

# NSNotification Class Cluster

## Class Cluster Description

NSNotification objects contain notification information communicated from an object to that object's observers. The NSNotification cluster's single public class, NSNotification, declares the programmatic interface for notifications.

The objects you create using this class are referred to as *notification objects* (or simply, *notifications*). They are immutable objects. Because of the nature of class clusters, notification objects are not actual instances of the NSNotification class but of one of its private subclasses. Although a notification object's class is private, its interface is public, as declared by the abstract superclass, NSNotification. (See <sup>a</sup>Class Clusters<sup>o</sup> in the introduction to the Foundation Kit for more information on class clusters and creating subclasses within a cluster.)

You instantiate a notification object directly by sending the **notificationWithName:object:** message to the NSNotification class object. You can also create notifications indirectly, through the NSNotificationCenter class.

## NSNotification

**Inherits From:**        NSObject  
**Conforms To:**        NSCopying

**Declared In:** foundation/NSNotification.h

## Class Description

NSNotification is an abstract class that defines the public interface for notification objects (hereafter, simply referred to as *notifications*). Objects post notifications to the Notification Center to notify their observers (other objects) that a specific condition has occurred. Notifications encapsulate the name and the originator (or sender) of a notification. Accessor methods allow you to obtain both of these things. Additionally, you can create your own, ready-made notifications with the class method, **notificationWithName:sender:**.

However, you do not usually create your own notifications objects this directly. The NSNotificationCenter methods **postNotificationName:object:** and **postNotification:** create and then post notification objects.

You can subclass NSNotification to contain information in addition to name and sender.

## Instance Variables

None declared in this class.

## Adopted Protocols

NSCopying	- copy
	- copyWithZone:

## Method Types

Obtaining notification objects    + **allocWithZone:**  
   + **notificationWithName:object:**

Getting information about a notification

- **notificationName**
- **notificationObject**

## Class Methods

### **allocWithZone:**

+ **allocWithZone:**(NXZone \*)*zone*

Returns an uninitialized concrete notification object. You are responsible for deallocating notification objects that you create with this method.

Typically, you create dictionary objects using the **notification...** class methods, not the **alloc...** and **init...** methods. Note that it's your responsibility to release (with either **release** or **autorelease**) those objects created with the **alloc...** methods.

### **notificationWithName:object:**

+ **notificationWithName:**(NSString \*)*name* **object:***anObject*

Returns a notification object initialized with *name* and *anObject*.

## Instance Methods

### **notificationName**

- (NSString \*)**notificationName**

Returns the name of the notification. Examples of this might be `^PortIsInvalid^` or `^PhoneRinging^`. Typically, you invoke this method on the notification object passed in to your notification-handling or notification-dispatch method.

You can compose notification names with any string. To avoid name collisions, however, you might want to use a prefix that is specific to your application.

## **notificationObject**

### **- notificationObject**

Returns the object that initiated the notification or that is somehow connected to the notification. Typically, you invoke this method on the notification object passed in to your notification-handling or notification-dispatch method.

To illustrate this, let's assume that you've added your object as an observer of a port. When that port dies, the object monitoring that port will post an `^NSPortInvalid^` notification to the Notification Center. The Notification Center then invokes the method you've registered to handle that notification:

```
- (void)handlePortDeath:(NSNotification *)notification
{
    // ...
    [self reclaimResourcesForPort:[notification object]];
    // ...
}
```