Introduction

Interspire Knowledge Manager uses an innovative templating system, which allows you to choose from a list of templates, rearrange your content, edit the style of your site, and edit or create your own panels, which -- like the panels provided -- can be placed anywhere on your site.
The Interspire Knowledge Manager release package contains a number of built-in templates that you are free to use. It is simple to install additional templates. The site templates are located in the /templates/ directory of the Interspire Knowledge Manager application. A template with the name CVS will not be selectable from the settings page so avoid using this name for your template.
A number of elements make up the Interspire Knowledge Manager templates. These are explained below:

- Layout Files - HTML template files are used to design the layout of each page. The files consist of HTML with place holders to indicate where the content will be rendered. Different types of content include panels, page data, and config data.
- Panels - Panels contain PHP code to render static or dynamic content, or to perform a specific task. They are positioned on your pages by the HTML template files. Panels may contain other panels, however you want to be careful to limit the depth of including panels or the performance of your site will suffer. Panels are kept in the Interspire Knowledge Manager/templates/ <TemplateName>/Panels directory.
- Images - Each template has its own images to match. These are kept in the Interspire Knowledge Manager/templates/<TemplateName>/Images/ directory.
- Snippets - Snippets are fragments of html which are usually used multiple times. They may be things like a <ol> tag or a table row. Depending on the snippet they may also have placeholders in them. Snippets are kept in the Interspire Knowledge Manager/templates/<TemplateName>/ Snippets directory.
- Style Sheets - Style sheets are used to help with the layout of each page and to apply the theme, including fonts, colors, sizes, etc.
- Misc Files - The Interspire Knowledge Manager templates each require a preview file called Preview.jpg. This image is located in the Interspire Knowledge Manager/templates/ <TemplateName>/ directory. It can be viewed when selecting a template in the installer or on the settings page. Javascript files are kept in the Interspire Knowledge Manager/templates/ <TemplateName/js directory (for the supplied templates).

Layout, Panel and Snippet files may contain placeholders for dynamically generated content. The name and description of each type of placeholder is given here.

- Global variables - Allows you to define a variable in PHP's global scope to be output as part of a panel. For example, if you had this code in your ParsePanel function: $GLOBALS["Test"] = "testing"; Then you would add this place holder into your panels HTML file to output its value: %%GLOBAL_Test%%
- Page data - At this time only the %%Page.Title%% variable is available in Interspire Knowledge Manager
- Config data - The following variables are available
    - %%Config.ImagePath%% - The path to the Images directory for the current templates
    - %%Config.HomePath%% - The path to the Interspire Knowledge Manager installation
    - %%Config.Current.KnowledgebaseUrl%% - The path to the Interspire Knowledge Manager installation (the same as %%Config.HomePath%%, left in for compatability reasons)
    - %%Config.KnowledgebaseName%% - The name of the knowledge base
- Language data - Allows you to insert a language constant into your panel. Language variables are defined in the /includes/language/front_language.ini file. To output an already defined

language variable, you would add this placeholder to your HTML panel file: %%LNG_hpPopularArticles%%

- Snippet data - If you want to include snippet placeholders in your code, you must define the entry in the $GLOBALS['SNIPPETS'] associative array for the placeholder. You can then use the value of that entry by using %%SNIPPET_Name%% where Name is the key in the associative array for the data you want.
- Panel Files - Defined by using %%Panel.PanelName%% where PanelName is the name of the panel in the templates/<TemplateName/Panels directory without the file extension.
- External Files - Defined by using %%Include.Path%% where Path is the path to the file you want to include. Path may be a local filesystem path such as /var/www/mywebsite/header.php or it may be a remote file specified with the full http://www.example.com/dir/file.html address. It can also refer to a file in the panels folder of the template.
  Local files may have PHP in them which will get executed, however any remote files (ones included with http://) that include PHP will not have the PHP code parsed. In both cases any placeholders in the file which are available to panels will be replaced.

Layout Files

Layout Files are responsible for starting the template initialization process. Layout files in Interspire Knowledge Manager don't contain any content, instead they contain placeholders indicating where other types of content should go.
The types of placeholders that can be used in a layout file are

- Panel
- Page data
- Config data
- Global data
- Language data

Layout files themselves can not be included inside of anything else.
Panels
Panels are made up of HTML and PHP to render that HTML. Panels can interact with the objects in the Interspire Knowledge Manager framework.
Each panel may have a PHP file in the display directory with the same name where you would setup any custom placeholders. So for PageFooter.html you would create a file named PageFooter.php in the display directory with the contents (to start with) of

view plain copy to clipboard print ?

```
<?php
  AKB_PAGEFOOTER extends AKB_PANEL
  {
    function SetPanelSettings()
    {
    }
  }
?>
```

The name of the class is the uppercase version of the file name with AKB_ prepended to the start and any instances of the word panel replaced with _PANEL. The class function that gets called is always SetPanelSettings inside of the panel's class, however this may call other functions inside or outside of the class to get whatever information you wish to display in the panel.
The types of placeholders that can be used in a panel file are:

- Panel

- Global data
- Language data

Panel files can be included in:

- Layout Files
- Panel Files

Snippets

The types of placeholders that can be used in a snippet file are:

- Page data
- Config data
- Global data
- Language data

Panel files can be include in:

- Layout Files
- Panel Files

Style Sheets

Style Sheet Files

Style sheets are used to help with the layout of each page and to apply the theme, including fonts, colors, sizes, etc. Each template contains 2 style sheets, located in the templates/<TemplateName>/Styles directory.

- Styles.css - General styles used on the pages of your site.
- WindowStyles.css - Styles used on all pop-up windows, such as 'Print Article'.

The Windows template contains 2 additional stylesheets relating to it's tree panel.

- tab.css - Styles for the look and feel of the tab layout of the left panel
- xtree.css - Styles relating to the look and feel of the javascript tree used for navigation

Container Styles

A couple of common container styles include (note, they aren't necessarily part of every template)

- GlobalContainer - This style is typically applied to a tag wrapped around the entire page immediately inside the <form> tags.
- PageContainer - This style is typically applied to a <table> tag that defines the position of the page header, page body, and page footer.
- HeaderContainer - This style is typically applied to a <td> tag inside the PageContainer that helps position the page header.
- BodyContainer - This style is typically applied to a <td> tag inside the PageContainer that helps position the body of your page.
- FooterContainer - This style is typically applied to a <td> tag inside the PageContainer that helps position the page footer.

Body Styles (Columns) The page body is typically divided into columns. These are cells of a table with the class "BodyColumns". The classes are:

- BodyColumns - The style typically applied to the <table> tag that defines the body columns.
- LeftColumn - The style typically applied to the leftmost <td> tag inside the BodyColumns table.
- CentreColumn - The style typically applied to the centre <td> tag inside the BodyColumns table.
- RightColumn - The style typically applied to the rightmost <td> tag inside the BodyColumns table.

The column styles can be adjusted to change background colors, borders, spacing, even the styles of child content.

Panel Styles

Note: The following instructions may not apply to all templates and are simple a rough guide based on the built-in templates.

Each panel has the class of 'Panel', as well a panel specific class, which is usually the name of the panel. You can edit the design of the panel by opening the Styles.css file and searching for the class name (ie. the name of the panel). If it doesn't exist, then you can create the class yourself.

To edit styles that are applied to all panels or panels in a specific area, you can use the 'Panel' class. You'll notice that the headings in the centre column are declared using .CentreColumn .Heading rather than .CentreColumn .Panel .Heading. This format is for simplicity, however both methods have the same effect. Form Styles The forms are broken down into the following styles:

- FieldLabel - The style of all form field labels.
- Field400 - The style of a field that is 400px wide.

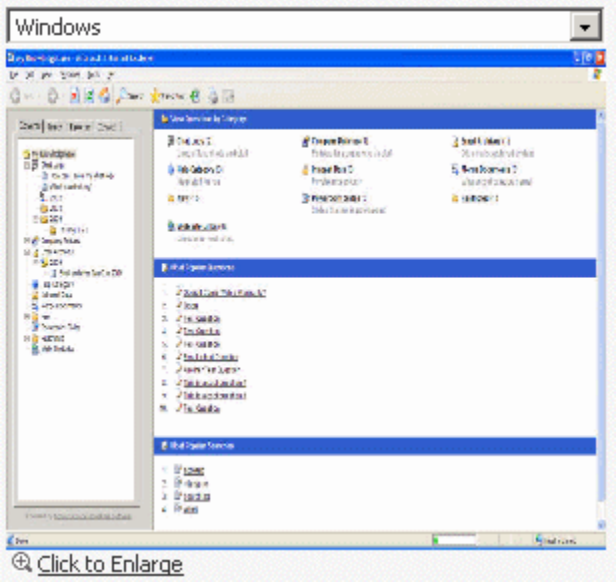
These width specific styles are used on forms with a uniformed field width.

- FieldFull - The style of a field that is 100% wide.
- Field - The style of any other field

FormButton - The style of all form submit and cancel buttons

Applying a Template

There are two places where you can select a template to apply to your site: in the installer and on the settings page in the control panel. When selecting a template you will see the preview image:



To view a larger preview, click the 'Click to Enlarge' link. This will bring up a new window containing the full size preview image. To select a template, simply select the template in the drop down list and then click on the form submit button.

Custom Templates

It's not difficult to set up your own totally customizable template and control its look, feel, and even behavior, of your site. There are two ways to get started: You can either create a new directory in the /templates/ directory of the Interspire Knowledge Manager application and use the instructions below to create all of the files, or you can copy an existing template and place it into the /templates/ directory then use the instructions below to make the changes you want.
Just follow the steps below to set up your template: Note: The default HTML forms already exists.

1. Panels - Panels are the content displayed on your site. You'll need to add these first or there won't be anything to display.

2. HTML Template Files - HTML template files are used to specify the layout of each page. Start by adding these to your template. See HTML Template Files for more.
3. Style Sheets - Style sheets are used to help with the layout of each page and to apply fonts, colors, sizes, etc. Now add these to your template. See Style Sheets for more.
4. Images - Each template typically has its own collection of images. You'll need to add these to the templates/<TemplateName>/Images/ directory.
5. Preview Image - Each template needs a preview image. Once your template is set up, take a screenshot of it and save it to the templates/<TemplateName>/Preview.jpg file.

Thinking Ahead

The most important thing to take into consideration before starting any customisation is how you should make customisations in a way that will allow you to upgrade Interspire Knowledge Manager when a new version is released, without losing your changes. The best way to do this is to create your own template. The reason that creating your own template is the best way to avoid your changes being overwritten is that when you get the new version, updating becomes a simple matter of uploading all the files (except the config.php one of course) over the top of your exisiting installation.
To start with, try out each of the included templates and then choose the one which most closely mirrors your needs. If you are unsure which to choose, the Default template uses css to create it's layout as well as it's look and feel so that should be a pretty good starting point.
Once you've chosen the template you're going to base your customised version on, make a copy of it and give it a name, such as "MyTemplate". This can be almost anything you want, but you must follow these guidelines:

- It can not be named CVS. The code on the settings page explicitly hides this directory to make things a little nicer on us during development.
- It must only contain the standard english characters (a-z, A-Z, 0-9, _ - or +).

Your custom template

Now that you have your own template, what's next? Well, the way most people get started with customising their template is to make changes to the style sheet. Say for example we decided that the Default template was what we were going to base our custom template off of, however we want our design to stretch out to fill the whole browser window. What we would do is edit the Styles.css for our custom template (in the Styles directory) and change the two instances of width: 750px; to width: 100%; to make the contents stretch to fill the whole window. For more details on how to get started with style sheets, there is a guide at Webmonkey that can help you get started.

Language Packs

Interspire Knowledge Manager is completely language packed and all language variables are stored in PHP INI files. The language variables for the front end of Interspire Knowledge Manager are stored in the /includes/language/front_language.ini file, whereas the language variables for the control panel are stored in the /includes/language/back_language.ini file. If you'd like to convert your site to a language other than English, you can simply edit these 2 language files to change the variables as needed. Also, if you notice a language placeholder in the panel files that you want to change, follow these steps:

1. Add your new language variable into the language file.
2. Edit the panel file, which is located in templates/{Template Name}/panels and add a place holder for the language variable. A language variable place holder looks like this: %%LNG_{Language_Variable_Name}%% For example: %%LNG_myTest%%

Integration

Table of Contents

Active Response System

Interspire Knowledge Manager has a built-in contact system that your knowledgebase visitors can use to send questions/comments. This can optionally be disabled from the control panel by unchecking the "Enable Contact Page" option.
Also, you can choose how to receive your visitors questions/comments: via email, or via email and having them saved as pending questions in your Interspire Knowledge Manager control panel. If you are not familiar with the Active Response System, please click here.
The power of Interspire Knowledge Manager's Active Response System lies in the ability to add it to any page on your web site. Follow the steps below to add Interspire Knowledge Manager's Active Response System to any page on your web site. For our example, we will create a simple form with a text box.

- Create a new web page. Let's call it ars_test.html
- In the <head> section of the page, add these lines:
  Example3.1.Example code required in head of file to enable ARS

```
view plain  copy to clipboard  print  ?
01. <script type="text/javascript">var akbPath = '[PATH_TO_KB]';</script>
02. <script type="text/javascript" src="[PATH_TO_KB]/javascript/ars.js"></script>
03. <link rel="stylesheet" type="text/css" href="[PATH_TO_KB]/templates/Classic/
    Styles.css">
```

  Replace [PATH_TO_KB] with the complete path to the directory where you have Interspire Knowledge Manager installed, such as http://www.mywebsite.com/Interspire Knowledge Manager -- Do not include any trailing slashes.

  > **Note**
  > Your form and Interspire Knowledge Manager MUST be the same domain otherwise the ARS system won't work because of the browsers security settings.

- Next, we'll create a text box. When you type anything into this text box, Interspire Knowledge Manager's Active Response System will use JavaScript and client-side XML to retrieve a list of questions in your knowledgebase that match what is typed. Add this text box to your page:
  Example3.2.Example code required to enable ARS in a text box

```
view plain  copy to clipboard  print  ?
01. <textarea
    name="Message" id="Message" class="Field300" rows="8" onkeyup="ARS(this.id);
    "></textarea>
02.
03. <div id="SearchResults"></div>
```

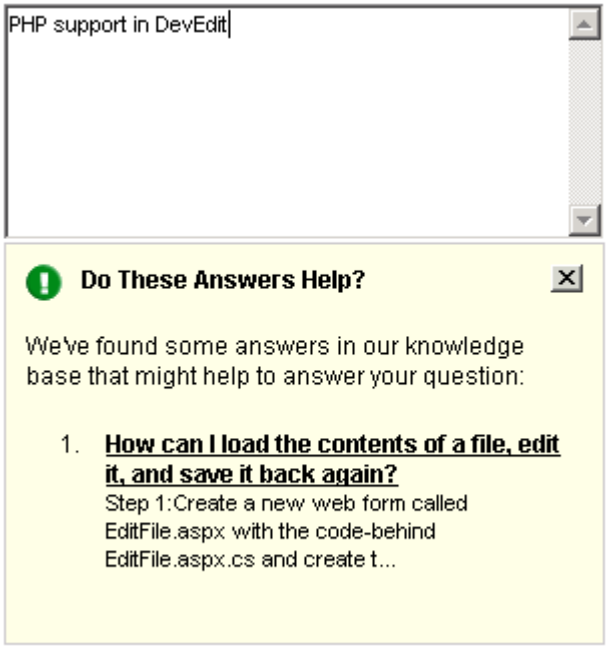
  Example3.2.5.Optional Example code to limit the ARS to a particular category*[Active KB NX2.6+ Only]*
  As of Interspire Knowledge Manager NX2.6, it's possible to specify a select box of categories to narrow down search results.
  To do this create a select box with an id of 'categories' and add the categories that you wish to search on. The value attribute of each category must match up with the category name from your Interspire Knowledge Manager database. Note that a blank value in the select box will search on all categories.

```
view plain  copy to clipboard  print  ?
```

```
<select id="categories">
<option value="">No Category</option>
<option value="Product Updates">Product Updates</option>

<option value="Registration">Registration</option>
</select>
```

- Here's an example using a copy of Interspire Knowledge Manager NX2 setup on Interspire.com. Note that this works on all modern web browsers, including Mozilla:



- Alternatively, if you didn't want to include the whole Interspire Knowledge Manager style sheet in the page with your text area, you could remove the line that references the stylesheet.
Example3.3.Line To remove to not include the Interspire Knowledge Manager style sheet
<link rel="stylesheet" type="text/css" href="[PATH_TO_KB]/templates/Classic/Styles.css">
Example3.4.Example styles to format the look of the ARS popup

```
<style type="text/css">
    .FieldInfo
    {
        width: 400px;
        padding: 10px;
        background-color: #FFFFE7;
        border: solid 1px #CACACA;
        overflow: auto;
        font: normal 12px Verdana;
    }
    .FieldInfo li
    {
        margin-bottom: 15px;
    }
    .CloseIcon
    {
```

```
            float: right;
            cursor: pointer;
        }
    </style>
```

Using the API

Introduction

Interspire Knowledge Manager uses an API (or Application Programming Interface) for its data creation, deletion and updating. The files for this API are in the Interspire Knowledge Manager/lib/api directory. The file class.api.php is the parent class which is inherited by each sub class and handles all data orientated tasks. Child classes are responsible for defining things like the table names to store data, which fields are valid and how to check if they are valid or not.

Creating

If you wanted to create a category programmatically, you would define the values you wanted to create in the $_POST PHP super global variable and then create a new API_CATEGORY class and call it's create() function, like this:

Example3.5.Creating a category programmatically

view plain copy to clipboard print ?

```php
<?php
    $_POST['name'] = 'My Test Category';
    $_POST['description'] = 'A test category created programmatically';
    $_POST['parentid'] = 0; // A parentid of 0 is a top level category
    $_POST['pass'] = ''; // Don't put a password on this category
    $_POST['icon'] = '1.gif'; // We'll just use the default icon

    // The init file handles all the things like connecting to the database and including our API files
    require_once('/path/to/Interspire Knowledge Manager/init.php');

    // We need to include the language file in case the API gives us an error
    $GLOBALS['AKB_LANG'] = parse_ini_file('/path/to/Interspire Knowledge Manager/includes/language/front_language.ini');

    $c = new API_CATEGORY();

    $newid = $c->create();

    if ($newid > 0) {
        echo "Category created successfully<br />\n";
    } else {
        echo "There was a problem creating the category, reason: ".$c->error."<br />\n";
    }
?>
```

Example3.6.Creating a comment programmatically

view plain copy to clipboard print ?

```php
<?php

    $_POST['questionid'] = 130; // Id of the question the comment is to be associated with
    $_POST['name'] = 'Joe Bloggs'; // Name of the commenter
    $_POST['email'] = 'joe@example.com'; // Email of the person commenting
    $_POST['comment'] = 'Creating comments programmatically is fun !'; // The users comment
```

```
    $_POST['status'] = 'pending'; // Can be pending, approved or disapproved

    // The init file handles all the things like connecting to the database and including our API files
    require_once('/path/to/Interspire Knowledge Manager/init.php');

    // We need to include the language file in case the API gives us an error
    $GLOBALS['AKB_LANG'] = parse_ini_file('/path/to/Interspire Knowledge Manager/includes/language/front_language.ini');

    $c = new API_COMMENT();

    $newid = $c->create();

    if ($newid > 0) {
        echo "Comment created successfully<br />\n";
    } else {
        echo "There was a problem creating the comment, reason: ".$c->error."<br />\n";
    }
?>
```

Example3.7.Creating a glossary term programmatically

view plain copy to clipboard print ?

```
<?php

$_POST['word'] = 'programmatically'; // The glossary term
$_POST['description'] = 'Using programming to accomplish a task.'; // The definition of the word

// The init file handles all the things like connecting to the database and including our API files
require_once('/path/to/Interspire Knowledge Manager/init.php');

// We need to include the language file in case the API gives us an error
$GLOBALS['AKB_LANG'] = parse_ini_file('/path/to/Interspire Knowledge Manager/includes/language/front_language.ini');

$g = new API_GLOSSARY();

$newid = $g->create();

if ($newid > 0) {
    echo "Glossary term created successfully<br />\n";
} else {
    echo "There was a problem creating the glossary term, reason: ".$g->error."<br />\n";
}
?>
```

Example3.8.Creating a group programmatically

view plain copy to clipboard print ?

```
<?php

    // The init file handles all the things like connecting to the database and including our API files
    require_once('/path/to/Interspire Knowledge Manager/init.php');

    // We need to include the language file in case the API gives us an error
```

```
07.    $GLOBALS['AKB_LANG'] = parse_ini_file('/path/to/Interspire Knowledge Manager/includes/
       language/front_language.ini');
08.
09.    $g = new API_GROUP();
10.
11.    $_POST['name'] = 'Sales and support'; // The name of the group
12.    $_POST['contactable'] = 1; // Optional, display the group on the contact page (0 = default, no, 1
       = yes)
13.
14.    // Set the group permissions to their defaults
15.    $g->perms = $g->defaultPerms;
16.    $g->applies = $g->defaultApplies;
17.
18.    $g->perms['question']['create'] = true; // Allow users in this group to create questions,
19.     // Refer to the lib/api/class.group.php file for the other permissions you can set
20.
21.    $newid = $g->create();
22.
23.    if ($newid > 0) {
24.        echo "Group created successfully<br />\n";
25.    } else {
26.        echo "There was a problem creating the group, reason: ".$g->error."<br />\n";
27.    }
28. ?>
```

Example3.9.Creating a question programmatically

view plain copy to clipboard print ?

```
01. <?php
02.
03.    $_POST['title'] = 'My dynamically created question'; // The title of the question
04.    $_POST['answer'] = '<b>Hooray</b> we have added a question !'; // The answer to the
       question
05.    $_POST['related'] = '1,3,5,7,9'; // Command seperated list of related question ids
06.    $_POST['detectrelated'] = 0; // Should we detect related questions or use those specified (0 =
       use specified, 1 = auto detect)
07.    $_POST['visible'] = 1; // Is the question visible or not (0 = hidden, 1 = visible)
08.    $_POST['userid'] = 1; // The user id of the person who created the question
09.    $_POST['views'] = 0; // How many times has this question been viewed
10.    $_POST['posvotes'] = 0; // Number of positive votes
11.    $_POST['negvotes'] = 0; // Number of negative votes
12.    $_POST['score'] = 0; // Score for ranking
13.    $_POST['emailed'] = 0; // Counter of how many times this question has been emailed to a
       friend
14.    $_POST['sortorder'] = 0; // Search results are ordered by sortorder before their name
15.    $_POST['metakeywords'] = 'dynamic, question, generated'; // Keywords for the meta tags
16.    $_POST['metadescription'] = 'Dynamically created question'; // Description for the meta tags
17.
18.    $_POST['lastupdated'] = '2006-10-26 11:00:00'; // Lastupdated is optional - if you don't specify it
       the current time will be used
19.
20.
21.    // The init file handles all the things like connecting to the database and including our API files
22.    require_once('/path/to/Interspire Knowledge Manager/init.php');
```

```
23.
24.    // We need to include the language file in case the API gives us an error
25.    $GLOBALS['AKB_LANG'] = parse_ini_file('/path/to/Interspire Knowledge Manager/includes/
       language/front_language.ini');
26.
27.    $q = new API_QUESTION();
28.
29.    $newid = $q->create();
30.
31.    if ($newid > 0) {
32.        echo "Question created successfully<br />\n";
33.    } else {
34.        echo "There was a problem creating the question, reason: ".$q->error."<br />\n";
35.    }
36. ?>
```

Example3.10.Creating a rating programmatically

A rating is used for keeping track of who has rated which question, to try and cut down on questions being rated by the same person thousands of times

view plain copy to clipboard print ?

```
01. <?php
02.
03.    // The init file handles all the things like connecting to the database and including our API files
04.    require_once('/path/to/Interspire Knowledge Manager/init.php');
05.
06.    // We need to include the language file in case the API gives us an error
07.    $GLOBALS['AKB_LANG'] = parse_ini_file('/path/to/Interspire Knowledge Manager/includes/
       language/front_language.ini');
08.
09.    $r = new API_RATING();
10.
11.    $_SERVER['REMOTE_ADDR'] = '127.0.0.1'; // Manually set the ip our request is coming from
12.    $_POST['questionid'] = 1337; // The questionid that was being rated
13.
14.    $newid = $r->create();
15.
16.    if ($newid > 0) {
17.     echo "Rating created successfully<br />\n";
18.    } else {
19.     echo "There was a problem creating the rating, reason: ".$r->error."<br />\n";
20.    }
21.
22. ?>
```

Example3.11.Creating a submitted question programmatically

view plain copy to clipboard print ?

```
01. <?php
02.
03.    $_POST['name'] = 'Joe Bloggs'; // The name of the submitter
04.    $_POST['email'] = 'joe@example.com'; // The submitter's email address
05.    $_POST['subject'] = 'I have a question'; // The subject of their question
06.    $_POST['question'] = 'Why is the sky blue ?'; // The actual question
07.
08.    // The init file handles all the things like connecting to the database and including our API files
```

```
09.    require_once('/path/to/Interspire Knowledge Manager/init.php');
10.
11.    // We need to include the language file in case the API gives us an error
12.    $GLOBALS['AKB_LANG'] = parse_ini_file('/path/to/Interspire Knowledge Manager/includes/language/front_language.ini');
13.
14.    $sq = new API_SUBMITTEDQUESTION();
15.
16.    $newid = $sq->create();
17.
18.    if ($newid > 0) {
19.        echo "Submitted question created successfully<br />\n";
20.    } else {
21.        echo "There was a problem creating the submitted question, reason: ".$sq->error."<br />\n";
22.    }
23. ?>
```

Example3.12.Creating a subscriber programmatically

When a subscriber is created, the date is recorded automatically. By default the users is set to be unconfirmed. If you want to confirm them you have to update the user after they have been created.

view plain copy to clipboard print ?

```
01. <?php
02.
03.    $_POST['email'] = 'joe@example.com'; // The subscriber's email address
04.    $_POST['questionid'] = 130; // The id of the question the person has subscribed to
05.
06.    // The init file handles all the things like connecting to the database and including our API files
07.    require_once('/path/to/Interspire Knowledge Manager/init.php');
08.
09.    // We need to include the language file in case the API gives us an error
10.    $GLOBALS['AKB_LANG'] = parse_ini_file('/path/to/Interspire Knowledge Manager/includes/language/front_language.ini');
11.
12.    $sub = new API_SUBSCRIBER();
13.
14.    $newid = $sub->create();
15.
16.    if ($newid > 0) {
17.        echo "Subscriber created successfully<br />\n";
18.    } else {
19.        echo "There was a problem creating the subscriber, reason: ".$sub->error."<br />\n";
20.    }
21.
22. ?>
```

Example3.13.Creating a user programmatically

view plain copy to clipboard print ?

```
01. <?php
02.
03.    $_POST['username'] = 'jbloggs'; // The users login
04.    $_POST['password'] = 'password'; // The user's password. This will be encrypted before saving so no need to encrypt it yourself
05.    $_POST['firstname'] = 'Joe'; // The users first name
06.    $_POST['lastname'] = 'Bloggs'; // The users last name
```

```
07.    $_POST['status'] = 1; // Is the user account active ? (0 = disabled, 1 = active)
08.
09.    // The init file handles all the things like connecting to the database and including our API files
10.    require_once('/path/to/Interspire Knowledge Manager/init.php');
11.
12.    // We need to include the language file in case the API gives us an error
13.    $GLOBALS['AKB_LANG'] = parse_ini_file('/path/to/Interspire Knowledge Manager/includes/
    language/front_language.ini');
14.
15.    $u = new API_USER();
16.
17.    $newid = $u->create();
18.
19.    if ($newid > 0) {
20.        echo "User created successfully<br />\n";
21.    } else {
22.        echo "There was a problem creating the user, reason: ".$u->error."<br />\n";
23.    }
24.
25. ?>
```

Updating

Let's say we wanted to take our previous example and update the the name of the category to "My Updated Test Category". Assuming we knew the id of the category we want to update, then updating it is simple.

Example3.14.Updating a category programmatically

view plain copy to clipboard print ?

```
01. <?php
02.
03.    // The init file handles all the things like connecting to the database and including our API files
04.    require_once('/path/to/Interspire Knowledge Manager/init.php');
05.
06.    // We need to include the language file in case the API gives us an error
07.    $GLOBALS['AKB_LANG'] = parse_ini_file('/path/to/Interspire Knowledge Manager/includes/
    language/front_language.ini');
08.
09.    $c = new API_CATEGORY();
10.    $c->load(130); // Assuming the category you want to update has the id 130 in the database
11.
12.    $_POST['name'] = 'My Updated Test Category';
13.    $_POST['metakeywords'] = 'dynamic, question, generated, updated';
14.
15.    if ($c->save()) {
16.        echo "Question updated successfully<br />\n";
17.    } else {
18.        echo "There was a problem updating the question, reason: ".$c->error."<br />\n";
19.    }
20.
21. ?>
```

Example3.15.Updating a comment programmatically

view plain copy to clipboard print ?

```
01. <?php
02.
```

```
03.    // The init file handles all the things like connecting to the database and including our API files
04.    require_once('/path/to/Interspire Knowledge Manager/init.php');
05.
06.    // We need to include the language file in case the API gives us an error
07.    $GLOBALS['AKB_LANG'] = parse_ini_file('/path/to/Interspire Knowledge Manager/includes/
       language/front_language.ini');
08.
09.    $c = new API_COMMENT();
10.    $c->load(1337); // Assuming the comment you want to update has the id 1337 in the database
11.
12.    $_POST['status'] = 'approved';
13.
14.    if ($c->save()) {
15.        echo "Question updated successfully<br />\n";
16.    } else {
17.        echo "There was a problem updating the question, reason: ".$c->error."<br />\n";
18.    }
19.
20. ?>
```

Example3.16.Updating a glossary term programmatically

view plain copy to clipboard print ?

```
01. <?php
02.
03.    // The init file handles all the things like connecting to the database and including our API files
04.    require_once('/path/to/Interspire Knowledge Manager/init.php');
05.
06.    // We need to include the language file in case the API gives us an error
07.    $GLOBALS['AKB_LANG'] = parse_ini_file('/path/to/Interspire Knowledge Manager/includes/
       language/front_language.ini');
08.
09.    $g = new API_GLOSSARY();
10.    $g->load(130); // Assuming the glossary term you want to update has the id 130 in the
       database
11.
12.    $_POST['description'] = $g->description.' In this case it is by using PHP'; // Append some text to
       our exisiting description
13.
14.    if ($g->save()) {
15.        echo "Glossary term updated successfully<br />\n";
16.    } else {
17.        echo "There was a problem updating the glossary term, reason: ".$g->error."<br />\n";
18.    }
19.
20. ?>
```

Example3.17.Updating a group programmatically

view plain copy to clipboard print ?

```
01. <?php
02.
03.    // The init file handles all the things like connecting to the database and including our API files
04.    require_once('/path/to/Interspire Knowledge Manager/init.php');
```

```
05.
06.    // We need to include the language file in case the API gives us an error
07.    $GLOBALS['AKB_LANG'] = parse_ini_file('/path/to/Interspire Knowledge Manager/includes/
       language/front_language.ini');
08.
09.    $g = new API_GROUP();
10.    $g->load(130); // Assuming the group you want to update has the id 130 in the database
11.
12.    $_POST['name'] = 'Sales';
13.
14.    if ($g->save()) {
15.        echo "Group updated successfully<br />\n";
16.    } else {
17.        echo "There was a problem updating the group, reason: ".$g->error."<br />\n";
18.    }
19.
20. ?>
```

Example3.18.Updating a question programmatically

view plain copy to clipboard print ?

```
01. <?php
02.
03.    // The init file handles all the things like connecting to the database and including our API files
04.    require_once('/path/to/Interspire Knowledge Manager/init.php');
05.
06.    // We need to include the language file in case the API gives us an error
07.    $GLOBALS['AKB_LANG'] = parse_ini_file('/path/to/Interspire Knowledge Manager/includes/
       language/front_language.ini');
08.
09.    $q = new API_QUESTION();
10.    $q->load(130); // Assuming the question you want to update has the id 130 in the database
11.
12.    $_POST['title'] = 'My Updated Question Title';
13.
14.    if ($q->save()) {
15.        echo "Category updated successfully<br />\n";
16.    } else {
17.        echo "There was a problem updating the category, reason: ".$q->error."<br />\n";
18.    }
19.
20. ?>
```

Example3.19.Updating a rating programmatically

view plain copy to clipboard print ?

```
01. <?php
02.
03.    // The init file handles all the things like connecting to the database and including our API files
04.    require_once('/path/to/Interspire Knowledge Manager/init.php');
05.
06.    // We need to include the language file in case the API gives us an error
07.    $GLOBALS['AKB_LANG'] = parse_ini_file('/path/to/Interspire Knowledge Manager/includes/
       language/front_language.ini');
08.
09.    $r = new API_RATING();
```

```
    $r->load(130); // Assuming the rating you want to update has the id 130 in the database

    $_POST['ip'] = '192.168.0.1';

    if ($r->save()) {
        echo "Rating updated successfully<br />\n";
    } else {
        echo "There was a problem updating the rating, reason: ".$r->error."<br />\n";
    }

?>
```

Example3.20.Updating a submitted question programmatically

view plain copy to clipboard print ?

```
<?php

    // The init file handles all the things like connecting to the database and including our API files
    require_once('/path/to/Interspire Knowledge Manager/init.php');

    // We need to include the language file in case the API gives us an error
    $GLOBALS['AKB_LANG'] = parse_ini_file('/path/to/Interspire Knowledge Manager/includes/language/front_language.ini');

    $sq = new API_SUBMITTEDQUESTION();
    $sq->load(130); // Assuming the submitted question you want to update has the id 130 in the database

    $_POST['email'] = 'mbloggs@example.com';
    $_POST['name'] = 'Mary Bloggs';
    $_POST['subject'] = 'Has Joe been using my computer again ?';

    if ($sq->save()) {
        echo "Submitted question updated successfully<br />\n";
    } else {
        echo "There was a problem updating the submitted question, reason: ".$sq->error."<br />\n";
    }

?>
```

Example3.21.Updating a subscriber programmatically

This example is a little more specific then the others to demonstrate how to access values after you have performed a load()

view plain copy to clipboard print ?

```
<?php

    // The init file handles all the things like connecting to the database and including our API files
    require_once('/path/to/Interspire Knowledge Manager/init.php');

    // We need to include the language file in case the API gives us an error
    $GLOBALS['AKB_LANG'] = parse_ini_file('/path/to/Interspire Knowledge Manager/includes/language/front_language.ini');

    $sub = new API_SUBSCRIBER();
```

```
10.    $sub->load(130); // Assuming the subscriber you want to update has the id 130 in the
    database
11.
12.    // Once we have loaded an entry, it's values are available as class variables
13.    if ($sub->confirmed == 0 && $sub->confirmcode == $_POST['confirmcode']) {
14.     $_POST['confirmd'] = 1;
15.    }
16.
17.    if ($sub->save()) {
18.        echo "Subscriber confirmed successfully<br />\n";
19.    } else {
20.        echo "There was a problem confirming the subscriber, reason: ".$sub->error."<br />\n";
21.    }
22.
23. ?>
```

Example3.22.Updating a user programmatically

This example also shows you how to use the find() function to find the row by any unique column on it's database table.

view plain copy to clipboard print ?

```
01. <?php
02.
03.    // The init file handles all the things like connecting to the database and including our API files
04.    require_once('/path/to/Interspire Knowledge Manager/init.php');
05.
06.    // We need to include the language file in case the API gives us an error
07.    $GLOBALS['AKB_LANG'] = parse_ini_file('/path/to/Interspire Knowledge Manager/includes/
    language/front_language.ini');
08.
09.    $u = new API_USER();
10.    $u->load(130); // Assuming the user you want to update has the id 130 in the database
11.
12.    $_POST['password'] = 'newpass';
13.
14.    if ($u->save()) {
15.        echo "Password updated successfully<br />\n";
16.    } else {
17.        echo "There was a problem updating the users password, reason: ".$u->error."<br />\n";
18.    }
19.
20.
21.    // Since the user table has a unique index on the username field, we can also load a value by
    name
22.    $u2 = new API_USER();
23.    $u2->find('username', 'jbloggs');
24.
25.    $_POST['password'] = 'joesnewpass';
26.    if ($u2->save()) {
27.        echo "Password updated successfully<br />\n";
28.    } else {
29.        echo "There was a problem updating the users password, reason: ".$u2->error."<br />\n";
30.    }
31.
```

```
32. ?>
```

Deleting
Deleting can be done in two ways. You can delete a single entry or you can delete multiple entries. The functions for doing this are delete and multiDelete respectively. Here is an example of each. Deleting is the same for all items using the API so only 4 examples are show for both styles of deleting to save space. All you have to do is adjust the line which creates a new API object to your specific needs.

Deleting a single item

Example3.23.Deleting a category programmatically

view plain copy to clipboard print ?

```
01. <?php
02.
03.    // The init file handles all the things like connecting to the database and including our API files
04.    require_once('/path/to/Interspire Knowledge Manager/init.php');
05.
06.    // We need to include the language file in case the API gives us an error
07.    $GLOBALS['AKB_LANG'] = parse_ini_file('/path/to/Interspire Knowledge Manager/includes/
    language/front_language.ini');
08.
09.    $c = new API_CATEGORY();
10.
11.    // Delete the category with the id 130
12.    if ($c->delete(130)) {
13.     echo "The category has been deleted successfully<br />\n";
14.    } else {
15.     echo "There was a problem deleting the category, reason: ".$c->error."<br />\n";
16.    }
17.
18. ?>
```

Example3.24.Deleting a comment programmatically

view plain copy to clipboard print ?

```
01. <?php
02.
03.    // The init file handles all the things like connecting to the database and including our
    API files
04.    require_once('/path/to/Interspire Knowledge Manager/init.php');
05.
06.    // We need to include the language file in case the API gives us an error
07.    $GLOBALS['AKB_LANG'] = parse_ini_file('/path/to/Interspire Knowledge Manager/
    includes/language/front_language.ini');
08.
09.    $c = new API_COMMENT();
10.
11.    // Delete the comment with the id 1337
12.    if ($c->delete(1337)) {
13.        echo "The comment has been deleted successfully<br />\n";
14.    } else {
15.        echo "There was a problem deleting the comment, reason: ".$c->error."<br />\n";
```

```
16.    }
17.
18. ?>
```

Example3.25.Deleting a glossary term programmatically

view plain copy to clipboard print ?

```
01. <?php
02.
03.    // The init file handles all the things like connecting to the database and including our
    API files
04.    require_once('/path/to/Interspire Knowledge Manager/init.php');
05.
06.    // We need to include the language file in case the API gives us an error
07.    $GLOBALS['AKB_LANG'] = parse_ini_file('/path/to/Interspire Knowledge Manager/
    includes/language/front_language.ini');
08.
09.    $g = new API_GLOSSARY();
10.
11.    // Delete the glossary term with the id 130
12.    if ($g->delete(130)) {
13.        echo "The glossary term has been deleted successfully<br />\n";
14.    } else {
15.        echo "There was a problem deleting the glossary term, reason: ".$g->error."<br />\n";
16.    }
17.
18. ?>
```

Example3.26.Deleting a question programmatically

view plain copy to clipboard print ?

```
01. <?php
02.
03.    // The init file handles all the things like connecting to the database and including our
    API files
04.    require_once('/path/to/Interspire Knowledge Manager/init.php');
05.
06.    // We need to include the language file in case the API gives us an error
07.    $GLOBALS['AKB_LANG'] = parse_ini_file('/path/to/Interspire Knowledge Manager/
    includes/language/front_language.ini');
08.
09.    $q = new API_QUESTION();
10.
11.    // Delete the question with the id 130
12.    if ($q->delete(130)) {
13.        echo "The question has been deleted successfully<br />\n";
14.    } else {
15.        echo "There was a problem deleting the question, reason: ".$q->error."<br />\n";
16.    }
17.
18. ?>
```

Deleting multiple items

Example3.27.Deleting multiple categories programmatically

view plain copy to clipboard print ?

```php
<?php

    // The init file handles all the things like connecting to the database and including our API files
    require_once('/path/to/Interspire Knowledge Manager/init.php');

    // We need to include the language file in case the API gives us an error
    $GLOBALS['AKB_LANG'] = parse_ini_file('/path/to/Interspire Knowledge Manager/includes/language/front_language.ini');

    $c = new API_CATEGORY();

    // Delete the categories with the id's 1, 2, 3, 4 or 5
    $ids_to_delete = array (1, 2, 3, 4, 5);

    if ($c->multiDelete($ids_to_delete);) {
        echo "The categories have been deleted successfully<br />\n";
    } else {
        echo "There was a problem deleting the categories, reason: ".$c->error."<br />\n";
    }

?>
```