BAD_PARAM, SystemException;
    public static final IE.Iona.OrbixWeb.CORBA.ContextList
        ContextList(org.omg.CORBA.ContextList cl)
            throws org.omg.CORBA.BAD_PARAM, SystemException;
    public static final IE.Iona.OrbixWeb.CORBA.ObjectRef
        Object(org.omg.CORBA.Object o)
            throws org.omg.CORBA.BAD_PARAM, SystemException;
    public static final IE.Iona.OrbixWeb.CORBA.Principal
        Principal(org.omg.CORBA.Principal p)
            throws org.omg.CORBA.BAD_PARAM, SystemException;
    public static final IE.Iona.OrbixWeb.CORBA.Request
        Request(org.omg.CORBA.Request r)
            throws org.omg.CORBA.BAD_PARAM, SystemException;
```

```
public static final IE.Iona.OrbixWeb.CORBA.DSIServerRequest
    ServerRequest(org.omg.CORBA.ServerRequest r)
    throws org.omg.CORBA.BAD_PARAM, SystemException;
public static final IE.Iona.OrbixWeb.CORBA.OrbCurrent
    Current()
    throws org.omg.CORBA.BAD_PARAM, SystemException;
```

## Any()

### Synopsis

```
public static IE.Iona.OrbixWeb.CORBA.Any
    Any(org.omg.CORBA.Any a);
```

### Description

Converts a reference of type `org.omg.CORBA.Any` to a reference of type `IE.Iona.OrbixWeb.CORBA.Any`.

### Parameters

a            The `Any` to be converted.

### Return Value

Returns an `Any` of type `IE.Iona.OrbixWeb.CORBA.Any`

### Notes

Orbix-specific.

### See Also

["Class IE.Iona.OrbixWeb.CORBA.Any" on page 47](#)

## Context()

### Synopsis

```
public static final IE.Iona.OrbixWeb.CORBA.Context
    Context(org.omg.CORBA.Context c)
```

### Description

Converts a reference of type `org.omg.CORBA.Context` to a reference of type `IE.Iona.OrbixWeb.CORBA.Context`.

### Parameters

c            The `Context` to be converted.

### Return Value

Returns a `Context` of type `IE.Iona.OrbixWeb.CORBA.Context`

### Notes

Orbix-specific.

### See Also

["Class org.omg.CORBA.Context" on page 9](#)  
["Class IE.Iona.OrbixWeb.CORBA.Context" on page 83](#)

## ContextList()

### Synopsis

```
public static final IE.Iona.OrbixWeb.CORBA.ContextList
    ContextList(org.omg.CORBA.ContextList c)
```

### Description

Converts a reference of type `org.omg.CORBA.ContextList` to a reference of type `IE.Iona.OrbixWeb.CORBA.ContextList`.

### Parameters

c            The `ContextList` to be converted.

### Return Value

Returns a `Context` of type `IE.Iona.OrbixWeb.CORBA.ContextList`

### Notes

Orbix-specific.

### See Also

["Class org.omg.CORBA.ContextList" on page 10](#)  
["Class IE.Iona.OrbixWeb.CORBA.ContextList" on page 90](#)

## Current()

### Synopsis

### Description

```
public static final IE.Iona.OrbixWeb.CORBA.OrbCurrent Current()
```

The current object contains information about the request currently being processed. This method can be validly called during the execution of the request itself as well as in filters and loaders. These operations include the following:

```
// Return the principal of the client.
```

```
public static org.omg.CORBA.Principal get_principal()
```

```
// Return the DSI server request.
```

```
public static org.omg.CORBA.ServerRequest get_request()
```

```
// Return the target of the request.
```

```
public static org.omg.CORBA.Object get_object()
```

```
// Return the protocol (IIOP or Orbix) in which the  
// request was transmitted.
```

```
public static int get_protocol()
```

```
// Return the connection object on which the request arrived  
// in the server.
```

```
public static java.lang.Object get_socket()
```

To ensure thread safety, if you expect the server to concurrently execute requests. You should set the configurable item `IT_MULTI_THREADED_SERVER` to `true` to ensure that these operations return the information associated with the current thread rather than another active thread.

### Return Value

`OrbCurrent`            The returned current object.

### Notes

Orbix-specific.

### See Also

["get\\_current\(\)" on page 69](#)

["Class IE.Iona.OrbixWeb.CORBA.OrbCurrent" on page 159](#)

["Class org.omg.CORBA.Current" on page 12](#)

## NamedValue()

### Synopsis

```
public static final IE.Iona.OrbixWeb.CORBA.NamedValue  
    NamedValue(org.omg.CORBA.NamedValue nv);
```

### Description

Converts a reference of type `org.omg.CORBA.NamedValue` to a reference of type `IE.Iona.OrbixWeb.CORBA.NamedValue`.

### Parameters

`nv`            The `NamedValue` to be converted.

### Return Value

Returns a `NamedValue` of type `IE.Iona.OrbixWeb.CORBA.NamedValue`

### Notes

Orbix-specific.

### See Also

["Class org.omg.CORBA.NamedValue" on page 16](#)  
["Class IE.Iona.OrbixWeb.CORBA.NamedValue" on page 98](#)

## NVList()

### Synopsis

```
public static final IE.Iona.OrbixWeb.CORBA.NVList  
    NVList(org.omg.CORBA.NVList nvl)
```

### Description

Converts a reference of type `org.omg.CORBA.NVList` to a reference of type `IE.Iona.OrbixWeb.CORBA.NVList`.

### Parameters

`nvl`            The `NVList` to be converted.

### Return Value

Returns an `NVList` of type `IE.Iona.OrbixWeb.CORBA.NVList`

### Notes

Orbix-specific.

### See Also

["Class org.omg.CORBA.NVList" on page 17](#)  
["Class IE.Iona.OrbixWeb.CORBA.NVList" page 102](#)

## Object()

### Synopsis

```
public static final IE.Iona.OrbixWeb.CORBA.ObjectRef  
    Object(org.omg.CORBA.Object o)
```

### Description

Converts a reference of type `org.omg.CORBA.Object` to a reference of type `IE.Iona.OrbixWeb.CORBA.ObjectRef`. This allows the use of extra Orbix Java specific methods with object references.

### Parameters

`o`            The `org.omg.CORBA.Object` to be converted.

### Return Value

Returns an `Object` of type `IE.Iona.OrbixWeb.CORBA.ObjectRef`

### Notes

Orbix-specific.

### See Also

["Interface org.omg.CORBA.Object" on page 18](#)  
["Interface IE.Iona.OrbixWeb.CORBA.ObjectRef" on page 109](#)



## Principal()

### Synopsis

```
public static final IE.Iona.OrbixWeb.CORBA.Principal  
Principal(org.omg.CORBA.Principal p)
```

### Description

Converts a reference of type `org.omg.CORBA.Principal` to a reference of type `IE.Iona.OrbixWeb.CORBA.Principal`.

### Parameters

`p` The `Principal` to be converted.

### Return Value

Returns a `Principal` of type `IE.Iona.OrbixWeb.CORBA.Principal`

### Notes

Orbix-specific.

### See Also

["Class org.omg.CORBA.Principal" on page 28](#)  
["Class IE.Iona.OrbixWeb.CORBA.Principal" on page 164](#)

## Request()

### Synopsis

```
public static final IE.Iona.OrbixWeb.CORBA.Request  
Request(org.omg.CORBA.Request r)
```

### Description

Converts a reference of type `org.omg.CORBA.Request` to a reference of type `IE.Iona.OrbixWeb.CORBA.Request`.

### Parameters

`r` The `Request` to be converted.

### Return Value

Returns a `Request` of type `IE.Iona.OrbixWeb.CORBA.Request`

### Notes

Orbix-specific.

### See Also

["Class org.omg.CORBA.Request" on page 29](#)  
["Class IE.Iona.OrbixWeb.CORBA.Request" on page 167](#)

## ServerRequest()

### Synopsis

```
public static final IE.Iona.OrbixWeb.CORBA.DSIServerRequest  
ServerRequest(org.omg.CORBA.ServerRequest sr)
```

### Description

Converts a reference of type `org.omg.CORBA.ServerRequest` to a reference of type `IE.Iona.OrbixWeb.CORBA.DSIServerRequest`.

### Parameters

`sr` The `ServerRequest` to be converted.

### Return Value

Returns a `ServerRequest` of type  
`IE.Iona.OrbixWeb.CORBA.DSIServerRequest`

### Notes

Orbix-specific.

### See Also

["Class org.omg.CORBA.ServerRequest" on page 31](#)

## TypeCode()

### Synopsis

```
public static final IE.Iona.OrbixWeb.OrbixProt.OrbixTypeCode  
    TypeCode(org.omg.CORBA.TypeCode t)
```

### Description

Converts a reference of type `org.omg.CORBA.TypeCode` to a reference of type `IE.Iona.OrbixWeb.OrbixProt.OrbixTypeCode`.

### Parameters

`t`                    The `TypeCode` to be converted.

### Return Value

Returns a `Typecode` of type `IE.Iona.OrbixWeb.OrbixProt.OrbixTypeCode`.

### Notes

Orbix-specific. This function is only for use with the Orbix protocol as the returned object is optimized for this protocol and has Orbix-specific additions, or for debug purposes as the returned class contains a `'toString()'` operation for outputting the `Typecode` in Orbix format. Normally the runtime is the only user of this class. There is no equivalent method for the IIOP protocol as the class `IE.Iona.OrbixWeb.CORBA.TypeCode` directly implements the `org.omg.CORBA.TypeCode` abstract base class only.

### See Also

["Class IE.Iona.OrbixWeb.CORBA.TypeCode" on page 190](#)

# Part V

## IDL Interface to the Interface Repository

### In this part

This part contains the following:

<a href="#">Common CORBA Data Types</a>	<a href="#">page 251</a>
<a href="#">CORBA::AliasDef</a>	<a href="#">page 252</a>
<a href="#">CORBA::ArrayDef</a>	<a href="#">page 253</a>
<a href="#">CORBA::AttributeDef</a>	<a href="#">page 254</a>
<a href="#">CORBA::ConstantDef</a>	<a href="#">page 256</a>
<a href="#">CORBA::Contained</a>	<a href="#">page 258</a>
<a href="#">CORBA::Container</a>	<a href="#">page 262</a>
<a href="#">CORBA::EnumDef</a>	<a href="#">page 270</a>
<a href="#">CORBA::ExceptionDef</a>	<a href="#">page 271</a>
<a href="#">CORBA::IDLType</a>	<a href="#">page 273</a>
<a href="#">CORBA::InterfaceDef</a>	<a href="#">page 274</a>
<a href="#">CORBA::IObject</a>	<a href="#">page 279</a>
<a href="#">CORBA::InterfaceDef</a>	<a href="#">page 274</a>
<a href="#">CORBA::IT_Repository</a>	<a href="#">page 281</a>
<a href="#">CORBA::ModuleDef</a>	<a href="#">page 283</a>
<a href="#">CORBA::OperationDef</a>	<a href="#">page 284</a>
<a href="#">CORBA::PrimitiveDef</a>	<a href="#">page 287</a>

<a href="#">CORBA::Repository</a>	<a href="#">page 288</a>
<a href="#">CORBA::SequenceDef</a>	<a href="#">page 291</a>
<a href="#">CORBA::StructDef</a>	<a href="#">page 293</a>
<a href="#">CORBA::StringDef</a>	<a href="#">page 295</a>
<a href="#">CORBA::TypedefDef</a>	<a href="#">page 296</a>
<a href="#">CORBA::UnionDef</a>	<a href="#">page 297</a>

# Common CORBA Data Types

## CORBA::DefinitionKind

**Synopsis**

```
enum DefinitionKind {
    dk_none, dk_all,
    dk_Attribute, dk_Constant, dk_Exception, dk_Interface,
    dk_Module, dk_Operation, dk_Typedef,
    dk_Alias, dk_Struct, dk_Union, dk_Enum,
    dk_Primitive, dk_String, dk_Sequence, dk_Array,
    dk_Repository
};
```

**Description** Each Interface Repository (IFR) object has an attribute (`def_kind`) of type `DefinitionKind` that records the kind of the IFR object. For example, the `def_kind` attribute of an `InterfaceDef` object is `dk_interface`. The enumeration constants `dk_none` and `dk_all` have special meanings when searching for an object in a repository.

**Notes** CORBA-defined.

## CORBA::Identifier

**Synopsis**

```
typedef string Identifier;
```

**Description** A simple name that identifies modules, interfaces, constants, typedefs, exceptions, attributes, and operations. An identifier is not necessarily unique within the entire Interface Repository; it is unique only within a particular `Repository`, `ModuleDef`, `InterfaceDef`, or `OperationDef`.

**Notes** CORBA-defined.

## CORBA::RepositoryId

**Synopsis**

```
typedef string RepositoryId;
```

**Description** A string that uniquely identifies a module, interface, constant, typedef, exception, attribute, or operation.

**Notes** CORBA-defined.

## CORBA::ScopedName

**Synopsis**

```
typedef string ScopedName;
```

**Description** A `ScopedName` gives an entity's name relative to a scope. A `ScopedName` that begins with `::` is an *absolute scoped name*; one that uniquely identifies an entity within a repository. An example is:

```
::Account::makeWithdrawal.
```

A `ScopedName` that does not begin with `::` is a relative scoped name; one that identifies an entity relative to some other entity. An example is:

```
makeWithdrawal
```

This is within the entity with the absolute scoped name `::Account`.

**Notes** CORBA-defined.

# CORBA::AliasDef

**Synopsis** The interface `AliasDef` describes an IDL typedef that aliases another definition. It is used to represent an IDL typedef.

```
CORBA // IDL
// In module CORBA.

interface AliasDef : TypedefDef {
    attribute IDLType original_type_def;
};
```

**Notes** CORBA-defined.

**See Also** ["CORBA::Contained"](#)  
["Container::create\\_alias\(\)"](#)

## AliasDef::describe()

**Synopsis** Description `describe()`;

**Description** Inherited from `Contained`, the `describe()` operation returns a structure of type `Contained::Description`:

```
struct Description {
    DefinitionKind kind;
    any value;
};
```

The `DefinitionKind` for the `kind` member is `dk_Alias`. The value member is an `any` whose `TypeCode` is `_tc_AliasDescription` and whose value is a structure of type `TypeDescription`:

```
// IDL
struct TypeDescription {
    Identifier name;
    RepositoryId id;
    RepositoryId defined_in;
    VersionSpec version;
    TypeCode type;
};
```

**See Also** ["TypedefDef::describe\(\)"](#)

## AliasDef:original\_type\_def

**Synopsis** attribute `IDLType original_type_def`;

**Description** Identifies the type being aliased.

Modifying the `original_type_def` attribute automatically updates the `type` attribute (the `type` attribute is inherited from `TypedefDef` which in turn inherits it from `IDLType`). Both attributes contain the same information.

**Notes** CORBA-defined.

**See Also** ["IDLType::type"](#)

# CORBA::ArrayDef

**Synopsis** The interface `ArrayDef` represents a one-dimensional array. A multi-dimensional array is represented by an `ArrayDef` with an element type that is another array definition. The final element type represents the type of element contained in the array. You can create an instance of interface `ArrayDef` using `CORBA::Repository::create_array()`.

```
CORBA // IDL
// In module CORBA

interface ArrayDef : IDLType {
    attribute unsigned long length;
    readonly attribute TypeCode element_type;
    attribute IDLType element_type_def;
};
```

**Notes** CORBA-defined.

**See Also** ["CORBA::IDLType" on page 273](#)  
["ArrayDef::element\\_type\\_def" on page 253](#)  
["Repository::create\\_array\(\)" on page 288](#)

## ArrayDef::element\_type

**Synopsis** `readonly attribute TypeCode element_type;`

**Description** Identifies the type of the element contained in the array. This contains the same information as the attribute `element_type_def`.

**Notes** CORBA-defined.

**See Also** ["ArrayDef::element\\_type\\_def" on page 253](#)

## ArrayDef::element\_type\_def

**Synopsis** `attribute IDLType element_type_def;`

**Description** Describes the type of the element contained within the array. This contains the same information as the attribute `element_type`.  
You can change the type of elements contained in the array by changing this attribute. Changing this attribute also changes the `element_type` attribute.

**Notes** CORBA-defined.

## ArrayDef::length

**Synopsis** `attribute unsigned long length;`

**Description** Specifies the number of elements in the array.

**Notes** CORBA-defined.

# CORBA::AttributeDef

**Synopsis** Interface AttributeDef describes an IDL attribute.

**CORBA**

```
// IDL
// In module CORBA.
enum AttributeMode { ATTR_NORMAL, ATTR_READONLY };

interface AttributeDef : Contained {
    readonly attribute TypeCode type;
    attribute IDLType type_def;
    attribute AttributeMode mode;
};
```

**Notes** CORBA-defined.

**See Also** ["CORBA::Contained"](#)  
["InterfaceDef::create\\_attribute\(\)"](#)

## AttributeDef::describe()

**Synopsis** Description describe();

**Description** Inherited from Contained, the describe() operation returns a structure of type Contained::Description:

```
struct Description {
    DefinitionKind kind;
    any value;
};
```

The DefinitionKind for the kind member is dk\_Attribute. The value member is an any whose TypeCode is \_tc\_AttributeDescription.

The value is a structure of type AttributeDescription:

```
// IDL
// In module CORBA.
struct AttributeDescription {
    Identifier name;
    RepositoryId id;
    RepositoryId defined_in;
    VersionSpec version;
    TypeCode type;
    AttributeMode mode;
};
```

**See Also** ["Contained::describe\(\)" on page 259](#)

## AttributeDef::mode

**Synopsis** attribute AttributeMode mode;

**Description** Specifies whether the attribute is read/write (ATTR\_NORMAL) or readonly (ATTR\_READONLY).

**Notes** CORBA-defined.



## AttributeDef::type

<b>Synopsis</b>	<code>readonly attribute TypeCode type;</code>
<b>Description</b>	Identifies the type of this attribute. The same information is contained in the <code>type_def</code> attribute.
<b>Notes</b>	CORBA-defined.
<b>See Also</b>	<a href="#">"AttributeDef::type_def"</a> <a href="#">"CORBA::TypedefDef"</a>

## AttributeDef::type\_def

<b>Synopsis</b>	<code>attribute IDLType type_def;</code>
<b>Description</b>	Describes the type for this attribute. The same information is contained in the <code>type</code> attribute. Changing the <code>type_def</code> attribute automatically changes the <code>type</code> attribute.
<b>Notes</b>	CORBA-defined.
<b>See Also</b>	<a href="#">"AttributeDef::type_def"</a> <a href="#">"CORBA::TypedefDef"</a>

# CORBA::ConstantDef

**Synopsis** Interface ConstantdefinedDef describes an IDL constant. The name of the constant is inherited from Contained.

```
CORBA // IDL
// in module CORBA.

interface ConstantDef : Contained {
    readonly attribute TypeCode type;
    attribute IDLType type_def;
    attribute any value;
};
```

**Notes** CORBA-defined.

**See Also** ["CORBA::Contained"](#)  
["Container::create\\_constant\(\)"](#)

## ConstantDef::describe()

**Synopsis** Description describe();

**Description** Inherited from Contained, the describe() operation returns a structure of type Contained::Description:

```
struct Description {
    DefinitionKind kind;
    any value;
};
```

The DefinitionKind for the kind member is dk\_Constant.

The value member is an any whose TypeCode is \_tc\_ConstantDescription and whose value is a structure of type ConstantDescription:

```
// IDL
struct ConstantDescription {
    Identifier name;
    RepositoryId id;
    RepositoryId defined_in;
    VersionSpec version;
    TypeCode type;
    any value;
};
```

**Notes** CORBA-defined.

**See Also** ["Contained::describe\(\)"](#)

## ConstantDef::type

**Synopsis** readonly attribute TypeCode type;

**Description** Identifies the type of this constant. The type must be a TypeCode for one of the simple types (such as long, short, float, char, string, double, boolean, unsigned long, and unsigned short). The same information is contained in the type\_def attribute.

**Notes** CORBA-defined.

**See Also** ["ConstantDef::type\\_def"](#)

## ConstantDef::type\_def

### Synopsis

attribute IDLType type\_def;

### Description

Identifies the type of the constant. The same information is contained in the `type` attribute.

You can change the type of a constant by changing its `type_def` attribute. This also changes its `type` attribute.

### Notes

CORBA-defined.

### See Also

[“ConstantDef::type”](#)

## ConstantDef::value

### Synopsis

attribute any value;

### Description

Contains the value for this constant. When changing the `value` attribute, the `TypeCode` of the `any` must be the same as the `type` attribute.

### Notes

CORBA-defined.

# CORBA::Contained

## Synopsis

Interface Contained is an abstract interface that describes Interface Repository objects that can be contained in a module, interface, or repository. It is a base interface for the following interfaces:

```
ModuleDef
InterfaceDef
ConstantDef
TypedefDef
ExceptionDef
AttributeDef
OperationDef
StructDef
EnumDef
UnionDef
AliasDef
```

## CORBA

```
// IDL
// In module CORBA.

typedef string VersionSpec;

interface Contained : IObject {
    attribute RepositoryId id;
    attribute Identifier name;
    attribute VersionSpec version;

    readonly attribute Container defined_in;
    readonly attribute ScopedName absolute_name;
    readonly attribute Repository containing_repository;

    struct Description {
        DefinitionKind kind;
        any value;
    };
    Description describe();
    void move (
        in Container new_container,
        in Identifier new_name,
        in VersionSpec new_version);
};
```

## Notes

CORBA-defined.

## See Also

["CORBA::Container"](#)  
["CORBA::IObject"](#)

## Contained::absolute\_name()

### Synopsis

```
readonly attribute ScopedName absolute_name;
```

### Description

Gives the absolute scoped name of an object.

### Notes

CORBA-defined.

## Contained::containing\_repository()

**Synopsis** `readonly attribute Repository containing_repository;`  
**Description** Gives the `Repository` within which the object is contained.  
**Notes** CORBA-defined.

## Contained::defined\_in

**Synopsis** `attribute Container defined_in;`  
**Description** Specifies the `Repository ID` for the `Interface Repository` object in which the object is contained.  
An `IFR` object is said to be contained by the `IFR` object in which it is defined. For example, an `InterfaceDef` object is contained by the `ModuleDef` in which it is defined.  
Contained also applies to objects of type `AttributeDef` or `OperationDef`. These objects may also be said to be contained in an `InterfaceDef` object if they are inherited into that interface. Note that inheritance of operations and attributes across the boundaries of different modules is also allowed.  
**Notes** CORBA-defined.  
**See Also** ["Container::contents\(\)"](#)

## Contained::describe()

**Synopsis** `Description describe();`  
**Description** Returns a structure of type `Contained::Description`:  

```
// IDL
struct Description {
    DefinitionKind kind
    any value;
};
```

This is a generic form of description used as a wrapper for another structure stored in the `value` field. Depending on the type of the `Contained` object, the `value` field contains a corresponding description structure:

```
ConstantDescription
ExceptionDescription
AttributeDescription
OperationDescription
ModuleDescription
InterfaceDescription
TypeDescription
```

The last of these, `TypeDescription` is used for objects of type `StructDef`, `UnionDef`, `EnumDef`, and `AliasDef` (it is associated with interface `TypedefDef` from which these four listed interfaces inherit).

The `kind` field contains the same value as the `def_kind` attribute that `Contained` inherits from `IRObject`.

**Notes** CORBA-defined.  
**See Also** ["Container::describe\\_contents\(\)"](#)

## Contained::id

**Synopsis** `attribute RepositoryId id;`

**Description** A `RepositoryId` provides an alternative method of naming an object which is independent of the `ScopedName`. To be CORBA-compliant, you should follow the naming conventions specified for CORBA `RepositoryIds`. Changing the `id` attribute changes the global identity of the contained object. It is an error to change the `id` to a value that currently exists in the contained object's `Repository`.

**Notes** CORBA-defined.

## Contained::move ()

**Synopsis** `void move(in Container new_container,  
in Identifier new_name, in VersionSpec new_version);`

**Description** Removes this object from its container, and adds it to the container specified by `new_container`. The new container must:

- Be in the same repository.
- Be capable of containing an object of this type.
- Not contain an object of the same name (unless multiple versions are supported).

The `name` attribute of the object being moved is changed to that specified by the `new_name` parameter. The `version` attribute is changed to that specified by the `new_version` parameter.

**Notes** CORBA-defined.

**See Also** ["CORBA::Container"](#)

## Contained::name

**Synopsis** `attribute Identifier name;`

**Description** The name of the object within its scope. For example, in the following definition:

```
// IDL
interface Example {
    void op();
};
```

the names are `Example` and `op`. A `name` must be unique within its scope but is not necessarily unique within an `Interface Repository`. You can change the `name` attribute, however, it is an error to change it to a value that is currently in use within the object's `Container`.

**Notes** CORBA-defined.

**See Also** ["Contained::id"](#)

## Contained::version

### Synopsis

attribute VersionSpec version;

### Description

The version number for this object. Each interface object is identified by a version that distinguishes it from other objects of the same name.

### Notes

CORBA-defined.

# CORBA::Container

## Synopsis

Interface Container describes objects that can contain other objects. Such objects are:

- Repository
- ModuleDef
- InterfaceDef

## CORBA

```
// IDL
// In module CORBA.
typedef sequence <Contained> ContainerSeq;

interface Container : IRObject {

    Contained lookup(in ScopedName search_name);

    ContainedSeq contents(
        in DefinitionKind limit_type,
        in boolean exclude_inherited);

    ContainedSeq lookup_name(
        in Identifier search_name,
        in long levels_to_search,
        in DefinitionKind limit_type,
        in boolean exclude_inherited);

    struct Description {
        Contained contained_object;
        DefinitionKind kind;
        any value;
    };

    typedef sequence<Description> DescriptionSeq;

    DescriptionSeq describe_contents(
        in DefinitionKind limit_type,
        in boolean exclude_inherited,
        in long max_returned_objs);

    ModuleDef create_module(
        in RepositoryId id,
        in Identifier name,
        in VersionSpec version
    );

    ConstantDef create_constant(
        in RepositoryId id,

        in Identifier name,
        in VersionSpec version,
        in IDLType type,
        in any value);

    StructDef create_struct(
        in RepositoryId id,
        in Identifier name,
        in VersionSpec version,
        in StructMemberSeq members);
```



```

UnionDef create_union(
    in RepositoryId id,
    in Identifier name,
    in VersionSpec version,
    in IDLType discriminator_type,
    in UnionMemberSeq members);

EnumDef create_enum(
    in RepositoryId id,
    in Identifier name,
    in VersionSpec version,
    in EnumMemberSeq members);

AliasDef create_alias(
    in RepositoryId id,
    in Identifier name,
    in VersionSpec version,
    in IDLType original_type);
InterfaceDef create_interface(
    in RepositoryId id,
    in Identifier name,
    in VersionSpec version,
    in InterfaceDefSeq base_interfaces);

ExceptionDef create_exception(
    in RepositoryId id;
    in Identifier name,
    in VersionSpec version,
    in StructMemberSeq members);
};

```

**Notes** CORBA-defined.

**See Also** ["CORBA::IObject"](#)

## Container::contents()

**Synopsis** ContainedSeq contents(in DefinitionKind limit\_type,  
in boolean exclude\_inherited);

**Description** Returns a sequence of Contained objects that are directly contained in (defined in or inherited into) the target object. You can use this operation to navigate through the hierarchy of definitions—starting, for example, at a Repository.

### Parameters

limit_type	If set to dk_all, all of the contained Interface Repository objects are returned. If set to the DefinitionKind for a specific interface type, it returns only interfaces of that type. For example, if set to, dk_Operation, it returns contained operations only.
exclude_inherited	Applies only to interfaces. If set to TRUE, inherited objects are not returned. If set to FALSE, objects are returned even if they are inherited.

**Notes** CORBA-defined.

**See Also** ["Container::describe\\_contents\(\)"](#)

## Container::create\_alias()

### Synopsis

```
UnionDef create_alias(in RepositoryId id,  
                     in Identifier name,  
                     in VersionSpec version,  
                     in IDLType original_type);
```

### Description

Creates a new `AliasDef` object within the target `Container`. The `defined_in` attribute is set to the target `Container`. The `containing_repository` attribute is set to the `Repository` in which the new `AliasDef` object is defined.

### Parameters

<code>id</code>	The <code>RepositoryId</code> for the new <code>AliasDef</code> object. An error is returned if an <code>Interface Repository</code> object with the same <code>id</code> already exists within the object's <code>Repository</code> .
<code>name</code>	The name for the new <code>AliasDef</code> object. It is an error to specify a name that already exists within the object's <code>Container</code> when multiple versions are not supported.
<code>version</code>	A version for the new <code>AliasDef</code> .
<code>original_type</code>	The original type that is being aliased.

### Notes

CORBA-defined.

### See Also

["CORBA::AliasDef"](#)

## Container::create\_constant()

### Synopsis

```
ConstantDef create_constant(in RepositoryId id,  
                           in Identifier name,  
                           in VersionSpec version,  
                           in IDLType type,  
                           in any value);
```

### Description

Creates a `ConstantDef` object within the target `Container`. The `defined_in` attribute is set to the target `Container`. The `containing_repository` attribute is set to the `Repository` in which the new `ConstantDef` object is defined.

### Parameters

<code>id</code>	The <code>RepositoryId</code> of the new <code>ConstantDef</code> object. It is an error to specify an <code>id</code> that already exists within the object's <code>Repository</code> .
<code>name</code>	The name of the new <code>ConstantDef</code> object. It is an error to specify a name that already exists within the object's <code>Container</code> when multiple versions are not supported.
<code>version</code>	The version number of the new <code>ConstantDef</code> object.
<code>type</code>	The type of the defined constant. This must be one of the simple types ( <code>long</code> , <code>short</code> , <code>ulong</code> , <code>ushort</code> , <code>float</code> , <code>double</code> , <code>char</code> , <code>string</code> , <code>boolean</code> ).
<code>value</code>	The value of the defined constant.

### Notes

CORBA-defined.

### See Also

["CORBA::ConstantDef"](#)

## Container::create\_enum()

### Synopsis

```
EnumDef create_enum(in RepositoryId id,  
                   in Identifier name,  
                   in VersionSpec version,  
                   in EnumMemberSeq members);
```

### Description

Creates a new `EnumDef` object within the target `Container`. The `defined_in` attribute is set to `Container`. The `containing_repository` attribute is set to the `Repository` in which the new `EnumDef` object is defined.

### Parameters

<code>id</code>	The <code>RepositoryId</code> of the new <code>EnumDef</code> object. It is an error to specify an <code>id</code> that already exists within the <code>Repository</code> .
<code>name</code>	The name of the <code>EnumDef</code> object. It is an error to specify a name that already exists within the object's <code>Container</code> when multiple versions are not supported.
<code>version</code>	The version number of the new <code>EnumDef</code> object.
<code>members</code>	A sequence of <code>EnumMember</code> structures that describe each member of the new <code>EnumDef</code> object.

### Notes

CORBA-defined.

### See Also

["CORBA::EnumDef"](#)

## Container::create\_exception()

### Synopsis

```
ExceptionDef create_exception(in RepositoryId id,  
                             in Identifier name,  
                             in VersionSpec version,  
                             in StructMemberSeq members);
```

### Description

Creates a new `ExceptionDef` object within the target `Container`. The `defined_in` attribute is set to the target `Container`. The `containing_repository` attribute is set to the `Repository` in which the new `ExceptionDef` object is defined. The `type` attribute of the `StructMember` structures is ignored and should be set to `_tc_void`.

### Parameters

<code>id</code>	The <code>RepositoryId</code> of the new <code>ExceptionDef</code> object. It is an error to specify an <code>id</code> that already exists within the object's <code>Repository</code> .
<code>name</code>	The name of the new <code>ExceptionDef</code> object. It is an error to specify a name that already exists within the object's <code>Container</code> when multiple versions are not supported.
<code>version</code>	A version number for the new <code>ExceptionDef</code> object.
<code>members</code>	A sequence of <code>StructMember</code> structures that describe each member of the new <code>ExceptionDef</code> object.

### Notes

CORBA-defined.

### See Also

["CORBA::ExceptionDef"](#) on page 271

## Container::create\_interface()

### Synopsis

```
InterfaceDef create_interface(  
    in RepositoryId id,  
    in Identifier name,  
    in VersionSpec version,  
    in InterfaceDefSeq base_interfaces);
```

### Description

Creates a new empty `InterfaceDef` object within the target `Container`. The `defined_in` attribute is set to the target `Container`. The `containing_repository` attribute is set to the `Repository` in which the new `InterfaceDef` object is defined.

### Parameters

<code>id</code>	The <code>RepositoryId</code> of the new <code>InterfaceDef</code> object. It is an error to specify an <code>id</code> that already exists within the object's <code>Repository</code> .
<code>name</code>	The name of the new <code>InterfaceDef</code> object. It is an error to specify a name that already exists within the object's <code>Container</code> when multiple versions are not supported.
<code>version</code>	A version for the new <code>InterfaceDef</code> object.
<code>base_interfaces</code>	A sequence of <code>InterfaceDef</code> objects from which the new interface inherits.

### Notes

CORBA-defined.

### See Also

["CORBA::InterfaceDef"](#)

## Container::create\_module()

### Synopsis

```
ModuleDef create_module(  
    in RepositoryId id,  
    in Identifier name,  
    in VersionSpec version);
```

### Description

Creates an empty `ModuleDef` object within the target `Container`. The `defined_in` attribute is set to `Container`. The `containing_repository` attribute is set to the `Repository` in which the newly-created `ModuleDef` object is defined.

### Parameters

<code>id</code>	The <code>RepositoryId</code> of the new <code>ModuleDef</code> object. It is an error to specify an <code>id</code> that already exists within the object's <code>Repository</code> .
<code>name</code>	The name of the new <code>ModuleDef</code> object. It is an error to specify a name that already exists within the object's <code>Container</code> when multiple versions are not supported.
<code>version</code>	A version for the <code>ModuleDef</code> object to be created.

### Notes

CORBA-defined.

## Container::create\_struct()

### Synopsis

```
StructDef create_struct(  
    in RepositoryId id,  
    in Identifier name,  
    in VersionSpec version,  
    in StructMemberSeq members);
```

### Description

Creates a new `StructDef` object within the target `Container`. The `defined_in` attribute is set to the target `Container`. The `containing_repository` attribute is set to the `Repository` in which the new `StructDef` object is defined. The `type` attribute of the `StructMember` structures is ignored and should be set to `_tc_void`.

### Parameters

<code>id</code>	The <code>RepositoryId</code> of the new <code>StructDef</code> object. It is an error to specify an <code>id</code> that already exists within the object's <code>Repository</code> .
<code>name</code>	The name of the new <code>StructDef</code> object. It is an error to specify a name that already exists within the object's <code>Container</code> when multiple versions are not supported.
<code>version</code>	A version for the new <code>StructDef</code> object.
<code>members</code>	A sequence of <code>StructMember</code> structures that describe each member of the new <code>StructDef</code> object.

### Notes

CORBA-defined.

### See Also

["CORBA::StructDef"](#)

## Container::create\_union()

### Synopsis

```
UnionDef create_union(  
    in RepositoryId id,  
    in Identifier name,  
    in VersionSpec version,  
    in IDLType discriminator_type,  
    in UnionMemberSeq members);
```

### Description

Creates a new `UnionDef` object within the target `Container`. The `defined_in` attribute is set to the target `Container`. The `Containing_repository` attribute is set to the `Repository` in which the new `UnionDef` object is defined. The `type` attribute of the `UnionMember` structures is ignored and should be set to `_tc_void`.

### Parameters

<code>id</code>	The <code>RepositoryId</code> of the new <code>UnionDef</code> object. It is an error to specify an <code>id</code> that already exists within the object's <code>Repository</code> .
<code>name</code>	The name of the new <code>UnionDef</code> object. It is an error to specify a name that already exists within the object's <code>Container</code> when multiple versions are not supported.
<code>version</code>	A version for the new <code>UnionDef</code> object.
<code>discriminator_type</code>	The type of the <code>Union</code> discriminator.

members                      A sequence of `UnionMember` structures that describe each member of the new `UnionDef` object.

**Notes**                      CORBA-defined.

**See Also**                    ["CORBA::UnionDef"](#)

## Container::describe\_contents()

**Synopsis**

```
DescriptionSeq describe_contents(
    in DefinitionKind limit_type,
    in boolean exclude_inherited,
    in long max_returned_objs);
```

**Description**                A combination of the operation `Contained::describe()` and the operation `Container::contents()`, the `describe_contents()` operation returns a sequence of structures of type `Container::Description`:

```
// IDL
struct Description {
    Contained contained_object;
    DefinitionKind kind;
    any value;
};
```

Each of these structures gives the object reference of a contained object, together with its `kind` and `value`.

**Notes**                      CORBA-defined.

**See Also**                    ["Container::contents\(\)"](#)  
["Contained::describe\(\)"](#)

## Container::lookup()

**Synopsis**

```
Contained lookup(in ScopedName search_name);
```

**Description**                Locates an object name within the target container. The objects can be directly or indirectly defined in or inherited into the target container.

### Parameters

`search_name`                The name of the object to search for relative to the target container. If a relative name is given, the object is looked up relative to the target container. If `search_name` is an absolute scoped name (prefixed by `'::'`), the object is located relative to the containing Repository.

**Notes**                      CORBA-defined.

**See Also**                    ["Container::lookup\\_name\(\)" on page 269](#)

## Container::lookup\_name()

### Synopsis

```
ContainedSeq lookup_name(  
    in Identifier search_name,  
    in long levels_to_search,  
    in DefinitionKind limit_type,  
    in boolean exclude_inherited);
```

### Description

Locates an object or objects by name within the target container. The named objects can be directly or indirectly defined in or inherited into the target container.

### Parameters

<code>search_name</code>	The simple name of the object to search for. The use of the wildcard character "*" is also allowed (Orbis specific).
<code>levels_to_search</code>	Defines whether the search is confined to the current object or should include all Interface Repository objects contained by the object. If set to -1, the current object and all contained Interface Repository objects are searched. If set to 1, only the current object is searched.
<code>limit_type</code>	If this is set to <code>dk_all</code> , all the contained Interface Repository objects are returned. If set to the <code>DefinitionKind</code> for a particular Interface Repository kind, it returns only objects of that kind. For example, if set to <code>dk_Operation</code> , it returns contained operations only.
<code>exclude_inherited</code>	Applies only to interfaces. If set to <code>TRUE</code> , inherited objects are not returned. If set to <code>FALSE</code> , objects are returned even if they are inherited.

### Return Value

Returns a sequence of contained objects; more than one object, having the same simple name can exist within a nested scope structure.

### Notes

CORBA-defined.

# CORBA::EnumDef

**Synopsis** Interface EnumDef describes an IDL enumeration definition.

```
CORBA // IDL
// In module CORBA.

typedef sequence <Identifier> EnumMemberSeq;

interface EnumDef : TypedefDef {
    attribute EnumMemberSeq members;
};
```

**Notes** CORBA-defined.

**See Also** ["CORBA::TypedefDef"](#)

## EnumDef::describe()

**Synopsis** Description describe();

**Description** Inherited from Contained, the describe() operation returns a structure of type Contained::Description:

```
struct Description {
    DefinitionKind kind;
    any value;
};
```

The DefinitionKind for the kind member is dk\_Enum. The value member is an any whose TypeCode is \_tc\_TypeDescription and whose value is a structure of type TypeDescription:

```
// IDL
struct TypeDescription {
    Identifier name;
    RepositoryId id;
    RepositoryId defined_in;
    VersionSpec version;
    TypeCode type;
};
```

The type field of the struct gives the TypeCode of the defined Enum.

**See Also** ["TypedefDef::describe\(\)"](#)

## EnumDef::members

**Synopsis** attribute EnumMemberSeq members;

**Description** Contains the enumeration's list of identifiers (its enumerated constants).

You can change the set of enumerated constants by changing this attribute.

**Notes** CORBA-defined.



# CORBA::ExceptionDef

**Synopsis** Interface `ExceptionDef` describes an IDL exception. It inherits from interface `Contained`.

**CORBA**

```
// IDL
// In module CORBA.
struct StructMember {
    Identifier name;
    TypeCode type;
    IDLType type_def;
};
typedef sequence <StructMember> StructMemberSeq;

interface ExceptionDef : Contained {
    readonly attribute TypeCode type;
    attribute StructMemberSeq members;
};
```

**Notes** CORBA-defined.

**See Also** ["CORBA::Contained"](#)

## ExceptionDef::describe()

**Synopsis** Description `describe()`;

**Description** Inherited from `Contained`, the `describe()` operation returns a structure of type `Contained::Description`:

```
struct Description {
    DefinitionKind kind;
    any value;
};
```

The `DefinitionKind` for the `kind` member is `dk_Exception`. The `value` member is an `any` whose `TypeCode` is `_tc_ExceptionDescription` and whose value is a structure of type `ExceptionDescription`:

```
// IDL
struct ExceptionDescription {
    Identifier name;
    RepositoryId id;
    RepositoryId defined_in;
    VersionSpec version;
    TypeCode type;
};
```

The `type` field of the struct gives the `TypeCode` of the defined exception.

**Notes** CORBA-defined.

**See Also** ["Contained::describe\(\)"](#)

## ExceptionDef::members

<b>Synopsis</b>	<code>attribute StructMemberSeq members;</code>
<b>Description</b>	<p>In a sequence of <code>StructMember</code> structures, the <code>members</code> attribute describes the exception's members.</p> <p>You can modify the <code>members</code> attribute to change the structure's members. You should set only the <code>name</code> and <code>type_def</code> fields of each <code>StructMember</code>. The <code>type</code> field should be set to <code>_tc_void</code>, and it is set automatically to the <code>TypeCode</code> of the <code>type_def</code> field.</p>
<b>Notes</b>	CORBA-defined.
<b>See Also</b>	<a href="#">"CORBA::StructDef"</a> <a href="#">"ExceptionDef::type"</a>

## ExceptionDef::type

<b>Synopsis</b>	<code>readonly attribute TypeCode type;</code>
<b>Description</b>	<p>The type of the exception (from which the definition of the exception can be understood). The <code>TypeCode</code> kind for an exception is <code>tk_except</code>.</p>
<b>Notes</b>	CORBA-defined.
<b>See Also</b>	<a href="#">"ExceptionDef::members"</a>

# CORBA::IDLType

## Synopsis

The abstract interface `IDLType` describes Interface Repository objects that represent interfaces, type definitions, structures, unions, enumerations, aliases (that is, IDL typedef), primitives, bounded strings, sequences, and array types. Thus, it is a base interface for the following interfaces:

- `ArrayDef`
- `InterfaceDef`
- `PrimitiveDef`
- `StringDef`
- `SequenceDef`
- `TypedefDef`
- `AliasDef`
- `StructDef`
- `UnionDef`
- `EnumDef`

## CORBA

```
// IDL
// In module CORBA.
interface IDLType : IRObject {
    readonly attribute TypeCode type;
};
```

## Notes

CORBA-defined.

## See Also

[“CORBA::IRObject”](#)

## IDLType::type

### Synopsis

```
readonly attribute TypeCode type;
```

### Description

Encodes the type information of an Interface Repository object. Most type information can also be extracted using operations and attributes defined on derived types of `IDLType`.

## Notes

CORBA-defined.

# CORBA::InterfaceDef

## Synopsis

Interface `InterfaceDef` describes an IDL interface definition. It may contain lists of constants, typedefs, exceptions, operations, and attributes and inherits from the interfaces `Container`, `Contained` and `IDLType`.

The `_get_interface()` function on an object reference returns a reference to the `InterfaceDef` object that defines the CORBA object's interface.

## CORBA

```
// IDL
// In module CORBA.

interface InterfaceDef;

typedef sequence <InterfaceDef> InterfaceDefSeq;
typedef sequence <RepositoryId> RepositoryIdSeq;
typedef sequence <OperationDescription> OpDescriptionSeq;
typedef sequence <AttributeDescription> AttrDescriptionSeq;

interface InterfaceDef : Container, Contained, IDLType {
    attribute InterfaceDefSeq base_interfaces;

    boolean is_a (in RepositoryId interface_id);

    struct FullInterfaceDescription {
        Identifier name;
        RepositoryId id;
        RepositoryId defined_in;
        VersionSpec version;
        OpDescriptionSeq operations;
        AttrDescriptionSeq attributes;
        RepositoryIdSeq base_interfaces;
        TypeCode type;
    };

    FullInterfaceDescription describe_interface();

    AttributeDef create_attribute (
        in RepositoryId id,
        in Identifier name,
        in VersionSpec version,
        in IDLType type,
        in AttributeMode mode);
    OperationDef create_operation (
        in RepositoryId id,
        in Identifier name,
        in VersionSpec version,
        in IDLType result,
        in OperationMode mode,
        in ParDescriptionSeq params,
        in ExceptionDefSeq exceptions,
        in ContextIdSeq contexts);
};
```

## Notes

CORBA-defined.

## See Also

["CORBA::Contained"](#)  
["CORBA::Container"](#)

## InterfaceDef::base\_interfaces

**Synopsis** `attribute InterfaceDefSeq base_interfaces;`

**Description** In a sequence of `InterfaceDefs`, the `base_interfaces` attribute lists the interfaces from which this interface inherits.

You can change the inheritance specification of an `InterfaceDef` object by changing its `base_interfaces` attribute. It is an error if the name of any definition contained in the interface conflicts with the name of a definition in any of the base interfaces.

**Notes** CORBA-defined.

## InterfaceDef::create\_attribute()

**Synopsis** `AttributeDef create_attribute(  
in RepositoryId id,  
in Identifier name,  
in VersionSpec version,  
in IDLType type,  
in AttributeMode mode);`

**Description** Creates a new `AttributeDef` within the target `InterfaceDef`. The `defined_in` attribute of the new `AttributeDef` is set to the target `InterfaceDef`.

### Parameters

<code>id</code>	The <code>RepositoryId</code> of the new attribute. It is an error to specify an <code>id</code> that already exists within the target object's <code>Repository</code> .
<code>name</code>	The name of the attribute. It is an error to specify a name that already exists within this <code>InterfaceDef</code> .
<code>version</code>	A version for this attribute.
<code>type</code>	The <code>IDLType</code> for this attribute.
<code>mode</code>	Specifies whether the attribute is read-only ( <code>ATTR_READONLY</code> ) or read/write ( <code>ATTR_NORMAL</code> ).

**Notes** CORBA-defined.

**See Also** ["CORBA::AttributeDef"](#)

## InterfaceDef::create\_operation()

### Synopsis

```
OperationDef create_operation(  
    in RepositoryId id,  
    in Identifier name,  
    in VersionSpec version,  
    in IDLType result,  
    in OperationMode mode,  
    in ParDescriptionSeq params,  
    in ExceptionDefSeq exceptions,  
    in ContextIdSeq contexts);
```

### Description

Creates a new `OperationDef` within the target `InterfaceDef`. The `defined_in` attribute of the new `OperationDef` is set to the target `InterfaceDef`.

### Parameters

<code>id</code>	The <code>RepositoryId</code> of the new attribute. It is an error to specify an <code>id</code> that already exists within the target object's <code>Repository</code> .
<code>name</code>	The name of the attribute. It is an error to specify a name that already exists within this <code>InterfaceDef</code> .
<code>version</code>	A version number for this operation.
<code>result</code>	The return type for this operation.
<code>mode</code>	Specifies whether this operation is normal ( <code>OP_NORMAL</code> ) or oneway ( <code>OP_ONEWAY</code> ).
<code>params</code>	A sequence of <code>ParameterDescription</code> structures which describe the parameters to this operation.
<code>exceptions</code>	A sequence of <code>ExceptionDef</code> objects that describe the exceptions this operation can raise.
<code>contexts</code>	A sequence of context identifiers for this operation.

### Notes

CORBA-defined.

### See Also

["CORBA::OperationDef"](#)  
["CORBA::ExceptionDef"](#)

## InterfaceDef::describe()

### Synopsis

```
Description describe();
```

### Description

Inherited from `Contained`, the `describe()` operation returns a structure of type `Contained::Description`:

```
struct Description {
    DefinitionKind kind;
    any value;
};
```

The `DefinitionKind` for the `kind` member is `dk_Interface`. The value member is an any whose `TypeCode` is `_tc_InterfaceDescription` and whose value is a structure of type `InterfaceDescription`:

```
// IDL
struct InterfaceDescription {
    Identifier name;
    RepositoryId id;
    RepositoryId defined_in;
    VersionSpec version;
    RepositoryIdSeq base_interfaces;
};
```

### Notes

CORBA-defined.

### See Also

["Contained::describe\(\)"](#)

## InterfaceDef::describe\_interface ()

### Synopsis

```
FullInterfaceDescription describe_interface();
```

### Description

Returns a description of the interface, including its operations, attributes, and base interfaces in a structure of type `FullInterfaceDescription`:

```
struct FullInterfaceDescription {
    Identifier name;
    RepositoryId id;
    RepositoryId defined_in;
    VersionSpec version;
    OpDescriptionSeq operations;
    AttrDescriptionSeq attributes;
    RepositoryIdSeq base_interfaces;
    TypeCode type;
};
```

You can determine details of exceptions and contexts via the returned sequence of `OperationDescription` structures.

### Notes

CORBA-defined.

### See Also

["OperationDef::describe\(\)"](#)  
["AttributeDef::describe\(\)"](#)

## InterfaceDef::is\_a ()

### Synopsis

```
boolean is_a(in RepositoryId interface_id);
```

### Description

Returns `TRUE` if the interface is either identical to or inherits, directly or indirectly, from the `InterfaceDef` object whose `RepositoryId` is passed in the parameter `interface_id`. Otherwise it returns `FALSE`.

### Notes

CORBA-defined.



# CORBA::IObject

**Synopsis** The interface `IObject` provides a base interface from which all Interface Repository interfaces are derived.

```
CORBA // IDL
// In module CORBA.

interface IObject {
    readonly attribute DefinitionKind def_kind;
    void destroy ();
};
```

**Notes** CORBA-defined.

## IObject::def\_kind

**Synopsis** `readonly attribute DefinitionKind def_kind`

**Description** Identifies the kind of an Interface Repository (IFR) object. For example, an `OperationDef` object, describing an IDL operation, has the kind `dk_Operation`.

**Notes** CORBA-defined.

## IObject::destroy()

**Synopsis** `void destroy();`

**Description** Deletes an IFR object. This also deletes any objects contained within the target object. It is an error to invoke the `destroy()` operation on a `Repository` or on a `PrimitiveDef` object.

**Notes** CORBA-defined.

# CORBA::IT\_InterfaceDef

**Synopsis** The interface `IT_InterfaceDef` adds one extra function to the standard `InterfaceDef` object.

**CORBA**

```
// IDL
// In module CORBA.

interface IT_InterfaceDef : InterfaceDef {
    InterfaceDefSeq derived_interfaces();
};
```

**Notes** Orbix-specific.

**See Also** ["CORBA::InterfaceDef"](#)

## IT\_InterfaceDef::derived\_interfaces()

**Synopsis** `InterfaceDefSeq derived_interfaces();`

**Description** Returns a list of all the interfaces that are derived from the given interface. They are returned as a sequence of `InterfaceDef` objects.

**Notes** Orbix-specific.

**See Also** ["InterfaceDef::base\\_interfaces"](#)

# CORBA::IT\_Repository

**Synopsis** The interface `IT_Repository` provides transactional access to the Interface Repository. These operations can be used to help ensure that the repository is left in a consistent state. In the present implementation only one transaction may be active at a time.

**Note:** This transactional access is a feature of the Interface Repository and should not be confused with transactions in OrbixOTS.

**CORBA**

```
// IDL
// In module CORBA.

interface IT_Repository : Repository {
    unsigned long start();
    void commit(in unsigned long transaction_id);
    void rollBack(in unsigned long transaction_id);
    typedef sequence <unsigned long> transactions;
    transactions active_transactions();
};
```

**Notes** Orbix-specific.

**See Also** ["CORBA::Container"](#)  
["CORBA::Repository"](#)

## IT\_Repository::start()

**Synopsis** `unsigned long start();`

**Description** Starts a new transaction.

**Return Value** Returns a transaction identifier for the transaction started or 0 if the transaction could not be started.

**Notes** Orbix-specific.

## IT\_Repository::commit()

**Synopsis** `void commit(in unsigned long transaction_id);`

**Description** Commits a transaction.

**Parameters**

`transaction_id` The identifier for the transaction.

**Notes** Orbix-specific.

**See Also** ["IT\\_Repository::start\(\)"](#)

## IT\_Repository::rollBack()

### Synopsis

```
void rollBack(in unsigned long transaction_id);
```

### Description

Rolls back all changes made during the transaction to the previous commit point.

### Parameters

`transaction_id` The identifier for the transaction to be rolled back.

### Notes

Orbix-specific.

### See Also

["IT\\_Repository::commit\(\)"](#) on page 281

## IT\_Repository::active\_transactions()

### Synopsis

```
transactions active_transactions();
```

### Description

Returns a sequence of active `transaction_ids`. If no transaction is active, an empty sequence is returned.

### Notes

Orbix-specific.

# CORBA::ModuleDef

**Synopsis** The interface `ModuleDef` describes an IDL module. It inherits from the interfaces `Container` and `Contained`.

**CORBA**

```
// IDL
// In module CORBA.
interface ModuleDef : Container, Contained {
};
```

**Notes** CORBA-defined.

**See Also** ["CORBA::Contained"](#)  
["CORBA::Container"](#)

## ModuleDef::describe()

**Synopsis** `Description describe();`

**Description** Inherited from `Contained`, the `describe()` operation returns a structure of type `Contained::Description`:

```
struct Description {
    DefinitionKind kind;
    any value;
};
```

The `DefinitionKind` for the `kind` member is `dk_Module`. The value member is an any whose `TypeCode` is `_tc_ModuleDescription` and whose value is a structure of type `ModuleDescription`:

```
struct ModuleDescription {
    Identifier name;
    RepositoryId id;
    RepositoryId defined_in;
    VersionSpec version;
};
```

**Notes** CORBA-defined.

**See Also** ["Contained::describe\(\)"](#)

# CORBA::OperationDef

**Synopsis** Interface `OperationDef` describes an IDL operation that is defined in an IDL interface.

One use of `OperationDef` is to construct an `NVList` for a specific operation for use in the Dynamic Invocation Interface. See `CORBA::ORB::create_operation_list()` for details.

## CORBA

```
// IDL
// In module CORBA.

enum OperationMode { OP_NORMAL, OP_ONEWAY };

enum ParameterMode { PARAM_IN, PARAM_OUT, PARAM_INOUT };

struct ParameterDescription {
    Identifier name;
    TypeCode type;
    IDLType type_def;
    ParameterMode mode;
};

typedef sequence <ParameterDescription> ParDescriptionSeq;

typedef Identifier ContextIdentifier;
typedef sequence <ContextIdentifier> ContextIdSeq;

typedef sequence <ExceptionDescription> ExcDescriptionSeq;

interface OperationDef : Contained {
    readonly attribute TypeCode result;
    attribute IDLType result_def;
    attribute ParDescriptionSeq params;
    attribute OperationMode mode;
    attribute ContextIdSeq contexts;
    attribute ExceptionDefSeq exceptions;
};
```

**Notes** CORBA-defined.

**See Also** ["CORBA::Contained"](#)  
["CORBA::ExceptionDef"](#)

## OperationDef::contexts

**Synopsis** attribute `ContextIdSeq contexts;`

**Description** The list of context identifiers specified in the context clause of the operation.

**Notes** CORBA-defined.

## OperationDef::exceptions

**Synopsis** attribute `ExceptionDefSeq`

**Description** The list of exceptions that the operation can raise.

**Notes** CORBA-defined.

**See Also** ["CORBA::ExceptionDef"](#)

## OperationDef::describe()

**Synopsis** Description describe();

**Description** Inherited from Contained, the describe() operation returns a structure of type Contained::Description:

```
struct Description {
    DefinitionKind kind;
    any value;
};
```

The DefinitionKind for the kind member is dk\_Operation. The value member is an any whose TypeCode is \_tc\_OperationDescription and whose value is a structure of type OperationDescription:

```
struct OperationDescription {
    Identifier name;
    RepositoryId id;
    RepositoryId defined_in;
    VersionSpec version;
    TypeCode result;
    OperationMode mode;
    ContextIdSeq contexts;
    ParDescriptionSeq parameters;
    ExcDescriptionSeq exceptions;
};
```

**Notes** CORBA-defined.

**See Also** ["Contained::describe\(\)"](#)  
["CORBA::ExceptionDef"](#)

## OperationDef::mode

**Synopsis** attribute OperationMode mode;

**Description** Specifies whether the operation is normal (OP\_NORMAL) or oneway (OP\_ONEWAY). You can set the mode attribute set to OP\_ONEWAY only if the result is \_tc\_void and all parameters have a mode of PARAM\_IN.

**Notes** CORBA-defined.

## OperationDef::params

**Synopsis** `attribute ParDescriptionSeq params;`

**Description** Specifies the parameters for this operation. It is a sequence of structures of type `ParameterDescription`:

```
struct ParameterDescription {
    Identifier name;
    TypeCode type;
    IDLType type_def;
    ParameterMode mode;
};
```

The `name` member provides the name for the parameter. The `type` member identifies the `TypeCode` for the parameter. The `type_def` member identifies the definition of the type for the parameter. The `mode` specifies whether the parameter is an `in` (`PARAM_IN`), an `out` (`PARAM_OUT`) or an `inout` (`PARAM_INOUT`) parameter. The order of the `ParameterDescription`s is significant.

**Notes** CORBA-defined.

**See Also** ["IDLType::type"](#)

## OperationDef::result

**Synopsis** `readonly attribute TypeCode result;`

**Description** The return type of this operation. The attribute `result_def` contains the same information.

**Notes** CORBA-defined.

**See Also** ["OperationDef::result\\_def"](#)

## OperationDef::result\_def

**Synopsis** `attribute IDLType result_def;`

**Description** Describes the return type for this operation. The attribute `result` contains the same information.

Setting the `result_def` attribute also updates the `result` attribute.

**Notes** CORBA-defined.

**See Also** ["IDLType::type"](#)  
["OperationDef::result"](#)



# CORBA::PrimitiveDef

## Synopsis

Interface `PrimitiveDef` represents a primitive type such as `short` or `long`. `PrimitiveDef` objects are anonymous (unnamed) and owned by the Repository.

You cannot create objects of type `PrimitiveDef`. When needed, you can obtain a reference to a `PrimitiveDef` through a call to the operation `CORBA::Repository::get_primitive()`.

```
// IDL
// In module CORBA.

enum PrimitiveKind {
    pk_null, pk_void, pk_short, pk_long, pk_ushort, pk_ulong,
    pk_float, pk_double, pk_boolean, pk_char, pk_octet,
    pk_any, pk_TypeCode, pk_Principal, pk_string, pk_objref
};

interface PrimitiveDef : IDLType {
    readonly attribute PrimitiveKind kind;
};
```

## Notes

CORBA-defined.

## See Also

[“IDLType::type”](#)

## PrimitiveDef::kind

### Synopsis

```
readonly attribute PrimitiveKind kind;
```

### Description

Identifies which of the primitive types is represented by this `PrimitiveDef`. A `PrimitiveDef` with a kind of type `pk_string` represents an unbounded string, a bounded string is represented by the interface `StringDef`. A `PrimitiveDef` with a kind of type `pk_objref` represents the IDL type `Object`.

## Notes

CORBA-defined.

## See Also

[“IDLType::type”](#)  
[“CORBA::StringDef”](#)

# CORBA::Repository

## Synopsis

The Interface Repository itself is a container for IDL type definitions. Its interface is described by the `Repository` interface. It can be used to look up any definition, by either name or identity, that is defined in the global name space or within an interface or module.

## CORBA

```
// IDL
// In module CORBA.

interface Repository : Container {
    Contained lookup_id(
        in RepositoryId search_id);

    PrimitiveDef get_primitive (in PrimitiveKind kind);

    StringDef create_string (in unsigned long bound);

    SequenceDef create_sequence (
        in unsigned long bound,
        in IDLType element_type);

    ArrayDef create_array (
        in unsigned long length,
        in IDLType element_type);
};
```

## Notes

CORBA-defined.

## See Also

["CORBA::Container"](#)

## Repository::create\_array()

### Synopsis

```
ArrayDef create_array(in unsigned long length,
    in IDLType element_type);
```

### Description

Returns a new array object defining an anonymous (unnamed) type. The new array object must be used in the definition of exactly one other object; it is deleted when the object it is contained in is deleted. It is the application's responsibility to delete any anonymous type object it creates if subsequently that object is not successfully used in the definition of a `Contained` object.

### Parameters

`length`            The number of elements in the array.  
`element_type`    The type of element that the array contains.

## Notes

CORBA-defined.

## See Also

["CORBA::ArrayDef"](#)  
["CORBA::IObject"](#)

## Repository::create\_sequence()

**Synopsis** `SequenceDef create_sequence (in unsigned long bound,  
in IDLType element_type);`

**Description** Returns a new sequence object defining an anonymous (unnamed) type. The new sequence object must be used in the definition of exactly one other object; it is deleted when the object it is contained in is deleted. It is the application's responsibility to delete any anonymous type object it creates if subsequently that object is not successfully used in the definition of a Contained object.

### Parameters

`bound` The number of elements in the sequence. A bound of 0 indicates an unbounded sequence.  
`element_type` The type of element that the sequence contains.

**Notes** CORBA-defined.

**See Also** ["CORBA::SequenceDef"](#)

## Repository::create\_string()

**Synopsis** `StringDef create_string (in unsigned long bound);`

**Description** Returns a new string object defining an anonymous (unnamed) type. The new string object must be used in the definition of exactly one other object; it is deleted when the object it is contained in is deleted. It is the application's responsibility to delete any anonymous type object it creates if subsequently that object is not successfully used in the definition of a Contained object.

### Parameters

`bound` The maximum number of characters in the string. This cannot be 0.

**Notes** CORBA-defined.

**See Also** ["CORBA::StringDef"](#)

## Repository::get\_primitive()

**Synopsis** `PrimitiveDef get_primitive(in PrimitiveKind kind);`

**Description** Returns a reference to a PrimitiveDef of the specified PrimitiveKind. All PrimitiveDefs are owned by the Repository, one primitive object per primitive type (for example, short, long, unsigned short, unsigned long and so on).

**Notes** CORBA-defined.

**See Also** ["CORBA::PrimitiveDef"](#)

## Repository::describe\_contents()

### Synopsis

```
sequence<Description> describe_contents (
    in InterfaceName restrict_type,
    in boolean    exclude_inherited,
    in long    max_returned_objs);
```

### Description

The operation `describe_contents()` is inherited from interface `Container`. It returns a sequence of `Container::Description` structures; one such structure for each top level item in the repository. The structure is defined as:

```
// IDL
struct Description {
    Contained contained_object;
    DefinitionKind kind;
    any value;
};
```

Each structure has the following members:

<code>contained_object</code>	The object reference, of type <code>Contained</code> , of the contained top level object. You can call <code>describe()</code> an object reference of type <code>Contained</code> , to get further information on a top level object in the <code>Repository</code> .
<code>kind</code>	The kind of the object being described.
<code>value</code>	An any that may contain one of the following structS: <ul style="list-style-type: none"><li>• <code>ModuleDescription</code></li><li>• <code>ConstantDescription</code></li><li>• <code>TypeDescription</code></li><li>• <code>ExceptionDescription</code></li><li>• <code>AttributeDescription</code></li><li>• <code>ParameterDescription</code></li><li>• <code>OperationDescription</code></li><li>• <code>InterfaceDescription</code></li></ul>

### Notes

CORBA-defined.

### See Also

["Container::describe\\_contents\(\)"](#)

## Repository::lookup\_id()

### Synopsis

```
Contained lookup_id(in RepositoryId search_id);
```

### Description

Returns an object contained within the `Repository` given its `RepositoryId`.

### Notes

CORBA-defined.

### See Also

["CORBA::Contained"](#)

# CORBA::SequenceDef

**Synopsis** Interface SequenceDef represents an IDL sequence definition. It inherits from the interface IDLType.

```
CORBA // IDL
// In module CORBA.
interface SequenceDef : IDLType {
    attribute unsigned long bound;
    readonly attribute TypeCode element_type;
    attribute IDLType element_type_def;
};
```

**Notes** CORBA-defined.

**See Also** ["CORBA::IDLType"](#)  
["Repository::create\\_sequence\(\)"](#)

## SequenceDef::bound

**Synopsis** attribute unsigned long bound;

**Description** The bound of the sequence. A bound of 0 indicates an unbounded sequence type.

Changing the bound attribute also updates the inherited type attribute.

**Notes** CORBA-defined.

**See Also** ["SequenceDef::type"](#)

## SequenceDef::element\_type

**Synopsis** readonly attribute TypeCode element\_type;

**Description** Describes the type of the element contained within this sequence. The attribute element\_type\_def contains the same information.

**Notes** CORBA-defined.

**See Also** ["SequenceDef::element\\_type\\_def"](#)

## SequenceDef::element\_type\_def

**Synopsis** attribute IDLType element\_type\_def;

**Description** Describes the type of element contained within this sequence. The attribute element\_type contains the same information. Setting the element\_type\_def attribute also updates the element\_type and IDLType::type attributes.

**Notes** CORBA-defined.

**See Also** ["SequenceDef::element\\_type"](#)  
["IDLType::type"](#)

## SequenceDef::type

### Synopsis

readonly attribute TypeCode type;

### Description

The `type` attribute is inherited from interface `IDLType`. This attribute is a `tk_sequence` `TypeCode` that describes the sequence. It is updated automatically whenever the attributes `bound` or `element_type_def` are changed.

### Notes

CORBA-defined.

### See Also

["SequenceDef::element\\_type"](#)  
["SequenceDef::bound"](#)

# CORBA::StructDef

**Synopsis** Interface StructDef describes an IDL structure.

**CORBA**

```
// IDL
// In module CORBA.
struct StructMember {
    Identifier name;
    TypeCode type;
    IDLType type_def;
};

typedef sequence<StructMember> StructMemberSeq;

interface StructDef : TypedefDef {
    attribute StructMemberSeq members;
};
```

**Notes** CORBA-defined.

**See Also** ["CORBA::Contained"](#)  
["Container::create\\_struct\(\)"](#)

## StructDef::describe()

**Synopsis** Description describe();

**Description** Inherited from Contained, the describe() operation returns a structure of type Contained::Description:

```
struct Description {
    DefinitionKind kind;
    any value;
};
```

The DefinitionKind for the kind member is dk\_Struct. The value member is an any whose TypeCode is \_tc\_TypeDescription and whose value is a structure of type TypeDescription:

```
// IDL
// In module CORBA.
struct TypeDescription {
    Identifier name;
    RepositoryId id;
    RepositoryId defined_in;
    VersionSpec version;
    TypeCode type;
};
```

**Notes** CORBA-defined.

**See Also** ["TypedefDef::describe\(\)"](#)

## StructDef::members

### Synopsis

attribute StructMemberSeq members;

### Description

Describes the members of the structure.

You can modify this attribute to change the members of a structure. Only the `name` and `type_def` fields of each `StructMember` should be set (the `type` field should be set to `_tc_void` and it is set automatically to the `TypeCode` of the `type_def` field).

### Notes

CORBA-defined.

### See Also

["CORBA::TypedefDef"](#)



# CORBA::StringDef

**Synopsis** Interface `StringDef` represents a bounded string type. Unbounded strings are primitive types. A `StringDef` object is anonymous, that is, unnamed.

**CORBA**

```
// IDL
// In module CORBA.
interface StringDef : IDLType {
    attribute unsigned long bound;
};
```

**Notes** CORBA-defined.

**See Also** ["CORBA::IDLType"](#)  
["CORBA::PrimitiveDef"](#)  
["Repository::create\\_string\(\)"](#)

## StringDef::bound

**Synopsis** `attribute unsigned long bound;`

**Description** Specifies the bound of the string. This cannot be zero.

**Notes** CORBA-defined.

# CORBA::TypedefDef

## Synopsis

The abstract interface `TypedefDef` is inherited by all Interface Repository interfaces (except `InterfaceDef`) that define named types. Named types are types for which a name must appear in their definition such as structures, unions, enumerations and aliases.

Anonymous types (`PrimitiveDef`, `StringDef`, `SequenceDef` and `ArrayDef`) do not inherit from `TypedefDef`.

The role of interface `TypedefDef` in the Interface Repository is not of particular importance; it is merely a base interface for `StructDef`, `UnionDef`, `EnumDef` and `AliasDef`.

## CORBA

```
// IDL
// In module CORBA.

interface TypedefDef : Contained, IDLType {
};
```

## Notes

CORBA-defined.

## See Also

["CORBA::Contained"](#)

## TypedefDef::describe()

### Synopsis

Description `describe()`;

### Description

Inherited from `Contained`, the `describe()` operation returns a structure of type `Contained::Description`:

```
struct Description {
    DefinitionKind kind;
    any value;
};
```

The `DefinitionKind` for the `kind` member is `dk_Typedef`. The value member is an any whose `TypeCode` is `_tc_TypeDescription` and whose value is a structure of type `TypeDescription`:

```
// IDL
// In module CORBA.
struct TypeDescription {
    Identifier name;
    RepositoryId id;
    RepositoryId defined_in;
    VersionSpec version;
    TypeCode type;
};
```

## Notes

CORBA-defined.

## See Also

["Contained::describe\(\)"](#)

# CORBA::UnionDef

**Synopsis** Interface UnionDef represents an IDL union.

**CORBA**

```
// IDL
// In module CORBA.

struct UnionMember {
    Identifier name;
    any label;
    TypeCode type;
    IDLType type_def;
};

typedef sequence <UnionMember> UnionMemberSeq;

interface UnionDef : TypedefDef {
    readonly attribute TypeCode discriminator_type;
    attribute IDLType discriminator_type_def;
    attribute UnionMemberSeq members;
};
```

**Notes** CORBA-defined.

**See Also** ["CORBA::Contained"](#)  
["CORBA::TypedefDef"](#)  
["Container::create\\_union\(\)"](#)

## UnionDef::describe()

**Synopsis** Description describe();

**Description** Inherited from Contained, the describe() operation returns a structure of type Contained::Description:

```
struct Description {
    DefinitionKind kind;
    any value;
};
```

The DefinitionKind for the kind member is dk\_Union. The value member is an any whose TypeCode is \_tc\_TypeDescription and whose value is a structure of type TypeDescription:

```
// IDL
struct TypeDescription {
    Identifier name;
    RepositoryId id;
    RepositoryId defined_in;
    VersionSpec version;
    TypeCode type;
};
```

**Notes** CORBA-defined.

**See Also** ["TypedefDef::describe\(\)"](#)

## UnionDef::discriminator\_type

<b>Synopsis</b>	<code>readonly attribute TypeCode discriminator_type;</code>
<b>Description</b>	Describes the discriminator type for this union. For example, if the union currently contains a long, the <code>discriminator_type</code> is <code>_tc_long</code> . The attribute <code>discriminator_type_def</code> contains the same information.
<b>Notes</b>	CORBA-defined.

## UnionDef::discriminator\_type\_def

<b>Synopsis</b>	<code>attribute IDLType discriminator_type_def;</code>
<b>Description</b>	Describes the discriminator type for this union. The attribute <code>discriminator_type</code> contains the same information. Changing this attribute automatically updates the <code>discriminator_type</code> attribute and the <code>IDLType::type</code> attribute.
<b>Notes</b>	CORBA-defined.
<b>See Also</b>	<a href="#">"IDLType::type"</a> <a href="#">"UnionDef::discriminator_type"</a>

## UnionDef::members

<b>Synopsis</b>	<code>attribute UnionMemberSeq members;</code>
<b>Description</b>	Contains a description of each union member: its name, label and type ( <code>type</code> and <code>type_def</code> contain the same information). The <code>members</code> attribute can be modified to change the union's members. You should set only the <code>name</code> , <code>label</code> and <code>type_def</code> fields of each <code>UnionMember</code> . You should set the <code>type</code> field to <code>_tc_void</code> , and it is then set automatically to the <code>TypeCode</code> of the <code>type_def</code> field.
<b>Notes</b>	CORBA-defined.
<b>See Also</b>	<a href="#">"CORBA::TypedefDef"</a>

# Part VI

## IDL Interface to the Orbix Java Daemon

### **In this part**

This part contains the following:

<a href="#">IDL Interface to the Orbix Daemon</a>
---------------------------------------------------

page 291
----------



# IDL Interface to the Orbix Java Daemon

## Synopsis

The Orbix Java daemon is itself an Orbix Java server whose IDL interface is called `IT_daemon`. The Orbix Java daemon is responsible for launching servers (if an appropriate server is not already running) and dispatching operation requests. The daemon is involved, if at all, only with the first operation request from a client—it is not involved with subsequent requests. Two Orbix Java daemon executables are available: `orbixd` and `orbixdj` (the Java daemon).

The Orbix Java daemon is also responsible for managing the Implementation Repository. It accepts requests from the Orbix Java utilities—`putit`, `catit`, `lsit`, and so on. `orbixdj` implements a subset of the `IT_daemon` operations, and these are indicated clearly in the operation descriptions in this section.

## Orbix Java

```
// IDL
interface IT_daemon{
    boolean lookUp(in string service,
                  out stringSeq hostList,
                  in octet hops,
                  in string tag);

    boolean addHostsToServer(in string server,
                            in stringSeq hostList);
    boolean addHostsToGroup(in string group,
                            in stringSeq hostList);
    boolean addGroupsToServer(in string server,
                              in stringSeq groupList);
    boolean delHostsFromServer(in string server,
                              in stringSeq hostList);
    boolean delHostsFromGroup(in string group,
                              in stringSeq hostList);
    boolean delGroupsFromServer(in string server,
                                in stringSeq groupList);

    boolean listHostsInServer(in string server,
                              out stringSeq hostList);
    boolean listHostsInGroup(in string group,
                              out stringSeq hostList);
    boolean listGroupsInServer(in string server,
                               out stringSeq groupList);

    enum LaunchStatus {
        inActive,
        manualLaunch,
        automaticLaunch
    };

    struct serverDetails {
        string server;
        string marker;
        string principal;
        string code;
        string comms;
        string port;
        unsigned long OSspecific;
    };
};
```

```

        LaunchStatus status;
    };

void listActiveServers(out serverDetailsSeq servers);

void killServer(in string name, in string marker);
void newSharedServer(in string serverName,
    in stringSeq marker,
    in stringSeq launchCommand,
    in unsigned long mode_flags);
public void newSharedServer2(String serverName,
    String[] marker,
    String[] launchCommand,
    int mode_flags,
    int nservers,
    int wellKnownPort);

void newUnSharedServer(in string serverName,
    in stringSeq marker,
    in stringSeq launchCommand,
    in unsigned long mode_flags);

void newPerMethodServer(in string serverName,
    in stringSeq method,
    in stringSeq launchCommand);

void listServers(in string subDir,
    out stringSeq servers);

void deleteServer(in string serverName);

boolean serverExists(in string serverName);

public void getIIOPDetails(
    String serverName,
    String markerName,
    String methodName,
    org.omg.CORBA.StringHolder iiopPort,
    org.omg.CORBA.StringHolder
    activationPolicy);

public void getImplementationDetails(
    String serverName,
    String markerName,
    String methodName,
    org.omg.CORBA.StringHolder codeProtocol,
    org.omg.CORBA.StringHolder
    commsProtocol,
    org.omg.CORBA.StringHolder commsPort,
    org.omg.CORBA.StringHolder
    activationPolicy ) ;

void getServer(in string serverName,
    out string commsProtocol,
    out string codeProtocol,
    out string activationPolicy,
    out unsigned long mode_flags,
    out string owner,
    out string invokeList,
    out string launchList,
    out stringSeq markers,

```



```

        out stringSeq methods,
        out stringSeq commands);

public void getServer2(String serverName,
                      org.omg.CORBA.StringHolder
        commsProtocol,
                      org.omg.CORBA.StringHolder
        codeProtocol,
                      org.omg.CORBA.StringHolder
        activationPolicy,
                      org.omg.CORBA.IntHolder mode_flags,
                      org.omg.CORBA.StringHolder owner,
                      org.omg.CORBA.StringHolder
        invokeList,
                      org.omg.CORBA.StringHolder
        launchList,
                      org.omg.CORBA.IntHolder nservers,
                      org.omg.CORBA.IntHolder port,
                      stringSeqHolder markers,
                      stringSeqHolder methods,
                      stringSeqHolder commands) ;

void addUnsharedMarker(in string serverName,
                      in string markerName,
                      in string newCommand);
void removeUnsharedMarker(in string serverName,
                          in string markerName);
void addSharedMarker(in string serverName,
                    in string markerName,
                    in string newCommand);
void removeSharedMarker(in string serverName,
                        in string markerName);

void addMethod(in string serverName,
              in string methodName,
              in string newCommand);
void removeMethod(in string serverName,
                 in string methodName);
void newDirectory(in string dirName);
void deleteDirectory(in string dirName,
                    in boolean deleteChildren);
void changeOwnerServer(in string new_owner,
                      in string serverName);
void addInvokeRights(in string userGroup,
                    in string serverName);

public void registerPersistentServer(
                      String serverName,
                      int serverPid,
                      org.omg.CORBA.StringHolder
        codeProtocol,
                      org.omg.CORBA.StringHolder
        commsProtocol,
                      org.omg.CORBA.StringHolder
        commsPort);
void removeInvokeRights(in string userGroup,
                       in string serverName);
void addLaunchRights(in string userGroup,
                    in string serverName);
void removeLaunchRights(in string userGroup,

```

```

        in string serverName);
void addInvokeRightsDir(in string userGroup,
    in string dirName);
void removeInvokeRightsDir(in string userGroup,
    in string dirName);
void addLaunchRightsDir(in string userGroup,
    in string dirName);
void removeLaunchRightsDir(in string userGroup,
    in string dirName);
};

```

**Notes** Orbix-specific.

## IT\_daemon::addLaunchRightsDir()

**Synopsis** void addLaunchRightsDir (in string userGroup,  
in string dirName);

**Description** Adds the user or group in `userGroup` to the list of owners for the directory `dirName`.

**Notes** Orbix-specific.

## IT\_daemon::addInvokeRights()

**Synopsis** void addInvokeRights(  
in string userGroup,  
in string serverName);

**Description** Adds the user or group in `userGroup` to the *invoke* access control list (ACL) for the server `serverName`. A user who has invoke rights on a server can invoke operations on any object controlled by that server. By default, only the owner of an Implementation Repository entry has invoke rights on the server registered.

**Notes** Orbix-specific.

**See Also** ["IT\\_daemon::removeInvokeRights\(\)" on page 317](#)  
["IT\\_daemon::addLaunchRights\(\)" on page 305](#)  
["IT\\_daemon::addInvokeRightsDir\(\)" on page 305](#)

## IT\_daemon::addInvokeRightsDir()

<b>Synopsis</b>	<pre>void addInvokeRightsDir(     in string userGroup,     in string dirName);</pre>
<b>Description</b>	Adds the user or group in <code>userGroup</code> to the <i>invoke</i> access control list (ACL) for the directory <code>dirName</code> .
<b>Notes</b>	Orbix-specific.
<b>See Also</b>	<a href="#">"IT_daemon::removeInvokeRightsDir()" on page 317</a> <a href="#">"IT_daemon::addInvokeRightsDir()" on page 304</a>

## IT\_daemon::addLaunchRights()

<b>Synopsis</b>	<pre>void addLaunchRights(     in string userGroup,     in string serverName);</pre>
<b>Description</b>	Adds the user or group in <code>userGroup</code> to the <i>launch</i> access control list for the server <code>serverName</code> . By default, only the owner of an Implementation Repository entry has launch rights on the server registered.
<b>Notes</b>	Orbix-specific.
<b>See Also</b>	<a href="#">"IT_daemon::removeLaunchRights()" on page 317</a>

## IT\_daemon::addLaunchRightsDir()

<b>Synopsis</b>	<pre>void addLaunchRightsDir(     in string userGroup,     in string dirName);</pre>
<b>Description</b>	Adds the user or group in <code>userGroup</code> to the <i>launch</i> access control list for the directory <code>dirName</code> .
<b>Notes</b>	Orbix-specific.
<b>See Also</b>	<a href="#">"IT_daemon::addLaunchRights()" on page 305</a> <a href="#">"IT_daemon::removeLaunchRightsDir()" on page 318</a>

## IT\_daemon::addMethod()

<b>Synopsis</b>	<pre>void addMethod(     in string serverName,     in string methodName,     in string newCommand);</pre>
<b>Description</b>	Adds an <i>activation order</i> to the Implementation Repository entry for the per-method server, <code>serverName</code> . This activation order specifies that an invocation of a method whose name matches the method (or method pattern) indicated in <code>methodName</code> should cause the server to be launched using the command <code>newCommand</code> .
<b>Notes</b>	Orbix-specific. Not supported by <code>orbixdj</code> .
<b>See Also</b>	<a href="#">"IT_daemon::removeMethod()" on page 318</a>

## IT\_daemon::addSharedMarker()

### Synopsis

```
void addSharedMarker(  
    in string serverName,  
    in string markerName,  
    in string newCommand);
```

### Description

Adds an *activation order* to the Implementation Repository entry for the shared server, `serverName`. This activation order specifies that an invocation for an object whose marker matches the marker (or marker pattern) indicated in `markerName` should cause the server to be launched (if not already running) using the command, `newCommand`.

### Notes

Orbix-specific.

Not supported by `orbixdj`.

### See Also

["IT\\_daemon::removeSharedMarker\(\)" on page 318](#)

## IT\_daemon::addUnsharedMarker()

### Synopsis

```
void addUnsharedMarker(  
    in string serverName,  
    in string markerName,  
    in string newCommand);
```

### Description

Adds an *activation order* to the Implementation Repository entry for the unshared server, `serverName`. This activation order specifies that an invocation for an object whose marker matches the marker (or marker pattern) indicated in `markerName` should cause the server to be launched using the command, `newCommand`.

### Notes

Orbix-specific.  
Not supported by `orbixdj`.

### See Also

["IT\\_daemon::removeUnsharedMarker\(\)" on page 318](#)

## IT\_daemon::changeOwnerServer()

### Synopsis

```
void changeOwnerServer(  
    in string new_owner,  
    in string serverName);
```

### Description

Changes the ownership of the Implementation Repository entry for the server `serverName`. The `principal` (user) invoking this operation must be the current owner of the Implementation Repository entry.

### Notes

Orbix-specific.

## IT\_daemon::deleteDirectory()

**Synopsis**

```
void deleteDirectory(  
    in string dirName,  
    in boolean deleteChildren);
```

**Description** Removes a registration directory from the Implementation Repository. If `deleteChildren` is `true`, server entries and sub-directories are also deleted.

**Notes** Orbix-specific.

**See Also** ["IT\\_daemon::newDirectory\(\)" on page 313](#)

## IT\_daemon::deleteServer()

**Synopsis**

```
void deleteServer(  
    in string serverName);
```

**Description** Deletes the entry for the server, `serverName`, from the Implementation Repository.

**Notes** Orbix-specific.

**See Also** ["IT\\_daemon::newPerMethodServer\(\)" on page 313](#)  
["IT\\_daemon::newSharedServer\(\)" on page 313](#)  
["IT\\_daemon::newUnSharedServer\(\)" on page 315](#)

## IT\_daemon::getServer()

**Synopsis**

```
void getServer(  
    in string serverName,  
    out string commsProtocol,  
    out string codeProtocol,  
    out string activationPolicy,  
    out unsigned long mode_flags,  
    out string owner,  
    out string invokeList,  
    out string launchList,  
    out stringSeq markers,  
    out stringSeq methods,  
    out stringSeq commands);
```

**Description** Gets full information about the Implementation Repository entry for `serverName`.

**Notes** Orbix-specific.

## IT\_daemon::getServer2()

**Synopsis**

```
public void getServer2(String serverName,  
                       org.omg.CORBA.StringHolder  
                       commsProtocol,  
                       org.omg.CORBA.StringHolder  
                       codeProtocol,  
                       org.omg.CORBA.StringHolder  
                       activationPolicy,  
                       org.omg.CORBA.IntHolder mode_flags,  
                       org.omg.CORBA.StringHolder owner,  
                       org.omg.CORBA.StringHolder  
                       invokeList,
```

```

launchList,
org.omg.CORBA.StringHolder
org.omg.CORBA.IntHolder nservers,
org.omg.CORBA.IntHolder port,
stringSeqHolder markers,
stringSeqHolder methods,
stringSeqHolder commands) ;;

```

**Description** Gets full information about the Implementation Repository entry for `serverName`. This method was added to the `IT_daemon` interface to allow users to query the daemon for information on flags that were added to the `putitj` command, for example, `putitj -n` or `putitj -port`.

**Notes** Orbix-specific.

## IT\_daemon::getIIOPDetails

**Synopsis**

```

public void getIIOPDetails(String serverName,
String markerName,
String methodName,
org.omg.CORBA.StringHolder iiopPort,
org.omg.CORBA.StringHolder
activationPolicy);

```

**Description** This method get information about the server specified by `serverName` from the Implementation Repository. If the server is not currently running then the daemon launches the server. If the `serverName` or the `markerName` are null an `org.omg.CORBA.BAD_PARAM` exception is raised.

This method should only be used to launch server for the IIOP Protocol.

### Parameters

<code>serverName</code>	The name of the server.
<code>markerName</code>	The marker name associated with this launch command.
<code>methodName</code>	The method name associated with this launch command.
<code>iiopPort</code>	The returned port number on which the server is listening.
<code>activationPolicy</code>	Further activation mode details: 0 (shared activation mode) 1 (unshared activation mode) 2 (per-method activation mode)

**Notes** Orbix-specific.

**See Also** ["IT\\_daemon::getImplementationDetails" on page 309](#)

## IT\_daemon::getImplementationDetails

**Synopsis**

```

public void getImplementationDetails(
String serverName,
String markerName,
String methodName,
org.omg.CORBA.StringHolder codeProtocol,

```

```
        org.omg.CORBA.StringHolder  
commsProtocol,        org.omg.CORBA.StringHolder commsPort,  
                    org.omg.CORBA.StringHolder  
activationPolicy ) ;
```

## Description

This method gets information about the server specified by `serverName` from the Implementation Repository. If the server is not currently running, the daemon launches the server. If the `serverName` or the `markerName` are null, an `org.omg.CORBA.BAD_PARAM` exception is raised.

This method should only be used to launch a server for the Orbix protocol.



## Parameters

<code>serverName</code>	The name of the server.
<code>markerName</code>	The marker name associated with this launch command.
<code>methodName</code>	The method name associated with this launch command.
<code>codeProtocol</code>	The protocol used for encoding object sent across the network (for example, xdr encoding)
<code>commsProtocol</code>	The returned communications protocol used (for example, TCP).
<code>commsPort</code>	The returned port number the server is listening on.
<code>activationPolicy</code>	Further activation mode details: 0 (shared activation mode) 1 (unshared activation mode) 2 (per-method activation mode).

**Notes** Orbix-specific.

**See Also** ["IT\\_daemon::getIOPDetails"](#) on page 309

## IT\_daemon::killServer()

**Synopsis**

```
void killServer(  
    in string name,  
    in string marker);
```

**Description** Kills a server process. Where there is more than one server process, the marker parameter is used to select between different processes. The marker parameter is required when killing a process in the unshared mode.

**Notes** Orbix-specific.

## IT\_daemon::LaunchStatus

**Synopsis**

```
enum LaunchStatus (  
    inActive,  
    manuallaunch,  
    automaticLaunch  
};
```

**Description** Possible values for the `launchStatus` of a server.

**Notes** Orbix-specific.

## IT\_daemon::listActiveServers()

**Synopsis**

```
typedef sequence<serverDetails> serverDetailsSeq;  
void listActiveServers(  
    out serverDetailsSeq servers);
```

**Description** Returns a list of active server processes known to the Orbix daemon and includes information about each process.

**Notes** Orbix-specific.

See Also ["IT\\_daemon::serverDetails" on page 318](#)

## IT\_daemon::listHostsInServer()

**Synopsis** `boolean listHostsInServer(  
in string server,  
out stringSeq hostList);`

**Description** Returns a list of the hosts on which the server, `server`, runs as listed in the `server location` configuration file.

**Notes** Orbix-specific.

## IT\_daemon::listServers()

**Synopsis** `void listServers(  
in string subdir,  
out stringSeq servers);`

**Description** Lists all servers in the Implementation Repository directory `subdir`.

**Notes** Orbix-specific.

## IT\_daemon::lookUp()

**Synopsis**

```
boolean lookUp(  
    in string service,  
    out stringSeq hostList,  
    in octet hops,  
    in string tag)
```

**Description** Invokes the corresponding `lookUp()` function on the locator. This is normally the default locator—unless an alternative locator has been installed.

**Notes** Orbix-specific.

**See Also**

## IT\_daemon::newDirectory()

**Synopsis**

```
void newDirectory(in string dirName);
```

**Description** Creates a new Implementation Repository directory. The name is specified in `dirName` and may be a new directory or a subdirectory of an existing directory. Use the `'/'` character to indicate a subdirectory—for example, the name `"server/banks"` is a valid directory name.

**Notes** Orbix-specific.

**See Also** ["IT\\_daemon::deleteDirectory\(\)" on page 308](#)

## IT\_daemon::newPerMethodServer()

**Synopsis**

```
void newPerMethodServer(  
    in string serverName,  
    in stringSeq methods,  
    in stringSeq launchCommands);
```

**Description** Creates a new entry in the Implementation Repository for the per-method server `serverName`. The new entry has an activation order for each element of the sequences in `methods` and `launchCommands`.

**Parameters**

<code>serverName</code>	The name of the server.
<code>methods</code>	A sequence of methods. Each element in the sequence has a corresponding element in the sequence <code>launchCommands</code> .
<code>launchCommands</code>	A sequence of launch commands (the full path name of an executable file). Each element in the sequence has a corresponding element in the sequence <code>methods</code> .

**Notes** Orbix-specific.  
Not supported by `orbixdj`.

## IT\_daemon::newSharedServer()

**Synopsis**

```
void newSharedServer(  
    in string serverName,  
    in stringSeq methods,  
    in stringSeq launchCommands);
```

```
in string serverName,  
in stringSeq markers,  
in stringSeq launchCommands,  
in unsigned long mode_flags);
```

## Description

Creates a new Implementation Repository entry for the shared server `serverName`. The new entry has an activation order for each element of the sequences in `markers` and `launchCommands`.

## Parameters

<code>serverName</code>	The name of the server.
<code>markers</code>	A sequence of markers. Each element in the sequence has a corresponding element in the sequence <code>launchCommands</code> .
<code>launchCommands</code>	A sequence of launch commands (the full path name of an executable file and possibly command line switches). Each element in the sequence has a corresponding element in the sequence <code>markers</code> .
<code>mode_flags</code>	Further activation mode details: <ul style="list-style-type: none"><li>0 Multiple-client activation mode.</li><li>1 Per-client activation mode.</li><li>2 Per-client-process activation mode.</li></ul>

## Notes

Orbix-specific.

## IT\_daemon::newSharedServer2()

## Synopsis

```
void newSharedServer2(String serverName,  
String[] marker,  
String[] launchCommand,  
int mode_flags,  
int nservers,  
int wellKnownPort);
```

## Description

Creates a new Implementation Repository entry for the shared server `serverName`. The new entry has an activation order for each element of the sequences in `markers` and `launchCommands`. This method allows servers to be registered with extra information equivalent to the `putitj -port` and `putitj -n` commands.

## Parameters

<code>serverName</code>	The name of the server.
<code>markers</code>	A sequence of markers. Each element in the sequence has a corresponding element in the sequence <code>launchCommands</code> .
<code>launchCommands</code>	A sequence of launch commands (the full path name of an executable file and possibly command line switches). Each element in the sequence has a corresponding element in the sequence <code>markers</code> .
<code>mode_flags</code>	Further activation mode details: <ul style="list-style-type: none"><li>0 Multiple-client activation mode.</li><li>1 Per-client activation mode.</li><li>2 Per-client-process activation mode.</li></ul>

<code>nServers</code>	Specifies the numbers of servers to use for round-robin load balancing on this host; equivalent to <code>putitj -n</code> command.
<code>wellKnownPort</code>	Specifies the well-known port number that this method should use in its IORs when exporting object references.

**Notes** Orbix-specific.

## IT\_daemon::newUnSharedServer()

### Synopsis

```
void newUnSharedServer(
    in string serverName,
    in stringSeq markers,
    in stringSeq launchCommands,
    in unsigned long mode_flags);
```

### Description

Creates a new Implementation Repository entry for the unshared server `serverName`. The new entry has an activation order for each element of the sequences in `markers` and `launchCommands`.

### Parameters

<code>serverName</code>	The name of the server.
<code>markers</code>	A sequence of markers. Each element in the sequence has a corresponding element in the sequence <code>launchCommands</code> .
<code>launchCommands</code>	A sequence of launch commands (the full path name of an executable file and possibly command line switches). Each element in the sequence has a corresponding element in the sequence <code>markers</code> .
<code>mode_flags</code>	Further activation mode details: <ul style="list-style-type: none"> <li>0 Multiple-client activation mode.</li> <li>1 Per-client activation mode.</li> <li>2 Per-client-process activation mode.</li> </ul>

**Notes** Orbix-specific.  
Not supported by `orbixdj`.

## IT\_daemon::registerPersistentServer()

### Synopsis

```
public void registerPersistentServer(
    String serverName,
    int serverPid,
    org.omg.CORBA.StringHolder
    codeProtocol,
    org.omg.CORBA.StringHolder
    commsProtocol,
    org.omg.CORBA.StringHolder
    commsPort);
```

### Description

This method contacts the daemon with the servers' `serverName` and `serverPid`. The daemon returns all the information the server needs to know in order to listen for incoming events—which protocols to use and what port to listen on.

## Parameters

<code>serverName</code>	The name of this server.
<code>serverPid</code>	A process ID to be associated with this server.
<code>codeProtocol</code>	The returned coding protocol that this server should use.
<code>commsProtocol</code>	The communications protocol that this server should use.
<code>commsPort</code>	The port that this server should listen on.

**Notes** Orbix-specific.

## **IT\_daemon::removeDirRights()**

**Synopsis** `removeDirRights(in string userGroup, in string dirName);`

**Description** Removes the user or group in `userGroup` to the list of owners for the directory `dirName`.

**Notes** Orbix-specific.

## IT\_daemon::removeInvokeRights()

<b>Synopsis</b>	<pre>void removeInvokeRights(     in string userGroup,     in string serverName);</pre>
<b>Description</b>	Removes the user or group in <code>userGroup</code> from the <code>invoke</code> access control list for server <code>serverName</code> .
<b>Notes</b>	Orbix-specific.
<b>See Also</b>	<a href="#">"IT_daemon::addInvokeRights()" on page 304</a>

## IT\_daemon::removeInvokeRightsDir()

<b>Synopsis</b>	<pre>void removeInvokeRightsDir(     in string userGroup,     in string dirName);</pre>
<b>Description</b>	Removes the user or group in <code>userGroup</code> from the <i>invoke</i> access control list for directory <code>dirName</code> .
<b>Notes</b>	Orbix-specific.
<b>See Also</b>	<a href="#">"IT_daemon::addInvokeRightsDir()" on page 305</a>

## IT\_daemon::removeLaunchRights()

<b>Synopsis</b>	<pre>void removeLaunchRights(     in string userGroup,     in string serverName);</pre>
<b>Description</b>	Removes the user or group in <code>userGroup</code> from the <i>launch</i> access control list for server <code>serverName</code> .
<b>Notes</b>	Orbix-specific.
<b>See Also</b>	<a href="#">"IT_daemon::addLaunchRights()" on page 305</a>

## IT\_daemon::removeLaunchRightsDir()

<b>Synopsis</b>	<pre>void removeLaunchRightsDir(     in string userGroup,     in string dirName);</pre>
<b>Description</b>	Removes the user or group in <code>userGroup</code> from the <i>launch</i> access control list for the directory <code>dirName</code> .
<b>Notes</b>	Orbix-specific.
<b>See Also</b>	<a href="#">“IT_daemon::addLaunchRightsDir()” on page 305</a>

## IT\_daemon::removeMethod()

<b>Synopsis</b>	<pre>void removeMethod(     in string serverName,     in string methodName);</pre>
<b>Description</b>	Removes the activation order for the method (or method pattern) in <code>methodName</code> from the Implementation Repository entry for the per-method server, <code>serverName</code> .
<b>Notes</b>	Orbix-specific. Not supported by <code>orbixdj</code> .
<b>See Also</b>	<a href="#">“IT_daemon::addMethod()” on page 305</a>

## IT\_daemon::removeSharedMarker()

<b>Synopsis</b>	<pre>void removeSharedMarker(     in string serverName,     in string markerName);</pre>
<b>Description</b>	Removes the activation order for the marker (or marker pattern) in <code>markerName</code> from the Implementation Repository entry for the shared server, <code>serverName</code> .
<b>Notes</b>	Orbix-specific. Not supported by <code>orbixdj</code> .
<b>See Also</b>	<a href="#">“IT_daemon::addSharedMarker()” on page 306</a>

## IT\_daemon::removeUnsharedMarker()

<b>Synopsis</b>	<pre>void removeUnsharedMarker(     in string serverName,     in string markerName);</pre>
<b>Description</b>	Removes the activation order for the marker (or marker pattern) in <code>markerName</code> from the Implementation Repository entry for the unshared server, <code>serverName</code> .
<b>Notes</b>	Orbix-specific. Not supported by <code>orbixdj</code> .
<b>See Also</b>	<a href="#">“IT_daemon::addUnsharedMarker()” on page 307</a>

## IT\_daemon::serverDetails

<b>Synopsis</b>	<pre>struct serverDetails {     string server;</pre>
-----------------	----------------------------------------------------------



```

    string marker;
    string principal;
    string code;
    string comms;
    string port;
    unsigned long OSspecific;
    LaunchStatus status;
};

```

**Description**

The members of the struct are as follows:

server	The name of the server.
marker	The marker (if any).
principal	The name of the principal (end-user) for whom the server was launched. This is null if the server is not a per-client server.
code	The encoding protocol (for example, xdr).
comms	The transport protocol (for example, TCP/IP).
port	The port used by the communications system.
OSspecific	On UNIX, this is the operating system process identifier of the server process.
status	One of the enumerated values, <code>inactive</code> , <code>manualLaunch</code> or <code>automaticLaunch</code> .

**Notes**

Orbix-specific.

**See Also**

["IT\\_daemon::LaunchStatus" on page 311](#)

**IT\_daemon::serverExists()**

**Synopsis**

```
boolean serverExists(in string serverName);
```

**Description**

Determines whether there is an entry for the server `serverName` in the Implementation Repository.

**Notes**

Orbix-specific.



# Part VII

## Appendices

### In this part

This part contains the following:

<a href="#">IDL Grammar</a>	<a href="#">page 323</a>
<a href="#">Naming Service: IDL Definitions</a>	<a href="#">page 329</a>
<a href="#">System Exceptions</a>	<a href="#">page 331</a>



# IDL Reference

*This appendix presents reference material on the Interface Definition Language.*

## IDL Grammar

This section presents the grammar of IDL.

The notation is as follows:

Note that the two characters >> are always interpreted as a right shift operator. This means that a declaration of the form:

```
// IDL
typedef sequence<sequence<long>>  sslong;
cannot be written without a white space between the two >
characters:
```

```
// IDL
// Illegal
typedef sequence<sequence<long>>  sslong;
```

## IDL Grammar: EBNF

- (1) <specification> ::= <definition>+
- (2) <definition> ::= <type\_dcl> ";"  
| <const\_dcl> ";"  
| <except\_dcl> ";"  
| <interface> ";"  
| <module> ";"
- (3) <module> ::= "module" <identifier>  
              "{ " <definition>+ " }
- (4) <interface> ::= <interface\_dcl>  
                  | <forward\_dcl>
- (5) <interface\_dcl> ::= <interface\_header>  
                      "{ " <interface-body> " }
- (6) <forward\_dcl> ::= "interface" <identifier>
- (7) <interface\_header> ::= "interface" <identifier>  
                          [ <inheritance\_spec> ]
- (8) <interface\_body> ::= <export>\*
- (9) <export> ::= <type\_dcl> ";"  
              | <const\_dcl> ";"  
              | <except\_dcl> ";"  
              | <attr\_dcl> ";"  
              | <op\_dcl> ";"
- (10) <inheritance\_spec> ::= ":" <scoped\_name> { "," <scoped\_name> }\*
- (11) <scoped\_name> ::= <identifier>  
                      | "::" <identifier>  
                      | <scoped\_name> "::" <identifier>
- (12) <const\_dcl> ::= "const" <const\_type> <identifier>  
                      "=" <const\_exp>

```

(13) <const_type> ::= <integer_type>
                    | <char_type>
                    | <boolean_type>
                    | <floating_pt_type>
                    | <string_type>
                    | <scoped_name>
(14) <const_exp> ::= <or_expr>
(15) <or_expr> ::= <xor_expr>
                    | <or_expr> "|" <xor_expr>
(16) <xor_expr> ::= <and_expr>
                    | <xor_expr> "^" <and_expr>
(17) <and_expr> ::= <shift_expr>
                    | <and_expr> "&" <shift_expr>
(18) <shift_expr> ::= <add_expr>
                    | <shift_expr> ">>" <add_expr>
                    | <shift_expr> "<<" <add_expr>
(19) <add_expr> ::= <mult_expr>
                    | <add_expr> "+" <mult_expr>
                    | <add_expr> "-" <mult_expr>
(20) <mult_expr> ::= <unary_expr>
                    | <mult_expr> "*" <unary_expr>
                    | <mult_expr> "/" <unary_expr>
                    | <mult_expr> "%" <unary_expr>
(21) <unary_expr> ::= <unary_operator> <primary_expr>
                    | <primary_expr>
(22) <unary_operator> ::= "-"
                    | "+"
                    | "~"
(23) <primary_expr> ::= <scoped_name>
                    | <literal>
                    | "(" <const_exp> ")"
(24) <literal> ::= <integer_literal>
                    | <string_literal>
                    | <character_literal>
                    | <floating_pt_literal>
                    | <boolean_literal>
(25) <boolean_literal> ::= "TRUE"
                    | "FALSE"
(26) <positive_int_const> ::= <const_exp>
(27) <type_dcl> ::= "typedef" <type_declarator>
                    | <struct_type>
                    | <union_type>
                    | <enum_type>
(28) <type_declarator> ::= <type_spec> <declarators>
(29) <type_spec> ::= <simple_type_spec>
                    | <constr_type_spec>
(30) <simple_type_spec> ::= <base_type_spec>

```

```

| <template_type_spec>
| <scoped_name>
(31) <base_type_spec> ::= <floating_pt_type>
| <integer_type>
| <char_type>
| <boolean_type>
| <octet_type>
| <any_type>
(32) <template_type_spec> ::= <sequence_type>
| <string_type>
(33) <constr_type_spec> ::= <struct_type>
| <union_type>
| <enum_type>
(34) <declarators> ::= <declarator> { "," <declarator> }*
(35) <declarator> ::= <simple_declarator>
| <complex_declarator>
(36) <simple_declarator> ::= <identifier>
(37) <complex_declarator> ::= <array_declarator>
(38) <floating_pt_type> ::= "float"
| "double"
(39) <integer_type> ::= <signed_int>
| <unsigned_int>
(40) <signed_int> ::= <signed_long_int>
| <signed_short_int>
(41) <signed_long_int> ::= "long"
(42) <signed_short_int> ::= "short"
(43) <unsigned_int> ::= <unsigned_long_int>
| <unsigned_short_int>
(44) <unsigned_long_int> ::= "unsigned" "long"
(45) <unsigned_short_int> ::= "unsigned" "short"
(46) <char_type> ::= "char"
(47) <boolean_type> ::= "boolean"
(48) <octet_type> ::= "octet"
(49) <any_type> ::= "any"
(50) <struct_type> ::= "struct" <identifier>
| "{" <member_list> "}"
(51) <member_list> ::= <member>+
(52) <member> ::= <type_spec> <declarators> ";"
(53) <union_type> ::= "union" <identifier> "switch"
| "(" <switch_type_spec> ")"
| "{" <switch_body> "}"
(54) <switch_type_spec> ::= <integer_type>
| <char_type>
| <boolean_type>
| <enum_type>
| <scoped_name>
(55) <switch_body> ::= <case>+

```

```

(56) <case> ::= <case_label>+ <element_spec> ";"
(57) <case_label> ::= "case" <const_exp> ":"
      | "default" ":"
(58) <element_spec> ::= <type_spec> <declarator>
(59) <enum_type> ::= "enum" <identifier> "{" <enumerator>
      { "," <enumerator> }* "}"
(60) <enumerator> ::= <identifier>
(61) <sequence_type> ::= "sequence" "<" <simple_type_spec>
      "," <positive_int_const> ">"
      | "sequence" "<" <simple_type_spec> ">"
(62) <string_type> ::= "string" "<" <positive_int_const> ">"
      | "string"
(63) <array_declarator> ::= <identifier> <fixed_array_size>+
(64) <fixed_array_size> ::= "[" <positive_int_const> "]"
(65) <attr_dcl> ::= ["readonly"] "attribute"
      <param_type_spec>
      <simple_declarator>
      { "," <simple_declarator> }*
(66) <except_dcl> ::= "exception" <identifier>
      "{" <member>* "}"
(67) <op_dcl> ::= [<op_attribute>] <op_type_spec>
      <identifier>
      <parameter_dcls>
      [<raises_expr>] [<context_expr>]
(68) <op_attribute> ::= "oneway"
(69) <op_type_spec> ::= <param_type_spec>
      | "void"
(70) <parameter_dcls> ::= "(" <param_dcl> {"," <param_dcl>}* ")"
      | "(" ")"
(71) <param_dcl> ::= <param_attribute> <param_type_spec>
      <simple_declarator>
(72) <param_attribute> ::= "in"
      | "out"
      | "inout"
(73) <raises_expr> ::= "raises" "(" <scoped_name>
      { "," <scoped_name> }* ")"
(74) <context_expr> ::= "context" "(" <string_literal>
      { "," <string_literal>}* ")"
(75) <param_type_spec> ::= <base_type_spec> <string_type>
      <scoped_name>

```



## Keywords

The following are keywords in IDL.

any	default	interface	readonly	unsigned
attribute	double	long	sequence	union
boolean	enum	module	short	void
case	exception	octet	string	FALSE
char	float	oneway	struct	Object
const	in	out	switch	TRUE
context	inout	raises	typedef	

You must write keywords exactly as shown. For example, writing `Boolean` rather than `boolean` gives a compiler error.



# Naming Service: IDL Definitions

*This appendix lists the IDL definitions in the Naming Service CosNaming module.*

## The CosNaming Module

```
// IDL
module CosNaming {

    typedef string Istring;
    struct NameComponent {
        Istring id;
        Istring kind;
    };
    typedef sequence<NameComponent> Name;

    enum BindingType {nobject, ncontext};
    struct Binding {
        Name        binding_name;
        BindingType binding_type;
    };
    typedef sequence <Binding> BindingList;

    interface BindingIterator;

    interface NamingContext {
        enum NotFoundReason {missing_node,
                             not_context, not_object};
        exception NotFound {
            NotFoundReason why;
            Name           rest_of_name;
        };
        exception CannotProceed {
            NamingContext cxt;
            Name           rest_of_name;
        };
        exception InvalidName {};
        exception AlreadyBound {};
        exception NotEmpty {};

        void bind(in Name n, in Object obj)
            raises (NotFound, CannotProceed,
                  InvalidName, AlreadyBound);
        void rebind(in Name n, in Object obj)
            raises (NotFound, CannotProceed,
                  InvalidName);
        void bind_context(in Name n,
                          in NamingContext nc)
            raises (NotFound, CannotProceed,
                  InvalidName, AlreadyBound);
        void rebind_context(in Name n,
                            in NamingContext nc)
            raises (NotFound, CannotProceed,
```

```

        InvalidName);
Object resolve(in Name n)
    raises (NotFound, CannotProceed,
           InvalidName);
void unbind(in Name n)
    raises (NotFound, CannotProceed,
           InvalidName);
NamingContext new_context();
NamingContext bind_new_context(in Name n)
    raises (NotFound, CannotProceed,
           InvalidName, AlreadyBound);
void destroy() raises (NotEmpty);
void list(in unsigned long how_many,
          out BindingList bl,
          out BindingIterator bi);

interface BindingIterator {
    boolean next_one(out Binding b);
    boolean next_n(in unsigned long how_many,
                  out BindingList bl);

    void destroy();
};
};

```

# System Exceptions

The following tables shows the system exceptions defined by CORBA, and the system exceptions that are specific to Orbix Java.

## System Exceptions Defined by CORBA

*Table 1: CORBA System Exceptions*

Exception	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.
Bounds	Bounds exception.
BAD_TYPECODE	Bad TypeCode.
COMM_FAILURE	Communication failure.
DATA_CONVERSION	Data conversion error.
IMP_LIMIT	Violated implementation limit.
INITIALIZE	ORB initialization failure.
INTERNAL	ORB internal error.
INTF_REPOS	Error accessing interface repository.
INV_IDENT	Invalid identifier syntax.
INV_FLAG	Invalid flag was specified.
INV_OBJREF	Invalid object reference.
MARSHAL	Request marshalling error.
NO_MEMORY	Dynamic memory allocation failure.
NO_PERMISSION	No permission for attempted operation.
NO_IMPLEMENT	Operation implementation unavailable.
NO_RESOURCES	Insufficient resources for request.
NO_RESPONSE	Response to request not yet available.
OBJ_ADAPTOR	Failure detected by object adaptor.
PERSIST_STORE	Persistent storage failure.
TRANSACTION	Transaction exception.
TRANSIENT	Transient failure—reissue request.
UNKNOWN	The unknown exception.

# System Exceptions Specific to Orbix Java

*Table 2: Orbix Java-Specific System Exceptions*

<b>Orbix Java Exception</b>	<b>Description</b>
FILTER_SUPPRESS	Suppress exception raised in per-object pre-filter.

# Index

## A

absolute\_name() 258  
access\_name() 166  
active\_transactions() 282  
add() 91, 96, 104  
add\_arg() 171  
add\_in\_arg() 171  
add\_inout\_arg() 171  
addInvokeRights() 304  
addInvokeRightsDir() 305  
add\_item() 91, 96, 104  
addLaunchRights() 305  
addLaunchRightsDir() 304, 305  
addMethod() 305  
add\_named\_in\_arg() 172  
add\_named\_inout\_arg() 172  
add\_named\_out\_arg() 172  
add\_out\_arg() 173  
addSharedMarker() 306  
addUnsharedMarker() 307  
add\_value() 105  
any 47  
Any() 49, 50, 51  
    \_OrbixWeb conversion 244  
Any class 47  
anyClientsConnected() 63  
ARG\_IN class 3  
ARG\_INOUT class 4  
ARG\_OUT class 5  
arguments() 173  
AuthenticationFilter() 204  
AuthenticationFilter class 203

## B

BadKind() 40  
base\_interfaces 275  
baseInterfacesOf() 127  
Basic Object Adapter 59  
BOA interface 59  
bound 291, 295  
Bounds() 6, 41  
Bounds class 6, 41

## C

changeImplementation() 63  
changeOwnerServer() 307  
clear() 94  
clone() 51, 100, 104  
CloseCallback() 218  
closeConnection() 127  
collocated() 127, 128  
commit() 281  
compare() 196

CompletionStatus class 7  
Config() 205  
config() 128  
Config class 205  
connect() 64  
containing\_repository() 259  
containsType() 51  
contents() 263  
content\_type() 199  
Context()  
    \_OrbixWeb conversion 244  
Context() constructor 85  
Context class 9, 83  
ContextIterator() 92  
ContextIterator class 92  
ContextList()  
    \_OrbixWeb conversion 244  
ContextList class 10, 90  
context\_name() 86  
contexts 284  
contexts() 173  
continueThreadDispatch() 65  
copy() 52  
CORBA::AliasDef::  
    describe() 252  
    original\_type\_def 252  
CORBA::ArrayDef::  
    element\_type 253  
    element\_type\_def 253  
    length 253  
CORBA::AttributeDef::  
    describe() 254  
    mode 254  
    type 255  
    type\_def 255  
CORBA::ConstantDef::  
    describe() 256  
    type 256  
    type\_def 257  
    value 257  
CORBA::Contained::  
    absolute\_name() 258  
    containing\_repository() 259  
    defined\_in 259  
    describe() 259  
    id 260  
    move() 260  
    name() 260  
    version 261  
CORBA::Container::  
    contents() 263  
    create\_alias() 264  
    create\_constant() 264  
    create\_enum() 265

- create\_exception() 265
- create\_interface() 266
- create\_module() 266
- create\_struct() 267
- create\_union() 267
- describe\_contents() 268
- lookup() 268
- lookup\_name() 269
- CORBA::DefinitionKind 251
- CORBA::EnumDef::
  - describe() 270
  - members 270
- CORBA::ExceptionDef::
  - describe() 271
  - members 272
  - type 272
- CORBA::Identifier 251
- CORBA::IDLType::
  - type 273
- CORBA::InterfaceDef::
  - base\_interfaces 275
  - create\_attribute() 275
  - create\_operation() 276
  - describe() 277
  - describe\_interface() 277
  - is\_a() 278
- CORBA::IObject::
  - def\_kind 279
  - destroy() 279
- CORBA::IT\_Repository::
  - active\_transactions() 282
  - commit() 281
  - start() 281
- CORBA::IT\_Repository::rollback() 282
- CORBA::ModuleDef::
  - describe() 283
- CORBA::OperationDef::
  - contexts 284
  - describe() 285
  - exceptions 284
  - mode 285
  - params 286
  - result 286
  - result\_def 286
- CORBA::PrimitiveDef::
  - kind 287
- CORBA::Repository::
  - create\_array() 288
  - create\_sequence() 289
  - create\_string() 289
  - describe\_contents() 290
  - get\_primitive() 289
  - lookup\_id() 290
- CORBA::RepositoryId 251
- CORBA::ScopedName 251
- CORBA::SequenceDef::
  - bound 291
  - element\_type 291
  - element\_type\_def 291
  - type 292
- CORBA::StringDef::
  - bound 295
- CORBA::StructDef::
  - members 294
- CORBA::StructDef::describe() 293
- CORBA::TypedefDef::
  - describe() 296
- CORBA::UnionDef::
  - describe() 297
  - discriminator\_type() 298
  - discriminator\_type\_def() 298
  - members 298
- \_CORBA class 239
- count() 91, 96, 105
- create() 66
- create\_alias() 264
- create\_alias\_tc() 129, 187
- create\_any() 130, 183
- create\_array() 288
- create\_array\_tc() 129, 187
- create\_attribute() 275
- create\_child() 86
- create\_constant() 264
- create\_context\_list() 131, 184
- create\_enum() 265
- create\_enum\_tc() 129, 187
- create\_environment() 131, 184
- create\_exception() 265
- create\_exception\_list() 132, 185
- create\_exception\_tc() 129, 187
- create\_input\_stream() 52, 173
- create\_interface() 266
- create\_interface\_tc() 129, 187
- create\_list() 132, 185
- create\_module() 266
- create\_named\_value() 133, 186
- create\_operation() 276
- create\_operation\_list() 133
- create\_output\_stream() 52, 135, 174, 186
- create\_recursive\_sequence\_tc() 129, 187
- \_create\_request() 18, 111
- create\_sequence() 289
- create\_sequence\_tc() 129, 187
- create\_string() 289
- create\_string\_tc() 129, 187
- create\_struct() 267
- create\_struct\_tc() 129, 188
- create\_tc() 129, 187
- create\_union() 267
- create\_union\_tc() 129, 188
- create\_wstring\_tc() 129, 188
- ctx() 32, 174
- CTX\_RESTRICT\_SCOPE class 11
- Current()
  - \_OrbixWeb conversion 244, 245
- Current class 12

## D

- daemon
  - IDL definition 301
- deactivate\_impl() 66
- deactivate\_obj() 67
- defaultConfigFile() 229



- default\_index() 199
- defaultTxTimeout() 135
- defined\_in 259
- def\_kind 279
- \_delete() 209
- deleteDirectory() 308
- deleteServer() 308
- delete\_values() 86
- \_deref() 112
- describe() 252, 254, 256, 259, 270, 271, 277, 283, 285, 293, 296, 297
- describe\_contents() 268, 290
- describe\_interface() 277
- destroy() 279
- disconnect() 67
- discriminator\_type() 198, 298
- discriminator\_type\_def() 298
- dispose() 68
- documentation
  - .pdf format xviii
  - updates on the web xviii
- DynamicImplementation class 13

## E

- element\_type 253, 291
- element\_type\_def 253, 291
- enableLoaders() 68
- enableProxyServer() 68
- env() 174
- Environment class 14, 94
- equal() 53, 196
- equals() 100, 103, 195
- except() 34
- exception() 94
- ExceptionList() 95
- ExceptionList class 15, 95
- Exceptions
  - system exceptions 331
- exceptions 284
- exceptions() 175
- explicit\_call 240
- extract() 53
- extract\_any() 53
- extract\_boolean() 53
- extract\_char() 53
- extract\_double() 53
- extract\_float() 53
- extract\_long() 53
- extract\_longlong() 53
- extract\_Object() 53
- extract\_octet() 53
- extract\_Principal() 53
- extract\_short() 53
- extract\_Streamable() 53
- extract\_string() 53
- extract\_TypeCode() 53
- extract\_ulong() 53
- extract\_wchar() 53
- extract\_wstring() 53

## F

- Filter() 208
- Filter class 207
- finalize() 69, 136
- flags() 101
- from\_int() 8, 37
- fromString() 54
- fully-functional ORB 181

## G

- getClientConnection() 175
- getConfigFile() 230
- getConfigItem() 136, 205, 230
- getConfiguration() 137, 230
- get\_count() 87
- get\_count\_all() 87
- get\_current() 69
- getDaemonConnections() 137
- get\_default\_context() 138, 187
- \_getException() 175
- getHostPort() 138
- get\_id() 70
- \_get\_implementation() 112
- getImplementationDetails() 309
- \_get\_interface\_def() 112
- getMessageLength() 175
- get\_my\_principal() 139
- getMyReqTransformer() 139
- get\_next\_response() 140
- get\_object() 161
- get\_primitive() 289
- get\_principal() 70, 140, 161
- get\_principal\_string() 70, 141, 162
- get\_protocol() 162
- get\_request() 162
- getResponse() 176
- getServer() 308
- get\_server() 163
- getServer2() 308
- get\_socket() 163
- getTransformer() 140
- get\_values() 87

## H

- \_hash() 113
- \_hasValidOpenChannel() 113
- hasValidOpenChannel() 141
- \_host() 113

## I

- id 260
- \_id() 113
- id() 197
- IDL
  - definitions for Implementation Repository 301
  - grammar 323
  - keywords 327
- IE.Iona.OrbixWeb
  - \_CORBA 239
  - explicit\_call 240

- IT\_INFINITE\_TIMEOUT 240
- IT\_INTEROPERABLE\_OR\_KIND 240
- IT\_ORBIX\_OR\_KIND 240
- \_MAX\_LOCATOR\_HOPS 240
- objectDeletion 241
- \_ORBIX\_VERSION 240
- processTermination 242
- CORBA
  - Any 47, 55
    - Any() 49, 50
    - clone() 51
    - containsType() 51
    - copy() 52
    - create\_input\_stream() 52
    - create\_output\_stream() 52
    - equal() 53
    - extract() 53
    - extract\_any() 53
    - extract\_boolean() 53
    - extract\_char() 53
    - extract\_double() 53
    - extract\_float() 53
    - extract\_long() 53
    - extract\_longlong() 53
    - extract\_Object() 53
    - extract\_octet() 53
    - extract\_Principal() 53
    - extract\_short() 53
    - extract\_Streamable() 53
    - extract\_string() 53
    - extract\_TypeCode() 53
    - extract\_ulong() 53
    - extract\_ulonglong() 53
    - extract\_wchar() 53
    - extract\_wstring() 53
    - fromString() 54
    - insert() 55
    - insert\_any() 55
    - insert\_boolean() 55
    - insert\_char() 55
    - insert\_double() 55
    - insert\_long() 55
    - insert\_longlong() 55
    - insert\_Object() 55
    - insert\_octet() 55
    - insert\_Principal() 55
    - insert\_short() 55
    - insert\_Streamable() 55
    - insert\_string() 55
    - insert\_ulong() 55
    - insert\_ulonglong() 55
    - insert\_ushort() 55
    - insert\_wchar() 55
    - insert\_wstring() 55
    - read\_value() 56
    - reset() 57
    - toString() 57
    - type() 57
    - write\_value() 58
  - Any() 50, 51
  - BOA 59
    - anyClientsConnected () 63
    - change\_implementation() 63
    - connect() 64
    - continueThreadDispatch() 65
    - create() 66
    - deactivate\_impl() 66
    - deactivate\_obj() 67
    - disconnect() 67
    - dispose() 68
    - enableLoaders() 68
    - enableProxyServer() 68
    - finalize() 69
    - get\_current() 69
    - get\_id() 70
    - get\_principal() 70
    - get\_principal\_string() 70
    - impl\_is\_ready() 71
    - isEventPending() 73
    - myActivationMode() 73
    - myHost() 74
    - myHostIP() 74
    - myImplementationName() 74
    - myMarkerName() 75
    - myMarkerPattern() 75
    - myMethodName() 75
    - numClientsConnected() 75
    - obj\_is\_ready() 76
    - processEvents() 77
    - processNextEvent() 79
    - setNoHangup() 80
    - setProxyServer() 81
    - setServerName() 81
    - shutdown() 81, 136
    - toString() 82
  - Context 83
    - Context() 85
    - context\_name() 86
    - create\_child() 86
    - delete\_values() 86
    - get\_count() 87
    - get\_count\_all() 87
    - get\_values() 87
    - IT\_create() 88
    - \_nil() 86
    - parent() 88
    - set\_one\_value() 88
    - set\_values() 88
  - ContextIterator 92
    - ContextIterator() 92
    - next() 92
    - setList() 93
  - ContextList 90
    - add() 91
    - ContextList() 90
    - count() 91
    - item() 91
    - remove() 91
  - Environment 94
    - clear() 94
    - exception() 94
  - ExceptionList 95
    - add() 96
    - count() 96

- ExceptionList() 95
- item() 96
- remove() 96
- InitService 97
  - initialise() 97
  - InitService() 97
- NamedValue 98
  - clone() 100
  - equals() 100
  - flags() 101
  - name() 100
  - NamedValue() 99
  - \_nil() 101
  - toString() 100
  - value() 101
- NVList 102
  - add() 104
  - add\_item() 91, 96, 104
  - add\_value() 105
  - clone() 104
  - count() 105
  - equals() 103
  - item() 105
  - \_nil() 106
  - NVList() 103
  - remove() 106
- NVList() 90, 95, 103
- NVListIterator 107
  - next() 108
  - NVListIterator() 107
- ObjectRef 109
  - \_create\_request() 110, 111
  - \_deref() 112
  - \_get\_implementation() 112
  - \_get\_interface\_def() 112
  - \_hash() 113
  - \_hasValidOpenChannel() 113
  - \_host() 113
  - \_id() 113
  - \_implementation() 114
  - \_interfaceHost() 114
  - \_interfaceImplementation() 114
  - \_interfaceMarker() 114
  - \_is\_a() 115
  - \_is\_equivalent() 115
  - \_isRemote() 116
  - \_loader() 116
  - \_marker() 116
  - \_name() 117
  - \_non\_existent() 117
  - \_object\_to\_string() 118
  - \_port() 117
  - \_request() 117
  - \_save() 118
- ORB 120, 129, 130
  - baseInterfacesOf() 127
  - closeConnection() 127
  - collocated() 127, 128
  - config() 128
  - create\_alias\_tc() 129
  - create\_array\_tc() 129
  - create\_context\_list() 131
  - create\_enum\_tc() 129
  - create\_environment() 131
  - create\_exception\_list() 132
  - create\_exception\_tc() 129
  - create\_interface\_tc() 129
  - create\_list() 132
  - create\_named\_value() 133
  - create\_operation\_list() 133
  - create\_output\_stream() 135
  - create\_recursive\_sequence\_tc() 129
  - create\_sequence\_tc() 129
  - create\_string\_tc() 129
  - create\_struct\_tc() 129
  - create\_tc() 129
  - create\_wstring\_tc() 129
  - defaultTxTimeout() 135
  - finalize() 136
  - getConfigItem() 136
  - getConfiguration() 137
  - getDaemonConnections() 137
  - get\_default\_context() 138
  - getHostPort() 138
  - get\_my\_principal() 139
  - getMyReqTransformer() 139
  - get\_next\_response() 140
  - get\_principal() 140
  - get\_principal\_string() 141
  - getTransformer() 140
  - hasValidOpenChannel() 141
  - init() 142
  - isBaseInterfaceOf() 144
  - list\_initial\_services() 144
  - locator() 145
  - makeIOR() 145
  - maxConnectRetries() 146
  - myHost() 147
  - \_nil() 147
  - noReconnectOnFailure() 147
  - object\_to\_string() 148
  - pingDuringBind() 148
  - poll\_next\_response() 149
  - registerIOCallback() 150
  - resizeConnectionTable() 149
  - resolve\_initial\_references() 150
  - send\_multiple\_requests\_deferred( ) 151
  - send\_multiple\_requests\_oneway() 152
  - getConfigItem() 152
  - setConfiguration() 153
  - setDiagnostics() 153
  - setHostPort() 154
  - setMyReqTransformer() 154
  - set\_parameters() 155
  - set\_principal() 155
  - setReqTransformer() 155
  - string\_to\_object() 157
  - toString() 158
  - unregisterIOCallback() 158
- OrbCurrent 159
  - get\_object() 161

- get\_principal() 161
- get\_principal\_string() 162
- get\_protocol() 162
- get\_request() 162
- get\_server() 163
- get\_socket() 163
- Principal 164
  - access\_name 166
  - name() 166
  - Principal() 165
  - toString() 166
- Request 59, 167
  - add\_arg() 171
  - add\_in\_arg () 171
  - add\_inout\_arg() 171
  - add\_named\_in\_arg () 172
  - add\_named\_inout\_arg () 172
  - add\_named\_out\_arg () 172
  - add\_out\_arg () 173
  - arguments() 173
  - contexts() 173
  - create\_input\_stream() 173
  - create\_output\_stream() 174
  - ctx() 174
  - env() 174
  - exceptions() 175
  - getClientConnection() 175
  - \_getException() 175
  - getMessageLength() 175
  - getResponse() 176
  - invoke() 176
  - isDynamic() 176
  - isException() 176
  - \_nil() 177
  - operation() 177
  - poll\_response() 177
  - Request() 169, 170
  - reset() 177
  - result() 178
  - return\_value() 178
  - send\_deferred() 178
  - send\_oneway() 179
  - setOperation() 179
  - set\_return\_type() 179
  - setTarget() 180
  - target() 180
- singletonORB 181
  - create\_alias\_tc() 187
  - create\_any() 183
  - create\_array\_tc() 187
  - create\_context\_list() 184
  - create\_enum\_tc() 187
  - create\_environment() 184
  - create\_exception\_list() 185
  - create\_exception\_tc() 187
  - create\_interface\_tc() 187
  - create\_list() 185
  - create\_named\_value() 186
  - create\_output\_stream() 186
  - create\_recursive\_sequence\_tc() 187
  - create\_sequence\_tc() 187
- create\_string\_tc() 187
- create\_struct\_tc() 188
- create\_tc() 187
- create\_union\_tc() 188
- create\_wstring\_tc() 188
- get\_default\_context() 187
- TypeCode 190
  - compare() 196
  - content\_type() 199
  - default\_index() 199
  - discriminator\_type() 198
  - equal() 196
  - equals() 195
  - id() 197
  - kind() 195
  - length() 199
  - member\_count() 197
  - member\_label() 198
  - member\_name() 198
  - member\_type() 197
  - name() 197
  - orbixTypeCode() 200
  - toString() 200
  - TypeCode() 194, 195
- Features
  - AuthenticationFilter 203
    - AuthenticationFilter() 204
  - Config 205
    - Config() 205
    - getConfigItem() 205
    - setConfigItem() 206
  - Filter 207
    - \_delete() 209
    - Filter() 208
    - inReplyFailure() 209
    - inReplyPostMarshal() 209
    - inReplyPreMarshal() 210
    - inRequestPostMarshal() 211
    - inRequestPreMarshal() 211
    - outReplyFailure() 212
    - outReplyPostMarshal() 213
    - outReplyPreMarshal() 214
    - outRequestPostMarshal() 215
    - outRequestPreMarshal() 216
  - ioCallback 217
    - CloseCallback() 218
    - OpenCallback() 218
  - IT\_reqTransformer 219
    - transform() 220
    - transform\_error() 221
  - LoaderClass 222
    - load() 223
    - LoaderClass() 222, 223
    - record() 225
    - rename() 225
    - save() 226
  - locatorClass 227
    - locator() 227
    - lookUp() 227
  - OrbConfig 229
    - defaultConfigFile() 229
    - getConfigFile() 230

- getConfigItem() 230
- getConfiguration() 230
- OrbConfig() 229
- setConfigItem() 231
- zeroConfiguration() 231
- ProxyFactory 232
  - New() 233
  - ProxyFactory() 232
- ThreadFilter 234
  - inRequestPreMarshal() 235
  - ThreadFilter() 234
- \_OrbixWeb 243
  - Any() 244
  - Context() 244
  - ContextList() 244
  - Current() 244, 245
  - NamedValue() 246
  - NVList() 246
  - Object() 246
  - Principal() 247
  - Request() 247
  - ServerRequest() 247
  - TypeCode() 248
- \_implementation() 114
- impl\_is\_ready() 71
- init() 22, 23, 142
- initialise() 97
- InitService() 97
- InitService class 97
- InputStream, class 25
- inReplyFailure() 209
- inReplyPostMarshal() 209
- inReplyPreMarshal() 210
- inRequestPostMarshal() 211
- inRequestPreMarshal() 211, 235
- insert() 55
- insert\_any() 55
- insert\_boolean() 55
- insert\_char() 55
- insert\_double() 55
- insert\_float() 55
- insert\_long() 55
- insert\_longlong() 55
- insert\_Object() 55
- insert\_octet() 55
- insert\_Principal() 55
- insert\_short() 55
- insert\_Streamable() 55
- insert\_string() 55
- insert\_TypeCode() 55
- insert\_ulong() 55
- insert\_ulonglong() 55
- insert\_ushort() 55
- insert\_wchar() 55
- insert\_wstring() 55
- \_interfaceHost() 114
- \_interfaceImplementation() 114
- \_interfaceMarker() 114
- InterfaceName 290
- InvalidName class 24
- invoke() 176
- ioCallback interface 217

- \_is\_a() 115
- is\_a() 278
- isBaseInterfaceOf() 144
- isDynamic() 176
- \_is\_equivalent() 115
- isEventPending() 73
- isException() 176
- \_isRemote() 116
- IT\_create()
  - context 88
- IT\_daemon 301
- operations
  - addInvokeRights() 304
  - addInvokeRightsDir() 305
  - addLaunchRights() 305
  - addLaunchRightsDir() 304, 305
  - addMethod() 305
  - addSharedMarker() 306
  - addUnsharedMarker() 307
  - changeOwnerServer() 307
  - deleteDirectory() 308
  - deleteServer() 308
  - getImplementationDetails() 309
  - getServer() 308
  - getServer2() 308
  - killServer() 311
  - LaunchStatus 311
  - listActiveServers() 311
  - listHostsInServer() 312
  - listServers() 312
  - lookup() 313
  - newDirectory() 313
  - newPerMethodServer() 313
  - newSharedServer() 313
  - newSharedServer2() 314
  - newUnSharedServer() 315
  - removeDirRights() 316
  - removeInvokeRights() 317
  - removeInvokeRightsDir() 317
  - removeLaunchRights() 317
  - removeLaunchRightsDir() 318
  - removeMethod() 318
  - removeSharedMarker() 318
  - removeUnsharedMarker() 318
  - serverExists() 319
  - serverDetails 318
- item() 91, 96, 105
- IT\_INFINITE\_TIMEOUT 240
- IT\_INTEROPERABLE\_OR\_KIND 240
- IT\_ORBIX\_OR\_KIND 240
- IT\_reqTransformer class 219

## **K**

- killServer() 311
- kind 287
- kind() 195

## **L**

- LaunchStatus 311
- length 253
- length() 199

- listActiveServers() 311
- listHostsInServer() 312
- list\_initial\_services() 144
- listServers() 312
- load() 223
- \_loader() 116
- LoaderClass class 222
- loaders 222
- locator() 145
- locatorClass class 227
- locators
  - default locator 301
- lookup() 227, 313
- lookup() 268
- lookup\_id() 290
- lookup\_name() 269

## M

- makeIOR() 145
- \_marker() 116
- maxConnectRetries() 146
- \_MAX\_LOCATOR\_HOPS 240
- max\_returned\_objs 290
- member\_count() 197
- member\_label() 198
- member\_name() 198
- members 270, 272, 294, 298
- member\_type() 197
- mode 254, 285
- move() 260
- myActivationMode() 73
- myHost() 74, 147
- myHostIP() 74
- myImplementationName() 74
- myMarkerName() 75
- myMarkerPattern() 75
- myMethodName() 75

## N

- \_name() 117
- name() 100, 166, 197, 260
- NamedValue() 99
  - \_OrbixWeb conversion 246
- NamedValue class 16, 98
- New() 233
- newDirectory() 313
- newMethodPerServer() 313
- newSharedServer() 313
- newSharedServer2() 314
- newUnsharedServer() 315
- next() 92, 108
- \_nil() 86, 101, 106, 147, 177
- \_non\_existent() 117
- noReconnectOnFailure() 147
- numClientsConnected() 75
- NVList() 90, 95, 103
  - \_OrbixWeb conversion 246
- NVList class 17, 102
- NVListIterator() 107
- NVListIterator class 107

## O

- Object()
  - \_OrbixWeb conversion 246
- Object class 18
- objectDeletion 241
- ObjectRef interface 109
- \_object\_to\_string() 118
- object\_to\_string() 148
- obj\_is\_ready() 76
- OpenCallback() 218
- operation() 177
- op\_name() 32
- ORB class 19, 120
- OrbConfig() 229
- OrbConfig class 229
- OrbCurrent class 159
- orbixd
  - IDL definition 301
- orbixTypeCode() 200
- \_ORBIX\_VERSION 240
- \_OrbixWeb class 243
- org.omg.CORBA
  - ARG\_IN 3
  - ARG\_INOUT 4
  - ARG\_OUT 5
  - Bounds 6
    - Bounds() 6
  - CompletionStatus 7
    - from\_int() 8
    - value() 7
  - Context 9
  - ContextList 10
  - ContextList class 10
    - \_create\_request() 18
  - CTX\_RESTRICT\_SCOPE 11
  - Current 12
  - DynamicImplementation 13
  - Environment 14
  - ExceptionList 15
  - NamedValue 16
  - NVList 17
  - Object 18
  - ORB 19
    - init() 22, 23
  - ORBPackge
    - InvalidName 24
    - InvalidName() 24
  - portable
    - InputStream 25
    - OutputStream 26
    - Streamable 27
  - Principal 28
  - Request 29
  - ServerRequest
    - ctx() 32
    - except() 34
    - op\_name() 32
    - params() 32
    - result() 34
  - SystemException 35
  - TCKind 36

- from\_int() 37
- value() 37
- TypeCode 38
- TypeCodePackage
  - BadKind() 40
  - BadKind class 40
  - Bounds 41
  - Bounds() 41
- UnknownUserException 42
- UnknownUserException() 42
- UserException 43
- original\_type\_def 252
- OutputStream class 26
- outReplyFailure() 212
- outReplyPostMarshal() 213
- outReplyPreMarshal() 214
- outRequestPostMarshal() 215
- outRequestPreMarshal() 216

## P

- params 286
- params() 32
- parent() 88
- pingDuringBind() 148
- poll\_next\_response() 149
- poll\_response() 177
- \_port() 117
- Principal() 165
  - \_OrbixWeb conversion 247
- Principal class 28, 164
- processEvents() 77
- processNextEvent() 79
- processTermination 242
- ProxyFactory() 232
- ProxyFactory class 232

## R

- read\_value() 56
- record() 225
- registerIOCallback() 150
- remove() 91, 96, 106
- removeDirRights() 316
- removeInvokeRights() 317
- removeInvokeRightsDir() 317
- removeLaunchRights() 317
- removeLaunchRightsDir() 318
- removeMethod() 318
- removeSharedMarker() 318
- removeUnsharedMarker() 318
- rename() 225
- Request() 169, 170
  - \_OrbixWeb conversion 247
- \_request() 110, 117
- Request class 29, 59, 167
- reset() 57, 177
- reSizeConnectionTable() 149
- resolve\_initial\_references() 150
- result 286
- result() 34, 178
- return\_value() 178
- rollback() 282

## S

- \_save() 118
- save() 226
- send\_deferred() 178
- send\_multiple\_requests\_deferred() 151
- send\_multiple\_requests\_oneway() 152
- send\_oneway() 179
- serverDetails 318
- serverExists() 319
- ServerRequest()
  - \_OrbixWeb conversion 247
- ServerRequest class 31
- setConfigItem() 152, 206, 231
- setConfiguration() 153
- setDiagnostics() 153
- setHostPort() 154
- setList() 93
- setMyReqTransformer() 154
- setNoHangup() 80
- set\_one\_value() 88
- setOperation() 179
- set\_parameters() 155
- set\_principal() 155
- setProxyServer() 81
- setReqTransformer() 155
- set\_return\_type() 179
- setServerName() 81
- setTarget() 180
- set\_values() 88
- shutdown() 81, 136
- singleton ORB 181
- start() 281
- Streamable class 27
- string\_to\_object() 157
- SystemException class 35

## T

- target() 180
- TCKind class 36
- ThreadFilter() 234
- ThreadFilter class 234
- threads
  - used by ioCallbacks 217
- toString() 57, 82, 100, 158, 166, 200
- transform() 220
- transform\_error() 221
- type 255, 256, 272, 273, 292
- type() 57
- TypeCode
  - constants 239
- TypeCode, Class 190
- TypeCode() 194, 195
  - \_OrbixWeb conversion 248
- TypeCode class 38
- type\_def 255, 257

## U

- UnknownUserException() 42
- UnknownUserException class 42
- unregisterIOCallback() 158
- UserException class 43

## **V**

value 257  
value() 7, 37, 101  
version 261

## **W**

write\_value() 58

## **Z**

zeroConfiguration() 231