# EdRepo Back-end Writer's Guide

This is a quick guide to writing storage back-ends for EdRepo.

## EdRepo Structure

For the purposes of this guide, EdRepo is organized into three parts: The front-end, the data-manager, and the back-end. The front-end is the part of EdRepo which users directly interact with. Data storage, organization, and retrial is performed by a back-end, which is free to internally work however it chooses. The front-end communicates with the back-end through a number of functions found in the data-manager. The data-manager is in charge of passing requests by the front-end to the back-end and returning the results back to the front-end in a standardized, uniform way. Since the data-manager is partly specific to the back-end (since, although every data-manager looks the same tot he front-end, its method(s) of passing data between the front-end and back-end may differ between back-ends), data-managers are shipped as part of a back-end, and in fact back-end writers may choose to integrate their data-managers and back-end components very closely together.

EdRepo's front-end supports working with only one data-manager/back-end at a time, and has no support for moving data between data-managers/back-ends. The front-end loads a back-end by importing its data-manager, and then using the functions provided by that data-manager, which in turn is responsible for correctly interfacing with the back-end. Each data-manager/back-end is stored in a subdirectory of the "backends" subdirectory of an EdRepo install. The file "backends/backend.php" is used to select a data-manager/back-end by simply importing the data-manager of the back-end to use.

## Writing a Data-Manager

A data-manager is simply a file (usually named "datamanager.php") which contains a set list of functions. The purpose of the data-manager is to accept input from the front-end through these functions, perform the action requested via the back-end, and return the result in the format expected by the front-end. Back-end writers may decide to include most of their back-end code within the data-manager file (by writing back-end code directly in the data-manager functions). This is perfectly acceptable.

Data-managers also must advertise which features their back-end supports and basic information about the back-end (name, version, author, etc) to the front-end. These tasks are accomplished via the getBackendCapabilities() and getBackendBasicInformation() functions, which must return arrays with the proper indexes.

All other functions are expected to either return an expected value to the front-end, an error, or the string "NotImplimented" if the feature being called by the front-end is not handled by the back-end.

Please see the MySQL database back-end code for an explanation of every data-manager function, along with its parameters and allowable values it may pass back to the front-end. Your data-manager should accept the same parameters and be able to return the same data, in the same format, as the MySQL back-end (except for any features you choose not to support).

As a general rule, error handling, and handling of back-end features you choose not to implement are done in the following way:

- Returning FALSE by a data-manager function is ALWAYS considered to be an error. Some back-end functions may be able to return more detailed error messages. However, most can not, and all front-end components recognize FALSE as an error (note: this is the boolean type FALSE, not the string "FALSE").
- Returning the string "NotImplimented" indicates that the back-end does not support the function called by the front-end. For example, if your back-end does not support creating materials, its data-manager must still handle the createMaterial() function. However, since the action is not supported, it would simply return "NotImplimented", indicating to the front-end that the feature is not supported. Note that some features must be supported for the front-end to be usable. See the chart below.

It is suggested that you begin writing your data-manager by coping the data-manager from the MySQL database back-end, removing the contents of the functions, and using the resulting function definitions and source code comments as a template for your data-manager.

The MySQL database back-end and data-manager are the reference back-end/data-manager to use when writing your own data-manager. Your data-manager should be able to return the exact same things as the MySQL data-manager can return. In particular, this means that whenever an array is returned in the MySQL data-manager, your data-manager MUST return the same array with the same keys and structure! If your back-end does not support one or more of the keys returned, you must still include them in the returned array, filled with an empty string. For example, when getting using information, the returned array contains the keys "authorFirstName" and "authorLastName". If your back-end only includes on "name" (does not split the name into two parts), it still must return both the "authorFirstName" and "authorLastName" keys, although it may set one to an empty string. It might, for example, return the array (... "authorFirstName"=>"John Doe", "authorLastName"=>"" ...), putting the entire name in the "authorFirstName" key. If you are doing something like this, thoroughly test your back-end with the front-end to ensure there is no unexpected or unwanted behavior.

## Functions and Features your Back-end MUST Support

Although the EdRepo front-end attempts to adapt itself to back-ends which do not support every possibly feature, there is a set of features, some technically "optional," which back-ends must support for all features of the front-end to work properly. This is a list of these features, along with which part(s) of the front-end which depend on these back-end features.

| Feature | Example Data-manager Call | Required For |
|---|---|---|
| Return list of back-end capabilities. | getBackendCapabilities() | All front-end features. No part of the front-end will function if this feature is not supported. |
| Return information about the back-end. | getBackendBasicInformation() | All front-end features. |
| Search modules by date. | searchModules(array("minDate"=>"a_date")) | All aspects of harvesting with OAI-PMH. Back-ends must support the "minDate" and "maxDate" parameters to the searchModules() function for OAI-PMH harvesting to work. |
| Search materials by date | searchMaterials(array("maxDate"=>"a_date")) | All aspects of harvesting with OAI-PMH. Back-ends must support the "minDate" and "maxDate" parameters to the searchMaterials() function for OAI-PMH harvesting to work. |
| Using modules and materials in read mode | getModuleByID(an_id) getMaterialByID(an_id) | All aspects of the front-end. The front-end is useless if using modules and materials is not supported in read mode. |
| Searching modules by the first letter of the title. | searchModules(array("titleStartsWith"=>"A")) | Browsing modules. |