**Common core hooks** <https://drupal.org/comment/3560674#comment-3560674>

Here's the start of a classification of hook by function or area. I think what's really needed in a mind-graph or other classification system.

**System**

* hook\_cron   
  Perform periodic actions.
* hook\_flush\_caches   
  Add a list of cache tables to be cleared.
* hook\_locale   
  Allows modules to define their own text groups that can be translated.
* hook\_mail   
  Prepare a message based on parameters; called from drupal\_mail().
* hook\_mail\_alter   
  Alter an email message created with the drupal\_mail() function

**Module Lifetime**

* hook\_enable   
  Perform necessary actions after module is enabled.
* hook\_disable   
  Perform necessary actions before module is disabled.

**Node Persistence**

* hook\_delete   
  Respond to node deletion
* hook\_insert   
  Respond to creation of a new node.

**Permissions**

* hook\_access   
  hook\_node\_access Control access to a node.  
  hook\_search\_access Define access to a custom search routine.  
  hook\_field\_access Determine whether the user has access to a given field.
* hook\_permission   
  Define user permissions.

**Node Loading**

* hook\_load hook\_node\_load  
  Act on arbitrary nodes being loaded from the database.

**Path / URL**

* hook\_menu   
  Define menu items and page callbacks.
* hook\_menu\_alter   
  Alter the data being saved to the {menu\_router} table after hook\_menu is invoked.
* hook\_menu\_link\_alter   
  Alter the data being saved to the {menu\_links} table by menu\_link\_save().
* hook\_file\_download   
  Control access to private file downloads and specify HTTP headers.

**Processing Requests**

* hook\_boot   
  Perform setup tasks for all page requests.   
    
  This hook is run at the beginning of the page request. It is typically used to set up global parameters that are needed later in the request.

Only use this hook if your code must run even for cached page views. This hook is called before the theme, modules, or most include files are loaded into memory. It happens while Drupal is still in bootstrap mode.

* hook\_init  
  Perform setup tasks for non-cached page requests.  
    
  This hook is run at the beginning of the page request. It is typically used to set up global parameters that are needed later in the request. When this hook is called, the theme and all modules are already loaded in memory.

This hook is not run on cached pages.

To **add CSS or JS** that should be present on all pages, modules **should not implement this hook**, but **declare these files in their .info file**.

* hook\_url\_inbound\_alter  
  Alters inbound URL requests.
* hook\_url\_outbound\_alter  
  Alters outbound URLs
* hook\_exit  
  Perform cleanup tasks.

This hook is run at the end of most regular page requests. It is often used for page logging and specialized cleanup. This **hook MUST NOT print anything** because **by the time it runs the response is already sent to the browser**.

Only use this hook if your code must run even for cached page views. If you have code which **must run once on all non-cached pages**, **use hook\_init() instead**. That is the usual case. If you implement this hook and see an error like 'Call to undefined function', it is likely that you are depending on the presence of a module which has not been loaded yet. It is not loaded because Drupal is still in bootstrap mode.

**Block**

* hook\_block

|  |  |
| --- | --- |
| [hook\_block\_cid\_parts\_alter](https://api.drupal.org/api/drupal/modules%21block%21block.api.php/function/hook_block_cid_parts_alter/7.x) | Act on block cache ID (cid) parts before the cid is generated. |
| [hook\_block\_configure](https://api.drupal.org/api/drupal/modules%21block%21block.api.php/function/hook_block_configure/7.x) | Define a configuration form for a block. |
| [hook\_block\_info](https://api.drupal.org/api/drupal/modules%21block%21block.api.php/function/hook_block_info/7.x) | Define all blocks provided by the module. |
| [hook\_block\_info\_alter](https://api.drupal.org/api/drupal/modules%21block%21block.api.php/function/hook_block_info_alter/7.x) | Change block definition before saving to the database. |
| [hook\_block\_list\_alter](https://api.drupal.org/api/drupal/modules%21block%21block.api.php/function/hook_block_list_alter/7.x) | Act on blocks prior to rendering. |
| [hook\_block\_save](https://api.drupal.org/api/drupal/modules%21block%21block.api.php/function/hook_block_save/7.x) | Save the configuration options from [hook\_block\_configure](https://api.drupal.org/api/drupal/modules%21block%21block.api.php/function/hook_block_configure/7.x" \o "Define a configuration form for a block.)(). |
| [hook\_block\_view](https://api.drupal.org/api/drupal/modules%21block%21block.api.php/function/hook_block_view/7.x) | Return a rendered or renderable view of a block. |
| [hook\_block\_view\_alter](https://api.drupal.org/api/drupal/modules%21block%21block.api.php/function/hook_block_view_alter/7.x) | Perform alterations to the content of a block. |
| [hook\_block\_view\_MODULE\_DELTA\_alter](https://api.drupal.org/api/drupal/modules%21block%21block.api.php/function/hook_block_view_MODULE_DELTA_alter/7.x) | Perform alterations to a specific block. |

**Actions/Triggers**

* hook\_action\_info   
  Declares information about actions.
* hook\_action\_info\_alter   
  Alters the actions declared by another module.
* hook\_actions\_delete   
  Executes code after an action is deleted.
* hook\_hook\_info   
  Defines one or more hooks that are exposed by a module.

Normally hooks do not need to be explicitly defined. However, by declaring a hook explicitly, a module may define a "group" for it. Modules that implement a hook may then place their implementation in either $module.module or in $module.$group.inc. If the hook is located in $module.$group.inc, then that file will be automatically loaded when needed. In general, hooks that are rarely invoked and/or are very large should be placed in a separate include file, while hooks that are very short or very frequently called should be left in the main module file so that they are always available.

* hook\_hook\_info\_alter   
  Alter information from hook\_hook\_info().

**Database**

* hook\_schema   
  Define the current version of the database schema.  
  A Drupal schema definition is an array structure representing one or more tables and their related keys and indexes. A schema is defined by hook\_schema() which must live in your module's .install file.

This hook is called at install and uninstall time, and in the latter case, it cannot rely on the .module file being loaded or hooks being known. If the .module file is needed, it may be loaded with drupal\_load().

The tables declared by this hook will be automatically created when the module is first enabled, and removed when the module is uninstalled. This happens before hook\_install() is invoked, and after hook\_uninstall() is invoked, respectively.

By declaring the tables used by your module via an implementation of hook\_schema(), these tables will be available on all supported database engines. You don't have to deal with the different SQL dialects for table creation and alteration of the supported database engines.

* hook\_install   
  Perform setup tasks when the module is installed.   
    
  If the module implements hook\_schema(), the database tables will be created before this hook is fired.

Implementations of this hook are by convention declared in the module's .install file. The implementation can rely on the .module file being loaded. The hook will only be called the first time a module is enabled or after it is re-enabled after being uninstalled. The module's schema version will be set to the module's greatest numbered update hook. Because of this, any time a hook\_update\_N() is added to the module, this function needs to be updated to reflect the current version of the database schema.

* hook\_uninstall()

Remove any information that the module sets.

The information that the module should remove includes:

* + variables that the module has set using variable\_set() or system\_settings\_form()
  + modifications to existing tables

The module should not remove its entry from the {system} table. Database tables defined by hook\_schema() will be removed automatically.

The uninstall hook must be implemented in the module's .install file. It will fire when the module gets uninstalled but before the module's database tables are removed, allowing your module to query its own tables during this routine.

When hook\_uninstall() is called, your module will already be disabled, so its .module file will not be automatically included. If you need to call API functions from your .module file in this hook, use drupal\_load() to make them available. (Keep this usage to a minimum, though, especially when calling API functions that invoke hooks, or API functions from modules listed as dependencies, since these may not be available or work as expected when the module is disabled.)

**Form**

* hook\_form  
  Display a node editing form.  
    
  This is a node-type-specific hook, which is invoked only for the node type being affected. See Node API hooks for more information.

Use hook\_form\_BASE\_FORM\_ID\_alter(), with base form ID 'node\_form', to alter node forms for all node types.

This hook, implemented by node modules, is called to retrieve the form that is displayed to create or edit a node. This form is displayed at path node/add/[node type] or node/[node ID]/edit.

The submit and preview buttons, administrative and display controls, and sections added by other modules (such as path settings, menu settings, comment settings, and fields managed by the Field UI module) are displayed automatically by the node module. This hook just needs to return the node title and form editing fields specific to the node type.

* hook\_forms   
  Map form\_ids to form builder functions.
* hook\_form\_alter   
  Perform alterations before a form is rendered.

One popular use of this hook is to add form elements to the node form. When altering a node form, the node object can be accessed at $form['#node'].

In addition to hook\_form\_alter(), which is called for all forms, there are two more specific form hooks available. The first, hook\_form\_BASE\_FORM\_ID\_alter(), allows targeting of a form/forms via a base form (if one exists). The second, hook\_form\_FORM\_ID\_alter(), can be used to target a specific form directly.

The call order is as follows: all existing form alter functions are called for module A, then all for module B, etc., followed by all for any base theme(s), and finally for the theme itself. The module order is determined by system weight, then by module name.

Within each module, form alter hooks are called in the following order:   
first, hook\_form\_alter();   
second, hook\_form\_BASE\_FORM\_ID\_alter();   
third, hook\_form\_FORM\_ID\_alter().   
So, for each module, the more general hooks are called first followed by the more specific.

* hook\_form\_FORM\_ID\_alter   
  Provide a form-specific alteration instead of the global [hook\_form\_alter](https://api.drupal.org/api/drupal/modules%21system%21system.api.php/function/hook_form_alter/7.x" \o "Perform alterations before a form is rendered.)().
* hook\_form\_BASE\_FORM\_ID\_alter  
  Provide a form-specific alteration for shared ('base') forms.

Modules can implement hook\_form\_BASE\_FORM\_ID\_alter() to modify a specific base form, rather than implementing hook\_form\_alter() and checking for conditions that would identify the shared form constructor.

To identify the base form ID for a particular form (or to determine whether one exists) check the $form\_state. The base form ID is stored under $form\_state['build\_info']['base\_form\_id'].

**Documentation**

* hook\_help  
  Provide online user help.

By implementing hook\_help(), a module can make documentation available to the user for the module as a whole, or for specific paths. Help for developers should usually be provided via function header comments in the code, or in special API example files.

The page-specific help information provided by this hook appears as a system help block on that page.   
case 'admin/structure/block':

The module overview help information is displayed by the Help module. It can be accessed from the page at admin/help or from the Modules page.   
case 'admin/help**#block'**: