Loading

Triggering Events

Using the plugin system in your add-on is fairly simple. The most important part is good planning because, to some degree, you're defining an interface for other people to use.

The first thing you need to do is to load your plug-in group. This is done via the following code:

```
JPluginHelper::importPlugin( 'myplugingroup' );
```

This will load all enabled plug-ins that have defined themselves as part of your group. The next thing you need to do is get an instance of the JDispatcher class like so:

```
$dispatcher =& JDispatcher::getInstance();
```

Notice two things here. First, we are using the getInstance() method, not "new" to create a new instance. That is because we need to get the global singleton instance of the JDispatcher object which contains a list of all the plug-ins available. Second, we are using the =& construct to make sure we have a reference to the instance of the JDispatcher and not a copy. Of course this really only applies to PHP version 4, but since you are a good cross-version developer, you will allow for PHP 4 users.

Next, we need to trigger our custom event:

```
$results = $dispatcher->trigger( 'onCdAddedToLibrary', array( &$artist
, &$title ) );
```

Here we have triggered the event 'onCdAddedToLibrary' and passed in the artist name and title of the track. All plug-ins will receive these parameters, process them and optionally pass back information. You can then handle that information however you like.

In summary, here's the complete example code:

```
JPluginHelper::importPlugin( 'myplugingroup' );
$dispatcher =& JDispatcher::getInstance();
```

```
$results = $dispatcher->trigger( 'onCdAddedToLibrary', array( &$artist
, &$title ) );
```

Note: One thing to notice about the trigger method is that there is nothing defining which group of plug-ins should be notified. In actuality, all plug-ins that have been loaded are notified regardless of the group they are in. So, it's important to make sure you have an event name that does not conflict with any other plug-in group's event name. Most of the time this is not an issue because your component is the one that is loading the plug-in group, so you know which ones are loaded, however be aware that the "system" plugin group is loaded very close to the beginning of the request, so you have to make sure you don't have any event naming conflicts with the system events.