Senior CAP Project

Administrator Documentation

16 May 2011

Mount Vernon Nazarene University

Grant Gipson

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision Number | Date of Release | Owner | Summary of Changes |
| 1.0 | May 14, 2011 | Grant Gipson | Initial draft of document |
| 1.1 | May 15, 2011 | Grant Gipson | Finished document |
| 1.2 | May 16, 2011 | Grant Gipson | Added instructions on renaming, deletions, and changing the server address |

Table of Contents

[Introduction 3](#_Toc293168137)

[Purpose 3](#_Toc293168138)

[Scope 3](#_Toc293168139)

[System Overview 3](#_Toc293168140)

[Client-side 3](#_Toc293168141)

[Server-side 3](#_Toc293168142)

[Administration 4](#_Toc293168143)

[CAPManage 4](#_Toc293168144)

[Data and Log Directories 5](#_Toc293168145)

[System Configuration File 5](#_Toc293168146)

[Database 6](#_Toc293168147)

[End-user Instructions 7](#_Toc293168148)

[Client Plug-in 7](#_Toc293168149)

[Web Interface 8](#_Toc293168150)

[Troubleshooting 8](#_Toc293168151)

[Download Requests 8](#_Toc293168152)

[Archive Requests 9](#_Toc293168153)

[Glossary 9](#_Toc293168154)

# Introduction

## Purpose

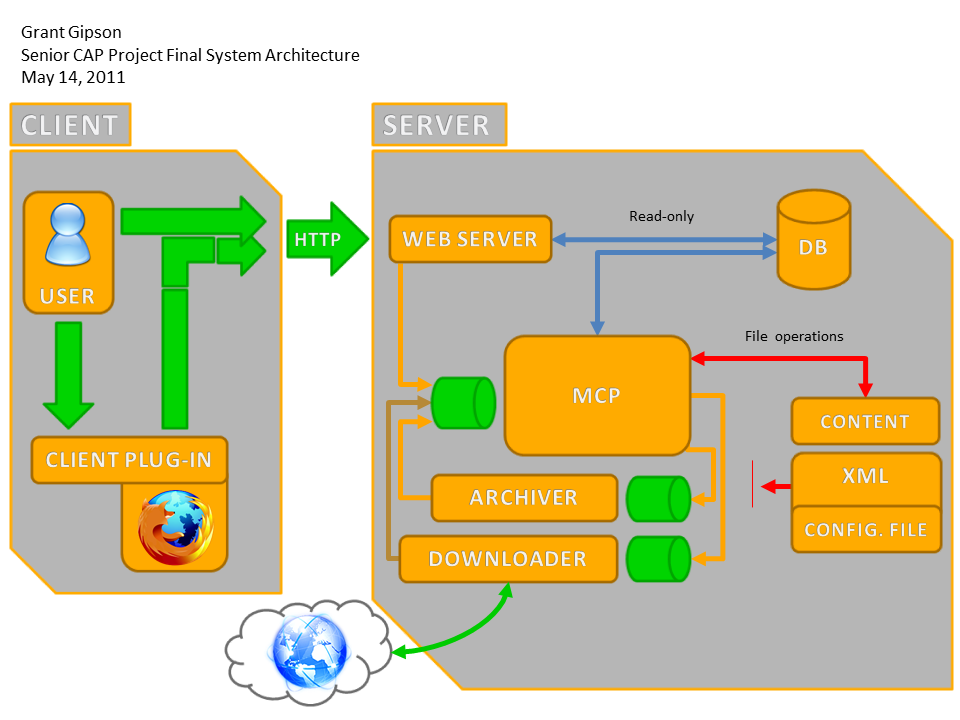
The purpose of this document is to instruct users of the system in both its administration and its general use by the end-user. For the administrator, it explains how to manage the system’s control flow, the design of the database, and general troubleshooting tips. For the end-user, it explains how to set up the local machine for use, access the web interface on the server, and how to use the system in general.

## Scope

This document is intended to be used by system administrators. The instructions herein are more pertinent to someone managing the system and are of less benefit to the typical end-user. Although there are instructions on use of the system by end-users, these are written here for the benefit of the administrator. End-users should be able to receive all the instruction they need from the system’s help options.

## System Overview

This document contains instructions on how the administrator is to install and operate the system, a detailed description of the overall system design, and tips on how this knowledge can be used in conjunction with documented issues in order to solve problems.



### Client-side

The entire client side consists of the Client Plug-in (CP) which is used to send download requests to the server. The CP is a Firefox Extension which adds a toolbar to the browser. When the user is a viewing a page they wish to download, they click the aptly-named Download Page button and the CP sends the request information to the server. The CP stores the server address in Firefox’s preferences which can be accessed in Firefox with the URL, “about:config”.

It is important to note how the user is informed of system malfunctions. First, the CGI scripts, which receive requests from the client-side, output any errors to a log file. If an error occurs before a log file can be opened then an error message is sent back to the client and displayed to the user. The result is that the user may view some rather technical error messages. Although this behavior is undesirable, it is more desirable than dropping the error out of politeness to the user. Second, with the system’s current design of inter-component communications there is no way to inform CGI scripts that an error has occurred. Therefore, unless the entire system is out of operation the user will not receive an error message—even if the entire request is dropped. Until this design flaw is remedied the administrator will need to be diligent in filtering log files for error messages so that system malfunctions are remedied quickly.

### Server-side

Requests from the client side and from the web server are handled by CGI scripts which forward these requests on to the Master (Control) Program. This program sits at the heart of the system architecture and handles all communications between components. It coordinates everything that happens in the system. No other component has the ability to make changes to the system. Here is a list of the major server-side components:

* CGI scripts
  + Client requests
  + Archive requests
* Web server
* Master Control Program
* Downloader
* Archiver
* Database
* Configuration file
* CAPManage
* Data directory
  + Communication pipes
  + Content directory
* Log directory

# Administration

Most of the administrator’s duties in the system will consist of interacting with the Master Control Program, possible maintenance of data and the database, and system troubleshooting.

## CAPManage

Management of the system will typically be done with CAPManage. This component is a Perl script which receives commands from the administrator and sends them as messages to the Master Control Program. CAPManage refers to the system configuration file to find the locations of other components. The location of this file is hardcoded to be “/var/cap/capconf.xml”, but because it is a Perl script, changing the location and name of the file is straightforward.

Command line:

**start [{component}|all]**  
Starts the given component or all of them if ‘all’ or no arguments are specified.

**stop [{component}|all]**  
Same arguments as start but stops the component. NOTE: currently if ‘stop all’ is called and one of the components is not running, then the script will block indefinitely waiting for this component which is not running. If this is the case, then individually shut each component down.

**kick [{component}|all]**  
If a component is waiting for a message, then this command “kicks” it out of its waiting state to perform any non-message processing before waiting for another message.

Components for command line:

* master -Master Control Program
* downloader
* archiver

## Data and Log Directories

The data directory contains all variable and runtime data used by the system; it also contains the content directory. If the system is not functioning properly, then a good place to start investigating is whether these components exist and have the proper permissions. These are the components and their descriptions:

* *content* directory  
  Contains all of the content downloaded onto the system as well as created archives. Every file is named after its ID in the database (with a minimum width of 10 digits) and suffixed with the appropriate extension. All downloaded content has an HTML extension and all archives have a ZIP extension. Ex: content/0000000001.html would be a content item of ID 1 in the database.
* *download* directory  
  The Downloader initially places downloaded content into this directory; it is later moved to *content* by the Master Control Program.
* *archive* directory  
  The Master Control Program copies content to be archived into this directory where the Archiver then proceeds to combine them into an archive. The Master Control Program will later copy the archive into the *content* directory.
* System configuration file
* Communication pipes  
  These are the pipes used to communicate between components. They have a FIFO extension.
* PID file  
  Stores the Master Control Program’s process ID and is used to indicate whether it is running.

The log directory contains a log file for the CGI scripts, Master Control Program, Downloader, and Archiver. These components can be rather verbose in their logging if instructed to do so in the system configuration file.

## System Configuration File

These are the settings stored in the system configuration file. All of these settings may be changed as needed by the system administrator.

* components
  + master\_program: location of Master Control Program
  + downloader: location of Downloader
  + downloader\_dir: directory downloads are made to
  + content\_dir: directory content is stored in
  + archiver: location of Archiver
  + archiver\_dir: directory archiving is done in
* database
  + connect: connection string to database, “[protocol]://[address][:port]/cap”
  + master\_user: Master Control Program’s user name in database
  + master\_password: password for user name
* log\_files: location of log files
  + master\_log
  + clientreq\_log
  + downloader\_log
  + archreq\_log
  + archiver\_log
* log\_priority\_write  
  The lowest level of priority message which may be written to Master Control Program log. Values: 0-No logging, 1-Fatal errors, 2-Errors, 3-Warnings, 4-Information. Ex: specifying 3 will allow fatal errors, non-fatal errors, and warnings to be logged.
* pid\_file: location of Master Control Program’s PID file
* pipes: names of pipes used for communications
  + pipes\_dir: directory pipes are stored in
  + pipes\_master
  + pipes\_downloader
  + pipes\_archiver

## Database

The USER table stores personal information about users of the system.

|  |  |
| --- | --- |
| ID | User ID |
| FIRST\_NAME | First name of user |
| LAST\_NAME | Last name of user |
| EMAIL | E-mail address of user |

The JOB table stores information about requests to download content. A job which has not yet been completed will have no completion date. See the JOB\_STATUS and JOB\_TYPE tables for descriptions of these fields.

|  |  |
| --- | --- |
| ID | Job ID |
| USER\_ID | ID of user job belongs to |
| TYPE | Type of job (see JOB\_TYPE) |
| STATUS | Status of the job (see JOB\_STATUS) |
| CMPL\_DATE | Date job was completed |
| URL | URL to be downloaded |

The CONTENT table stores information about all the content in the system. Archives also have a CONTENT record; the only thing that distinguishes them from downloaded content is that they also have a record in the ARCHIVE table.

|  |  |
| --- | --- |
| ID | Content ID |
| FOLDER\_ID | ID of folder that content is located in |
| USER\_ID | ID of user who owns content |
| STATUS | Status of content (see CONTENT\_STATUS) |
| TITLE | Title of content displayed to user |
| ADD\_DATE | Date content was added to system |

The ARCHIVE\_CONTENT table stores relationships between archives and content.

|  |  |
| --- | --- |
| ARCHIVE\_ID | ID of archive containing content |
| CONTENT\_ID | ID of content contained within archive |

## server address in web interface

The server address is hardcoded into the web interface. Therefore, if the server’s address changes then functions such as client and archive requests will fail. In such a situation, change the address specified in cap/server.js. Other scripts which contact the server reference this global variable.

# End-user Instructions

These instructions explain how the end-user is to set up the Client Plug-in, make download requests, manage their content, and create archives.

## Client Plug-in

Installation of the Client Plug-in is straightforward. The user needs to only navigate to the web server using Firefox, click the Help button, and then install the Client Plug-in package. As long as the user is running Firefox then the installation should be handled very easily. Once it is installed, Firefox will restart and the CP will be ready for use.

The user may need to change the location of the server which requests are sent to; this can be done by clicking the Settings button and entering in the server address. When the CP sends a request, it follows the format “http://[server address]/cap/clientreq.pl”. So these additional strings do not need to be specified by the user.

The Home button simply loads the server’s home page for the user to view the web interface.

The Download Page button is used to send download requests to the server. When the user clicks this button, the URL which is currently being visited is packaged in a download request and sent to the server. The only feedback to the user occurs if an error occurs. No response from clicking the button indicates success.

## Web Interface

The web interface for the server consists of a header, a toolbar (or menu bar), and a content area. The header displays the currently-logged in user in the top-right corner. And the toolbar consists of a series of menu options as well as input for a filter.

Here is an explanation of the content-related options on the toolbar. The Content option displays all of the user’s content which has not been deleted. Content on the system is not truly deleted—it is only marked deleted in the CONTENT table’s STATUS field. The Delete option proceeds to mark the currently highlighted content item as deleted. Again, nothing is deleted but the content is no longer displayed to the user and is removed from any archives it is related to. The View option either opens the currently highlighted content in a new window (downloaded content) or, if it is an archive, it lists the content which is in that archive. The Download option is only available for archives. It is a hyperlink which allows the user to download the archive to their local machine.

The Archive option displays all of the content items which the user has selected via the checkbox under the Content option. Any items which are checked, or unchecked, in Content are included, or excluded, in Archive. Likewise any item which is unchecked in Archive is unchecked in Content. Once the user is satisfied with the selection, the name of the archive is entered and the Create button is clicked; an archive request is sent to the server. Once again, the user only receives a notification if there was a failure.

Double-clicking a content item will present input for renaming the content. Once the user has finished and the input has lost the focus, then a client request is sent to the server which renames the content item.

The filter is applied to whatever rows are currently being displayed and is used on all the columns. The filter does not change the selection of items to be archived—it only hides them from display.

# Troubleshooting

These instructions are intended to give some direction when problems arise in the system. Most problems will occur when a user makes a request only to (eventually) see that it has not been satisfied. It is advisable to start from the client-side and then follow the control flow to the desired conclusion of the request in order to hopefully find the problem.

## CLIENT Requests

There are three types of client requests which may be sent to the server: download (CP), delete, and rename (web interface). All three of these types of request use the same mechanism and any problems they experience will be very similar to one another.

The user may receive these error messages:

“ERROR: Server cannot be reached”: the web server is inaccessible, check its status.

“SERVER ERROR: …”: clientreq.pl received the request but an error occurred. View the contents of the script to determine the exact cause of the message.

“UNKNOWN ERROR: …”: Server is accessible, but the CGI script did not respond.

If the user receives no error message, then the request was written to the pipe to the Master Control Program. View the client request log and the master log to look for where the request may have been lost. Also check that the pipe exists and has the proper permissions.

If the request made it to the Master Control Program then check the database; the rest of this section only pertains to downloads. If the job does not exist in the JOB table then check the master log for an error message. If the job does exist, but has not been completed then it will not have a CMPL\_DATE and the status will be pending. Also check if the status is failed. Non-completion most likely indicates that it has not been selected for download yet. A failed job will not be attempted again by the downloader. In either case check the master and downloader logs. If the job failed, it can be attempted again by setting the status to pending and waiting for it to be selected for download—or using CAPManage to send the kick command to the Master Control Program if it is waiting for messages.

If the job has been completed but does not exist in the CONTENT table then there may have been miscommunication between the Master Control Program and downloader. Check their error logs.

If the job has been completed but does not exist in the content directory, then check the master log because a file operation may have failed.

If none of the above has yielded a solution, then the problem might be that the web interface is not properly selecting and displaying the content.

## Archive Requests

The user may receive the message, “SERVER ERROR: …” when attempting to create an archive. View the contents of the archreq.pl script to determine the cause of the error.

The steps for troubleshooting archive requests are similar to those for download requests. When checking in the database, examine the CONTENT and ARCHIVE tables to find a matching ID as this indicates that the content is an archive. Also look in the ARCHIVE\_CONTENT table for expected relationships between the archive and content.

# Glossary

|  |  |
| --- | --- |
| Archive | A collection of content items within one ZIP file. |
| Archiver | Server component which combines specified content into an archive. |
| Client | The machine which the end-user is operating from. |
| Client plug-in (CP) | The Firefox extension which adds a toolbar to the user’s browser; it sends download requests from the user’s machine to the server for processing. |
| Configuration file | An XML file which stores the settings used by the system during operation. These files are only edited by the system administrator—the system, itself, does not touch them. |
| Content | Any data downloaded or archived by user. HTML pages and archives are both considered content. |
| Downloader | Crawls the Internet and downloads the specified URL to create new content. |
| Firefox | The Mozilla® Firefox web browser. |
| Master Control Program | Program which sits at the center of the server architecture receiving and sending messages, and coordinating system tasks. |
| Pipe | A communications device which is used to communicate between components. The system uses a specific type of pipe called a FIFO. |
| Server | The machine which hosts all of the system components and their requirements. |