| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/EventContext.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   [**NEXT CLASS**](http://docs.google.com/javax/naming/event/EventDirContext.html) | [**FRAMES**](http://docs.google.com/index.html?javax/naming/event/EventContext.html)    [**NO FRAMES**](http://docs.google.com/EventContext.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | CONSTR | [METHOD](#tyjcwt) | DETAIL: [FIELD](#1t3h5sf) | CONSTR | [METHOD](#3rdcrjn) |

## **javax.naming.event**

Interface EventContext

**All Superinterfaces:** [Context](http://docs.google.com/javax/naming/Context.html) **All Known Subinterfaces:** [EventDirContext](http://docs.google.com/javax/naming/event/EventDirContext.html)

public interface **EventContext**extends [Context](http://docs.google.com/javax/naming/Context.html)

Contains methods for registering/deregistering listeners to be notified of events fired when objects named in a context changes.

#### Target

The name parameter in the addNamingListener() methods is referred to as the *target*. The target, along with the scope, identify the object(s) that the listener is interested in. It is possible to register interest in a target that does not exist, but there might be limitations in the extent to which this can be supported by the service provider and underlying protocol/service.

If a service only supports registration for existing targets, an attempt to register for a nonexistent target results in a NameNotFoundException being thrown as early as possible, preferably at the time addNamingListener() is called, or if that is not possible, the listener will receive the exception through the NamingExceptionEvent.

Also, for service providers that only support registration for existing targets, when the target that a listener has registered for is subsequently removed from the namespace, the listener is notified via a NamingExceptionEvent (containing a NameNotFoundException).

An application can use the method targetMustExist() to check whether a EventContext supports registration of nonexistent targets.

#### Event Source

The EventContext instance on which you invoke the registration methods is the *event source* of the events that are (potentially) generated. The source is *not necessarily* the object named by the target. Only when the target is the empty name is the object named by the target the source. In other words, the target, along with the scope parameter, are used to identify the object(s) that the listener is interested in, but the event source is the EventContext instance with which the listener has registered.

For example, suppose a listener makes the following registration:

NamespaceChangeListener listener = ...;  
 src.addNamingListener("x", SUBTREE\_SCOPE, listener);

When an object named "x/y" is subsequently deleted, the corresponding NamingEvent (evt) must contain:

evt.getEventContext() == src  
 evt.getOldBinding().getName().equals("x/y")

Furthermore, listener registration/deregistration is with the EventContext *instance*, and not with the corresponding object in the namespace. If the program intends at some point to remove a listener, then it needs to keep a reference to the EventContext instance on which it invoked addNamingListener() (just as it needs to keep a reference to the listener in order to remove it later). It cannot expect to do a lookup() and get another instance of a EventContext on which to perform the deregistration.

#### Lifetime of Registration

A registered listener becomes deregistered when:

* It is removed using removeNamingListener().
* An exception is thrown while collecting information about the events. That is, when the listener receives a NamingExceptionEvent.
* Context.close() is invoked on the EventContext instance with which it has registered.

Until that point, a EventContext instance that has outstanding listeners will continue to exist and be maintained by the service provider.

#### Listener Implementations

The registration/deregistration methods accept an instance of NamingListener. There are subinterfaces of NamingListener for different of event types of NamingEvent. For example, the ObjectChangeListener interface is for the NamingEvent.OBJECT\_CHANGED event type. To register interest in multiple event types, the listener implementation should implement multiple NamingListener subinterfaces and use a single invocation of addNamingListener(). In addition to reducing the number of method calls and possibly the code size of the listeners, this allows some service providers to optimize the registration.

#### Threading Issues

Like Context instances in general, instances of EventContext are not guaranteed to be thread-safe. Care must be taken when multiple threads are accessing the same EventContext concurrently. See the [package description](http://docs.google.com/package-summary.html#THREADING) for more information on threading issues.

**Since:** 1.3

| **Field Summary** | |
| --- | --- |
| static int | [**OBJECT\_SCOPE**](http://docs.google.com/javax/naming/event/EventContext.html#OBJECT_SCOPE)            Constant for expressing interest in events concerning the object named by the target. |
| static int | [**ONELEVEL\_SCOPE**](http://docs.google.com/javax/naming/event/EventContext.html#ONELEVEL_SCOPE)            Constant for expressing interest in events concerning objects in the context named by the target, excluding the context named by the target. |
| static int | [**SUBTREE\_SCOPE**](http://docs.google.com/javax/naming/event/EventContext.html#SUBTREE_SCOPE)            Constant for expressing interest in events concerning objects in the subtree of the object named by the target, including the object named by the target. |

| **Fields inherited from interface javax.naming.**[**Context**](http://docs.google.com/javax/naming/Context.html) |
| --- |
| [APPLET](http://docs.google.com/javax/naming/Context.html#APPLET), [AUTHORITATIVE](http://docs.google.com/javax/naming/Context.html#AUTHORITATIVE), [BATCHSIZE](http://docs.google.com/javax/naming/Context.html#BATCHSIZE), [DNS\_URL](http://docs.google.com/javax/naming/Context.html#DNS_URL), [INITIAL\_CONTEXT\_FACTORY](http://docs.google.com/javax/naming/Context.html#INITIAL_CONTEXT_FACTORY), [LANGUAGE](http://docs.google.com/javax/naming/Context.html#LANGUAGE), [OBJECT\_FACTORIES](http://docs.google.com/javax/naming/Context.html#OBJECT_FACTORIES), [PROVIDER\_URL](http://docs.google.com/javax/naming/Context.html#PROVIDER_URL), [REFERRAL](http://docs.google.com/javax/naming/Context.html#REFERRAL), [SECURITY\_AUTHENTICATION](http://docs.google.com/javax/naming/Context.html#SECURITY_AUTHENTICATION), [SECURITY\_CREDENTIALS](http://docs.google.com/javax/naming/Context.html#SECURITY_CREDENTIALS), [SECURITY\_PRINCIPAL](http://docs.google.com/javax/naming/Context.html#SECURITY_PRINCIPAL), [SECURITY\_PROTOCOL](http://docs.google.com/javax/naming/Context.html#SECURITY_PROTOCOL), [STATE\_FACTORIES](http://docs.google.com/javax/naming/Context.html#STATE_FACTORIES), [URL\_PKG\_PREFIXES](http://docs.google.com/javax/naming/Context.html#URL_PKG_PREFIXES) |

| **Method Summary** | |
| --- | --- |
| void | [**addNamingListener**](http://docs.google.com/javax/naming/event/EventContext.html#addNamingListener(javax.naming.Name,%20int,%20javax.naming.event.NamingListener))([Name](http://docs.google.com/javax/naming/Name.html) target, int scope, [NamingListener](http://docs.google.com/javax/naming/event/NamingListener.html) l)            Adds a listener for receiving naming events fired when the object(s) identified by a target and scope changes. |
| void | [**addNamingListener**](http://docs.google.com/javax/naming/event/EventContext.html#addNamingListener(java.lang.String,%20int,%20javax.naming.event.NamingListener))([String](http://docs.google.com/java/lang/String.html) target, int scope, [NamingListener](http://docs.google.com/javax/naming/event/NamingListener.html) l)            Adds a listener for receiving naming events fired when the object named by the string target name and scope changes. |
| void | [**removeNamingListener**](http://docs.google.com/javax/naming/event/EventContext.html#removeNamingListener(javax.naming.event.NamingListener))([NamingListener](http://docs.google.com/javax/naming/event/NamingListener.html) l)            Removes a listener from receiving naming events fired by this EventContext. |
| boolean | [**targetMustExist**](http://docs.google.com/javax/naming/event/EventContext.html#targetMustExist())()            Determines whether a listener can register interest in a target that does not exist. |

| **Methods inherited from interface javax.naming.**[**Context**](http://docs.google.com/javax/naming/Context.html) |
| --- |
| [addToEnvironment](http://docs.google.com/javax/naming/Context.html#addToEnvironment(java.lang.String,%20java.lang.Object)), [bind](http://docs.google.com/javax/naming/Context.html#bind(javax.naming.Name,%20java.lang.Object)), [bind](http://docs.google.com/javax/naming/Context.html#bind(java.lang.String,%20java.lang.Object)), [close](http://docs.google.com/javax/naming/Context.html#close()), [composeName](http://docs.google.com/javax/naming/Context.html#composeName(javax.naming.Name,%20javax.naming.Name)), [composeName](http://docs.google.com/javax/naming/Context.html#composeName(java.lang.String,%20java.lang.String)), [createSubcontext](http://docs.google.com/javax/naming/Context.html#createSubcontext(javax.naming.Name)), [createSubcontext](http://docs.google.com/javax/naming/Context.html#createSubcontext(java.lang.String)), [destroySubcontext](http://docs.google.com/javax/naming/Context.html#destroySubcontext(javax.naming.Name)), [destroySubcontext](http://docs.google.com/javax/naming/Context.html#destroySubcontext(java.lang.String)), [getEnvironment](http://docs.google.com/javax/naming/Context.html#getEnvironment()), [getNameInNamespace](http://docs.google.com/javax/naming/Context.html#getNameInNamespace()), [getNameParser](http://docs.google.com/javax/naming/Context.html#getNameParser(javax.naming.Name)), [getNameParser](http://docs.google.com/javax/naming/Context.html#getNameParser(java.lang.String)), [list](http://docs.google.com/javax/naming/Context.html#list(javax.naming.Name)), [list](http://docs.google.com/javax/naming/Context.html#list(java.lang.String)), [listBindings](http://docs.google.com/javax/naming/Context.html#listBindings(javax.naming.Name)), [listBindings](http://docs.google.com/javax/naming/Context.html#listBindings(java.lang.String)), [lookup](http://docs.google.com/javax/naming/Context.html#lookup(javax.naming.Name)), [lookup](http://docs.google.com/javax/naming/Context.html#lookup(java.lang.String)), [lookupLink](http://docs.google.com/javax/naming/Context.html#lookupLink(javax.naming.Name)), [lookupLink](http://docs.google.com/javax/naming/Context.html#lookupLink(java.lang.String)), [rebind](http://docs.google.com/javax/naming/Context.html#rebind(javax.naming.Name,%20java.lang.Object)), [rebind](http://docs.google.com/javax/naming/Context.html#rebind(java.lang.String,%20java.lang.Object)), [removeFromEnvironment](http://docs.google.com/javax/naming/Context.html#removeFromEnvironment(java.lang.String)), [rename](http://docs.google.com/javax/naming/Context.html#rename(javax.naming.Name,%20javax.naming.Name)), [rename](http://docs.google.com/javax/naming/Context.html#rename(java.lang.String,%20java.lang.String)), [unbind](http://docs.google.com/javax/naming/Context.html#unbind(javax.naming.Name)), [unbind](http://docs.google.com/javax/naming/Context.html#unbind(java.lang.String)) |

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

### OBJECT\_SCOPE

static final int **OBJECT\_SCOPE**

Constant for expressing interest in events concerning the object named by the target.

The value of this constant is 0.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.naming.event.EventContext.OBJECT_SCOPE)

### ONELEVEL\_SCOPE

static final int **ONELEVEL\_SCOPE**

Constant for expressing interest in events concerning objects in the context named by the target, excluding the context named by the target.

The value of this constant is 1.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.naming.event.EventContext.ONELEVEL_SCOPE)

### SUBTREE\_SCOPE

static final int **SUBTREE\_SCOPE**

Constant for expressing interest in events concerning objects in the subtree of the object named by the target, including the object named by the target.

The value of this constant is 2.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.naming.event.EventContext.SUBTREE_SCOPE)

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

### addNamingListener

void **addNamingListener**([Name](http://docs.google.com/javax/naming/Name.html) target,  
 int scope,  
 [NamingListener](http://docs.google.com/javax/naming/event/NamingListener.html) l)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Adds a listener for receiving naming events fired when the object(s) identified by a target and scope changes. The event source of those events is this context. See the class description for a discussion on event source and target. See the descriptions of the constants OBJECT\_SCOPE, ONELEVEL\_SCOPE, and SUBTREE\_SCOPE to see how scope affects the registration.

target needs to name a context only when scope is ONELEVEL\_SCOPE. target may name a non-context if scope is either OBJECT\_SCOPE or SUBTREE\_SCOPE. Using SUBTREE\_SCOPE for a non-context might be useful, for example, if the caller does not know in advance whether target is a context and just wants to register interest in the (possibly degenerate subtree) rooted at target.

When the listener is notified of an event, the listener may in invoked in a thread other than the one in which addNamingListener() is executed. Care must be taken when multiple threads are accessing the same EventContext concurrently. See the [package description](http://docs.google.com/package-summary.html#THREADING) for more information on threading issues.

**Parameters:**target - A nonnull name to be resolved relative to this context.scope - One of OBJECT\_SCOPE, ONELEVEL\_SCOPE, or SUBTREE\_SCOPE.l - The nonnull listener. **Throws:** [NamingException](http://docs.google.com/javax/naming/NamingException.html) - If a problem was encountered while adding the listener.**See Also:**[removeNamingListener(javax.naming.event.NamingListener)](http://docs.google.com/javax/naming/event/EventContext.html#removeNamingListener(javax.naming.event.NamingListener))

### addNamingListener

void **addNamingListener**([String](http://docs.google.com/java/lang/String.html) target,  
 int scope,  
 [NamingListener](http://docs.google.com/javax/naming/event/NamingListener.html) l)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Adds a listener for receiving naming events fired when the object named by the string target name and scope changes. See the overload that accepts a Name for details.

**Parameters:**target - The nonnull string name of the object resolved relative to this context.scope - One of OBJECT\_SCOPE, ONELEVEL\_SCOPE, or SUBTREE\_SCOPE.l - The nonnull listener. **Throws:** [NamingException](http://docs.google.com/javax/naming/NamingException.html) - If a problem was encountered while adding the listener.**See Also:**[removeNamingListener(javax.naming.event.NamingListener)](http://docs.google.com/javax/naming/event/EventContext.html#removeNamingListener(javax.naming.event.NamingListener))

### removeNamingListener

void **removeNamingListener**([NamingListener](http://docs.google.com/javax/naming/event/NamingListener.html) l)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Removes a listener from receiving naming events fired by this EventContext. The listener may have registered more than once with this EventContext, perhaps with different target/scope arguments. After this method is invoked, the listener will no longer receive events with this EventContext instance as the event source (except for those events already in the process of being dispatched). If the listener was not, or is no longer, registered with this EventContext instance, this method does not do anything.

**Parameters:**l - The nonnull listener. **Throws:** [NamingException](http://docs.google.com/javax/naming/NamingException.html) - If a problem was encountered while removing the listener.**See Also:**[addNamingListener(javax.naming.Name, int, javax.naming.event.NamingListener)](http://docs.google.com/javax/naming/event/EventContext.html#addNamingListener(javax.naming.Name,%20int,%20javax.naming.event.NamingListener))

### targetMustExist

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

Determines whether a listener can register interest in a target that does not exist.

**Returns:**true if the target must exist; false if the target need not exist. **Throws:** [NamingException](http://docs.google.com/javax/naming/NamingException.html) - If the context's behavior in this regard cannot be determined.

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/EventContext.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   [**NEXT CLASS**](http://docs.google.com/javax/naming/event/EventDirContext.html) | [**FRAMES**](http://docs.google.com/index.html?javax/naming/event/EventContext.html)    [**NO FRAMES**](http://docs.google.com/EventContext.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | CONSTR | [METHOD](#tyjcwt) | DETAIL: [FIELD](#1t3h5sf) | CONSTR | [METHOD](#3rdcrjn) |

[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).