

## Contents

- [Using the Transfer Event Model](#)
  - [AfterNew](#)
  - [BeforeCreate](#)
  - [AfterCreate](#)
  - [BeforeUpdate](#)
  - [AfterUpdate](#)
  - [BeforeDelete](#)
  - [AfterDelete](#)
  - [See Also](#)

## Using the Transfer Event Model

Transfer has an event model that can be taken advantage of to notify other CFC's of updates, creates and deletes within the Transfer system, and essentially, the database.

Since ColdFusion 7 does not have interfaces (and we don't really *need* to have them anyway), the methods required are set by convention. The object that is added as an observer must have the appropriate method defined as set below for it to work correctly.

Each event action method, takes an argument of type [transfer.com.events.TransferEvent](#) as an argument. This gives them access to the TransferObject that the event has fired for.

### AfterNew

This event is fired after a TransferObject is created. This is after 'init()' and 'configure()' have been called on the TransferObject.

Required method on the observer:

```
<cffunction name="actionAfterNewTransferEvent" hint="Actions an event before a create happens" access="public" returnType="void" output="false">
  <cfargument name="event" hint="The event object" type="transfer.com.events.TransferEvent" required="Yes">
</cffunction>
```

To add or remove a Observer for a AfterNewEvent:

[Transfer.addAfterNewObserver\(component\)](#)

[Transfer.removeAfterNewObserver\(component\)](#)

### BeforeCreate

This event is fired just before a TransferObject is inserted into the database.

Required method on the observer:

```
<cffunction name="actionBeforeCreateTransferEvent" hint="Actions an event before a create happens" access="public" returnType="void" output="false">
  <cfargument name="event" hint="The event object" type="transfer.com.events.TransferEvent" required="Yes">
</cffunction>
```

To add or remove a Observer for a BeforeCreateEvent:

[Transfer.addBeforeCreateObserver\(component\)](#)

[Transfer.removeBeforeCreateObserver\(component\)](#)

### AfterCreate

This event is fired after a TransferObject is inserted into the database.

Required method on the observer:

```
<cffunction name="actionAfterCreateTransferEvent" hint="Actions an event after a create happens" access="public" returnType="void" output="false">
  <cfargument name="event" hint="The event object" type="transfer.com.events.TransferEvent" required="Yes">
</cffunction>
```

To add or remove an Observer for a AfterCreateEvent:

[Transfer.addAfterCreateObserver\(component\)](#)

[Transfer.removeAfterCreateObserver\(component\)](#)

### BeforeUpdate

This event is fired just before a TransferObject is updated in the database.

Required method on the observer:

```
<cffunction name="actionBeforeUpdateTransferEvent" hint="Actions an event before a update happens" access="public" returnType="void" output="false">
  <cfargument name="event" hint="The event object" type="transfer.com.events.TransferEvent" required="Yes">
</cffunction>
```

Do note, that if there is no change to the updated TransferObject, it will not be updated, and therefore the BeforeUpdate Event will not be fired.

To add or remove a Observer for a BeforeUpdateEvent:

[Transfer.addBeforeUpdateObserver\(component\)](#)

[Transfer.removeBeforeUpdateObserver\(component\)](#)

### AfterUpdate

This event is fired after a TransferObject is updated in the database.

Required method on the observer:

```
<cffunction name="actionAfterUpdateTransferEvent" hint="Actions an event after a update happens" access="public" returnType="void" output="false">
  <cfargument name="event" hint="The event object" type="transfer.com.events.TransferEvent" required="Yes">
</cffunction>
```

Do note, that if there is no change to the updated TransferObject, it will not be updated, and therefore the AfterUpdate Event will not be fired.

To add or remove a Observer for a AfterUpdateEvent:

[Transfer.addAfterUpdateObserver\(component\)](#)

[Transfer.removeAfterUpdateObserver\(component\)](#)

### BeforeDelete

This event is fired just before a TransferObject is deleted in the database.

Required method on the observer:

```
<cffunction name="actionBeforeDeleteTransferEvent" hint="Actions an event before a delete happens" access="public" returnType="void" output="false">
  <cfargument name="event" hint="The event object" type="transfer.com.events.TransferEvent" required="Yes">
</cffunction>
```

To add or remove a Observer for a BeforeDeleteEvent:

[Transfer.addBeforeDeleteObserver\(component\)](#)

[Transfer.removeBeforeDeleteObserver\(component\)](#)

### AfterDelete

This event is fired after a TransferObject is deleted in the database.

Required method on the observer:

```
<cffunction name="actionAfterDeleteTransferEvent" hint="Actions an event after a delete happens" access="public" returnType="void" output="false">
  <cfargument name="event" hint="The event object" type="transfer.com.events.TransferEvent" required="Yes">
</cffunction>
```

To add or remove a Observer for a AfterDeleteEvent:

[Transfer.addAfterDeleteObserver\(component\)](#)

[Transfer.removeAfterDeleteObserver\(component\)](#)

### See Also

- [Working with Transfer ORM: An Event Model Example](#) (Paul Marcotte)

- Paul gives a good overview on how to use the Transfer Event model to provide basic dependency injection
- [My Take on Transfer ORM Event Model - BeforeCreate Example](#) (Bob Silverberg)  
Utilising the Transfer Event Model to set Created and Updated dates on Transfer Objects automatically.

Categories: • [Events](#)