| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Context.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 CLASS**](http://docs.google.com/javax/naming/ConfigurationException.html)   [**NEXT CLASS**](http://docs.google.com/javax/naming/ContextNotEmptyException.html) | [**FRAMES**](http://docs.google.com/index.html?javax/naming/Context.html)    [**NO FRAMES**](http://docs.google.com/Context.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#tyjcwt) | CONSTR | [METHOD](#3dy6vkm) | DETAIL: [FIELD](#1t3h5sf) | CONSTR | [METHOD](#1ci93xb) |

## **javax.naming**

Interface Context

**All Known Subinterfaces:** [DirContext](http://docs.google.com/javax/naming/directory/DirContext.html), [EventContext](http://docs.google.com/javax/naming/event/EventContext.html), [EventDirContext](http://docs.google.com/javax/naming/event/EventDirContext.html), [LdapContext](http://docs.google.com/javax/naming/ldap/LdapContext.html) **All Known Implementing Classes:** [InitialContext](http://docs.google.com/javax/naming/InitialContext.html), [InitialDirContext](http://docs.google.com/javax/naming/directory/InitialDirContext.html), [InitialLdapContext](http://docs.google.com/javax/naming/ldap/InitialLdapContext.html)

public interface **Context**

This interface represents a naming context, which consists of a set of name-to-object bindings. It contains methods for examining and updating these bindings.

#### Names

Each name passed as an argument to a Context method is relative to that context. The empty name is used to name the context itself. A name parameter may never be null.

Most of the methods have overloaded versions with one taking a Name parameter and one taking a String. These overloaded versions are equivalent in that if the Name and String parameters are just different representations of the same name, then the overloaded versions of the same methods behave the same. In the method descriptions below, only one version is fully documented. The second version instead has a link to the first: the same documentation applies to both.

For systems that support federation, String name arguments to Context methods are composite names. Name arguments that are instances of CompositeName are treated as composite names, while Name arguments that are not instances of CompositeName are treated as compound names (which might be instances of CompoundName or other implementations of compound names). This allows the results of NameParser.parse() to be used as arguments to the Context methods. Prior to JNDI 1.2, all name arguments were treated as composite names.

Furthermore, for systems that support federation, all names returned in a NamingEnumeration from list() and listBindings() are composite names represented as strings. See CompositeName for the string syntax of names.

For systems that do not support federation, the name arguments (in either Name or String forms) and the names returned in NamingEnumeration may be names in their own namespace rather than names in a composite namespace, at the discretion of the service provider.

#### Exceptions

All the methods in this interface can throw a NamingException or any of its subclasses. See NamingException and their subclasses for details on each exception.

#### Concurrent Access

A Context instance is not guaranteed to be synchronized against concurrent access by multiple threads. Threads that need to access a single Context instance concurrently should synchronize amongst themselves and provide the necessary locking. Multiple threads each manipulating a different Context instance need not synchronize. Note that the [lookup](http://docs.google.com/javax/naming/Context.html#lookup(javax.naming.Name)) method, when passed an empty name, will return a new Context instance representing the same naming context.

For purposes of concurrency control, a Context operation that returns a NamingEnumeration is not considered to have completed while the enumeration is still in use, or while any referrals generated by that operation are still being followed.

#### Parameters

A Name parameter passed to any method of the Context interface or one of its subinterfaces will not be modified by the service provider. The service provider may keep a reference to it for the duration of the operation, including any enumeration of the method's results and the processing of any referrals generated. The caller should not modify the object during this time. A Name returned by any such method is owned by the caller. The caller may subsequently modify it; the service provider may not.

#### Environment Properties

JNDI applications need a way to communicate various preferences and properties that define the environment in which naming and directory services are accessed. For example, a context might require specification of security credentials in order to access the service. Another context might require that server configuration information be supplied. These are referred to as the *environment* of a context. The Context interface provides methods for retrieving and updating this environment.

The environment is inherited from the parent context as context methods proceed from one context to the next. Changes to the environment of one context do not directly affect those of other contexts.

It is implementation-dependent when environment properties are used and/or verified for validity. For example, some of the security-related properties are used by service providers to "log in" to the directory. This login process might occur at the time the context is created, or the first time a method is invoked on the context. When, and whether this occurs at all, is implementation-dependent. When environment properties are added or removed from the context, verifying the validity of the changes is again implementation-dependent. For example, verification of some properties might occur at the time the change is made, or at the time the next operation is performed on the context, or not at all.

Any object with a reference to a context may examine that context's environment. Sensitive information such as clear-text passwords should not be stored there unless the implementation is known to protect it.

#### Resource Files

To simplify the task of setting up the environment required by a JNDI application, application components and service providers may be distributed along with *resource files.* A JNDI resource file is a file in the properties file format (see [java.util.Properties](http://docs.google.com/java/util/Properties.html#load(java.io.Reader))), containing a list of key/value pairs. The key is the name of the property (e.g. "java.naming.factory.object") and the value is a string in the format defined for that property. Here is an example of a JNDI resource file:

java.naming.factory.object=com.sun.jndi.ldap.AttrsToCorba:com.wiz.from.Person  
 java.naming.factory.state=com.sun.jndi.ldap.CorbaToAttrs:com.wiz.from.Person  
 java.naming.factory.control=com.sun.jndi.ldap.ResponseControlFactory

The JNDI class library reads the resource files and makes the property values freely available. Thus JNDI resource files should be considered to be "world readable", and sensitive information such as clear-text passwords should not be stored there.

There are two kinds of JNDI resource files: *provider* and *application*.

##### Provider Resource Files

Each service provider has an optional resource that lists properties specific to that provider. The name of this resource is:[*prefix*/]jndiprovider.propertieswhere *prefix* is the package name of the provider's context implementation(s), with each period (".") converted to a slash ("/"). For example, suppose a service provider defines a context implementation with class name com.sun.jndi.ldap.LdapCtx. The provider resource for this provider is named com/sun/jndi/ldap/jndiprovider.properties. If the class is not in a package, the resource's name is simply jndiprovider.properties.

Certain methods in the JNDI class library make use of the standard JNDI properties that specify lists of JNDI factories:

* java.naming.factory.object
* java.naming.factory.state
* java.naming.factory.control
* java.naming.factory.url.pkgs

The JNDI library will consult the provider resource file when determining the values of these properties. Properties other than these may be set in the provider resource file at the discretion of the service provider. The service provider's documentation should clearly state which properties are allowed; other properties in the file will be ignored.

##### Application Resource Files

When an application is deployed, it will generally have several codebase directories and JARs in its classpath. Similarly, when an applet is deployed, it will have a codebase and archives specifying where to find the applet's classes. JNDI locates (using [ClassLoader.getResources()](http://docs.google.com/java/lang/ClassLoader.html#getResources(java.lang.String))) all *application resource files* named jndi.properties in the classpath. In addition, if the file *java.home*/lib/jndi.properties exists and is readable, JNDI treats it as an additional application resource file. (*java.home* indicates the directory named by the java.home system property.) All of the properties contained in these files are placed into the environment of the initial context. This environment is then inherited by other contexts.

For each property found in more than one application resource file, JNDI uses the first value found or, in a few cases where it makes sense to do so, it concatenates all of the values (details are given below). For example, if the "java.naming.factory.object" property is found in three jndi.properties resource files, the list of object factories is a concatenation of the property values from all three files. Using this scheme, each deployable component is responsible for listing the factories that it exports. JNDI automatically collects and uses all of these export lists when searching for factory classes.

##### Search Algorithm for Properties

When JNDI constructs an initial context, the context's environment is initialized with properties defined in the environment parameter passed to the constructor, the system properties, the applet parameters, and the application resource files. See [InitialContext](http://docs.google.com/InitialContext.html#ENVIRONMENT) for details. This initial environment is then inherited by other context instances.

When the JNDI class library needs to determine the value of a property, it does so by merging the values from the following two sources, in order:

1. The environment of the context being operated on.
2. The provider resource file (jndiprovider.properties) for the context being operated on.

For each property found in both of these two sources, JNDI determines the property's value as follows. If the property is one of the standard JNDI properties that specify a list of JNDI factories (listed [above](#2et92p0)), the values are concatenated into a single colon-separated list. For other properties, only the first value found is used.

When a service provider needs to determine the value of a property, it will generally take that value directly from the environment. A service provider may define provider-specific properties to be placed in its own provider resource file. In that case it should merge values as described in the previous paragraph.

In this way, each service provider developer can specify a list of factories to use with that service provider. These can be modified by the application resources specified by the deployer of the application or applet, which in turn can be modified by the user.

**Since:** 1.3

| **Field Summary** | |
| --- | --- |
| static [String](http://docs.google.com/java/lang/String.html) | [**APPLET**](http://docs.google.com/javax/naming/Context.html#APPLET)            Constant that holds the name of the environment property for specifying an applet for the initial context constructor to use when searching for other properties. |
| static [String](http://docs.google.com/java/lang/String.html) | [**AUTHORITATIVE**](http://docs.google.com/javax/naming/Context.html#AUTHORITATIVE)            Constant that holds the name of the environment property for specifying the authoritativeness of the service requested. |
| static [String](http://docs.google.com/java/lang/String.html) | [**BATCHSIZE**](http://docs.google.com/javax/naming/Context.html#BATCHSIZE)            Constant that holds the name of the environment property for specifying the batch size to use when returning data via the service's protocol. |
| static [String](http://docs.google.com/java/lang/String.html) | [**DNS\_URL**](http://docs.google.com/javax/naming/Context.html#DNS_URL)            Constant that holds the name of the environment property for specifying the DNS host and domain names to use for the JNDI URL context (for example, "dns://somehost/wiz.com"). |
| static [String](http://docs.google.com/java/lang/String.html) | [**INITIAL\_CONTEXT\_FACTORY**](http://docs.google.com/javax/naming/Context.html#INITIAL_CONTEXT_FACTORY)            Constant that holds the name of the environment property for specifying the initial context factory to use. |
| static [String](http://docs.google.com/java/lang/String.html) | [**LANGUAGE**](http://docs.google.com/javax/naming/Context.html#LANGUAGE)            Constant that holds the name of the environment property for specifying the preferred language to use with the service. |
| static [String](http://docs.google.com/java/lang/String.html) | [**OBJECT\_FACTORIES**](http://docs.google.com/javax/naming/Context.html#OBJECT_FACTORIES)            Constant that holds the name of the environment property for specifying the list of object factories to use. |
| static [String](http://docs.google.com/java/lang/String.html) | [**PROVIDER\_URL**](http://docs.google.com/javax/naming/Context.html#PROVIDER_URL)            Constant that holds the name of the environment property for specifying configuration information for the service provider to use. |
| static [String](http://docs.google.com/java/lang/String.html) | [**REFERRAL**](http://docs.google.com/javax/naming/Context.html#REFERRAL)            Constant that holds the name of the environment property for specifying how referrals encountered by the service provider are to be processed. |
| static [String](http://docs.google.com/java/lang/String.html) | [**SECURITY\_AUTHENTICATION**](http://docs.google.com/javax/naming/Context.html#SECURITY_AUTHENTICATION)            Constant that holds the name of the environment property for specifying the security level to use. |
| static [String](http://docs.google.com/java/lang/String.html) | [**SECURITY\_CREDENTIALS**](http://docs.google.com/javax/naming/Context.html#SECURITY_CREDENTIALS)            Constant that holds the name of the environment property for specifying the credentials of the principal for authenticating the caller to the service. |
| static [String](http://docs.google.com/java/lang/String.html) | [**SECURITY\_PRINCIPAL**](http://docs.google.com/javax/naming/Context.html#SECURITY_PRINCIPAL)            Constant that holds the name of the environment property for specifying the identity of the principal for authenticating the caller to the service. |
| static [String](http://docs.google.com/java/lang/String.html) | [**SECURITY\_PROTOCOL**](http://docs.google.com/javax/naming/Context.html#SECURITY_PROTOCOL)            Constant that holds the name of the environment property for specifying the security protocol to use. |
| static [String](http://docs.google.com/java/lang/String.html) | [**STATE\_FACTORIES**](http://docs.google.com/javax/naming/Context.html#STATE_FACTORIES)            Constant that holds the name of the environment property for specifying the list of state factories to use. |
| static [String](http://docs.google.com/java/lang/String.html) | [**URL\_PKG\_PREFIXES**](http://docs.google.com/javax/naming/Context.html#URL_PKG_PREFIXES)            Constant that holds the name of the environment property for specifying the list of package prefixes to use when loading in URL context factories. |

| **Method Summary** | |
| --- | --- |
| [Object](http://docs.google.com/java/lang/Object.html) | [**addToEnvironment**](http://docs.google.com/javax/naming/Context.html#addToEnvironment(java.lang.String,%20java.lang.Object))([String](http://docs.google.com/java/lang/String.html) propName, [Object](http://docs.google.com/java/lang/Object.html) propVal)            Adds a new environment property to the environment of this context. |
| void | [**bind**](http://docs.google.com/javax/naming/Context.html#bind(javax.naming.Name,%20java.lang.Object))([Name](http://docs.google.com/javax/naming/Name.html) name, [Object](http://docs.google.com/java/lang/Object.html) obj)            Binds a name to an object. |
| void | [**bind**](http://docs.google.com/javax/naming/Context.html#bind(java.lang.String,%20java.lang.Object))([String](http://docs.google.com/java/lang/String.html) name, [Object](http://docs.google.com/java/lang/Object.html) obj)            Binds a name to an object. |
| void | [**close**](http://docs.google.com/javax/naming/Context.html#close())()            Closes this context. |
| [Name](http://docs.google.com/javax/naming/Name.html) | [**composeName**](http://docs.google.com/javax/naming/Context.html#composeName(javax.naming.Name,%20javax.naming.Name))([Name](http://docs.google.com/javax/naming/Name.html) name, [Name](http://docs.google.com/javax/naming/Name.html) prefix)            Composes the name of this context with a name relative to this context. |
| [String](http://docs.google.com/java/lang/String.html) | [**composeName**](http://docs.google.com/javax/naming/Context.html#composeName(java.lang.String,%20java.lang.String))([String](http://docs.google.com/java/lang/String.html) name, [String](http://docs.google.com/java/lang/String.html) prefix)            Composes the name of this context with a name relative to this context. |
| [Context](http://docs.google.com/javax/naming/Context.html) | [**createSubcontext**](http://docs.google.com/javax/naming/Context.html#createSubcontext(javax.naming.Name))([Name](http://docs.google.com/javax/naming/Name.html) name)            Creates and binds a new context. |
| [Context](http://docs.google.com/javax/naming/Context.html) | [**createSubcontext**](http://docs.google.com/javax/naming/Context.html#createSubcontext(java.lang.String))([String](http://docs.google.com/java/lang/String.html) name)            Creates and binds a new context. |
| void | [**destroySubcontext**](http://docs.google.com/javax/naming/Context.html#destroySubcontext(javax.naming.Name))([Name](http://docs.google.com/javax/naming/Name.html) name)            Destroys the named context and removes it from the namespace. |
| void | [**destroySubcontext**](http://docs.google.com/javax/naming/Context.html#destroySubcontext(java.lang.String))([String](http://docs.google.com/java/lang/String.html) name)            Destroys the named context and removes it from the namespace. |
| [Hashtable](http://docs.google.com/java/util/Hashtable.html)<?,?> | [**getEnvironment**](http://docs.google.com/javax/naming/Context.html#getEnvironment())()            Retrieves the environment in effect for this context. |
| [String](http://docs.google.com/java/lang/String.html) | [**getNameInNamespace**](http://docs.google.com/javax/naming/Context.html#getNameInNamespace())()            Retrieves the full name of this context within its own namespace. |
| [NameParser](http://docs.google.com/javax/naming/NameParser.html) | [**getNameParser**](http://docs.google.com/javax/naming/Context.html#getNameParser(javax.naming.Name))([Name](http://docs.google.com/javax/naming/Name.html) name)            Retrieves the parser associated with the named context. |
| [NameParser](http://docs.google.com/javax/naming/NameParser.html) | [**getNameParser**](http://docs.google.com/javax/naming/Context.html#getNameParser(java.lang.String))([String](http://docs.google.com/java/lang/String.html) name)            Retrieves the parser associated with the named context. |
| [NamingEnumeration](http://docs.google.com/javax/naming/NamingEnumeration.html)<[NameClassPair](http://docs.google.com/javax/naming/NameClassPair.html)> | [**list**](http://docs.google.com/javax/naming/Context.html#list(javax.naming.Name))([Name](http://docs.google.com/javax/naming/Name.html) name)            Enumerates the names bound in the named context, along with the class names of objects bound to them. |
| [NamingEnumeration](http://docs.google.com/javax/naming/NamingEnumeration.html)<[NameClassPair](http://docs.google.com/javax/naming/NameClassPair.html)> | [**list**](http://docs.google.com/javax/naming/Context.html#list(java.lang.String))([String](http://docs.google.com/java/lang/String.html) name)            Enumerates the names bound in the named context, along with the class names of objects bound to them. |
| [NamingEnumeration](http://docs.google.com/javax/naming/NamingEnumeration.html)<[Binding](http://docs.google.com/javax/naming/Binding.html)> | [**listBindings**](http://docs.google.com/javax/naming/Context.html#listBindings(javax.naming.Name))([Name](http://docs.google.com/javax/naming/Name.html) name)            Enumerates the names bound in the named context, along with the objects bound to them. |
| [NamingEnumeration](http://docs.google.com/javax/naming/NamingEnumeration.html)<[Binding](http://docs.google.com/javax/naming/Binding.html)> | [**listBindings**](http://docs.google.com/javax/naming/Context.html#listBindings(java.lang.String))([String](http://docs.google.com/java/lang/String.html) name)            Enumerates the names bound in the named context, along with the objects bound to them. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**lookup**](http://docs.google.com/javax/naming/Context.html#lookup(javax.naming.Name))([Name](http://docs.google.com/javax/naming/Name.html) name)            Retrieves the named object. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**lookup**](http://docs.google.com/javax/naming/Context.html#lookup(java.lang.String))([String](http://docs.google.com/java/lang/String.html) name)            Retrieves the named object. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**lookupLink**](http://docs.google.com/javax/naming/Context.html#lookupLink(javax.naming.Name))([Name](http://docs.google.com/javax/naming/Name.html) name)            Retrieves the named object, following links except for the terminal atomic component of the name. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**lookupLink**](http://docs.google.com/javax/naming/Context.html#lookupLink(java.lang.String))([String](http://docs.google.com/java/lang/String.html) name)            Retrieves the named object, following links except for the terminal atomic component of the name. |
| void | [**rebind**](http://docs.google.com/javax/naming/Context.html#rebind(javax.naming.Name,%20java.lang.Object))([Name](http://docs.google.com/javax/naming/Name.html) name, [Object](http://docs.google.com/java/lang/Object.html) obj)            Binds a name to an object, overwriting any existing binding. |
| void | [**rebind**](http://docs.google.com/javax/naming/Context.html#rebind(java.lang.String,%20java.lang.Object))([String](http://docs.google.com/java/lang/String.html) name, [Object](http://docs.google.com/java/lang/Object.html) obj)            Binds a name to an object, overwriting any existing binding. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**removeFromEnvironment**](http://docs.google.com/javax/naming/Context.html#removeFromEnvironment(java.lang.String))([String](http://docs.google.com/java/lang/String.html) propName)            Removes an environment property from the environment of this context. |
| void | [**rename**](http://docs.google.com/javax/naming/Context.html#rename(javax.naming.Name,%20javax.naming.Name))([Name](http://docs.google.com/javax/naming/Name.html) oldName, [Name](http://docs.google.com/javax/naming/Name.html) newName)            Binds a new name to the object bound to an old name, and unbinds the old name. |
| void | [**rename**](http://docs.google.com/javax/naming/Context.html#rename(java.lang.String,%20java.lang.String))([String](http://docs.google.com/java/lang/String.html) oldName, [String](http://docs.google.com/java/lang/String.html) newName)            Binds a new name to the object bound to an old name, and unbinds the old name. |
| void | [**unbind**](http://docs.google.com/javax/naming/Context.html#unbind(javax.naming.Name))([Name](http://docs.google.com/javax/naming/Name.html) name)            Unbinds the named object. |
| void | [**unbind**](http://docs.google.com/javax/naming/Context.html#unbind(java.lang.String))([String](http://docs.google.com/java/lang/String.html) name)            Unbinds the named object. |

| **Field Detail** |
| --- |

### INITIAL\_CONTEXT\_FACTORY

static final [String](http://docs.google.com/java/lang/String.html) **INITIAL\_CONTEXT\_FACTORY**

Constant that holds the name of the environment property for specifying the initial context factory to use. The value of the property should be the fully qualified class name of the factory class that will create an initial context. This property may be specified in the environment parameter passed to the initial context constructor, an applet parameter, a system property, or an application resource file. If it is not specified in any of these sources, NoInitialContextException is thrown when an initial context is required to complete an operation.

The value of this constant is "java.naming.factory.initial".

**See Also:**[InitialContext](http://docs.google.com/javax/naming/InitialContext.html), [InitialDirContext](http://docs.google.com/javax/naming/directory/InitialDirContext.html), [NamingManager.getInitialContext(java.util.Hashtable)](http://docs.google.com/javax/naming/spi/NamingManager.html#getInitialContext(java.util.Hashtable)), [InitialContextFactory](http://docs.google.com/javax/naming/spi/InitialContextFactory.html), [NoInitialContextException](http://docs.google.com/javax/naming/NoInitialContextException.html), [addToEnvironment(String, Object)](http://docs.google.com/javax/naming/Context.html#addToEnvironment(java.lang.String,%20java.lang.Object)), [removeFromEnvironment(String)](http://docs.google.com/javax/naming/Context.html#removeFromEnvironment(java.lang.String)), [APPLET](http://docs.google.com/javax/naming/Context.html#APPLET), [Constant Field Values](http://docs.google.com/constant-values.html#javax.naming.Context.INITIAL_CONTEXT_FACTORY)

### OBJECT\_FACTORIES

static final [String](http://docs.google.com/java/lang/String.html) **OBJECT\_FACTORIES**

Constant that holds the name of the environment property for specifying the list of object factories to use. The value of the property should be a colon-separated list of the fully qualified class names of factory classes that will create an object given information about the object. This property may be specified in the environment, an applet parameter, a system property, or one or more resource files.

The value of this constant is "java.naming.factory.object".

**See Also:**[NamingManager.getObjectInstance(java.lang.Object, javax.naming.Name, javax.naming.Context, java.util.Hashtable)](http://docs.google.com/javax/naming/spi/NamingManager.html#getObjectInstance(java.lang.Object,%20javax.naming.Name,%20javax.naming.Context,%20java.util.Hashtable)), [ObjectFactory](http://docs.google.com/javax/naming/spi/ObjectFactory.html), [addToEnvironment(String, Object)](http://docs.google.com/javax/naming/Context.html#addToEnvironment(java.lang.String,%20java.lang.Object)), [removeFromEnvironment(String)](http://docs.google.com/javax/naming/Context.html#removeFromEnvironment(java.lang.String)), [APPLET](http://docs.google.com/javax/naming/Context.html#APPLET), [Constant Field Values](http://docs.google.com/constant-values.html#javax.naming.Context.OBJECT_FACTORIES)

### STATE\_FACTORIES

static final [String](http://docs.google.com/java/lang/String.html) **STATE\_FACTORIES**

Constant that holds the name of the environment property for specifying the list of state factories to use. The value of the property should be a colon-separated list of the fully qualified class names of state factory classes that will be used to get an object's state given the object itself. This property may be specified in the environment, an applet parameter, a system property, or one or more resource files.

The value of this constant is "java.naming.factory.state".

**Since:** 1.3 **See Also:**[NamingManager.getStateToBind(java.lang.Object, javax.naming.Name, javax.naming.Context, java.util.Hashtable)](http://docs.google.com/javax/naming/spi/NamingManager.html#getStateToBind(java.lang.Object,%20javax.naming.Name,%20javax.naming.Context,%20java.util.Hashtable)), [StateFactory](http://docs.google.com/javax/naming/spi/StateFactory.html), [addToEnvironment(String, Object)](http://docs.google.com/javax/naming/Context.html#addToEnvironment(java.lang.String,%20java.lang.Object)), [removeFromEnvironment(String)](http://docs.google.com/javax/naming/Context.html#removeFromEnvironment(java.lang.String)), [APPLET](http://docs.google.com/javax/naming/Context.html#APPLET), [Constant Field Values](http://docs.google.com/constant-values.html#javax.naming.Context.STATE_FACTORIES)

### URL\_PKG\_PREFIXES

static final [String](http://docs.google.com/java/lang/String.html) **URL\_PKG\_PREFIXES**

Constant that holds the name of the environment property for specifying the list of package prefixes to use when loading in URL context factories. The value of the property should be a colon-separated list of package prefixes for the class name of the factory class that will create a URL context factory. This property may be specified in the environment, an applet parameter, a system property, or one or more resource files. The prefix com.sun.jndi.url is always appended to the possibly empty list of package prefixes.

The value of this constant is "java.naming.factory.url.pkgs".

**See Also:**[NamingManager.getObjectInstance(java.lang.Object, javax.naming.Name, javax.naming.Context, java.util.Hashtable)](http://docs.google.com/javax/naming/spi/NamingManager.html#getObjectInstance(java.lang.Object,%20javax.naming.Name,%20javax.naming.Context,%20java.util.Hashtable)), [NamingManager.getURLContext(java.lang.String, java.util.Hashtable)](http://docs.google.com/javax/naming/spi/NamingManager.html#getURLContext(java.lang.String,%20java.util.Hashtable)), [ObjectFactory](http://docs.google.com/javax/naming/spi/ObjectFactory.html), [addToEnvironment(String, Object)](http://docs.google.com/javax/naming/Context.html#addToEnvironment(java.lang.String,%20java.lang.Object)), [removeFromEnvironment(String)](http://docs.google.com/javax/naming/Context.html#removeFromEnvironment(java.lang.String)), [APPLET](http://docs.google.com/javax/naming/Context.html#APPLET), [Constant Field Values](http://docs.google.com/constant-values.html#javax.naming.Context.URL_PKG_PREFIXES)

### PROVIDER\_URL

static final [String](http://docs.google.com/java/lang/String.html) **PROVIDER\_URL**

Constant that holds the name of the environment property for specifying configuration information for the service provider to use. The value of the property should contain a URL string (e.g. "ldap://somehost:389"). This property may be specified in the environment, an applet parameter, a system property, or a resource file. If it is not specified in any of these sources, the default configuration is determined by the service provider.

The value of this constant is "java.naming.provider.url".

**See Also:**[addToEnvironment(String, Object)](http://docs.google.com/javax/naming/Context.html#addToEnvironment(java.lang.String,%20java.lang.Object)), [removeFromEnvironment(String)](http://docs.google.com/javax/naming/Context.html#removeFromEnvironment(java.lang.String)), [APPLET](http://docs.google.com/javax/naming/Context.html#APPLET), [Constant Field Values](http://docs.google.com/constant-values.html#javax.naming.Context.PROVIDER_URL)

### DNS\_URL

static final [String](http://docs.google.com/java/lang/String.html) **DNS\_URL**

Constant that holds the name of the environment property for specifying the DNS host and domain names to use for the JNDI URL context (for example, "dns://somehost/wiz.com"). This property may be specified in the environment, an applet parameter, a system property, or a resource file. If it is not specified in any of these sources and the program attempts to use a JNDI URL containing a DNS name, a ConfigurationException will be thrown.

The value of this constant is "java.naming.dns.url".

**See Also:**[addToEnvironment(String, Object)](http://docs.google.com/javax/naming/Context.html#addToEnvironment(java.lang.String,%20java.lang.Object)), [removeFromEnvironment(String)](http://docs.google.com/javax/naming/Context.html#removeFromEnvironment(java.lang.String)), [Constant Field Values](http://docs.google.com/constant-values.html#javax.naming.Context.DNS_URL)

### AUTHORITATIVE

static final [String](http://docs.google.com/java/lang/String.html) **AUTHORITATIVE**

Constant that holds the name of the environment property for specifying the authoritativeness of the service requested. If the value of the property is the string "true", it means that the access is to the most authoritative source (i.e. bypass any cache or replicas). If the value is anything else, the source need not be (but may be) authoritative. If unspecified, the value defaults to "false".

The value of this constant is "java.naming.authoritative".

**See Also:**[addToEnvironment(String, Object)](http://docs.google.com/javax/naming/Context.html#addToEnvironment(java.lang.String,%20java.lang.Object)), [removeFromEnvironment(String)](http://docs.google.com/javax/naming/Context.html#removeFromEnvironment(java.lang.String)), [Constant Field Values](http://docs.google.com/constant-values.html#javax.naming.Context.AUTHORITATIVE)

### BATCHSIZE

static final [String](http://docs.google.com/java/lang/String.html) **BATCHSIZE**

Constant that holds the name of the environment property for specifying the batch size to use when returning data via the service's protocol. This is a hint to the provider to return the results of operations in batches of the specified size, so the provider can optimize its performance and usage of resources. The value of the property is the string representation of an integer. If unspecified, the batch size is determined by the service provider.

The value of this constant is "java.naming.batchsize".

**See Also:**[addToEnvironment(String, Object)](http://docs.google.com/javax/naming/Context.html#addToEnvironment(java.lang.String,%20java.lang.Object)), [removeFromEnvironment(String)](http://docs.google.com/javax/naming/Context.html#removeFromEnvironment(java.lang.String)), [Constant Field Values](http://docs.google.com/constant-values.html#javax.naming.Context.BATCHSIZE)

### REFERRAL

static final [String](http://docs.google.com/java/lang/String.html) **REFERRAL**

Constant that holds the name of the environment property for specifying how referrals encountered by the service provider are to be processed. The value of the property is one of the following strings: "follow" follow referrals automatically "ignore" ignore referrals "throw" throw ReferralException when a referral is encountered. If this property is not specified, the default is determined by the provider.

The value of this constant is "java.naming.referral".

**See Also:**[addToEnvironment(String, Object)](http://docs.google.com/javax/naming/Context.html#addToEnvironment(java.lang.String,%20java.lang.Object)), [removeFromEnvironment(String)](http://docs.google.com/javax/naming/Context.html#removeFromEnvironment(java.lang.String)), [Constant Field Values](http://docs.google.com/constant-values.html#javax.naming.Context.REFERRAL)

### SECURITY\_PROTOCOL

static final [String](http://docs.google.com/java/lang/String.html) **SECURITY\_PROTOCOL**

Constant that holds the name of the environment property for specifying the security protocol to use. Its value is a string determined by the service provider (e.g. "ssl"). If this property is unspecified, the behaviour is determined by the service provider.

The value of this constant is "java.naming.security.protocol".

**See Also:**[addToEnvironment(String, Object)](http://docs.google.com/javax/naming/Context.html#addToEnvironment(java.lang.String,%20java.lang.Object)), [removeFromEnvironment(String)](http://docs.google.com/javax/naming/Context.html#removeFromEnvironment(java.lang.String)), [Constant Field Values](http://docs.google.com/constant-values.html#javax.naming.Context.SECURITY_PROTOCOL)

### SECURITY\_AUTHENTICATION

static final [String](http://docs.google.com/java/lang/String.html) **SECURITY\_AUTHENTICATION**

Constant that holds the name of the environment property for specifying the security level to use. Its value is one of the following strings: "none", "simple", "strong". If this property is unspecified, the behaviour is determined by the service provider.

The value of this constant is "java.naming.security.authentication".

**See Also:**[addToEnvironment(String, Object)](http://docs.google.com/javax/naming/Context.html#addToEnvironment(java.lang.String,%20java.lang.Object)), [removeFromEnvironment(String)](http://docs.google.com/javax/naming/Context.html#removeFromEnvironment(java.lang.String)), [Constant Field Values](http://docs.google.com/constant-values.html#javax.naming.Context.SECURITY_AUTHENTICATION)

### SECURITY\_PRINCIPAL

static final [String](http://docs.google.com/java/lang/String.html) **SECURITY\_PRINCIPAL**

Constant that holds the name of the environment property for specifying the identity of the principal for authenticating the caller to the service. The format of the principal depends on the authentication scheme. If this property is unspecified, the behaviour is determined by the service provider.

The value of this constant is "java.naming.security.principal".

**See Also:**[addToEnvironment(String, Object)](http://docs.google.com/javax/naming/Context.html#addToEnvironment(java.lang.String,%20java.lang.Object)), [removeFromEnvironment(String)](http://docs.google.com/javax/naming/Context.html#removeFromEnvironment(java.lang.String)), [Constant Field Values](http://docs.google.com/constant-values.html#javax.naming.Context.SECURITY_PRINCIPAL)

### SECURITY\_CREDENTIALS

static final [String](http://docs.google.com/java/lang/String.html) **SECURITY\_CREDENTIALS**

Constant that holds the name of the environment property for specifying the credentials of the principal for authenticating the caller to the service. The value of the property depends on the authentication scheme. For example, it could be a hashed password, clear-text password, key, certificate, and so on. If this property is unspecified, the behaviour is determined by the service provider.

The value of this constant is "java.naming.security.credentials".

**See Also:**[addToEnvironment(String, Object)](http://docs.google.com/javax/naming/Context.html#addToEnvironment(java.lang.String,%20java.lang.Object)), [removeFromEnvironment(String)](http://docs.google.com/javax/naming/Context.html#removeFromEnvironment(java.lang.String)), [Constant Field Values](http://docs.google.com/constant-values.html#javax.naming.Context.SECURITY_CREDENTIALS)

### LANGUAGE

static final [String](http://docs.google.com/java/lang/String.html) **LANGUAGE**

Constant that holds the name of the environment property for specifying the preferred language to use with the service. The value of the property is a colon-separated list of language tags as defined in RFC 1766. If this property is unspecified, the language preference is determined by the service provider.

The value of this constant is "java.naming.language".

**See Also:**[addToEnvironment(String, Object)](http://docs.google.com/javax/naming/Context.html#addToEnvironment(java.lang.String,%20java.lang.Object)), [removeFromEnvironment(String)](http://docs.google.com/javax/naming/Context.html#removeFromEnvironment(java.lang.String)), [Constant Field Values](http://docs.google.com/constant-values.html#javax.naming.Context.LANGUAGE)

### APPLET

static final [String](http://docs.google.com/java/lang/String.html) **APPLET**

Constant that holds the name of the environment property for specifying an applet for the initial context constructor to use when searching for other properties. The value of this property is the java.applet.Applet instance that is being executed. This property may be specified in the environment parameter passed to the initial context constructor. When this property is set, each property that the initial context constructor looks for in the system properties is first looked for in the applet's parameter list. If this property is unspecified, the initial context constructor will search for properties only in the environment parameter passed to it, the system properties, and application resource files.

The value of this constant is "java.naming.applet".

**Since:** 1.3 **See Also:**[addToEnvironment(String, Object)](http://docs.google.com/javax/naming/Context.html#addToEnvironment(java.lang.String,%20java.lang.Object)), [removeFromEnvironment(String)](http://docs.google.com/javax/naming/Context.html#removeFromEnvironment(java.lang.String)), [InitialContext](http://docs.google.com/javax/naming/InitialContext.html), [Constant Field Values](http://docs.google.com/constant-values.html#javax.naming.Context.APPLET)

| **Method Detail** |
| --- |

### lookup

[Object](http://docs.google.com/java/lang/Object.html) **lookup**([Name](http://docs.google.com/javax/naming/Name.html) name)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Retrieves the named object. If name is empty, returns a new instance of this context (which represents the same naming context as this context, but its environment may be modified independently and it may be accessed concurrently).

**Parameters:**name - the name of the object to look up **Returns:**the object bound to name **Throws:** [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered**See Also:**[lookup(String)](http://docs.google.com/javax/naming/Context.html#lookup(java.lang.String)), [lookupLink(Name)](http://docs.google.com/javax/naming/Context.html#lookupLink(javax.naming.Name))

### lookup

[Object](http://docs.google.com/java/lang/Object.html) **lookup**([String](http://docs.google.com/java/lang/String.html) name)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Retrieves the named object. See [lookup(Name)](http://docs.google.com/javax/naming/Context.html#lookup(javax.naming.Name)) for details.

**Parameters:**name - the name of the object to look up **Returns:**the object bound to name **Throws:** [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered

### bind

void **bind**([Name](http://docs.google.com/javax/naming/Name.html) name,  
 [Object](http://docs.google.com/java/lang/Object.html) obj)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Binds a name to an object. All intermediate contexts and the target context (that named by all but terminal atomic component of the name) must already exist.

**Parameters:**name - the name to bind; may not be emptyobj - the object to bind; possibly null **Throws:** [NameAlreadyBoundException](http://docs.google.com/javax/naming/NameAlreadyBoundException.html) - if name is already bound [InvalidAttributesException](http://docs.google.com/javax/naming/directory/InvalidAttributesException.html) - if object did not supply all mandatory attributes [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered**See Also:**[bind(String, Object)](http://docs.google.com/javax/naming/Context.html#bind(java.lang.String,%20java.lang.Object)), [rebind(Name, Object)](http://docs.google.com/javax/naming/Context.html#rebind(javax.naming.Name,%20java.lang.Object)), [DirContext.bind(Name, Object, javax.naming.directory.Attributes)](http://docs.google.com/javax/naming/directory/DirContext.html#bind(javax.naming.Name,%20java.lang.Object,%20javax.naming.directory.Attributes))

### bind

void **bind**([String](http://docs.google.com/java/lang/String.html) name,  
 [Object](http://docs.google.com/java/lang/Object.html) obj)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Binds a name to an object. See [bind(Name, Object)](http://docs.google.com/javax/naming/Context.html#bind(javax.naming.Name,%20java.lang.Object)) for details.

**Parameters:**name - the name to bind; may not be emptyobj - the object to bind; possibly null **Throws:** [NameAlreadyBoundException](http://docs.google.com/javax/naming/NameAlreadyBoundException.html) - if name is already bound [InvalidAttributesException](http://docs.google.com/javax/naming/directory/InvalidAttributesException.html) - if object did not supply all mandatory attributes [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered

### rebind

void **rebind**([Name](http://docs.google.com/javax/naming/Name.html) name,  
 [Object](http://docs.google.com/java/lang/Object.html) obj)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Binds a name to an object, overwriting any existing binding. All intermediate contexts and the target context (that named by all but terminal atomic component of the name) must already exist.

If the object is a DirContext, any existing attributes associated with the name are replaced with those of the object. Otherwise, any existing attributes associated with the name remain unchanged.

**Parameters:**name - the name to bind; may not be emptyobj - the object to bind; possibly null **Throws:** [InvalidAttributesException](http://docs.google.com/javax/naming/directory/InvalidAttributesException.html) - if object did not supply all mandatory attributes [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered**See Also:**[rebind(String, Object)](http://docs.google.com/javax/naming/Context.html#rebind(java.lang.String,%20java.lang.Object)), [bind(Name, Object)](http://docs.google.com/javax/naming/Context.html#bind(javax.naming.Name,%20java.lang.Object)), [DirContext.rebind(Name, Object, javax.naming.directory.Attributes)](http://docs.google.com/javax/naming/directory/DirContext.html#rebind(javax.naming.Name,%20java.lang.Object,%20javax.naming.directory.Attributes)), [DirContext](http://docs.google.com/javax/naming/directory/DirContext.html)

### rebind

void **rebind**([String](http://docs.google.com/java/lang/String.html) name,  
 [Object](http://docs.google.com/java/lang/Object.html) obj)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Binds a name to an object, overwriting any existing binding. See [rebind(Name, Object)](http://docs.google.com/javax/naming/Context.html#rebind(javax.naming.Name,%20java.lang.Object)) for details.

**Parameters:**name - the name to bind; may not be emptyobj - the object to bind; possibly null **Throws:** [InvalidAttributesException](http://docs.google.com/javax/naming/directory/InvalidAttributesException.html) - if object did not supply all mandatory attributes [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered

### unbind

void **unbind**([Name](http://docs.google.com/javax/naming/Name.html) name)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Unbinds the named object. Removes the terminal atomic name in name from the target context--that named by all but the terminal atomic part of name.

This method is idempotent. It succeeds even if the terminal atomic name is not bound in the target context, but throws NameNotFoundException if any of the intermediate contexts do not exist.

Any attributes associated with the name are removed. Intermediate contexts are not changed.

**Parameters:**name - the name to unbind; may not be empty **Throws:** [NameNotFoundException](http://docs.google.com/javax/naming/NameNotFoundException.html) - if an intermediate context does not exist [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered**See Also:**[unbind(String)](http://docs.google.com/javax/naming/Context.html#unbind(java.lang.String))

### unbind

void **unbind**([String](http://docs.google.com/java/lang/String.html) name)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Unbinds the named object. See [unbind(Name)](http://docs.google.com/javax/naming/Context.html#unbind(javax.naming.Name)) for details.

**Parameters:**name - the name to unbind; may not be empty **Throws:** [NameNotFoundException](http://docs.google.com/javax/naming/NameNotFoundException.html) - if an intermediate context does not exist [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered

### rename

void **rename**([Name](http://docs.google.com/javax/naming/Name.html) oldName,  
 [Name](http://docs.google.com/javax/naming/Name.html) newName)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Binds a new name to the object bound to an old name, and unbinds the old name. Both names are relative to this context. Any attributes associated with the old name become associated with the new name. Intermediate contexts of the old name are not changed.

**Parameters:**oldName - the name of the existing binding; may not be emptynewName - the name of the new binding; may not be empty **Throws:** [NameAlreadyBoundException](http://docs.google.com/javax/naming/NameAlreadyBoundException.html) - if newName is already bound [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered**See Also:**[rename(String, String)](http://docs.google.com/javax/naming/Context.html#rename(java.lang.String,%20java.lang.String)), [bind(Name, Object)](http://docs.google.com/javax/naming/Context.html#bind(javax.naming.Name,%20java.lang.Object)), [rebind(Name, Object)](http://docs.google.com/javax/naming/Context.html#rebind(javax.naming.Name,%20java.lang.Object))

### rename

void **rename**([String](http://docs.google.com/java/lang/String.html) oldName,  
 [String](http://docs.google.com/java/lang/String.html) newName)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Binds a new name to the object bound to an old name, and unbinds the old name. See [rename(Name, Name)](http://docs.google.com/javax/naming/Context.html#rename(javax.naming.Name,%20javax.naming.Name)) for details.

**Parameters:**oldName - the name of the existing binding; may not be emptynewName - the name of the new binding; may not be empty **Throws:** [NameAlreadyBoundException](http://docs.google.com/javax/naming/NameAlreadyBoundException.html) - if newName is already bound [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered

### list

[NamingEnumeration](http://docs.google.com/javax/naming/NamingEnumeration.html)<[NameClassPair](http://docs.google.com/javax/naming/NameClassPair.html)> **list**([Name](http://docs.google.com/javax/naming/Name.html) name)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Enumerates the names bound in the named context, along with the class names of objects bound to them. The contents of any subcontexts are not included.

If a binding is added to or removed from this context, its effect on an enumeration previously returned is undefined.

**Parameters:**name - the name of the context to list **Returns:**an enumeration of the names and class names of the bindings in this context. Each element of the enumeration is of type NameClassPair. **Throws:** [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered**See Also:**[list(String)](http://docs.google.com/javax/naming/Context.html#list(java.lang.String)), [listBindings(Name)](http://docs.google.com/javax/naming/Context.html#listBindings(javax.naming.Name)), [NameClassPair](http://docs.google.com/javax/naming/NameClassPair.html)

### list

[NamingEnumeration](http://docs.google.com/javax/naming/NamingEnumeration.html)<[NameClassPair](http://docs.google.com/javax/naming/NameClassPair.html)> **list**([String](http://docs.google.com/java/lang/String.html) name)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Enumerates the names bound in the named context, along with the class names of objects bound to them. See [list(Name)](http://docs.google.com/javax/naming/Context.html#list(javax.naming.Name)) for details.

**Parameters:**name - the name of the context to list **Returns:**an enumeration of the names and class names of the bindings in this context. Each element of the enumeration is of type NameClassPair. **Throws:** [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered

### listBindings

[NamingEnumeration](http://docs.google.com/javax/naming/NamingEnumeration.html)<[Binding](http://docs.google.com/javax/naming/Binding.html)> **listBindings**([Name](http://docs.google.com/javax/naming/Name.html) name)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Enumerates the names bound in the named context, along with the objects bound to them. The contents of any subcontexts are not included.

If a binding is added to or removed from this context, its effect on an enumeration previously returned is undefined.

**Parameters:**name - the name of the context to list **Returns:**an enumeration of the bindings in this context. Each element of the enumeration is of type Binding. **Throws:** [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered**See Also:**[listBindings(String)](http://docs.google.com/javax/naming/Context.html#listBindings(java.lang.String)), [list(Name)](http://docs.google.com/javax/naming/Context.html#list(javax.naming.Name)), [Binding](http://docs.google.com/javax/naming/Binding.html)

### listBindings

[NamingEnumeration](http://docs.google.com/javax/naming/NamingEnumeration.html)<[Binding](http://docs.google.com/javax/naming/Binding.html)> **listBindings**([String](http://docs.google.com/java/lang/String.html) name)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Enumerates the names bound in the named context, along with the objects bound to them. See [listBindings(Name)](http://docs.google.com/javax/naming/Context.html#listBindings(javax.naming.Name)) for details.

**Parameters:**name - the name of the context to list **Returns:**an enumeration of the bindings in this context. Each element of the enumeration is of type Binding. **Throws:** [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered

### destroySubcontext

void **destroySubcontext**([Name](http://docs.google.com/javax/naming/Name.html) name)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Destroys the named context and removes it from the namespace. Any attributes associated with the name are also removed. Intermediate contexts are not destroyed.

This method is idempotent. It succeeds even if the terminal atomic name is not bound in the target context, but throws NameNotFoundException if any of the intermediate contexts do not exist.

In a federated naming system, a context from one naming system may be bound to a name in another. One can subsequently look up and perform operations on the foreign context using a composite name. However, an attempt destroy the context using this composite name will fail with NotContextException, because the foreign context is not a "subcontext" of the context in which it is bound. Instead, use unbind() to remove the binding of the foreign context. Destroying the foreign context requires that the destroySubcontext() be performed on a context from the foreign context's "native" naming system.

**Parameters:**name - the name of the context to be destroyed; may not be empty **Throws:** [NameNotFoundException](http://docs.google.com/javax/naming/NameNotFoundException.html) - if an intermediate context does not exist [NotContextException](http://docs.google.com/javax/naming/NotContextException.html) - if the name is bound but does not name a context, or does not name a context of the appropriate type [ContextNotEmptyException](http://docs.google.com/javax/naming/ContextNotEmptyException.html) - if the named context is not empty [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered**See Also:**[destroySubcontext(String)](http://docs.google.com/javax/naming/Context.html#destroySubcontext(java.lang.String))

### destroySubcontext

void **destroySubcontext**([String](http://docs.google.com/java/lang/String.html) name)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Destroys the named context and removes it from the namespace. See [destroySubcontext(Name)](http://docs.google.com/javax/naming/Context.html#destroySubcontext(javax.naming.Name)) for details.

**Parameters:**name - the name of the context to be destroyed; may not be empty **Throws:** [NameNotFoundException](http://docs.google.com/javax/naming/NameNotFoundException.html) - if an intermediate context does not exist [NotContextException](http://docs.google.com/javax/naming/NotContextException.html) - if the name is bound but does not name a context, or does not name a context of the appropriate type [ContextNotEmptyException](http://docs.google.com/javax/naming/ContextNotEmptyException.html) - if the named context is not empty [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered

### createSubcontext

[Context](http://docs.google.com/javax/naming/Context.html) **createSubcontext**([Name](http://docs.google.com/javax/naming/Name.html) name)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Creates and binds a new context. Creates a new context with the given name and binds it in the target context (that named by all but terminal atomic component of the name). All intermediate contexts and the target context must already exist.

**Parameters:**name - the name of the context to create; may not be empty **Returns:**the newly created context **Throws:** [NameAlreadyBoundException](http://docs.google.com/javax/naming/NameAlreadyBoundException.html) - if name is already bound [InvalidAttributesException](http://docs.google.com/javax/naming/directory/InvalidAttributesException.html) - if creation of the subcontext requires specification of mandatory attributes [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered**See Also:**[createSubcontext(String)](http://docs.google.com/javax/naming/Context.html#createSubcontext(java.lang.String)), [DirContext.createSubcontext(javax.naming.Name, javax.naming.directory.Attributes)](http://docs.google.com/javax/naming/directory/DirContext.html#createSubcontext(javax.naming.Name,%20javax.naming.directory.Attributes))

### createSubcontext

[Context](http://docs.google.com/javax/naming/Context.html) **createSubcontext**([String](http://docs.google.com/java/lang/String.html) name)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Creates and binds a new context. See [createSubcontext(Name)](http://docs.google.com/javax/naming/Context.html#createSubcontext(javax.naming.Name)) for details.

**Parameters:**name - the name of the context to create; may not be empty **Returns:**the newly created context **Throws:** [NameAlreadyBoundException](http://docs.google.com/javax/naming/NameAlreadyBoundException.html) - if name is already bound [InvalidAttributesException](http://docs.google.com/javax/naming/directory/InvalidAttributesException.html) - if creation of the subcontext requires specification of mandatory attributes [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered

### lookupLink

[Object](http://docs.google.com/java/lang/Object.html) **lookupLink**([Name](http://docs.google.com/javax/naming/Name.html) name)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Retrieves the named object, following links except for the terminal atomic component of the name. If the object bound to name is not a link, returns the object itself.

**Parameters:**name - the name of the object to look up **Returns:**the object bound to name, not following the terminal link (if any). **Throws:** [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered**See Also:**[lookupLink(String)](http://docs.google.com/javax/naming/Context.html#lookupLink(java.lang.String))

### lookupLink

[Object](http://docs.google.com/java/lang/Object.html) **lookupLink**([String](http://docs.google.com/java/lang/String.html) name)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Retrieves the named object, following links except for the terminal atomic component of the name. See [lookupLink(Name)](http://docs.google.com/javax/naming/Context.html#lookupLink(javax.naming.Name)) for details.

**Parameters:**name - the name of the object to look up **Returns:**the object bound to name, not following the terminal link (if any) **Throws:** [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered

### getNameParser

[NameParser](http://docs.google.com/javax/naming/NameParser.html) **getNameParser**([Name](http://docs.google.com/javax/naming/Name.html) name)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Retrieves the parser associated with the named context. In a federation of namespaces, different naming systems will parse names differently. This method allows an application to get a parser for parsing names into their atomic components using the naming convention of a particular naming system. Within any single naming system, NameParser objects returned by this method must be equal (using the equals() test).

**Parameters:**name - the name of the context from which to get the parser **Returns:**a name parser that can parse compound names into their atomic components **Throws:** [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered**See Also:**[getNameParser(String)](http://docs.google.com/javax/naming/Context.html#getNameParser(java.lang.String)), [CompoundName](http://docs.google.com/javax/naming/CompoundName.html)

### getNameParser

[NameParser](http://docs.google.com/javax/naming/NameParser.html) **getNameParser**([String](http://docs.google.com/java/lang/String.html) name)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Retrieves the parser associated with the named context. See [getNameParser(Name)](http://docs.google.com/javax/naming/Context.html#getNameParser(javax.naming.Name)) for details.

**Parameters:**name - the name of the context from which to get the parser **Returns:**a name parser that can parse compound names into their atomic components **Throws:** [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered

### composeName

[Name](http://docs.google.com/javax/naming/Name.html) **composeName**([Name](http://docs.google.com/javax/naming/Name.html) name,  
 [Name](http://docs.google.com/javax/naming/Name.html) prefix)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Composes the name of this context with a name relative to this context. Given a name (name) relative to this context, and the name (prefix) of this context relative to one of its ancestors, this method returns the composition of the two names using the syntax appropriate for the naming system(s) involved. That is, if name names an object relative to this context, the result is the name of the same object, but relative to the ancestor context. None of the names may be null.

For example, if this context is named "wiz.com" relative to the initial context, then

composeName("east", "wiz.com")

might return "east.wiz.com". If instead this context is named "org/research", then

composeName("user/jane", "org/research")

might return "org/research/user/jane" while

composeName("user/jane", "research")

returns "research/user/jane".

**Parameters:**name - a name relative to this contextprefix - the name of this context relative to one of its ancestors **Returns:**the composition of prefix and name **Throws:** [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered**See Also:**[composeName(String, String)](http://docs.google.com/javax/naming/Context.html#composeName(java.lang.String,%20java.lang.String))

### composeName

[String](http://docs.google.com/java/lang/String.html) **composeName**([String](http://docs.google.com/java/lang/String.html) name,  
 [String](http://docs.google.com/java/lang/String.html) prefix)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Composes the name of this context with a name relative to this context. See [composeName(Name, Name)](http://docs.google.com/javax/naming/Context.html#composeName(javax.naming.Name,%20javax.naming.Name)) for details.

**Parameters:**name - a name relative to this contextprefix - the name of this context relative to one of its ancestors **Returns:**the composition of prefix and name **Throws:** [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered

### addToEnvironment

[Object](http://docs.google.com/java/lang/Object.html) **addToEnvironment**([String](http://docs.google.com/java/lang/String.html) propName,  
 [Object](http://docs.google.com/java/lang/Object.html) propVal)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Adds a new environment property to the environment of this context. If the property already exists, its value is overwritten. See class description for more details on environment properties.

**Parameters:**propName - the name of the environment property to add; may not be nullpropVal - the value of the property to add; may not be null **Returns:**the previous value of the property, or null if the property was not in the environment before **Throws:** [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered**See Also:**[getEnvironment()](http://docs.google.com/javax/naming/Context.html#getEnvironment()), [removeFromEnvironment(String)](http://docs.google.com/javax/naming/Context.html#removeFromEnvironment(java.lang.String))

### removeFromEnvironment

[Object](http://docs.google.com/java/lang/Object.html) **removeFromEnvironment**([String](http://docs.google.com/java/lang/String.html) propName)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Removes an environment property from the environment of this context. See class description for more details on environment properties.

**Parameters:**propName - the name of the environment property to remove; may not be null **Returns:**the previous value of the property, or null if the property was not in the environment **Throws:** [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered**See Also:**[getEnvironment()](http://docs.google.com/javax/naming/Context.html#getEnvironment()), [addToEnvironment(String, Object)](http://docs.google.com/javax/naming/Context.html#addToEnvironment(java.lang.String,%20java.lang.Object))

### getEnvironment

[Hashtable](http://docs.google.com/java/util/Hashtable.html)<?,?> **getEnvironment**()  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Retrieves the environment in effect for this context. See class description for more details on environment properties.

The caller should not make any changes to the object returned: their effect on the context is undefined. The environment of this context may be changed using addToEnvironment() and removeFromEnvironment().

**Returns:**the environment of this context; never null **Throws:** [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered**See Also:**[addToEnvironment(String, Object)](http://docs.google.com/javax/naming/Context.html#addToEnvironment(java.lang.String,%20java.lang.Object)), [removeFromEnvironment(String)](http://docs.google.com/javax/naming/Context.html#removeFromEnvironment(java.lang.String))

### close

void **close**()  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Closes this context. This method releases this context's resources immediately, instead of waiting for them to be released automatically by the garbage collector.

This method is idempotent: invoking it on a context that has already been closed has no effect. Invoking any other method on a closed context is not allowed, and results in undefined behaviour.

**Throws:** [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered

### getNameInNamespace

[String](http://docs.google.com/java/lang/String.html) **getNameInNamespace**()  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Retrieves the full name of this context within its own namespace.

Many naming services have a notion of a "full name" for objects in their respective namespaces. For example, an LDAP entry has a distinguished name, and a DNS record has a fully qualified name. This method allows the client application to retrieve this name. The string returned by this method is not a JNDI composite name and should not be passed directly to context methods. In naming systems for which the notion of full name does not make sense, OperationNotSupportedException is thrown.

**Returns:**this context's name in its own namespace; never null **Throws:** [OperationNotSupportedException](http://docs.google.com/javax/naming/OperationNotSupportedException.html) - if the naming system does not have the notion of a full name [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered**Since:** 1.3

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Context.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 CLASS**](http://docs.google.com/javax/naming/ConfigurationException.html)   [**NEXT CLASS**](http://docs.google.com/javax/naming/ContextNotEmptyException.html) | [**FRAMES**](http://docs.google.com/index.html?javax/naming/Context.html)    [**NO FRAMES**](http://docs.google.com/Context.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#tyjcwt) | CONSTR | [METHOD](#3dy6vkm) | DETAIL: [FIELD](#1t3h5sf) | CONSTR | [METHOD](#1ci93xb) |

[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).