

# ResourceSpace Function Reference

Dan Huby, March 2009.

## Application Overview

The application is coded in a procedural fashion for simplicity and for maximum portability. All application logic is contained in functions within the included files in the 'include' directory.

## Function List

**AltImageRotate** (\$src\_img, \$angle) {  
image\_processing.php

**CheckDBStruct** (\$path)

*Check the database structure against the text files stored in \$path. Add tables / columns / data / indices as necessary.*

db.php

**UserRatingDisplay** (rating, hiiclass)

user\_rating.php

**UserRatingSet** (rating)

user\_rating.php

**add\_alternative\_file** (\$resource, \$name)

resource\_functions.php

**add\_collection** (\$user, \$collection)

*Add a collection to a user's 'My Collections'*

collections\_functions.php

**add\_field\_option** (\$field, \$option)

resource\_functions.php

**add\_keyword\_mappings** (\$ref, \$string, \$resource\_type\_field)

*For each instance of a keyword in \$string, add a keyword->resource mapping. Create keywords that do not yet exist. Increase the hit count of each keyword that matches. Store the position and field the string was entered against for advanced searching.*

resource\_functions.php

**add\_resource\_to\_collection** (\$resource, \$collection)

collections\_functions.php

**add\_saved\_search** (\$collection)

collections\_functions.php

**add\_saved\_search\_items** (\$collection)

collections\_functions.php

**add\_to\_collection\_link** (\$resource, \$search="")  
*Generates a HTML link for adding a resource to a collection*  
collections\_functions.php

**allow\_multi\_edit** (\$collection)  
*Returns true or false, can this collection be edited as a multi-edit? All the resources must be of the same type and status for this to work.*  
collections\_functions.php

**average\_length** (\$array)  
*Returns the average length of the strings in an array*  
general.php

**base64\_to\_jpeg** ( \$imageData, \$outputfile ) {  
image\_processing.php

**bulk\_mail** (\$userlist, \$subject, \$text)  
general.php

**change\_collection\_link** (\$collection)  
*Generates a HTML link for adding a changing the current collection*  
collections\_functions.php

**change\_password** (\$password)  
*Sets a new password for the current user.*  
general.php

**check\_access\_key** (\$resource, \$key)  
*Verify a supplied external access key*  
db.php

**check\_access\_key\_collection** (\$collection, \$key)  
db.php

**check\_password** (\$password)  
*Checks that a password conforms to the configured paramaters. Returns true if it does, or a descriptive string if it doesn't.*  
general.php

**checkperm** (\$perm)  
*Check that the user has the \$perm permission*  
db.php

**cleanse\_string** (\$string, \$preserve\_separators)  
*Removes characters from a string prior to keyword splitting, for example full stops Also makes the string lower case ready for indexing.*  
general.php

**collection\_is\_research\_request** (\$collection)  
*Returns true if a collection is a research request*  
collections\_functions.php

**collection\_log** (\$collection, \$type, \$resource)  
collections\_functions.php

**collection\_writeable** (\$collection)  
*Returns true if the current user has write access to the given collection.*  
collections\_functions.php

**copy\_collection** (\$copied, \$current, \$remove\_existing=false)  
*Get all data from the collection to copy.*  
collections\_functions.php

**copy\_hitcount\_to\_live** ()  
*Copy the temporary hit count used for relevance matching to the live column so it's activated (see comment for Update\_resource\_keyword\_hitcount())*  
general.php

**copy\_resource** (\$from, \$resource\_type=-1)  
*Create a new resource, copying all data from the resource with reference \$from. Note this copies only the data and not any attached file. It's very unlikely the Same file would be in the system twice, however users may want to clone an existing resource To avoid reentering data if the resource is very similar. If \$resource\_type if specified then the resource type for the new resource will be set to \$resource\_type Rather than simply copied from the \$from resource.*  
resource\_functions.php

**create\_collection** (\$userid, \$name, \$allowchanges=0, \$cant\_delete=0)  
*Creates a new collection and returns the reference*  
collections\_functions.php

**create\_previews** (\$ref, \$thumbonly=false, \$extension="jpg", \$previewonly=false)  
image\_processing.php

**create\_previews\_using\_im** (\$ref, \$thumbonly=false, \$extension="jpg", \$previewonly=false)  
image\_processing.php

**create\_resource** (\$resource\_type, \$archive=-1, \$user=-1)  
*Create a new resource.*  
resource\_functions.php

**daily\_stat** (\$activity\_type, \$object\_ref)  
*Update the daily statistics after a loggable event. The daily\_stat table contains a counter for each 'activity type' (i.e. download) for each object (i.e. resource) Per day.*  
db.php

**delete\_alternative\_file** (\$resource, \$ref)  
*Delete any uploaded file.*  
resource\_functions.php

**delete\_collection** (\$ref)  
*Deletes the collection with reference \$ref*  
collections\_functions.php

**delete\_collection\_access\_key** (\$collection, \$access\_key)

*Deletes the given access key.*

collections\_functions.php

**delete\_exif\_tmpfile** (\$tmpfile)

resource\_functions.php

**delete\_resource** (\$ref)

resource\_functions.php

**do\_report** (\$ref, \$from\_y, \$from\_m, \$from\_d, \$to\_y, \$to\_m, \$to\_d)

*Run report with id \$ref for the date range specified. Returns a result array.*

reporting\_functions.php

**do\_search** (\$search, \$restypes="", \$order\_by="relevance", \$archive=0, \$fetchrows=-1)

*Takes a search string \$search, as provided by the user, and returns a results set Of matching resources. If there are no matches, instead returns an array of suggested searches. \$restypes is optionally used to specify which resource types to search.*

search\_functions.php

**email\_collection** (\$collection, \$collectionname, \$fromusername, \$userlist, \$message, \$feedback)

*Attempt to resolve all users in the string \$userlist to user references. Add \$collection to these user's 'My Collections' page Send them an e-mail linking to this collection*

collections\_functions.php

**email\_collection\_request** (\$ref, \$details)

*E-mails a collection request (posted) to the team*

general.php

**email\_reminder** (\$email)

general.php

**email\_resource** (\$resource, \$resourcename, \$fromusername, \$userlist, \$message)

*Attempt to resolve all users in the string \$userlist to user references. Add \$collection to these user's 'My Collections' page Send them an e-mail linking to this collection*

resource\_functions.php

**email\_resource\_request** (\$ref, \$details)

*E-mails a resource request (posted) to the team*

general.php

**email\_user\_request** ()

*E-mails the submitted user request form to the team.*

general.php

**errorhandler** (\$errno, \$errstr, \$errfile, \$errline)

db.php

**escape\_check** (\$text) #only escape a string if we need to, to prevent escaping an already escaped string

db.php

**extract\_exif\_comment** (\$ref, \$extension)

*Extract the EXIF comment from either the ImageDescription field or the UserComment Also parse IPTC headers and insert*

image\_processing.php

**extract\_indd\_thumb** (\$filename) {

image\_processing.php

**extract\_mean\_colour** (\$image, \$ref)

*For image \$image, calculate the mean colour and update this to the image\_red, image\_green, image\_blue tables In the resources table. Also - we insert the height and width of the thumbnail at this stage as all information is available and we Are already performing an update on the resource record.*

image\_processing.php

**extract\_text** (\$ref, \$extension)

*Extract text from the resource and save to the configured field.*

image\_processing.php

**formatfilesize** (\$bytes)

*Return a human-readable string representing \$bytes in either KB or MB.*

general.php

**generate\_collection\_access\_key** (\$collection, \$feedback=0, \$email="")

*For each resource in the collection, create an access key so an external user can access each resource.*

collections\_functions.php

**generate\_file\_checksum** (\$resource, \$extension)

image\_processing.php

**get\_active\_users** ()

*Returns a list of active users, i.e. users still logged on with a last-active time within the last 2 hours.*

general.php

**get\_advanced\_search\_fields** (\$archive=false)

*Returns a list of fields suitable for advanced searching.*

search\_functions.php

**get\_all\_image\_sizes** (\$internal=false, \$restricted=false)

*Returns all image sizes available.*

general.php

**get\_all\_site\_text** (\$find="")

*Returns a list of all available editable site text (content). If \$find is specified a search is performed across page, name and text fields.*

general.php

**get\_alternative\_file** (\$resource, \$ref)

*Returns the row for the requested alternative file*

resource\_functions.php

**get\_alternative\_files** (\$resource)

*Returns a list of alternative files for the given resource*

resource\_functions.php

**get\_breadcrumbs** ()

*Returns a HTML breadcrumb trail for display at the top of the screen.*

general.php

**get\_collection** (\$ref)

*Returns all data for collection \$ref*

collections\_functions.php

**get\_collection\_comments** (\$collection)

collections\_functions.php

**get\_collection\_external\_access** (\$collection)

*Return all external access given to a collection. Users, emails and dates could be multiple for a given access key, an in this case they are returned comma-separated.*

collections\_functions.php

**get\_collection\_log** (\$collection)

collections\_functions.php

**get\_collection\_resource\_comment** (\$resource, \$collection)

collections\_functions.php

**get\_collection\_resources** (\$collection)

*Returns all resources in collection For many cases (e.g. when displaying a collection for a user) a search is used instead so permissions etc. are honoured.*

collections\_functions.php

**get\_collection\_videocount** (\$ref)

collections\_functions.php

**get\_colour\_key** (\$image)

*Extracts a colour key for the image, like a soundex.*

image\_processing.php

**get\_custom\_access** (\$resource, \$usergroup)

resource\_functions.php

**get\_data\_by\_field** (\$resource, \$field)

*Return the resource data for field \$field in resource \$resource*

general.php

**get\_exiftool\_fields** (\$resource\_type)

*Returns a list of exiftool fields, which are basically fields with an 'exiftool field' set.*

resource\_functions.php

**get\_field** (\$field)

resource\_functions.php

**get\_field\_options** (\$ref)

*For the field with reference \$ref, return a sorted array of options.*

general.php

**get\_field\_options\_with\_stats** (\$field)

*For a given field, list all options with usage stats. This is for the 'manage field options' page.*

resource\_functions.php

**get\_fields\_with\_options** ()

*Returns a list of fields that have option lists (checking user permissions) Used for 'manage field options' page.*

resource\_functions.php

**get\_grouped\_related\_keywords** (\$find="", \$specific="")

*Returns each keyword and the related keywords grouped, along with the resolved keywords strings.*

general.php

**get\_image\_sizes** (\$ref, \$internal=false, \$extension="jpg", \$onlyifexists=true)

*Returns a table of available image sizes for resource \$ref. The original image file assumes the name of the 'nearest size (up)' in the table*

general.php

**get\_ip** ()

db.php

**get\_keyword\_from\_option** (\$option)

*For the given field option, return the keyword that will be indexed.*

resource\_functions.php

**get\_max\_resource\_ref** ()

*Returns the highest resource reference in use.*

resource\_functions.php

**get\_mycollection\_name** (\$userref)

*Fetches the next name for a new My Collection for the given user (My Collection 1, 2 etc.)*

collections\_functions.php

**get\_related\_keywords** (\$keyref)

*For a given keyword reference returns the related keywords Also reverses the process, returning keywords for matching related words And for matching related words, also returns other words related to the same keyword.*

general.php

**get\_related\_resources** (\$ref)

*Return an array of resource references that are related to resource \$ref*

general.php

**get\_reports** ()

*Returns all reports in a result array.*

reporting\_functions.php

**get\_research\_request** (\$ref)  
research\_functions.php

**get\_research\_request\_collection** (\$ref)  
research\_functions.php

**get\_research\_requests** (\$find="")  
research\_functions.php

**get\_resource\_custom\_access** (\$resource)  
*Return a list of usergroups with the custom access level for resource \$resource (if set)*  
resource\_functions.php

**get\_resource\_data** (\$ref, \$cache=true)  
*Returns basic resource data (from the resource table alone) for resource \$ref. For 'dynamic' field data, see [get\\_resource\\_field\\_data](#)*  
general.php

**get\_resource\_field\_data** (\$ref, \$multi=false)  
*Returns field data and field properties (resource\_type\_field and resource\_data tables) For this resource, for display in an edit / view form.*  
general.php

**get\_resource\_field\_data\_batch** (\$refs)  
*Returns field data and field properties (resource\_type\_field and resource\_data tables) For all the resource references in the array \$refs. This will use a single SQL query and is therefore a much more efficient way of gathering Resource data for a list of resources (e.g. search result display for a page of resources).*  
general.php

**get\_resource\_log** (\$resource)  
resource\_functions.php

**get\_resource\_path** (\$ref, \$getfilepath, \$size, \$generate, \$extension="jpg", \$scramble=-1, \$page=1, \$watermarked=false, \$file\_modified="", \$alternative=-1, \$includemodified=true)  
*Returns the correct path to resource \$ref of size \$size (\$size==empty string is original resource) If one or more of the folders do not exist, and \$generate=true, then they are generated*  
general.php

**get\_resource\_ref\_range** (\$lower, \$higher)  
*Returns an array of resource references in the range \$lower to \$upper.*  
resource\_functions.php

**get\_resource\_top\_keywords** (\$resource, \$count)  
*Return the top \$count keywords (by hitcount) used by \$resource. This is for the 'Find Similar' search. Keywords that are too short or too long, or contain numbers are dropped - they are probably not as meaningful in The contexts of this search (consider being offered "12" or "OKB-34" as an option?)*  
general.php

**get\_resource\_type\_name** (\$type)  
resource\_functions.php

**get\_resource\_types ()**

*Returns a list of resource types.*

general.php

**get\_resources\_matching\_keyword (\$keyword, \$field)**

*Returns an array of resource references for resources matching the given keyword string.*

resource\_functions.php

**get\_saved\_searches (\$collection)**

collections\_functions.php

**get\_section\_list (\$page)**

db.php

**get\_simple\_search\_fields ()**

*Returns a list of fields suitable for the simple search box.*

general.php

**get\_site\_text (\$page, \$name, \$language, \$group)**

*Returns a specific site text entry.*

general.php

**get\_smart\_theme\_headers ()**

*Returns a list of smart theme headers, which are basically fields with a 'smart theme name' set.*

collections\_functions.php

**get\_smart\_themes (\$field)**

*Returns a list of smart themes (which are really field options). The results are filtered so that only field options that are in use are returned.*

collections\_functions.php

**get\_stats\_activity\_types ()**

*Returns a list of activity types for which we have stats data (Search, User Session etc.)*

general.php

**get\_stats\_years ()**

*Returns a list of years for which we have statistics.*

general.php

**get\_suggested\_keywords (\$search)**

*For the given partial word, suggest complete existing keywords.*

general.php

**get\_theme\_headers (\$theme1="", \$theme2="")**

*Return a list of theme headers, i.e. theme categories Return sql\_array("select theme value,count(\*) c from collection where public=1 and length(theme)>0 group by theme order by theme");*

collections\_functions.php

**get\_theme\_image (\$theme, \$theme2="", \$theme3="")**

collections\_functions.php

**get\_themes** (\$theme, \$theme2="", \$theme3="")  
*Return a list of themes under a given header (theme category).*  
collections\_functions.php

**get\_themes\_by\_resource** (\$ref)  
resource\_functions.php

**get\_user** (\$ref)  
*Return a user's credentials.*  
general.php

**get\_user\_collections** (\$user, \$find="", \$order\_by="name", \$sort="ASC", \$fetchrows=-1)  
*Returns a list of user collections.*  
collections\_functions.php

**get\_user\_log** (\$user)  
general.php

**get\_usergroups** (\$usepermissions=false, \$find="")  
*Returns a list of user groups. Put anything starting with 'General Staff Users' at the top (e.g. General Staff)*  
general.php

**get\_users** (\$group=0, \$find="", \$order\_by="u.username", \$usepermissions=false, \$fetchrows=-1)  
*Returns a user list. Group or search term is optional.*  
general.php

**getuid** ()  
*Generate a unique ID*  
db.php

**getval** (\$val, \$default)  
*Return a value from get/post or a default if neither set*  
db.php

**getvalescaped** (\$val, \$default)  
*Return a value from get/post, escaped and SQL-safe*  
db.php

**highlightkeywords** (\$text, \$search)  
*Highlight searched keywords in \$text Optional - depends on \$highlightkeywords being set in config.php.*  
general.php

**hook** (\$name, \$pagename="", \$params=array ())  
*Plugin architecture. Look for a hook with this name and execute.*  
db.php

**i18n\_get\_indexable** (\$text)  
*For field names / values using the i18n syntax, return all language versions, as necessary for indexing.*  
general.php

**i18n\_get\_translated** (\$text)

*For field names / values using the i18n syntax, return the version in the current user's language Format is ~en:Somename~es:Someothername*

general.php

**i18n\_get\_translations** (\$value)

*For a string in the language format, return all translations as an associative array E.g. "en"->"English translation"; "fr"->"French translation"*

general.php

**image\_size\_restricted\_access** (\$id)

*Returns true if the indicated size is allowed for a restricted user.*

general.php

**import\_resource** (\$path, \$type, \$title)

*Import the resource at the given path This is used by staticsync.php and Camillo's SOAP API Note that the file will be used at it's present location and will not be copied.*

resource\_functions.php

**iptc\_return\_utf8** (\$text)

*For the given \$text, return the utf-8 equiv. Used for iptc headers to auto-detect the character encoding.*

image\_processing.php

**make\_password** ()

*Generate a password using the configured settings.*

general.php

**new\_user** (\$newuser)

*Username already exists?*

general.php

**newlines** (\$text)

*Replace escaped newlines with real newlines.*

general.php

**nicedate** (\$date, \$time=false, \$wordy=false)

*Format a MySQL ISO date in the UK style*

db.php

**notify\_user\_contributed\_submitted** (\$refs)

resource\_functions.php

**pagename** ()

db.php

**pager** (\$break=true)

general.php

**populate\_smart\_theme\_tree\_node** (\$tree, \$node, \$return, \$indent)

*When displaying category trees as smart themes, this function is used to recursively parse each node adding items sequentially with an appropriate indent level.*

collections\_functions.php

**quoted\_printable\_encode** (\$string, \$linelen = 0, \$linebreak="\r\n", \$breaklen = 0, \$encodecrlf = false) {  
general.php

**quoted\_printable\_encode\_subject** (\$string, \$encoding='UTF-8') {  
general.php

**redirect** (\$url)  
db.php

**refresh\_collection\_frame** ()  
*Refresh the collections frame Only works when we are using a frameset.*  
collections\_functions.php

**relate\_to\_array** (\$ref, \$array)  
*Relates a resource to each in a simple array of ref numbers*  
resource\_functions.php

**relate\_to\_collection** (\$ref, \$collection)  
*Relates every resource in \$collection to \$ref*  
collections\_functions.php

**remove\_collection** (\$user, \$collection)  
*Remove someone else's collection from a user's My Collections*  
collections\_functions.php

**remove\_from\_collection\_link** (\$resource, \$search="")  
*Generates a HTML link for removing a resource to a collection*  
collections\_functions.php

**remove\_keyword\_mappings** (\$ref, \$string, \$resource\_type\_field)  
*Removes one instance of each keyword->resource mapping for each occurrence of that Keyword in \$string. This is used to remove keyword mappings when a field has changed. We also decrease the hit count for each keyword.*  
resource\_functions.php

**remove\_resource\_from\_collection** (\$resource, \$collection)  
collections\_functions.php

**remove\_saved\_search** (\$collection, \$search)  
collections\_functions.php

**resolve\_keyword** (\$keyword, \$create=false)  
*Returns the keyword reference for \$keyword, or false if no such keyword exists.*  
general.php

**resolve\_soundex** (\$keyword)  
*Returns the most commonly used keyword that sounds like \$keyword, or failing a soundex match, The most commonly used keyword that starts with the same few letters.*  
search\_functions.php

**resolve\_user\_agent** (\$agent)  
db.php

**resolve\_userlist\_groups** (\$userlist)  
*Given a comma separated user list (from the user select include file) turn all Group: entries into fully resolved list of usernames.*  
general.php

**resolve\_users** (\$users)  
*For a given comma-separated list of user refs (e.g. returned from a group\_concat()), return a string of matching usernames.*  
general.php

**resource\_log** (\$resource, \$type, \$field)  
resource\_functions.php

**save\_alternative\_file** (\$resource, \$ref)  
*Saves the 'alternative file' edit form back to the database*  
resource\_functions.php

**save\_collection** (\$ref)  
collections\_functions.php

**save\_collection\_resource\_comment** (\$resource, \$collection, \$comment, \$rating)  
*Get data before update so that changes can be logged.*  
collections\_functions.php

**save\_field\_options** (\$field)  
*Save the field options after editing.*  
resource\_functions.php

**save\_related\_keywords** (\$keyword, \$related)  
general.php

**save\_research\_request** (\$ref)  
*Save*  
research\_functions.php

**save\_resource\_custom\_access** (\$resource)  
resource\_functions.php

**save\_resource\_data** (\$ref, \$multi)  
*Save all submitted data for resource \$ref. Also re-index all keywords from indexable fields.*  
resource\_functions.php

**save\_resource\_data\_multi** (\$collection)  
*Save all submitted data for collection \$collection, this is for the 'edit multiple resources' feature Loop through the field data and save (if necessary)*  
resource\_functions.php

**save\_site\_text** (\$page, \$name, \$language, \$group)  
*Saves the submitted site text changes to the database.*  
general.php

**save\_user** (\$ref)  
general.php

**search\_public\_collections** (\$search="", \$order\_by="name", \$sort="ASC")  
collections\_functions.php

**send\_collection\_feedback** (\$collection, \$comment)  
*Sends the feedback to the owner of the collection.*  
collections\_functions.php

**send\_mail** (\$email, \$subject, \$message, \$from="")  
*Send a mail - but correctly encode the message/subject in quoted-printable UTF-8.*  
general.php

**send\_research\_request** ()  
*Insert a search request into the requests table.*  
research\_functions.php

**send\_statistics** ()  
*If configured, send two metrics to Montala.*  
general.php

**set\_research\_collection** (\$research, \$collection)  
research\_functions.php

**set\_user\_collection** (\$user, \$collection)  
collections\_functions.php

**split\_keywords** (\$search, \$index=false)  
*Takes \$search and returns an array of individual keywords.*  
general.php

**sql\_array** (\$query)  
*Like sql\_value() but returns an array of all values found. The value returned must have the column name aliased to 'value'*  
db.php

**sql\_insert\_id** ()  
*Return last inserted ID (abstraction)*  
db.php

**sql\_query** (\$sql, \$cache=false, \$fetchrows=-1, \$dbstruct=true)  
*Sql\_query(sql) - execute a query and return the results as an array. Database functions are wrapped in this way so supporting a database server other than MySQL is Easier. \$cache is not used at this time - it was intended for disk based results caching which may be added in the future. If \$fetchrows is set we don't have to loop through all the returned rows. We Just fetch \$fetchrows row but pad the array to the full result set size with empty values. This has been added retroactively to support large result sets, yet a pager can work as if a*

*full Result set has been returned as an array (as it was working previously).*

db.php

**sql\_value** (\$query, \$default)

*Return a single value from a database query, or the default if no rows The value returned must have the column name aliased to 'value'*

db.php

**str\_highlight** (\$text, \$needle, \$options = null, \$highlight = null)

*Thanks to Aidan Lister Sourced from [http://aidanlister.com/repos/v/function.str\\_highlight.php](http://aidanlister.com/repos/v/function.str_highlight.php) on 2007-10-09 License on the website reads: "All code on this website resides in the Public Domain, you are free to use and modify it however you wish." [Http://aidanlister.com/repos/license/](http://aidanlister.com/repos/license/)*

general.php

**string\_similar** (\$string1, \$string2)

*Returns an integer score based on how similar the two strings are. This was used when importing data for "fuzzy" keyword/option matching.*

general.php

**suggest\_refinement** (\$refs, \$search)

*Given an array of resource references (\$refs) and the original Search query (\$search), produce a list of suggested search refinements to Reduce the result set intelligently.*

search\_functions.php

**swap\_collection\_order** (\$resource1, \$resource2, \$collection)

*Inserts \$resource1 into the position currently occupied by \$resource2 and moves \$resource2 And subsequent resources down a position.*

collections\_functions.php

**text** (\$name)

db.php

**tidy\_trim** (\$text, \$length)

*Trims \$text to \$length if necessary. Tries to trim at a space if possible. Adds three full stops If trimmed...*

general.php

**tidylist** (\$list)

*Takes a value as returned from a check-list field type and reformats to be more display-friendly. Check-list fields have a leading comma.*

general.php

**trim\_array** (\$array)

*Removes whitespace from the beginning/end of all elements in an array*

general.php

**trim\_spaces** (\$text)

*Replace multiple spaces with a single space*

general.php

**tweak\_preview\_images** (\$ref, \$rotateangle, \$gamma, \$extension="jpg")

*Tweak all preview images On the edit screen, preview images can be either rotated or gamma adjusted. We*

*keep the high(original) and low resolution print versions intact as these would be adjusted professionally when in use in the target application.*

image\_processing.php

**update\_field** (\$resource, \$field, \$value)

*Updates a field. Works out the previous value, so this is not efficient if we already know what this previous value is (hence it is not used for edit where multiple fields are saved)*

resource\_functions.php

**update\_resource\_keyword\_hitcount** (\$resource, \$search)

*For the specified \$resource, increment the hitcount for each matching keyword in \$search This is done into a temporary column first (new\_hit\_count) so existing results are not affected. Copy\_hitcount\_to\_live() is then executed at a set interval to make this data live.*

general.php

**update\_resource\_type** (\$ref, \$type)

resource\_functions.php

**upload\_file** (\$ref)

*Process file upload for resource \$ref*

image\_processing.php

**upload\_preview** (\$ref)

*Upload a preview image only.*

image\_processing.php

**user\_rating\_save** (\$ref, \$rating)

*Save a user rating for a given resource*

resource\_functions.php

**write\_metadata** (\$path, \$ref)

resource\_functions.php