



# Table of Contents

<b><u>Creating Events for Vtiger Actions</u></b> .....	<b>1</b>
<u>The event handler interface</u> .....	1
<u>Registering a handler</u> .....	1
<u>Supported events</u> .....	1
<u>vtiger.entity.beforesave</u> .....	1
<u>vtiger.entity.aftersave</u> .....	1

# Creating Events for Vtiger Actions

When you want to extend vtiger, for example add email notification for a particular action, you end up needing to modify core vtiger code to add your functionality.

The event api provides a simple way of adding actions without making any modification to vtiger.

## The event handler interface

To add a new event handler you will need to define a class that extends the `VTEventHandler` abstract class.

```
class SimpleHandler extends VTEventHandler{
    public function handleEvent($name, $data){
        echo "This handler has been called for the $name event";
    }
}
```

## Registering a handler

```
require 'include/events/include.inc';
$em = new VTEventsManager($adb);
$em->registerHandler($eventName, $filePath, $className);
```

## Supported events

Currently the api supports two events.

### **vtiger.entity.beforesave**

This event is fired before an entity is saved. You will be passed a `VTEntityData` object representing the saved entity. It should be noted that new objects will not have an id. You can modify the contents of `VTEntityObject`.

```
class Handler extends VTEventHandler{
    public function handleEvent($name, $data){
        $data->fieldName = "value";
    }
}
```

### **vtiger.entity.aftersave**

This event is fired after an entity is saved. This too provides a `VTEntityData` object.