Scheduled Tasks

Plugins

A set of tasks can be registered with the Cron component by making a plugin. Each plugin must respond to the "onCronEvents" trigger. The response from that trigger is an object (stdClass) that returns the plugin's name and an array of callable tasks (event triggers).

Registering Tasks

Plugins should be placed within the cron plugins folder:

```
/myhub
    /plugins
        /cron
```

Here is an example of a cron plugin that registers a set of "mytasks" events.

```
/**
 * Cron plugin for my tasks
 */
class plgCronMytasks extends JPlugin
{
 /**
  * Return a list of events
  *
  * @return     array
  */
 public function onCronEvents()
 {
  // Load the plugin's language file
  $this->loadLanguage();

  // Create the return object
  $obj = new stdClass();

  // Assign the plugin's name
  $obj->plugin = $this->_name;

  // Build the list of callable events
  $obj->events = array(
   array(
```

```
    'name'   => 'doSomething',  // The name of your task
    'label'  => JText::_('PLG_CRON_MYTASKS_DOSOMETHING'),  // Nice lab
el
    'params' => ''  // Name of the params group to load (optional)
   )
  );

  // Return the data
  return $obj;
 }
}
```

As shown in the previous example, each event consist of an array containing three keys: name, label, and params.

name

The plugin must implement a method with the same name as whatever is specified for the name key and the names should match *exactly*. That is, if a name of 'onJumpUpAndDown' is specified, then the plugin **must** have a method of onJumpUpAndDown();.

label

This is a nice, human readable name for the event trigger. It should be a language key with an associated string in the plugin's language file.

params

This is an optional value for specifying a params group (Joomla 1.5) or fieldset (Joomla 1.6+) containing parameters associated with the specific plugin event. This allows for multiple cron jobs calling the same event but with varying values. An example of this can be found in the support tickets cron plugin where the event sendTicketsReminder has a specified params group of 'ticketreminder'. Changing those params would allow, for instance, a job that sends ticket reminders one a month for all open tickets and a ticket reminder once a week for all open and *status: critical* tickets.

A snippet from the support plugin, specifying the list of available tasks:

```
/**
```

```
 * Cron plugin for support tickets
 */
class plgCronSupport extends JPlugin
{
 /**
  * Return a list of events
  *
  * @return     array
  */
 public function onCronEvents()
 {
  $this->loadLanguage();

  $obj = new stdClass();
  $obj->plugin = $this->_name;

  $obj->events = array(
   array(
    'name'   => 'onClosePending',
    'label'  => JText::_('PLG_CRON_SUPPORT_CLOSE_PENDING'),
    'params' => 'ticketpending'
   ),
   array(
    'name'   => 'sendTicketsReminder',
    'label'  =>  JText::_('PLG_CRON_SUPPORT_EMAIL_REMINDER'),
    'params' => 'ticketreminder'
   )
  );

  return $obj;
 }
 ...
}
```

In the support plugin's manifest:

```
...
<fieldset group="ticketreminder">
 <field name="support_ticketreminder_severity" type="list" defaul
t="all" label="Tickets with severity" description="Ticket severit
y to message users about.">
  <option value="all">All</option>
  <option value="critical,major">High</option>
```

```
  <option value="normal">Normal</option>
  <option value="minor,trivial">Low</option>
 </field>
 <field name="support_ticketreminder_group" type="text" menu="hid
e" label="For users in group" default="" description="Only users
within the group specified will be messaged." />
</fieldset>
...
```

Running Tasks

All tasks are run as standard plugin events. Tasks should return a boolean of true upon completion.

See the managers documentation on how to create and schedule jobs.