| | [**Overview**](http://docs.google.com/overview-summary.html) | **Package** | Class | [**Use**](http://docs.google.com/package-use.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV PACKAGE**](http://docs.google.com/org/omg/CORBA/TypeCodePackage/package-summary.html)   [**NEXT PACKAGE**](http://docs.google.com/org/omg/CosNaming/NamingContextExtPackage/package-summary.html) | [**FRAMES**](http://docs.google.com/index.html?org/omg/CosNaming/package-summary.html)    [**NO FRAMES**](http://docs.google.com/package-summary.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |

## Package org.omg.CosNaming

Provides a naming service for Java IDL.

**See:**

[**Description**](#3znysh7)

| **Interface Summary** | |
| --- | --- |
| [**BindingIterator**](http://docs.google.com/org/omg/CosNaming/BindingIterator.html) | The BindingIterator interface allows a client to iterate through the bindings using the next\_one or next\_n operations. |
| [**BindingIteratorOperations**](http://docs.google.com/org/omg/CosNaming/BindingIteratorOperations.html) | The BindingIterator interface allows a client to iterate through the bindings using the next\_one or next\_n operations. |
| [**NamingContext**](http://docs.google.com/org/omg/CosNaming/NamingContext.html) | A naming context is an object that contains a set of name bindings in which each name is unique. |
| [**NamingContextExt**](http://docs.google.com/org/omg/CosNaming/NamingContextExt.html) | NamingContextExt is the extension of NamingContext which contains a set of name bindings in which each name is unique and is part of Interoperable Naming Service. |
| [**NamingContextExtOperations**](http://docs.google.com/org/omg/CosNaming/NamingContextExtOperations.html) | NamingContextExt is the extension of NamingContext which contains a set of name bindings in which each name is unique and is part of Interoperable Naming Service. |
| [**NamingContextOperations**](http://docs.google.com/org/omg/CosNaming/NamingContextOperations.html) | A naming context is an object that contains a set of name bindings in which each name is unique. |

| **Class Summary** | |
| --- | --- |
| [**\_BindingIteratorImplBase**](http://docs.google.com/org/omg/CosNaming/_BindingIteratorImplBase.html) |  |
| [**\_BindingIteratorStub**](http://docs.google.com/org/omg/CosNaming/_BindingIteratorStub.html) | The BindingIterator interface allows a client to iterate through the bindings using the next\_one or next\_n operations. |
| [**\_NamingContextExtStub**](http://docs.google.com/org/omg/CosNaming/_NamingContextExtStub.html) | NamingContextExt is the extension of NamingContext which contains a set of name bindings in which each name is unique and is part of Interoperable Naming Service. |
| [**\_NamingContextImplBase**](http://docs.google.com/org/omg/CosNaming/_NamingContextImplBase.html) |  |
| [**\_NamingContextStub**](http://docs.google.com/org/omg/CosNaming/_NamingContextStub.html) | A naming context is an object that contains a set of name bindings in which each name is unique. |
| [**Binding**](http://docs.google.com/org/omg/CosNaming/Binding.html) | org/omg/CosNaming/Binding.java . |
| [**BindingHelper**](http://docs.google.com/org/omg/CosNaming/BindingHelper.html) | org/omg/CosNaming/BindingHelper.java . |
| [**BindingHolder**](http://docs.google.com/org/omg/CosNaming/BindingHolder.html) | org/omg/CosNaming/BindingHolder.java . |
| [**BindingIteratorHelper**](http://docs.google.com/org/omg/CosNaming/BindingIteratorHelper.html) | The BindingIterator interface allows a client to iterate through the bindings using the next\_one or next\_n operations. |
| [**BindingIteratorHolder**](http://docs.google.com/org/omg/CosNaming/BindingIteratorHolder.html) | The BindingIterator interface allows a client to iterate through the bindings using the next\_one or next\_n operations. |
| [**BindingIteratorPOA**](http://docs.google.com/org/omg/CosNaming/BindingIteratorPOA.html) | The BindingIterator interface allows a client to iterate through the bindings using the next\_one or next\_n operations. |
| [**BindingListHelper**](http://docs.google.com/org/omg/CosNaming/BindingListHelper.html) | List of Bindings. |
| [**BindingListHolder**](http://docs.google.com/org/omg/CosNaming/BindingListHolder.html) | List of Bindings. |
| [**BindingType**](http://docs.google.com/org/omg/CosNaming/BindingType.html) | Specifies whether the given binding is for a object (that is not a naming context) or for a naming context. |
| [**BindingTypeHelper**](http://docs.google.com/org/omg/CosNaming/BindingTypeHelper.html) | Specifies whether the given binding is for a object (that is not a naming context) or for a naming context. |
| [**BindingTypeHolder**](http://docs.google.com/org/omg/CosNaming/BindingTypeHolder.html) | Specifies whether the given binding is for a object (that is not a naming context) or for a naming context. |
| [**IstringHelper**](http://docs.google.com/org/omg/CosNaming/IstringHelper.html) | org/omg/CosNaming/IstringHelper.java . |
| [**NameComponent**](http://docs.google.com/org/omg/CosNaming/NameComponent.html) | org/omg/CosNaming/NameComponent.java . |
| [**NameComponentHelper**](http://docs.google.com/org/omg/CosNaming/NameComponentHelper.html) | org/omg/CosNaming/NameComponentHelper.java . |
| [**NameComponentHolder**](http://docs.google.com/org/omg/CosNaming/NameComponentHolder.html) | org/omg/CosNaming/NameComponentHolder.java . |
| [**NameHelper**](http://docs.google.com/org/omg/CosNaming/NameHelper.html) | A name is a sequence of name components. |
| [**NameHolder**](http://docs.google.com/org/omg/CosNaming/NameHolder.html) | A name is a sequence of name components. |
| [**NamingContextExtHelper**](http://docs.google.com/org/omg/CosNaming/NamingContextExtHelper.html) | NamingContextExt is the extension of NamingContext which contains a set of name bindings in which each name is unique and is part of Interoperable Naming Service. |
| [**NamingContextExtHolder**](http://docs.google.com/org/omg/CosNaming/NamingContextExtHolder.html) | NamingContextExt is the extension of NamingContext which contains a set of name bindings in which each name is unique and is part of Interoperable Naming Service. |
| [**NamingContextExtPOA**](http://docs.google.com/org/omg/CosNaming/NamingContextExtPOA.html) | NamingContextExt is the extension of NamingContext which contains a set of name bindings in which each name is unique and is part of Interoperable Naming Service. |
| [**NamingContextHelper**](http://docs.google.com/org/omg/CosNaming/NamingContextHelper.html) | A naming context is an object that contains a set of name bindings in which each name is unique. |
| [**NamingContextHolder**](http://docs.google.com/org/omg/CosNaming/NamingContextHolder.html) | A naming context is an object that contains a set of name bindings in which each name is unique. |
| [**NamingContextPOA**](http://docs.google.com/org/omg/CosNaming/NamingContextPOA.html) | A naming context is an object that contains a set of name bindings in which each name is unique. |

## Package org.omg.CosNaming Description

Provides a naming service for Java IDL. The Object Request Broker Daemon (ORBD) also includes both a transient and persistent naming service.

The package and all its classes and interfaces were generated by running the tool idlj on the file nameservice.idl, which is a module written in OMG IDL.

### Package Specification

For a precise list of supported sections of official specifications with which the Java[tm] Platform, Standard Edition 6, ORB complies, see [Official Specifications for CORBA support in Java[tm] SE 6](http://docs.google.com/CORBA/doc-files/compliance.html).

## Interfaces

The package org.omg.CosNaming contains two public interfaces and several auxiliary classes.

The interfaces are:

* NamingContext
* BindingIterator

These two interfaces provide the means to bind/unbind names and object references, to retrieve bound object references, and to iterate through a list of bindings. The NamingContext interface supplies the main functionality for the naming service, and BindingIterator provides a means of iterating through a list of name/object reference bindings.

## Auxiliary Classes

In order to map an OMG IDL interface to the Java programming language, the idlj compiler creates Java classes that can be thought of as auxiliary classes. Comments for the generated auxiliary classes used by the interfaces NamingContext and BindingIterator are included here.

### Classes Used by NamingContext and BindingIterator

The following are classes used by the naming service. (Helper and holder classes, which are generated for each of the classes listed here, are discussed below.)

* public final class **NameComponent** -- a building block for names. (Names are bound to object references in a naming context.)  
  A name is an array of one or more NameComponent objects. A name with a single NameComponent is called a *simple name*; a name with multiple NameComponent objects is called a *compound name*.  
  A **NameComponent** object consists of two fields:
  1. **id** -- a String used as an identifier
  2. **kind** -- a String that can be used for any descriptive purpose. Its importance is that it can be used to describe an object without affecting syntax. The C programming language, for example, uses the the syntactic convention of appending the extension ".c" to a file name to indicate that it is a source code file. In a NameComponent object, the kind field can be used to describe the type of object rather than a file extension or some other syntactic convention. Examples of the value of the kind field include the strings "c\_source", "object\_code", "executable", "postscript", and "". It is not unusual for the kind field to be the empty string.

In a name, each NameComponent object except the last denotes a NamingContext object; the last NameComponent object denotes the bound object reference. This is similar to a path name, in which the last name is the file name, and all names before it are directory names.

* public final class **Binding** -- an object that associates a name with an object reference or a naming context. A Binding object has two fields:
  1. **binding\_name** - an array of one or more NameComponent objects that represents the bound name
  2. **binding\_type** - a BindingType object indicating whether the binding is between a name and an object reference or between a name and a naming context

The interface NamingContext has methods for binding/unbinding names with object references or naming contexts, for listing bindings, and for resolving bindings (given a name, the method resolve returns the object reference bound to it).

* public final class **BindingType** -- an object that specifies whether the given Binding object is a binding between a name and an object reference (that is, not a naming context) or between a name and a naming context.  
  The classBindingType consists of two methods and four constants. Two of these constants are BindingType objects, and two are ints.  
  The BindingType objects can be passed to the constructor for the class Binding or used as parameters or return values. These BindingType objects are:
  + public static final BindingType **nobject** -- to indicate that the binding is with an object reference
  + public static final BindingType **ncontext** -- to indicate that the binding is with a naming context

The int constants can be supplied to the method from\_int to create BindingType objects, or they can be return values for the method value. These constants are:

* + public static final int **\_nobject**
  + public static final int **\_ncontext**

If the method from\_int is supplied with anything other than \_nobject or \_ncontext, it will throw the exception org.omg.CORBA.BAD\_PARAM.Usage is as follows: BindingType btObject = from\_int(\_nobject);  
 BindingType btContext = from\_int(\_ncontext);  
 The variable btObject refers to a BindingType object initialized to represent a binding with an object reference. The variable btContext refers to a BindingType object initialized to represent a binding with a NamingContex object.The method value returns either \_nobject or \_ncontext, so in the following line of code, the variable bt will contain \_nobject or \_ncontext: int bt = BindingType.value();

### Holder Classes

OMG IDL uses OUT and INOUT parameters for returning values from operations. The mapping to the Java programming language, which does not have OUT and INOUT parameters, creates a special class for each type, called a holder class. An instance of a holder class can be passed to a Java method as a parameter, and a value can be assigned to its value field. This allows it to perform the function of an OUT or INOUT parameter.

The following holder classes are generated for the package org.omg.CosNaming:

* NamingContextHolder
* BindingIteratorHolder
* BindingHolder
* BindingListHolder
* BindingTypeHolder
* NameComponentHolder
* NameHolder

Note that in the org.omg.CORBA package, there is a holder class for each of the basic Java types: IntHolder, ShortHolder, StringHolder, and so on.

Note also that there is a NameHolder class even though there is no Name class; similarly, there is a BindingListHolder class even though there is no BindingList class. This is true because in the OMG IDL interface, Name and BindingList are typedefs. There is no mapping from an IDL typedef to a Java construct, but holder classes are generated if the typedef is for a sequence or an array. As mapped to the Java programming language, Name is an array of NameComponent objects, and a BindingList is an array of Binding objects. All holder classes have at least two constructors and one field:

* **value** field -- an instance of the type being used as an OUT or INOUT parameter. For example, the value field of a NamingContextHolder will be a NamingContext object.
* default constructor -- a constructor that creates a new holder object initialized with the default value for the type. For example, a new BindingHolder object created with the default constructor will have its value field set to null because that is the default value for an object. Other defaults are false for boolean, 0 for numeric and char types, and null for object references.
* constructor from an instance -- a constructor that creates a new holder object whose value field is initialized with the instance supplied

A holder class for a user-defined type (a Java class) has three more methods, but application developers do not use them directly.

### Helper Classes

Helper classes, which are generated for all user-defined types in an OMG IDL interface, supply static methods needed to manipulate those types.

There is only one method in a helper class that an application programmer uses: the method narrow. Only Java interfaces mapped from IDL interfaces will have a helper class that includes a narrow method, so in the CosNaming package, only the classes NamingContextHelper and BindingIteratorHelper have a narrow method.

* public static NamingContext **narrow**(org.omg.CORBA.Object obj) -- converts the given CORBA object to a NamingContext object
* public static BindingIterator **narrow**(org.omg.CORBA.Object obj) -- converts the given CORBA object to a BindingIterator object

## Package org.omg.CosNaming.NamingContextPackage

This package supplies Helper and Holder classes for the exceptions used in the package org.omg.CosNaming and also for the class NotFoundReason, which supplies a reason for the exception NotFound.

There are Helper and Holder classes for the following exceptions:

* AlreadyBound
* CannotProceed
* InvalidName
* NotEmpty
* NotFound

## Naming Service Compatibility

Sun's implementation of the CosNaming package complies with the OMG COSNaming specification. In other words, the APIs in Sun's naming service are implemented according to the guidelines for a naming service provided by OMG. Therefore, if a third-party vendor has implemented a naming service that is OMG compliant, it is possible to switch between Sun's implementation of CosNaming and the third-party vendor's implementation. However, it is important to understand that there can be minor variations in the way different vendors implement the naming service, such as differences in the exception strings.

### Instructions for Using a Third Party's Naming Service

Although we encourage using an ORB and ORB services that are both from one vendor, it is possible to plug in a third party's COSNaming implementation with Sun's RMI-IIOP ORB. Here are the steps to follow:

1. Create a properties file for the Bootstrap server and give it two entries. For example, you could call this properties file /tmp/services and put the following in it: NameService, <Stringified IOR of the Root Naming Context>.  
   This associates NameService with the Root Naming Context of the CosNaming implementation that you want to use.
2. Start the standalone Bootstrap server using the following command:  
      
    java -classpath $(CLASSPATH)  
    com.sun.corba.ee.internal.CosNaming.BootstrapServer -InitialServicesFile  
    "/tmp/services" [-ORBInitialPort port]  
      
      
   Note that the square brackets at the end of the command indicate that specifying a port number is optional.

Now when an application calls the method org.omg.CORBA.ORB.resolve\_initial\_references, CORBA processes will contact the Bootstrap Server to get the Root Naming Context.

## Package Specification

* Interoperable Naming Service ([ptc/00-08-07](http://cgi.omg.org/cgi-bin/doc?ptc/00-08-07))

## Related Documentation

For an overview and examples of how to use the CosNaming API, please see:

* [Naming Service](http://docs.google.com/technotes/guides/idl/tnameserv.html)

For an overview of Java IDL, please see:

* [Java IDL home page](http://docs.google.com/technotes/guides/idl/index.html)

**Since:** JDK1.3

| | [**Overview**](http://docs.google.com/overview-summary.html) | **Package** | Class | [**Use**](http://docs.google.com/package-use.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV PACKAGE**](http://docs.google.com/org/omg/CORBA/TypeCodePackage/package-summary.html)   [**NEXT PACKAGE**](http://docs.google.com/org/omg/CosNaming/NamingContextExtPackage/package-summary.html) | [**FRAMES**](http://docs.google.com/index.html?org/omg/CosNaming/package-summary.html)    [**NO FRAMES**](http://docs.google.com/package-summary.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).