AmmarServer

Generated by Doxygen 1.8.6

Thu Jul 17 2014 15:44:54

Contents

Chapter 1

AmmarServer

Author

Ammar Qammaz a.k.a. AmmarkoV - http://ammar.gr

A lightweight extendable barebones HTTP server for linux Please see the wiki for more info on whats going on in this repository :) https://github.com/AmmarkoV/AmmarServer/wiki

One of the most basic philosophies behind this is to try to add as much functionality possible in a reusable and very fast way and *WITHOUT* overly increasing lines of code (loc) .. The biggest recent improvements have been actually trying to merge common functionality and reducing loc , and code complexity .

Website: https://github.com/AmmarkoV/RGBDAcquisition

1.1 Introduction and History

AmmarServer began as a small sockets project back on 2004 , its main use back then was serving as a portable executable that I could take with me to share static files between different machines without having administrator privilages , setting up shares , on different Operating Systems and network topologies..

Needless to say despite beeing "my own brainchild" , it wasn't a webserver particularly useful on anything but static content and I always used Apache , MySQL and PHP as infrastructure for serious web-development work which served me well .. until I started working on embedded systems..

The Apache web server is a wonderful piece of software with a very large collection of plugins and modules and a huge percentage of the internet gets served by it every day, it is robust , mature , well documented and it is secure.. But all these positive qualities also mean that it is big and it is complex requiring a relatively large deployment and configuration payload (for a LAMP installation).

Using PHP (or any other interpreted high-level language) felt right at home from the first time I used it. With its C-like language structure but more goodies like multi line strings and loose variable declaration rules . It proved to be an invaluable tool but gradually also proved a heavy task for computer hosts lacking many computing resources or serving a very large number of requests. The picture got even worse when services like Wordpress (which is also great) that have many thousands lines of code generate dynamic content.. The delays , wether they where Disk , CPU or Memory based summed up and this lead to a very bad user experience while accessing and browsing various site configurations. Of course I am not the only one that has observed this and there are many projects to improve the situation and combat performance overheads such as the Hip-Hop library developed by facebook that translated php to a C++ generating a compiled binary and reduced their loads by a respectable 50%.. Other "home-made" solutions I tried was operating on memfs or ramfs partitions and many other hacks which optimized things more and more..

At some point I thought.. All this is good but is it the best that can be done ? What would be the best way to do it ? The least overhead possible can only be achieved by closely coupling the webserver with the dynamic content it serves. Compiled php binaries offer a faster way to generate the content but this content is loosely tied with the server that actually sends it. Instead of having seperate "entities" for the webserver the architecture of AmmarServer statically links the webserver library with the dynamic content which is compiled into the same executable..

The simple implementation of AmmarServer

1.2 What Is it ?

So the question is , What is it exactly ? AmmarServer is a low level framework that allows the creation of binary executables which contain both their webserver and the ability to generate dynamic pages.

A sample application that demonstrates this concept and you can see , is my V4L2ToHTTP project (https://github.com/AmmarkoV/V4L2ToHTTP) that uses AmmarServer as its backbone. V4L2ToHTTP is aimed at a thin server that receives frames from a video device (i.e. webcam) encodes them into jpeg format in memory , and when a client requests a version of cam.jpg the callback dynamically snaps and uploads a new frame from the camera. The whole point of course is having the minimal possible internal "generation/communication" overhead and the lowest possible memory footprint since the frame is mmaped to the place where the kernel receives the USB camera frame data , it then uses libjpeg for a hardware optimized conversion and then just basically moves a pointer address which is utilized by the send socket command to send the frame. The datapath literally can't get any smaller..

The way to write a web-service using AmmarServer is somewhat different than writing a PHP webservice on Apache. Each service is a different executable (process) that binds a TCP/IP port , instead of a collection of scripts. In order to serve clients each service spawns its own maximum number of threads (and can get individually balanced by the kernel scheduler) instead of preforking seperate processes like Apache does. AmmarServer works with threads (see why) in order to use a lower overall amount of memory , and to make it easier for the programmer (and the Kernel process scheduler) to prioritize serving requests on tight budgets. WebHosting services like godaddy etc.. are not fit to use this model since every "page" would have to be a seperate "server" .. So this server doesnot target this deployment scenario but rather the "dedicated hosting" one.. It is not really meant for a server that hosts 1000 different PHP services (although I guess it could also be used that way ;P) but for a Facebook or Youtube like project when we want a few number of services like uploading , caching and serving content , browsing pictures , editing profile information etc in order to make each of these sub tasks a different "server" as efficient as possible and maintaining a conceptually simple and maintainable model for the developer.

1.3 Coding Style

Coding style helpers are kind of a stub for now , since there are key parts of the library that are missing , and providing easier calls , aliases etc for missing functionality is impossible , it is something that should really happen in the future.. What I basically want to say is that the Model->View abstractions of Rails , or other modeling techniques are a nice thing , and there isn't any relevant helper functions built in the framework , or even a coding template so unfortunately I can't tell you how to organize your content for now.. I would guess that AmmarServer would naturaly mix well with ECPG (PostgreSQL embedded for C) and that the state information could be kept there if you want to use SQL. Most of the visual things (CSS , images , videos , audio , JScripts) should be in static files , and that the callbacks for conent shouldn't be thousands lines of code but instead use external functions that fit the model of the problem you are trying to solve..

There is also extensive testing that has to be done and things related to static string allocation etc .. , things are in a moderate shape right now and can be improved (for sure)

AmmarServer is relatively stable , but not thoroughly tested (security , pentesting etc) I certainly hope you will find it an interesting and handy codebase..

1.4 Future Plans

Future Planned Projects using AmmarServer

- An opensource RomPager alternative or Webmin alternative
- A more efficient version of myloader (https://github.com/MasterEx/myloader)

- Replacing Apache as the Web Interface of GuarddoG robot
- Making a Video Surveillance daemon like zoneminder with emphasis on performance and small system footprint
- Replacing my WebServer (for http://ammar.gr) with a Rasberry pi running AmmarServer

1.5 Deployment

To download AmmarServer you can click here (https://github.com/AmmarkoV/AmmarServer/archive/master.zip) or issue "git clone http://github.com/AmmarkoV/AmmarServer.git" on your terminal To compile it issue mkdir build && cd build && cmake .. && make while beeing in the root directory of the repo.. To run it using the default settings issue ./run_ammarserver You should review the list of open issues to better inform yourself of the current state of the server

1.6 Dependencies

The projects build dependencies are the gcc compiler , pthreads and pretty basic things (see https://raw.githubusercontent.com/AmmarkoV/AmmarServer/master/doc/dependencies.jpg) which is one of the core reasons of the "beauty" of this project :)

So if you issue sudo apt-get install build-essential (assuming a Debian/Ubuntu based system) you should be able to compile it without problems..

Newer versions also support compression , so you might want to also apt-get install liblzma-dev if you enable ENABLE_COMPRESSION at server_configuration.h MyURL needs libjpeg in order to serve captchas , so to add it sudo apt-get install libjpeg-dev

Compilation is controlled using cmake , so to perform a compilation you just need to issue mkdir build cd build cmake .. make

To update your version of the project you can use the provided script that updates directly from github It will remove any changes you have made to any of the files in the repository ./update_from_git from the root directory

Chapter 2

Bug List

Global AMMAR_SERVER_HTTP_HEADER_SPEC

A potential bug might arise if the specs of the header file are changed and someone is linking with an older version libAmmServer.a thats why this value exists

Global AmmServer_ExecuteCommandLine (char ∗command, char ∗what2GetBack, unsigned int what2GetBackMaxSize)

Executing commands can be dangerous , always check and sanitize input before executing , Also be sure about the max size of output so that you don't lose a part of it , also make something like escapeshellcmd

Global AmmServer_ReplaceVarInMemoryFile (char ∗page, unsigned int pageLength, char ∗var, char ∗value)

Value should not be bigger than variable otherwise things won't fit in the same memory block

Global AmmServer_SelfCheck (struct AmmServer_Instance ∗instance)

Maybe remove AmmServer_SelfCheck

Global AmmServer_SignalCountAsBadClientBehaviour (struct AmmServer_Instance ∗instance, struct AmmServer_DynamicRequest ∗rqst)

Client behaviours etc are not implemented yet

File AmmServerlib.h

AmmarServer is not properly pentested yet

Global cache_GetResource (struct AmmServer_Instance ∗instance, struct HTTPHeader ∗request, unsigned int resourceCacheID, char ∗verified_filename, unsigned int ∗index, unsigned long ∗filesize, struct stat ∗last_modification, unsigned char ∗compressionSupported, unsigned char ∗freeContentAfterUsingIt)

If verified_filename , is not really verified (i.e. outside of the public_html root directory , this function could pose a security problem , since it will just blindly open and serve the filename given to it)

Return values

1=Ok,0=Failed	

File client_list.h

Client Lists are a stub and not implemented yet

File dynamic_requests.h

Compression should be improved

File file_caching.h

File caching relies on hashmap for storing data , so it relies on optimizations done there for seek time optimization , other than that there needs to be a clean-up and code quality improvement

File file_compression.h

Compression should be improved

Global GenerateDirectoryPage (char ∗system_path, char ∗client_path, unsigned long ∗memoryUsed)

GenerateDirectoryPage does not handle memory correctly , code is in very bad shape

File hashmap.h

This hashmap implementation uses serial searches for now , and needs a lot of work

File http_header_analysis.h

HTTP header analysis can be improved (code style etc) although the recent use of stringscanners has greatly improved it and reduced lines of code

Global LoadConfigurationFile (struct AmmServer_Instance *instance, char *conf_file)

LoadConfigurationFiles etc is not ready yet , although it relies on InputParser and should be easy to implement , there are just things missing still and that's why I postpone implementing it

Global MAX_CLIENTS_PER_IP

MAX_CLIENTS_PER_IP is not used if there is no client list declared

File post_header_analysis.h

POST header analysis is not fully implemented yet

File prespawnedThreads.h

Prespawned threads have race conditions ?

File server_configuration.h

Server configuration at some point should be ported from defines to a per instance configuration file , some of these defines will always remain since they control global allocations

Global StopHTTPServer (struct AmmServer_Instance *instance)

Stop web server should be improved , to make sure it unbinds the closing socket

Return values

1=Success,0=Failure	

Global TEMPLATE_INTERNAL_URI

Please note that the file server has limits for filenames so this should not be very long *asvres*/filename.jpg is OK a filename like *asvres*/filenamemplampla.jpg will return a 404

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Data Structure Documentation

5.1 AmmServer_DynamicRequest Struct Reference

When a call to a function that is a dynamic request is done this is the structure that holds the information.

`#include <AmmServerlib.h>`

Data Fields

- unsigned int headerResponse
- char * content
- unsigned long contentSize
- unsigned long MAXcontentSize
- char * compressedContent
- unsigned long compressedContentSize
- unsigned long MAXcompressedContentSize
- char * GET_request
- unsigned int GET_request_length
- char * POST_request
- unsigned int POST_request_length
- unsigned int clientID

5.1.1 Detailed Description

When a call to a function that is a dynamic request is done this is the structure that holds the information.

5.1.2 Field Documentation

5.1.2.1 unsigned int clientID

5.1.2.2 char* compressedContent

5.1.2.3 unsigned long compressedContentSize

5.1.2.4 char* content

5.1.2.5 unsigned long contentSize

5.1.2.6 char* GET_request

5.1.2.7 unsigned int GET_request_length

5.1.2.8 unsigned int headerResponse

5.1.2.9 unsigned long MAXcompressedContentSize

5.1.2.10 unsigned long MAXcontentSize

5.1.2.11 char* POST_request

5.1.2.12 unsigned int POST_request_length

The documentation for this struct was generated from the following file:

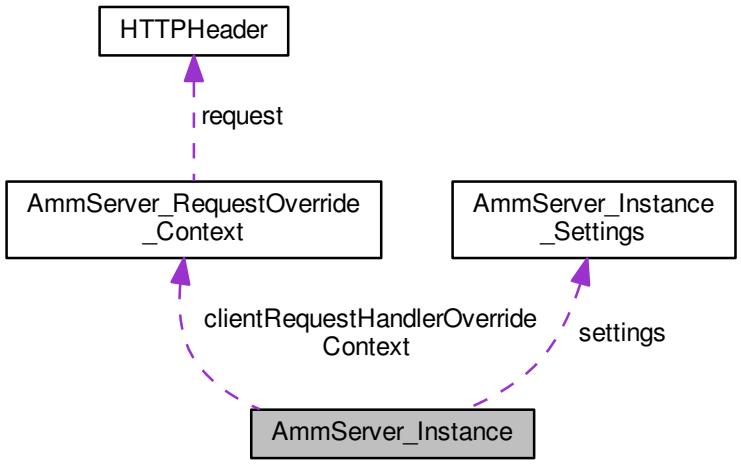
- src/AmmServerlib/AmmServerlib.h

5.2 AmmServer_Instance Struct Reference

This holds all the information about an Ammar Server Instance , sockets , thread pools , cache , memory , settings etc , this is the central structure for holding context.

`#include <AmmServerlib.h>`

Collaboration diagram for AmmServer_Instance:



Data Fields

- char instanceName [MAX_INSTANCE_NAME_STRING]
- struct AmmServer_Instance_Settings settings
- unsigned int prespawn_turn_to_serve
- unsigned int prespawn_jobs_started

- unsigned int prespawn_jobs_finished
- int files_open
- int serversock
- int server_running
- int pause_server
- int stop_server
- unsigned long loaded_cache_items_Kbytes
- unsigned int loaded_cache_items
- void * cache
- void * cacheHashMap
- void * clientList
- unsigned int CLIENT_THREADS_STARTED
- unsigned int CLIENT_THREADS_STOPPED
- pthread_t server_thread_id
- pthread_t * threads_pool
- void * prespawned_pool
- struct AmmServer_RequestOverride_Context * clientRequestHandlerOverrideContext
- char webserver_root [MAX_FILE_PATH]
- char templates_root [MAX_FILE_PATH]

5.2.1 Detailed Description

This holds all the information about an Ammar Server Instance , sockets , thread pools , cache , memory , settings etc , this is the central structure for holding context.

5.2.2 Field Documentation

5.2.2.1 void* cache

5.2.2.2 void* cacheHashMap

5.2.2.3 unsigned int CLIENT_THREADS_STARTED

5.2.2.4 unsigned int CLIENT_THREADS_STOPPED

5.2.2.5 void* clientList

5.2.2.6 struct AmmServer_RequestOverride_Context* clientRequestHandlerOverrideContext

5.2.2.7 int files_open

5.2.2.8 char instanceName[MAX_INSTANCE_NAME_STRING]

5.2.2.9 unsigned int loaded_cache_items

5.2.2.10 unsigned long loaded_cache_items_Kbytes

5.2.2.11 int pause_server

5.2.2.12 unsigned int prespawn_jobs_finished

5.2.2.13 unsigned int prespawn_jobs_started

5.2.2.14 unsigned int prespawn_turn_to_serve

5.2.2.15 void* prespawned_pool

5.2.2.16 int server_running

5.2.2.17 pthread_t server_thread_id

5.2.2.18 int serversock

5.2.2.19 struct AmmServer_Instance_Settings settings

5.2.2.20 int stop_server

5.2.2.21 char templates_root[MAX_FILE_PATH]

5.2.2.22 pthread_t* threads_pool

5.2.2.23 char webserver_root[MAX_FILE_PATH]

The documentation for this struct was generated from the following file:

- src/AmmServerlib/AmmServerlib.h

5.3 AmmServer_Instance_Settings Struct Reference

Each Instance of AmmarServer has some basic settings , which are stored in AmmServer_Instance_Settings.

`#include <AmmServerlib.h>`

Data Fields

- int PASSWORD_PROTECTION
- char * USERNAME
- char * PASSWORD
- char * BASE64PASSWORD
- int BINDING_PORT

5.3.1 Detailed Description

Each Instance of AmmarServer has some basic settings , which are stored in AmmServer_Instance_Settings.

5.3.2 Field Documentation

5.3.2.1 char* BASE64PASSWORD

5.3.2.2 int BINDING_PORT

5.3.2.3 char* PASSWORD

5.3.2.4 int PASSWORD_PROTECTION

5.3.2.5 char* USERNAME

The documentation for this struct was generated from the following file:

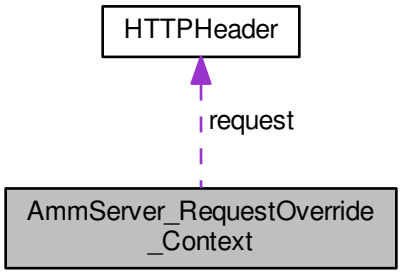
- src/AmmServerlib/AmmServerlib.h

5.4 AmmServer_RequestOverride_Context Struct Reference

We can override/intercept connections before the very fundamental HTTP stage using a request override context and AmmServer_AddRequestHandler This is the structure that holds the information and what to be called back to populate the response.

`#include <AmmServerlib.h>`

Collaboration diagram for AmmServer_RequestOverride_Context:



Data Fields

- char requestHeader [64]
- struct HTTPHeader * request
- void * request_override_callback

5.4.1 Detailed Description

We can override/intercept connections before the very fundamental HTTP stage using a request override context and AmmServer_AddRequestHandler This is the structure that holds the information and what to be called back to populate the response.

5.4.2 Field Documentation

5.4.2.1 struct HTTPHeader* request

5.4.2.2 void* request_override_callback

5.4.2.3 char requestHeader[64]

The documentation for this struct was generated from the following file:

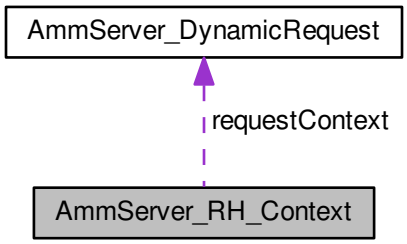
- src/AmmServerlib/AmmServerlib.h

5.5 AmmServer_RH_Context Struct Reference

We can override resources to respond with our own C function code , to do so a AmmServer_DynamicRequest must be populated using a AmmServer_AddResourceHandler.

`#include <AmmServerlib.h>`

Collaboration diagram for AmmServer_RH_Context:



Data Fields

- unsigned int RH_Scenario
- unsigned int last_callback
- unsigned int callback_every_x_msec
- char callback_cooldown
- void * dynamicRequestCallbackFunction
- char web_root_path [MAX_FILE_PATH]
- char resource_name [MAX_RESOURCE]
- struct AmmServer_DynamicRequest requestContext

5.5.1 Detailed Description

We can override resources to respond with our own C function code , to do so a AmmServer_DynamicRequest must be populated using a AmmServer_AddResourceHandler.

5.5.2 Field Documentation

5.5.2.1 char callback_cooldown

5.5.2.2 unsigned int callback_every_x_msec

5.5.2.3 void* dynamicRequestCallbackFunction

5.5.2.4 unsigned int last_callback

5.5.2.5 struct AmmServer_DynamicRequest requestContext

5.5.2.6 char resource_name[MAX_RESOURCE]

5.5.2.7 unsigned int RH_Scenario

5.5.2.8 char web_root_path[MAX_FILE_PATH]

The documentation for this struct was generated from the following file:

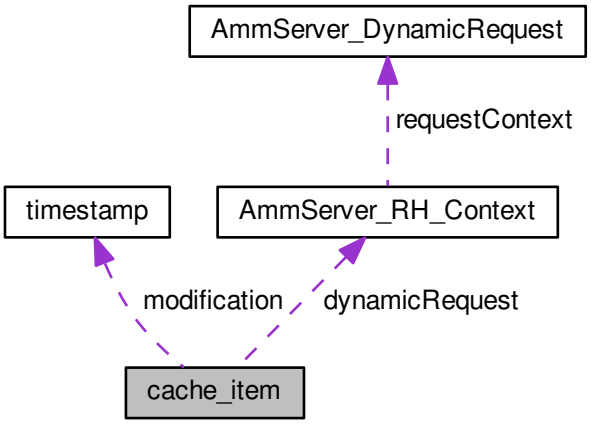
- src/AmmServerlib/AmmServerlib.h

5.6 cache_item Struct Reference

A cache item and all it's contents.

`#include <file_caching.h>`

Collaboration diagram for cache_item:



Data Fields

- void * dynamicRequestCallbackFunction
- struct AmmServer_RH_Context * dynamicRequest
- unsigned char doNOTCacheRule
- char * content
- unsigned long * contentSize
- char * compressedContent
- unsigned long * compressedContentSize
- contentType contentTypeID
- struct timestamp modification

5.6.1 Detailed Description

A cache item and all it's contents.

5.6.2 Field Documentation

5.6.2.1 char* compressedContent

5.6.2.2 unsigned long* compressedContentSize

5.6.2.3 char* content

5.6.2.4 unsigned long* contentSize

5.6.2.5 contentType contentTypeID

5.6.2.6 unsigned char doNOTCacheRule

5.6.2.7 struct AmmServer_RH_Context* dynamicRequest

5.6.2.8 void* dynamicRequestCallbackFunction

5.6.2.9 struct timestamp modification

The documentation for this struct was generated from the following file:

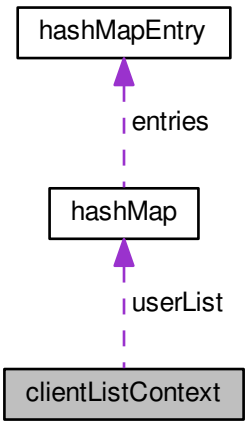
- src/AmmServerlib/cache/file_caching.h

5.7 clientListContext Struct Reference

The client list is just a hashmap (see hashmap.h)

`#include <client_list.h>`

Collaboration diagram for clientListContext:



Data Fields

- struct hashMap * userList

5.7.1 Detailed Description

The client list is just a hashmap (see hashmap.h)

5.7.2 Field Documentation

5.7.2.1 struct hashMap* userList

The documentation for this struct was generated from the following file:

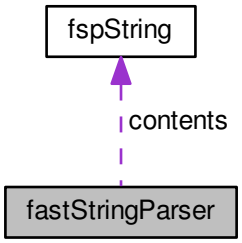
- src/AmmServerlib/cache/client_list.h

5.8 fastStringParser Struct Reference

Internal Structure that holds all the string parser context.

`#include <fastStringParser.h>`

Collaboration diagram for fastStringParser:



Data Fields

- struct fspString * contents
- unsigned int stringsLoaded
- unsigned int MAXstringsLoaded
- char * functionName
- unsigned int shortestStringLength
- unsigned int longestStringLength

5.8.1 Detailed Description

Internal Structure that holds all the string parser context.

5.8.2 Field Documentation

5.8.2.1 struct fspString* contents

5.8.2.2 char* functionName

5.8.2.3 unsigned int longestStringLength

5.8.2.4 unsigned int MAXstringsLoaded

5.8.2.5 unsigned int shortestStringLength

5.8.2.6 unsigned int stringsLoaded

The documentation for this struct was generated from the following file:

- src/StringRecognizer/fastStringParser.h

5.9 fspString Struct Reference

Internal Structure to hold a string and its id for further processing.

`#include <fastStringParser.h>`

Data Fields

- char * str
- char * strIDFriendly
- unsigned int strLength

5.9.1 Detailed Description

Internal Structure to hold a string and its id for further processing.

5.9.2 Field Documentation

5.9.2.1 char* str

5.9.2.2 char* strIDFriendly

5.9.2.3 unsigned int strLength

The documentation for this struct was generated from the following file:

- src/StringRecognizer/fastStringParser.h

5.10 guard_byte Struct Reference

`#include <InputParser_C.h>`

Data Fields

- unsigned int checksum

5.10.1 Field Documentation

5.10.1.1 unsigned int checksum

The documentation for this struct was generated from the following file:

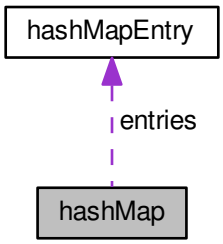
- src/AmmServerlib/InputParser/InputParser_C.h

5.11 hashMap Struct Reference

The central structure for the hash map.

`#include <hashmap.h>`

Collaboration diagram for hashMap:



Data Fields

- unsigned int maxNumberOfEntries
- unsigned int curNumberOfEntries
- unsigned int entryAllocationStep
- struct hashMapEntry * entries
- void * clearItemCallbackFunction
- pthread_mutex_t hm_addLock
- pthread_mutex_t hm_fileLock

5.11.1 Detailed Description

The central structure for the hash map.

5.11.2 Field Documentation

5.11.2.1 void* clearItemCallbackFunction

5.11.2.2 unsigned int curNumberOfEntries

5.11.2.3 struct hashMapEntry* entries

5.11.2.4 unsigned int entryAllocationStep

5.11.2.5 pthread_mutex_t hm_addLock

5.11.2.6 pthread_mutex_t hm_fileLock

5.11.2.7 unsigned int maxNumberOfEntries

The documentation for this struct was generated from the following file:

- src/AmmServerlib/hashmap/hashmap.h

5.12 hashMapEntry Struct Reference

An entry on the hash map flattened out for ease of use.

`#include <hashmap.h>`

Data Fields

- unsigned long keyHash
- unsigned int keyLength
- char * key
- unsigned int payloadLength
- void * payload
- unsigned int hits

5.12.1 Detailed Description

An entry on the hash map flattened out for ease of use.

5.12.2 Field Documentation

5.12.2.1 unsigned int hits

5.12.2.2 char* key

5.12.2.3 unsigned long keyHash

5.12.2.4 unsigned int keyLength

5.12.2.5 void* payload

5.12.2.6 unsigned int payloadLength

The documentation for this struct was generated from the following file:

- src/AmmServerlib/hashmap/hashmap.h

5.13 HTTPHeader Struct Reference

Each HTTP Request has a header , this is the internal structure that carries the information about the header of an HTTP request parsed and ready for easy for consumption by the various consumers of HTTP requests.

`#include <AmmServerlib.h>`

Data Fields

- char * headerRAW
- unsigned int headerRAWSize
- int requestType
- char resource [MAX_RESOURCE+1]
- char verified_local_resource [MAX_FILE_PATH+1]
- char GETquery [MAX_QUERY+1]
- char * POSTrequest
- unsigned long POSTrequestSize
- unsigned char authorized
- unsigned char keepalive
- unsigned char supports_compression
- unsigned long range_start
- unsigned long range_end
- unsigned long ContentLength
- char * cookie
- unsigned int cookieLength
- char * host
- unsigned int hostLength
- char * referer
- unsigned int refererLength
- char * eTag
- unsigned int eTagLength
- char * userAgent
- unsigned int userAgentLength
- char * contentType
- unsigned int contentTypeLength
- char * contentDisposition
- unsigned int contentDispositionLength
- char * boundary
- unsigned int boundaryLength

5.13.1 Detailed Description

Each HTTP Request has a header , this is the internal structure that carries the information about the header of an HTTP request parsed and ready for easy for consumption by the various consumers of HTTP requests.

5.13.2 Field Documentation

5.13.2.1 unsigned char authorized

5.13.2.2 char* boundary

5.13.2.3 unsigned int boundaryLength

5.13.2.4 char* contentDisposition

5.13.2.5 unsigned int contentDispositionLength

5.13.2.6 unsigned long ContentLength

5.13.2.7 char* contentType

5.13.2.8 unsigned int contentTypeLength

5.13.2.9 char* cookie

5.13.2.10 unsigned int cookieLength

5.13.2.11 char* eTag

5.13.2.12 unsigned int eTagLength

5.13.2.13 char GETquery[MAX_QUERY+1]

5.13.2.14 char* headerRAW

5.13.2.15 unsigned int headerRAWSize

5.13.2.16 char* host

5.13.2.17 unsigned int hostLength

5.13.2.18 unsigned char keepalive

5.13.2.19 char* POSTrequest

5.13.2.20 unsigned long POSTrequestSize

5.13.2.21 unsigned long range_end

5.13.2.22 unsigned long range_start

5.13.2.23 char* referer

5.13.2.24 unsigned int refererLength

5.13.2.25 int requestType

5.13.2.26 char resource[MAX_RESOURCE+1]

5.13.2.27 unsigned char supports_compression

5.13.2.28 char* userAgent

5.13.2.29 unsigned int userAgentLength

5.13.2.30 char verified_local_resource[MAX_FILE_PATH+1]

The documentation for this struct was generated from the following file:

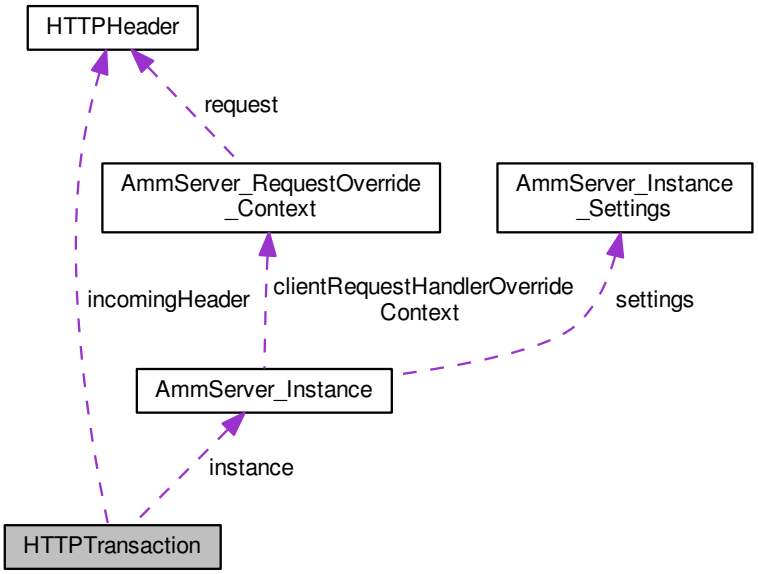
- src/AmmServerlib/AmmServerlib.h

5.14 HTTPTransaction Struct Reference

Structure to keep data for an HTTP Transaction.

`#include <AmmServerlib.h>`

Collaboration diagram for HTTPTransaction:



Data Fields

- struct AmmServer_Instance * instance
- struct HTTPHeader incomingHeader
- char * outgoingBody
- unsigned int outgoingBodySize
- unsigned int resourceCacheID
- int clientSock
- unsigned int clientListID
- unsigned int threadID
- int prespawnedThreadFlag

5.14.1 Detailed Description

Structure to keep data for an HTTP Transaction.

5.14.2 Field Documentation

5.14.2.1 unsigned int clientListID

5.14.2.2 int clientSock

5.14.2.3 struct HTTPHeader incomingHeader

5.14.2.4 struct AmmServer_Instance* instance

5.14.2.5 char* outgoingBody

5.14.2.6 unsigned int outgoingBodySize

5.14.2.7 int prespawnedThreadFlag

5.14.2.8 unsigned int resourceCacheID

5.14.2.9 unsigned int threadID

The documentation for this struct was generated from the following file:

- src/AmmServerlib/AmmServerlib.h

5.15 Image Struct Reference

`#include <imaging.h>`

Data Fields

- unsigned char * pixels
- unsigned int width
- unsigned int height
- unsigned int depth
- unsigned int imageSize

5.15.1 Field Documentation

5.15.1.1 unsigned int depth

5.15.1.2 unsigned int height

5.15.1.3 unsigned int imageSize

5.15.1.4 unsigned char* pixels

5.15.1.5 unsigned int width

The documentation for this struct was generated from the following file:

- src/AmmCaptcha/imaging.h

5.16 InputParser Class Reference

`#include <InputParser.h>`

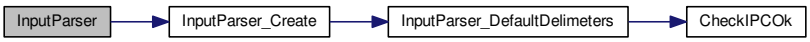
Public Member Functions

- const char * Version ()
- void DefaultDelimeterSetup ()
- InputParser ()
- ~InputParser ()
- void SetDelimeter (int num, char tmp)
- char GetDelimeter (int num)
- unsigned int GetWord (int num, char *thestr, unsigned int thestrsize)
- unsigned int GetUpcaseWord (int num, char *thestr, unsigned int thestrsize)
- unsigned int GetLowercaseWord (int num, char *thestr, unsigned int thestrsize)
- char GetWordChar (int num, int chr)
- signed int GetWordInt (int num)
- unsigned short GetWordLength (int num)
- int SeperateWords (char *inpt)
- int SeperateWordsCC (const char *inpt)
- int SeperateWordsUC (unsigned char *inpt)

5.16.1 Constructor & Destructor Documentation

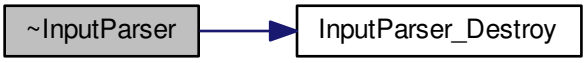
5.16.1.1 InputParser ( )

Here is the call graph for this function:



5.16.1.2 ~InputParser ( )

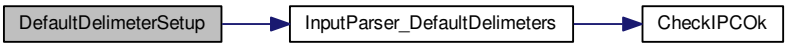
Here is the call graph for this function:



5.16.2 Member Function Documentation

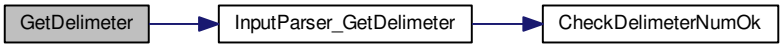
5.16.2.1 void DefaultDelimeterSetup ( )

Here is the call graph for this function:



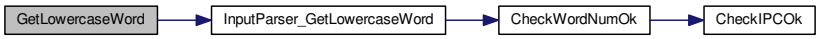
5.16.2.2 char GetDelimeter (int *num*)

Here is the call graph for this function:



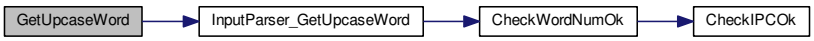
5.16.2.3 unsigned int GetLowercaseWord (int *num*, char * *thestr*, unsigned int *thestrsize*)

Here is the call graph for this function:



5.16.2.4 unsigned int GetUpcaseWord (int *num*, char * *thestr*, unsigned int *thestrsize*)

Here is the call graph for this function:

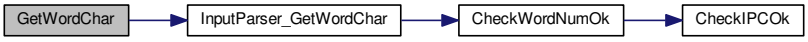


5.16.2.5 unsigned int GetWord (int *num*, char * *thestr*, unsigned int *thestrsize*)

Here is the call graph for this function:

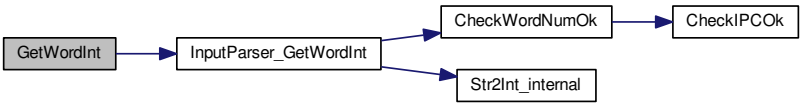
5.16.2.6 char GetWordChar (int *num*, int *chr*)

Here is the call graph for this function:



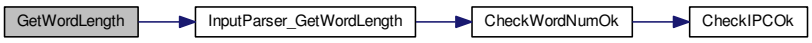
5.16.2.7 signed int GetWordInt (int *num*)

Here is the call graph for this function:



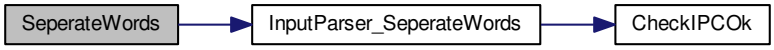
5.16.2.8 unsigned short GetWordLength (int *num*)

Here is the call graph for this function:



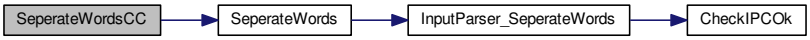
5.16.2.9 int SeperateWords (char * *inpt*)

Here is the call graph for this function:



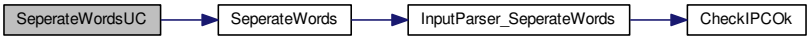
5.16.2.10 int SeperateWordsCC (const char * *inpt*)

Here is the call graph for this function:



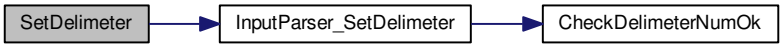
5.16.2.11 int SeperateWordsUC (unsigned char * *inpt*)

Here is the call graph for this function:



5.16.2.12 void SetDelimeter (int *num,* char *tmp*)

Here is the call graph for this function:



5.16.2.13 const char* Version ( )

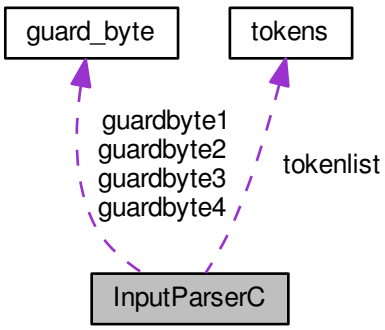
The documentation for this class was generated from the following files:

- src/AmmServerlib/InputParser/InputParser.h
- src/AmmServerlib/InputParser/InputParser.cpp

5.17 InputParserC Struct Reference

`#include <InputParser_C.h>`

Collaboration diagram for InputParserC:



Data Fields

- struct guard_byte guardbyte1
- unsigned int str_length
- unsigned char local_allocation
- char * str
- struct guard_byte guardbyte2
- unsigned short cur_container_count
- unsigned short max_container_count
- char * container_start
- char * container_end
- unsigned short cur_delimeter_count
- unsigned short max_delimeter_count
- char * delimeters
- struct guard_byte guardbyte3
- unsigned int tokens_max
- unsigned int tokens_count
- struct tokens * tokenlist
- struct guard_byte guardbyte4

5.17.1 Field Documentation

5.17.1.1 char* container_end

5.17.1.2 char* container_start

5.17.1.3 unsigned short cur_container_count

5.17.1.4 unsigned short cur_delimeter_count

5.17.1.5 char* delimeters

5.17.1.6 struct guard_byte guardbyte1

5.17.1.7 struct guard_byte guardbyte2

5.17.1.8 struct guard_byte guardbyte3

5.17.1.9 struct guard_byte guardbyte4

5.17.1.10 unsigned char local_allocation

5.17.1.11 unsigned short max_container_count

5.17.1.12 unsigned short max_delimeter_count

5.17.1.13 char* str

5.17.1.14 unsigned int str_length

5.17.1.15 struct tokens* tokenlist

5.17.1.16 unsigned int tokens_count

5.17.1.17 unsigned int tokens_max

The documentation for this struct was generated from the following file:

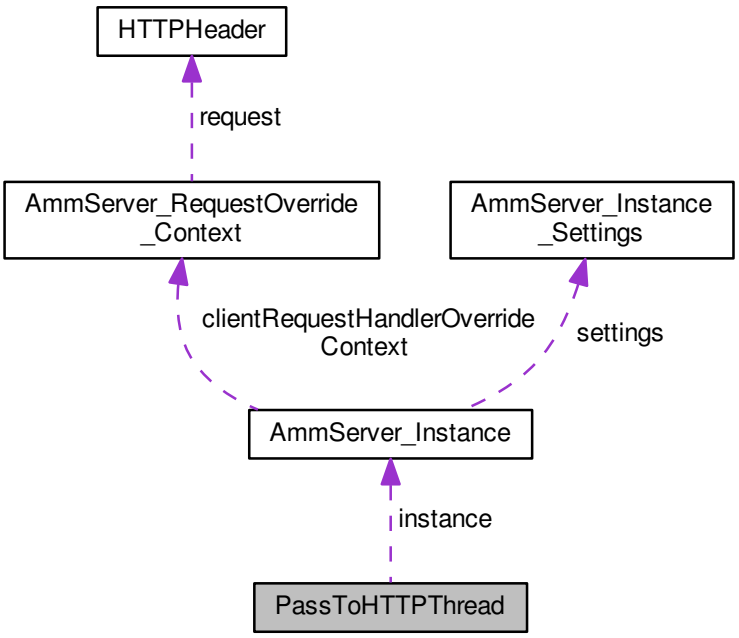
- src/AmmServerlib/InputParser/InputParser_C.h

5.18 PassToHTTPThread Struct Reference

A structure that holds information to be passed from the main thread to the new (fresh) thread.

`#include <freshThreads.h>`

Collaboration diagram for PassToHTTPThread:



Data Fields

- volatile int keep_var_on_stack
- struct sockaddr_in client
- unsigned int clientlen
- unsigned int thread_id
- unsigned int port
- int clientsock
- struct AmmServer_Instance * instance
- int pre_spawned_thread

5.18.1 Detailed Description

A structure that holds information to be passed from the main thread to the new (fresh) thread.

5.18.2 Field Documentation

5.18.2.1 struct sockaddr_in client

5.18.2.2 unsigned int clientlen

5.18.2.3 int clientsock

5.18.2.4 struct AmmServer_Instance* instance

5.18.2.5 volatile int keep_var_on_stack

5.18.2.6 unsigned int port

5.18.2.7 int pre_spawned_thread

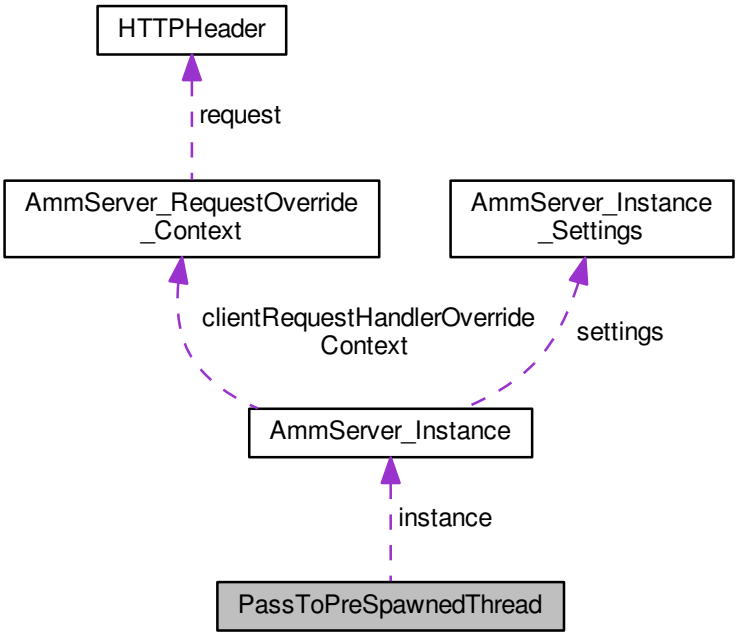
5.18.2.8 unsigned int thread_id

The documentation for this struct was generated from the following file:

- src/AmmServerlib/threads/freshThreads.h

5.19 PassToPreSpawnedThread Struct Reference

Collaboration diagram for PassToPreSpawnedThread:



Data Fields

- struct AmmServer_Instance * instance
- unsigned int i_adapt

5.19.1 Field Documentation

5.19.1.1 unsigned int i_adapt

5.19.1.2 struct AmmServer_Instance* instance

The documentation for this struct was generated from the following file:

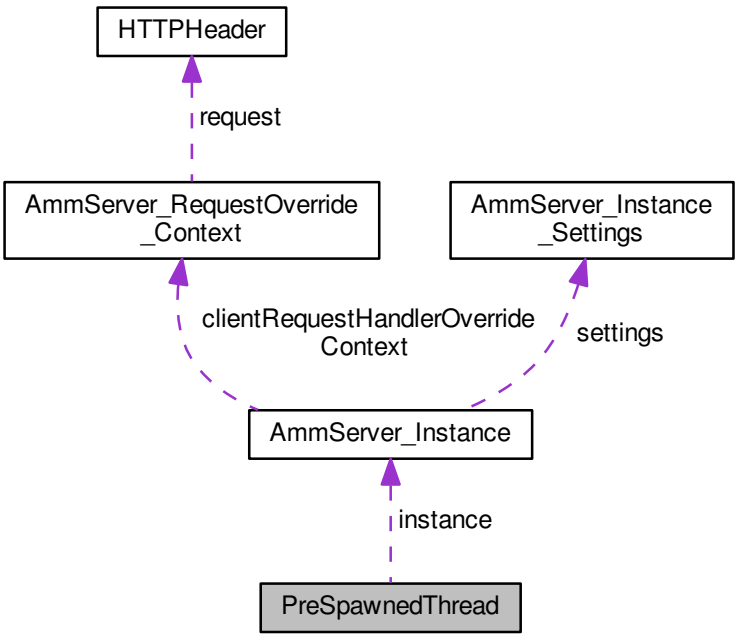
- src/AmmServerlib/threads/prespawnedThreads.c

5.20 PreSpawnedThread Struct Reference

A structure that holds information to be passed from the main thread to the new (prespawned) thread.

`#include <prespawnedThreads.h>`

Collaboration diagram for PreSpawnedThread:



Data Fields

- volatile unsigned char busy
- unsigned int threadNum
- struct AmmServer_Instance * instance
- pthread_t thread_id
- int clientsock
- struct sockaddr_in client
- unsigned int clientlen
- char webserver_root [MAX_FILE_PATH]
- char templates_root [MAX_FILE_PATH]

5.20.1 Detailed Description

A structure that holds information to be passed from the main thread to the new (prespawned) thread.

5.20.2 Field Documentation

5.20.2.1 volatile unsigned char busy

5.20.2.2 struct sockaddr_in client

5.20.2.3 unsigned int clientlen

5.20.2.4 int clientsock

5.20.2.5 struct AmmServer_Instance* instance

5.20.2.6 char templates_root[MAX_FILE_PATH]

5.20.2.7 pthread_t thread_id

5.20.2.8 unsigned int threadNum

5.20.2.9 char webserver_root[MAX_FILE_PATH]

The documentation for this struct was generated from the following file:

- src/AmmServerlib/threads/prespawnedThreads.h

5.21 time_snap Struct Reference

`#include <time_provider.h>`

Data Fields

- struct timeval starttime endtime difference

5.21.1 Field Documentation

5.21.1.1 struct timeval starttime endtime difference

The documentation for this struct was generated from the following file:

- src/AmmServerlib/tools/time_provider.h

5.22 timestamp Struct Reference

Timestamp for a cache item entry.

`#include <file_caching.h>`

Data Fields

- unsigned char hour
- unsigned char minute
- unsigned char second
- unsigned char wday
- unsigned char day
- unsigned char month
- unsigned int year

5.22.1 Detailed Description

Timestamp for a cache item entry.

5.22.2 Field Documentation

5.22.2.1 unsigned char day

5.22.2.2 unsigned char hour

5.22.2.3 unsigned char minute

5.22.2.4 unsigned char month

5.22.2.5 unsigned char second

5.22.2.6 unsigned char wday

5.22.2.7 unsigned int year

The documentation for this struct was generated from the following file:

- src/AmmServerlib/cache/file_caching.h

5.23 tokens Struct Reference

```
#include <InputParser_C.h>
```

Data Fields

- unsigned int token_start
- unsigned int length

5.23.1 Field Documentation

5.23.1.1 unsigned int length

5.23.1.2 unsigned int token_start

The documentation for this struct was generated from the following file:

- src/AmmServerlib/InputParser/InputParser_C.h

5.24 URLDB Struct Reference

Data Fields

- char * longURL
- char * shortURL
- unsigned long shortURLHash

5.24.1 Field Documentation

5.24.1.1 char* longURL

5.24.1.2 char* shortURL

5.24.1.3 unsigned long shortURLHash

The documentation for this struct was generated from the following file:

- src/Services/MyURL/main.c

Chapter 6

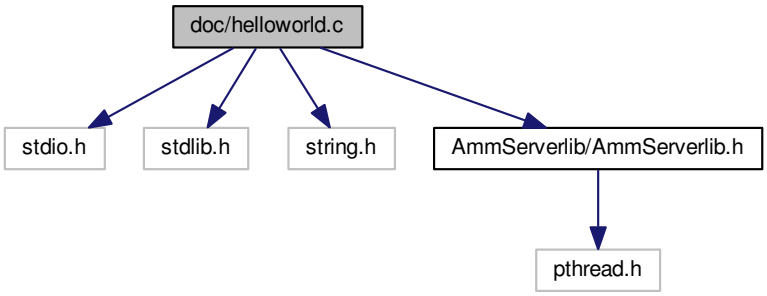
File Documentation

6.1 doc/DoxygenMainpage.h File Reference

6.2 doc/helloworld.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "AmmServerlib/AmmServerlib.h"
```

Include dependency graph for helloworld.c:



Functions

- void * prepare_helloworld_content_callback (unsigned int associated_vars)
- void init_dynamic_content ()
- void close_dynamic_content ()
- int main (int argc, char *argv[])

Variables

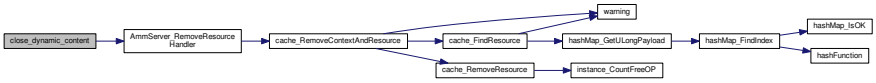
- char webserver_root [512] ="public_html/"
- char templates_root [512] ="public_html/templates/"
- struct AmmServer_RH_Context helloworld ={0}

- unsigned int helloworld_times_shown =0

6.2.1 Function Documentation

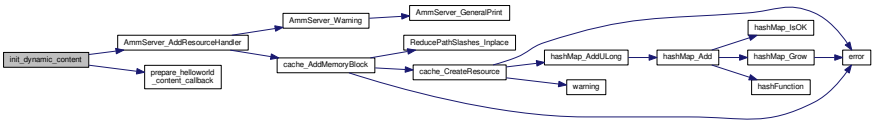
6.2.1.1 void close_dynamic_content ( )

Here is the call graph for this function:



6.2.1.2 void init_dynamic_content ( )

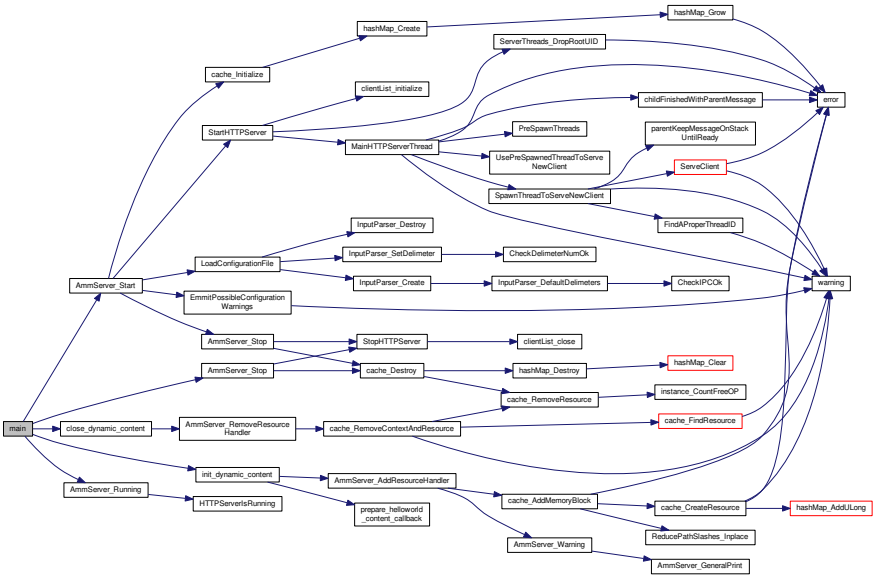
Here is the call graph for this function:



6.2.1.3 int main (int *argc,* char ∗ *argv[]*)

Dynamic content code ..! END ----------—

Here is the call graph for this function:



6.2.1.4 void* prepare_helloworld_content_callback (unsigned int *associated_vars*)

6.2.2 Variable Documentation

6.2.2.1 struct AmmServer_RH_Context helloworld ={0}

Dynamic content code ..! START!

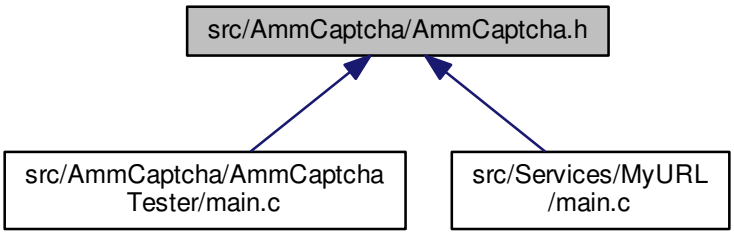
6.2.2.2 unsigned int helloworld_times_shown =0

6.2.2.3 char templates_root[512] ="public_html/templates/"

6.2.2.4 char webserver_root[512] ="public_html/"

6.3 src/AmmCaptcha/AmmCaptcha.h File Reference

This graph shows which files directly or indirectly include this file:



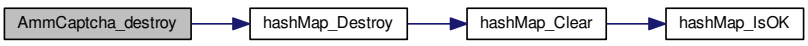
Functions

- int AmmCaptcha_initialize (char *font, char *dictFilename)
- int AmmCaptcha_destroy ()
- int AmmCaptcha_isReplyCorrect (unsigned int captchaID, char *reply)
- int AmmCaptcha_getCaptchaFrame (unsigned int captchaID, char *mem, unsigned long *mem_size)
- int testAmmCaptcha ()

6.3.1 Function Documentation

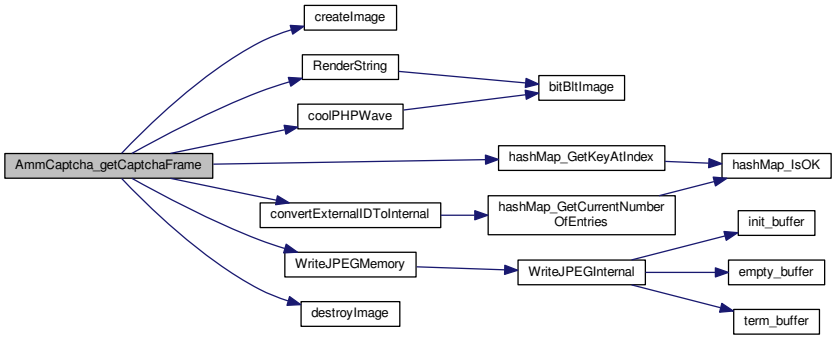
6.3.1.1 int AmmCaptcha_destroy ( )

Here is the call graph for this function:



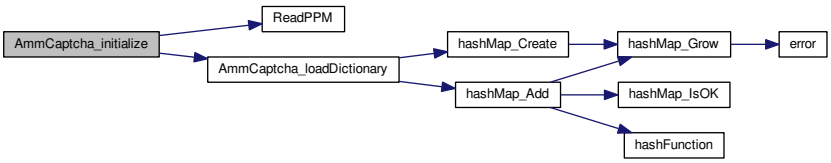
6.3.1.2 int AmmCaptcha_getCaptchaFrame (unsigned int *captchaID,* char * *mem,* unsigned long * *mem_size*)

Here is the call graph for this function:



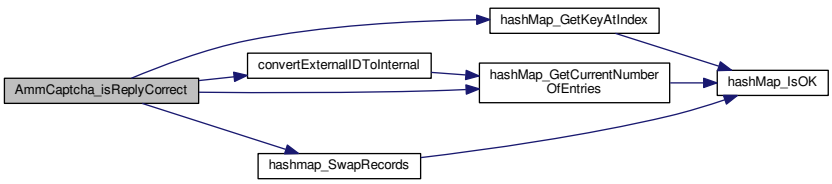
6.3.1.3 int AmmCaptcha_initialize (char * *font,* char * *dictFilename*)

Here is the call graph for this function:



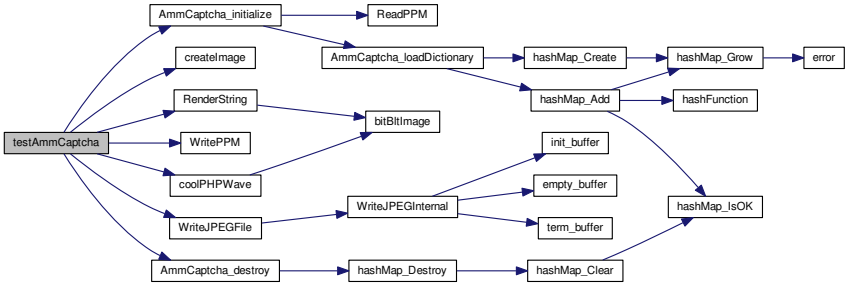
6.3.1.4 int AmmCaptcha_isReplyCorrect (unsigned int *captchaID,* char * *reply*)

Here is the call graph for this function:



6.3.1.5 int testAmmCaptcha ( )

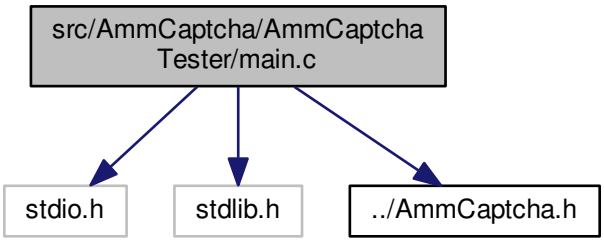
Here is the call graph for this function:



6.4 src/AmmCaptcha/AmmCaptchaTester/main.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include "../AmmCaptcha.h"
```

Include dependency graph for main.c:



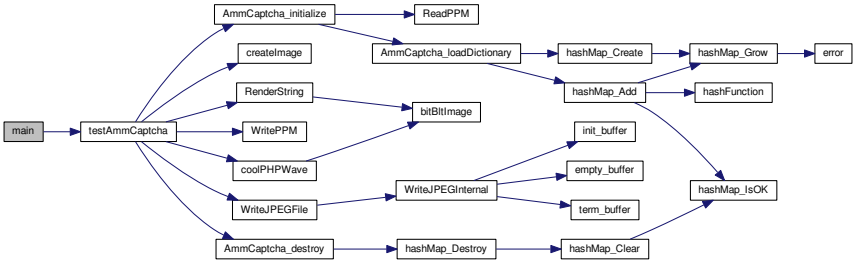
Functions

- int main ()

6.4.1 Function Documentation

6.4.1.1 int main ( )

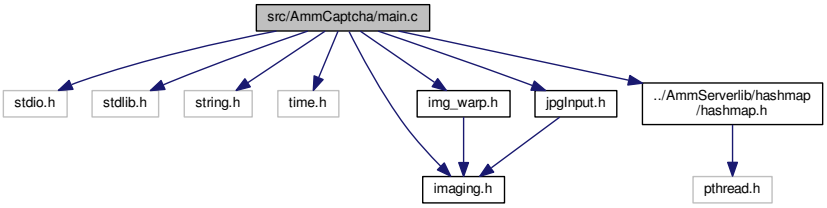
Here is the call graph for this function:



6.5 src/AmmCaptcha/main.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include "imaging.h"
#include "img_warp.h"
#include "jpgInput.h"
#include "../AmmServerlib/hashmap/hashmap.h"
```

Include dependency graph for main.c:



Macros

- #define RANDOMIZE_AFTER_FAILED_ATTEMPT 1

Functions

- int RenderString (struct Image *frame, struct Image *font, unsigned int x, unsigned int y, char *str)
- unsigned int convertExternalIDToInternal (unsigned int captchaID)
- int AmmCaptcha_isReplyCorrect (unsigned int captchaID, char *reply)
- int AmmCaptcha_getCaptchaFrame (unsigned int captchaID, char *mem, unsigned long *mem_size)
- int AmmCaptcha_copyCaptchaJPEGImageWithCopy (unsigned int captchaID, char *mem, unsigned long *mem_size)
- int AmmCaptcha_loadDictionary (char *dictFilename)
- int AmmCaptcha_initialize (char *font, char *dictFilename)

- int AmmCaptcha_destroy ()
- int testAmmCaptcha ()

Variables

- unsigned int fontX = 19
- unsigned int fontY = 22
- struct Image fontRAW ={0}
- struct hashMap * captchaStrings =0

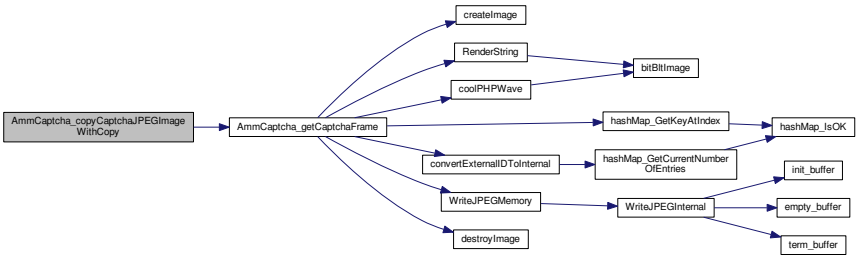
6.5.1 Macro Definition Documentation

6.5.1.1 #define RANDOMIZE_AFTER_FAILED_ATTEMPT 1

6.5.2 Function Documentation

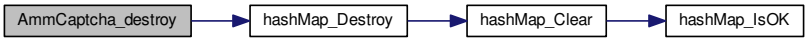
6.5.2.1 int AmmCaptcha_copyCaptchaJPEGImageWithCopy (unsigned int *captchaID,* char * *mem,* unsigned long * *mem_size*)

Here is the call graph for this function:



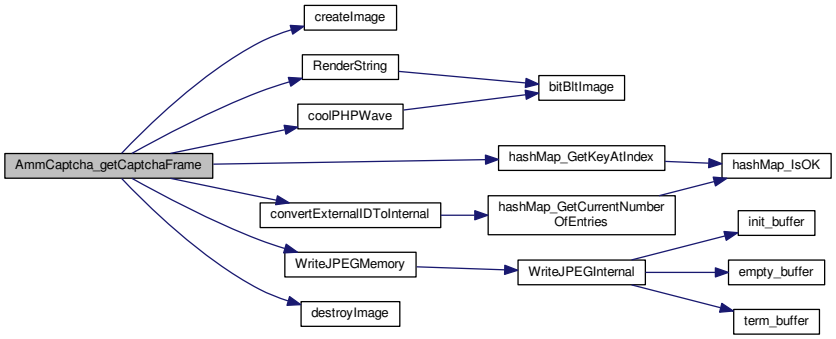
6.5.2.2 int AmmCaptcha_destroy ( )

Here is the call graph for this function:



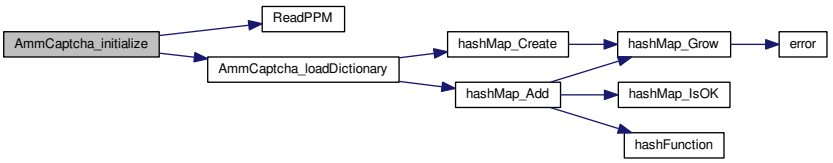
6.5.2.3 int AmmCaptcha_getCaptchaFrame (unsigned int *captchaID,* char * *mem,* unsigned long * *mem_size*)

Here is the call graph for this function:



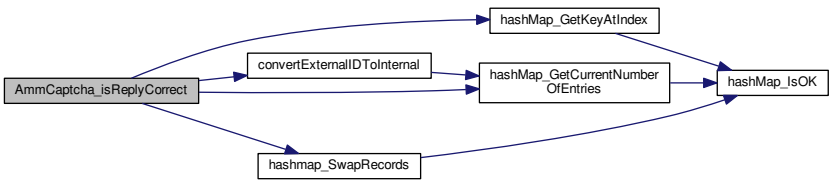
6.5.2.4 int AmmCaptcha_initialize (char * *font,* char * *dictFilename*)

Here is the call graph for this function:



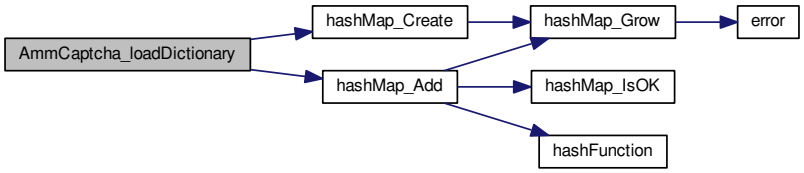
6.5.2.5 int AmmCaptcha_isReplyCorrect (unsigned int *captchaID,* char * *reply*)

Here is the call graph for this function:



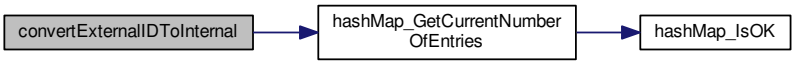
6.5.2.6 int AmmCaptcha_loadDictionary (char * *dictFilename*)

Here is the call graph for this function:



6.5.2.7 unsigned int convertExternalIDToInternal (unsigned int *captchaID*)

Here is the call graph for this function:

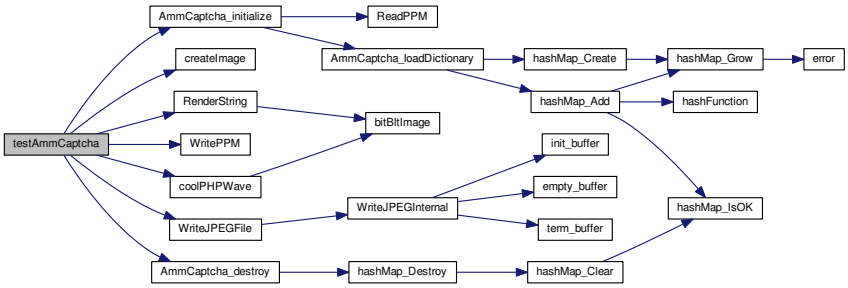


6.5.2.8 int RenderString (struct Image * *frame*, struct Image * *font*, unsigned int *x*, unsigned int *y*, char * *str*)

Here is the call graph for this function:

6.5.2.9 int testAmmCaptcha ( )

Here is the call graph for this function:



6.5.3 Variable Documentation

6.5.3.1 struct hashMap* captchaStrings =0

6.5.3.2 struct Image fontRAW ={0}

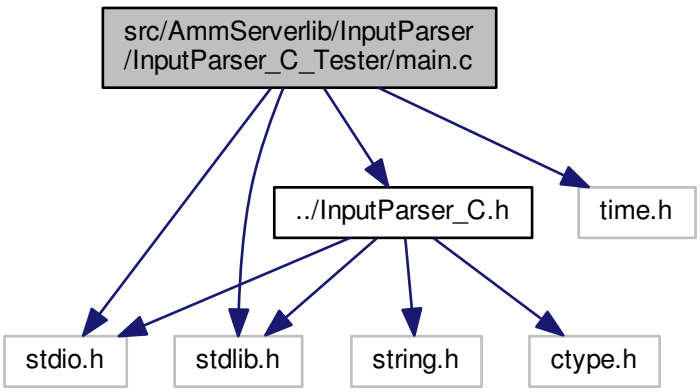
6.5.3.3 unsigned int fontX = 19

6.5.3.4 unsigned int fontY = 22

6.6 src/AmmServerlib/InputParser/InputParser_C_Tester/main.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include "../InputParser_C.h"
#include <time.h>
```

Include dependency graph for main.c:



Macros

- #define max_ret_word 256

Functions

- void ParseString (struct InputParserC *ipc, char *thestr)
- int IntermediateTests ()
- int main ()

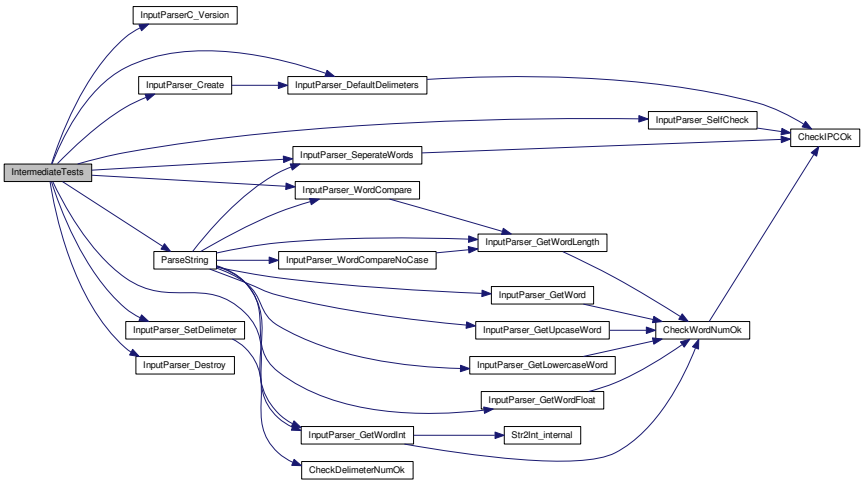
6.6.1 Macro Definition Documentation

6.6.1.1 #define max_ret_word 256

6.6.2 Function Documentation

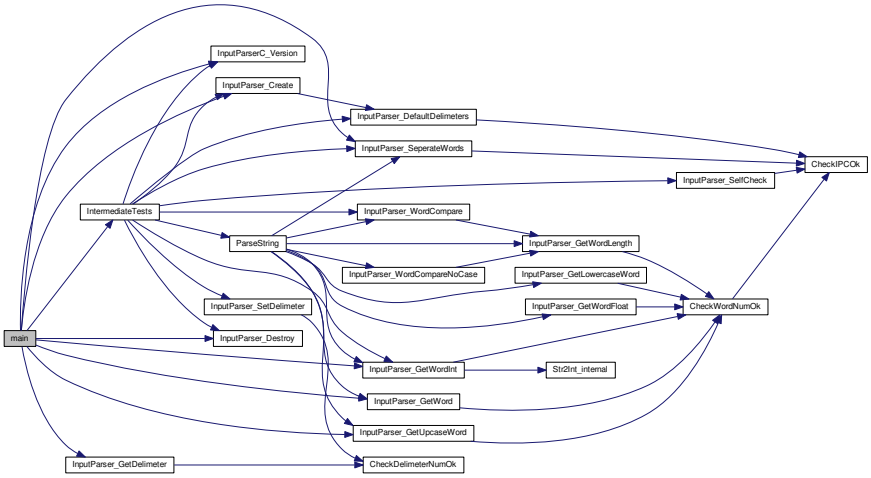
6.6.2.1 int IntermediateTests ( )

Here is the call graph for this function:



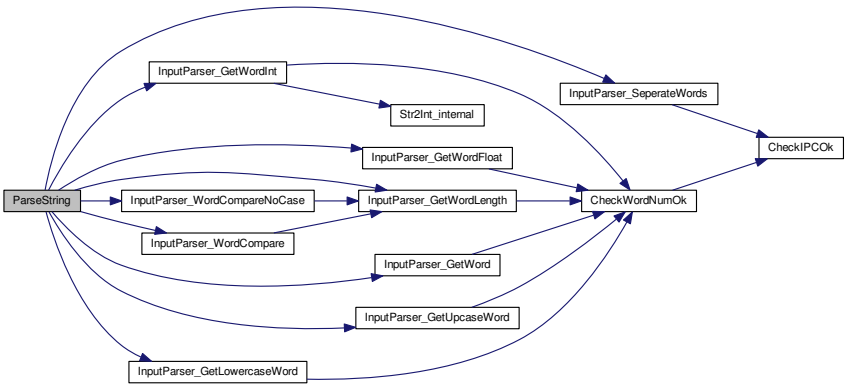
6.6.2.2 int main ( )

Here is the call graph for this function:



6.6.2.3 void ParseString (struct InputParserC * *ipc*, char * *thestr*)

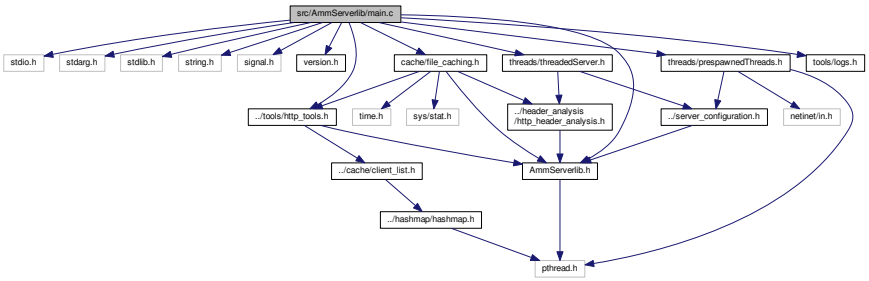
Here is the call graph for this function:



6.7 src/AmmServerlib/main.c File Reference

```
#include <stdio.h>
#include <stdarg.h>
#include <stdlib.h>
#include <string.h>
#include <signal.h>
#include "version.h"
#include "AmmServerlib.h"
#include "threads/threadedServer.h"
#include "threads/prespawnedThreads.h"
#include "cache/file_caching.h"
#include "tools/http_tools.h"
#include "tools/logs.h"
```

Include dependency graph for main.c:



Functions

- char * AmmServer_Version ()

  Returns a string with the version of AmmarServer , in case it returns NULL it means that we are linked to AmmarServerNULL which means a fake binary.
- int AmmServer_CheckIfHeaderBinaryAreTheSame (int headerSpec)

  Internal Check to compare against changes of the header files.
- void AmmServer_GeneralPrint (char *color, char *label, const char *format, va_list *arglist)
- void AmmServer_Warning (const char *format,...)

  Writes the C string pointed by format to stderr , as a warning (Yellow) and logs it to the appropriate log If format includes format specifiers (subsequences beginning with %), the additional arguments following format are formatted and inserted in the resulting string replacing their respective specifiers.
- void AmmServer_Error (const char *format,...)

  Writes the C string pointed by format to stderr , as an error (Red) and logs it to the appropriate log If format includes format specifiers (subsequences beginning with %), the additional arguments following format are formatted and inserted in the resulting string replacing their respective specifiers.
- void AmmServer_Success (const char *format,...)

  Writes the C string pointed by format to stderr , as a success (Green) and logs it to the appropriate log If format includes format specifiers (subsequences beginning with %), the additional arguments following format are formatted and inserted in the resulting string replacing their respective specifiers.
- int AmmServer_Stop (struct AmmServer_Instance *instance)

  Stop a Web Server , deallocate memory , free ports and free the server instance..
- struct AmmServer_Instance * AmmServer_Start (char *name, char *ip, unsigned int port, char *conf_file, char *web_root_path, char *templates_root_path)

  Start a Web Server , allocate memory , bind ports and return its instance..
- struct AmmServer_Instance * AmmServer_StartWithArgs (char *name, int argc, char **argv, char *ip, unsigned int port, char *conf_file, char *web_root_path, char *templates_root_path)

Start a Web Server , allocate memory , bind ports and return its instance , also process arguments (argc and argv from int main(int argc, char ∗argv[])) ..

- int AmmServer_Running (struct AmmServer_Instance ∗instance)

  Query if an instance of AmmarServer is initialized and running.

- int AmmServer_AddRequestHandler (struct AmmServer_Instance ∗instance, struct AmmServer_RequestOverride_Context ∗RequestOverrideContext, char ∗request_type, void ∗callback)

  Add a request handler to handle requests , before they get processed internally Calling this will bind a C function that will be called and produce output when someone asks for any resource using the specified method TODO : Improve this documenatation.

- int AmmServer_AddResourceHandler (struct AmmServer_Instance ∗instance, struct AmmServer_RH_Context ∗context, char ∗resource_name, char ∗web_root, unsigned int allocate_mem_bytes, unsigned int callback_every_x_msec, void ∗callback, unsigned int scenario)

  Add a request handler to handle dynamic requests , the core mechanic of AmmarServer Calling this will bind a C function that will be called and produce output when someone asks for a resource TODO : Improve this documenatation.

- int AmmServer_PreCacheFile (struct AmmServer_Instance ∗instance, char ∗filename)
- int AmmServer_DoNOTCacheResourceHandler (struct AmmServer_Instance ∗instance, struct AmmServer_RH_Context ∗context)

  Set resource handler to no-cache mode , this means whoever asks for it will never get a cached response.

- int AmmServer_DoNOTCacheResource (struct AmmServer_Instance ∗instance, char ∗resource_name)

  Set resource to no-cache mode , this means whoever asks for it will never get a cached response.

- int AmmServer_RemoveResourceHandler (struct AmmServer_Instance ∗instance, struct AmmServer_RH_Context ∗context, unsigned char free_mem)

  Remove a request handler that hanles dynamic requests.

- int AmmServer_GetInfo (struct AmmServer_Instance ∗instance, unsigned int info_type)

  Get an Integer out of the state of an instance , of course one can dive into the instance structure but this is a much more clean way to do this.

- int AmmServer_POSTArg (struct AmmServer_Instance ∗instance, struct AmmServer_DynamicRequest ∗rqst, char ∗var_id_IN, char ∗var_value_OUT, unsigned int max_var_value_OUT)

  Get a POST argument.

- int AmmServer_GETArg (struct AmmServer_Instance ∗instance, struct AmmServer_DynamicRequest ∗rqst, char ∗var_id_IN, char ∗var_value_OUT, unsigned int max_var_value_OUT)

  Get a GET argument.

- int AmmServer_FILES (struct AmmServer_Instance ∗instance, struct AmmServer_DynamicRequest ∗rqst, char ∗var_id_IN, char ∗var_value_OUT, unsigned int max_var_value_OUT)

  Access a FILE submitted by a dynamic requested.

- int _POST (struct AmmServer_Instance ∗instance, struct AmmServer_DynamicRequest ∗rqst, char ∗var_id_IN, char ∗var_value_OUT, unsigned int max_var_value_OUT)

  Shorthand/Shortcut for AmmServer_POSTArg()

- int _GET (struct AmmServer_Instance ∗instance, struct AmmServer_DynamicRequest ∗rqst, char ∗var_id_IN, char ∗var_value_OUT, unsigned int max_var_value_OUT)

  Shorthand/Shortcut for AmmServer_GETArg()

- int _FILES (struct AmmServer_Instance ∗instance, struct AmmServer_DynamicRequest ∗rqst, char ∗var_id_IN, char ∗var_value_OUT, unsigned int max_var_value_OUT)

  Shorthand/Shortcut for AmmServer_FILES()

- int AmmServer_SignalCountAsBadClientBehaviour (struct AmmServer_Instance ∗instance, struct AmmServer_DynamicRequest ∗rqst)

  Staged way to easily handle bad clients etc from the clients , currently a stub..!

- int AmmServer_SaveDynamicRequest (char ∗filename, struct AmmServer_Instance ∗instance, struct AmmServer_DynamicRequest ∗rqst)

  Save Dynamic Request to file.

- int AmmServer_GetIntSettingValue (struct AmmServer_Instance ∗instance, unsigned int set_type)

  Get an Integer out of the state of an instance , of course one can dive into the instance structure but this is a much more clean way to do this.

- int AmmServer_SetIntSettingValue (struct AmmServer_Instance *instance, unsigned int set_type, int set_value)

  Set an Integer inside the state of an instance , of course one can dive into the instance structure but this is a much more clean way to do this.
- char * AmmServer_GetStrSettingValue (struct AmmServer_Instance *instance, unsigned int set_type)

  Get a String out of the state of an instance , of course one can dive into the instance structure but this is a much more clean way to do this.
- int AmmServer_SetStrSettingValue (struct AmmServer_Instance *instance, unsigned int set_type, char *set_value)

  Set an string inside the state of an instance , of course one can dive into the instance structure but this is a much more clean way to do this.
- struct AmmServer_Instance * AmmServer_StartAdminInstance (char *ip, unsigned int port)

  Planned functionality for a default http administrator panel per server per instance , currently not implemented correctly.
- int AmmServer_SelfCheck (struct AmmServer_Instance *instance)

  Perform a sanity check on the instance of AmmarServer , this is mostly a dev debug tool and an entry point for code inside AmmServerlib.
- int AmmServer_ReplaceVarInMemoryFile (char *page, unsigned int pageLength, char *var, char *value)

  Hot-Replace a variable inside a memory block , typically used to replace placeholders inside text files , like $$$$$$$$-NAME$$$$$$$$, the value should be smaller or equal to the var beeing replaced.
- void AmmServer_GlobalTerminationHandler (int signum)
- int AmmServer_RegisterTerminationSignal (void *callback)

  Register a function to call a function that gracefully terminates a client when a SIGKILL or the time to stop the server comes.
- int AmmServer_ExecuteCommandLine (char *command, char *what2GetBack, unsigned int what2GetBackMaxSize)

  Execute a command and copy its output to the provided buffer.
- char * AmmServer_ReadFileToMemory (char *filename, unsigned int *length)

  Read a file and store it to a freshly allocated memory block.
- int AmmServer_WriteFileFromMemory (char *filename, char *memory, unsigned int memoryLength)

  Dump a memory block to a file.
- int AmmServer_DirectoryExists (char *filename)

  Check if directory Exists.
- int AmmServer_FileExists (char *filename)

  Check if file Exists.
- int AmmServer_EraseFile (char *filename)

  Erase a File.
- unsigned int AmmServer_StringIsHTMLSafe (char *str)

  Check if a string has html elements inside it , so if we append it to a web site we won't have html injected.

Variables

- void(* TerminationCallback)()=0

6.7.1 Function Documentation

6.7.1.1 int _FILES (struct AmmServer_Instance * *instance,* struct AmmServer_DynamicRequest * *rqst,* char * *var_id_IN,* char * *var_value_OUT,* unsigned int *max_var_value_OUT*)

Shorthand/Shortcut for AmmServer_FILES()

Here is the call graph for this function:

6.7.1.2 int _GET (struct AmmServer_Instance * *instance*, struct AmmServer_DynamicRequest * *rqst*, char * *var_id_IN*, char * *var_value_OUT*, unsigned int *max_var_value_OUT*)

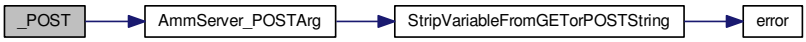
Shorthand/Shortcut for AmmServer_GETArg()

Here is the call graph for this function:

6.7.1.3 int _POST (struct AmmServer_Instance * *instance*, struct AmmServer_DynamicRequest * *rqst*, char * *var_id_IN*, char * *var_value_OUT*, unsigned int *max_var_value_OUT*)

Shorthand/Shortcut for AmmServer_POSTArg()

Here is the call graph for this function:



6.7.1.4 int AmmServer_AddRequestHandler (struct AmmServer_Instance * *instance*, struct AmmServer_RequestOverride_Context * *RequestOverrideContext*, char * *request_type*, void * *callback*)

Add a request handler to handle requests , before they get processed internally Calling this will bind a C function that will be called and produce output when someone asks for any resource using the specified method TODO : Improve this documenatation.

Parameters

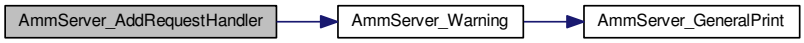
An	AmmarServer Instance

A	AmmServer_RequestOverride_Context to be populated
Request	Type
Pointer	to function callback

Return values

1=Success,0=Fail	

Here is the call graph for this function:



6.7.1.5 int AmmServer_AddResourceHandler (struct AmmServer_Instance * *instance*, struct AmmServer_RH_Context * *context*, char * *resource_name*, char * *web_root*, unsigned int *allocate_mem_bytes*, unsigned int *callback_every_x_msec*, void * *callback*, unsigned int *scenario*)

Add a request handler to handle dynamic requests , the core mechanic of AmmarServer Calling this will bind a C function that will be called and produce output when someone asks for a resource TODO : Improve this documenatation.

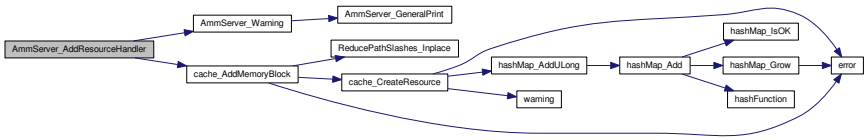
Parameters

An	AmmarServer Instance
An	AmmServer_RH_Context to be populated
Name	of resource that should get dynamic responses (i.e. "index.html")
Root	Path for the specific resource
Memory	chunk to allocate for responses , (this is the max response size)
Minimum	time between two calls of the function (0 = no minimum time)
Function	to be called and provides output when someone asks for resource
Scenario/Profile	of this resource (see RHScenarios)

Return values

1=Success,0=Fail	

Here is the call graph for this function:



6.7.1.6 int AmmServer_CheckIfHeaderBinaryAreTheSame (int *headerSpec*)

Internal Check to compare against changes of the header files.

Parameters

Header	(should be AMMAR_SERVER_HTTP_HEADER_SPEC)

Return values

1=Success,0=Failure	

6.7.1.7 int AmmServer_DirectoryExists (char * *filename*)

Check if directory Exists.

Parameters

Path	to directory

Return values

1=Exists,0=Does	not Exist

Here is the call graph for this function:

6.7.1.8 int AmmServer_DoNOTCacheResource (struct AmmServer_Instance * *instance,* char * *resource_name*)

Set resource to no-cache mode , this means whoever asks for it will never get a cached response.

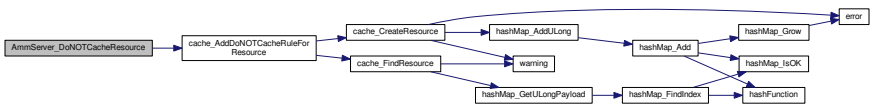
Parameters

Instance	of an AmmarServer
Resource	name that we want to always serve fresh

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.7.1.9 int AmmServer_DoNOTCacheResourceHandler (struct AmmServer_Instance * *instance,* struct AmmServer_RH_Context * *context*)

Set resource handler to no-cache mode , this means whoever asks for it will never get a cached response.

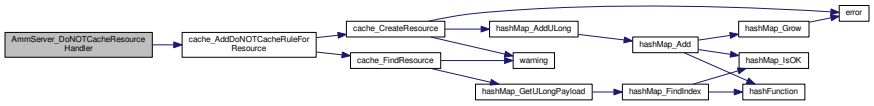
Parameters

Instance	of an AmmarServer
Resource	context that should always be served fresh (AmmServer_RH_Context)

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.7.1.10 int AmmServer_EraseFile (char * *filename*)

Erase a File.

Parameters

Path	to file

Return values

1=Success,0=Failure	

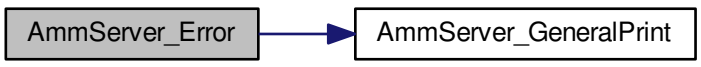
6.7.1.11 void AmmServer_Error (const char * *format,* ...)

Writes the C string pointed by format to stderr , as an error (Red) and logs it to the appropriate log If format includes format specifiers (subsequences beginning with %), the additional arguments following format are formatted and inserted in the resulting string replacing their respective specifiers.

Parameters

format,see	printf (http://www.cplusplus.com/reference/cstdio/printf/)
Arbitrary	number of other parameters that where defined in format

Here is the call graph for this function:



6.7.1.12 int AmmServer_ExecuteCommandLine (char * *command,* char * *what2GetBack,* unsigned int *what2GetBackMaxSize*)

Execute a command and copy its output to the provided buffer.

Parameters

Command	to execute
Allocated	memory to store the result
Size	of Allocated memory

Return values

1=Ok,0=Failed	

Bug Executing commands can be dangerous , always check and sanitize input before executing , Also be sure about the max size of output so that you don't lose a part of it , also make something like escapeshellcmd

6.7.1.13 int AmmServer_FileExists (char * *filename*)

Check if file Exists.

Parameters

Path	to file

Return values

1=Exists,0=Does	not Exist

Here is the call graph for this function:

6.7.1.14 int AmmServer_FILES (struct AmmServer_Instance * *instance,* struct AmmServer_DynamicRequest * *rqst,* char * *var_id_IN,* char * *var_value_OUT,* unsigned int *max_var_value_OUT*)

Access a FILE submitted by a dynamic requested.

Parameters

Instance	of an AmmarServer
Request	that contains the POST argument (see AmmServer_DynamicRequest)
Input	Name of argument we are looking for
Output	Pointer that will be copied with the value we were looking for
Maximum	Size for output Value

Return values

1=Success,0=Failure	

6.7.1.15 void AmmServer_GeneralPrint (char * *color,* char * *label,* const char * *format,* va_list * *arglist*)

6.7.1.16 int AmmServer_GETArg (struct AmmServer_Instance * *instance,* struct AmmServer_DynamicRequest * *rqst,* char * *var_id_IN,* char * *var_value_OUT,* unsigned int *max_var_value_OUT*)

Get a GET argument.

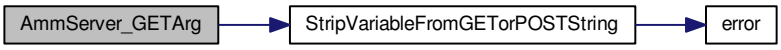
Parameters

Instance	of an AmmarServer
Request	that contains the POST argument (see AmmServer_DynamicRequest)
Input	Name of argument we are looking for
Output	Pointer that will be copied with the value we were looking for
Maximum	Size for output Value

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.7.1.17 int AmmServer_GetInfo (struct AmmServer_Instance * *instance*, unsigned int *info_type*)

Get an Integer out of the state of an instance , of course one can dive into the instance structure but this is a much more clean way to do this.

Parameters

An	AmmarServer Instance
An	ID about which info we want , see (AmmServInfos)

Return values

Value	of the integer we asked about

6.7.1.18 int AmmServer_GetIntSettingValue (struct AmmServer_Instance * *instance*, unsigned int *set_type*)

Get an Integer out of the state of an instance , of course one can dive into the instance structure but this is a much more clean way to do this.

Parameters

An	AmmarServer Instance
An	ID about which integer info we want , see (AmmServSettings)

Return values

Value	of the integer we asked about

6.7.1.19 char* AmmServer_GetStrSettingValue (struct AmmServer_Instance * *instance*, unsigned int *set_type*)

Get a String out of the state of an instance , of course one can dive into the instance structure but this is a much more clean way to do this.

Parameters

An	AmmarServer Instance
An	ID about which string info we want , see (AmmServStrSettings)

Return values

Value	of the string we asked about

6.7.1.20 void AmmServer_GlobalTerminationHandler (int *signum*)

6.7.1.21 int AmmServer_POSTArg (struct **AmmServer_Instance** * *instance*, struct **AmmServer_DynamicRequest** * *rqst*, char * *var_id_IN*, char * *var_value_OUT*, unsigned int *max_var_value_OUT*)

Get a POST argument.

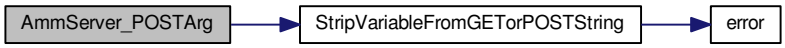
Parameters

Instance	of an AmmarServer
Request	that contains the POST argument (see AmmServer_DynamicRequest)
Input	Name of argument we are looking for
Output	Pointer that will be copied with the value we were looking for
Maximum	Size for output Value

Return values

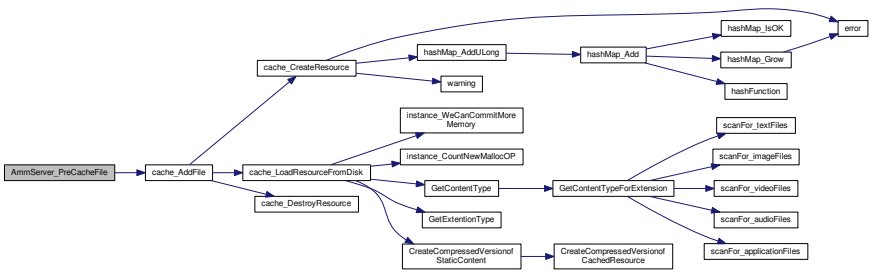
1=Success,0=Failure	

Here is the call graph for this function:



6.7.1.22 int AmmServer_PreCacheFile (struct **AmmServer_Instance** * *instance*, char * *filename*)

Here is the call graph for this function:



6.7.1.23 char* AmmServer_ReadFileToMemory (char * *filename,* unsigned int * *length*)

Read a file and store it to a freshly allocated memory block.

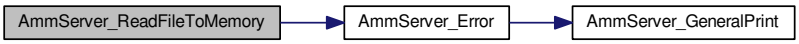
Parameters

Input	Filename
Output	Maximum Size

Return values

Pointer	to the new memory or 0=Failed

Here is the call graph for this function:



6.7.1.24 int AmmServer_RegisterTerminationSignal (void * *callback*)

Register a function to call a function that gracefully terminates a client when a SIGKILL or the time to stop the server comes.

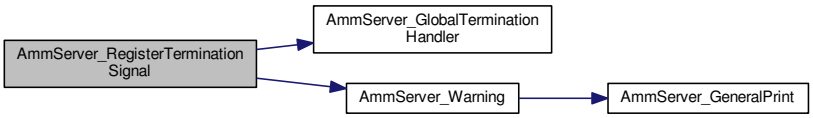
Parameters

Pointer	to function

Return values

1=Exists,0=Does	not Exist

Here is the call graph for this function:



6.7.1.25 int AmmServer_RemoveResourceHandler (struct **AmmServer_Instance** * *instance*, struct **AmmServer_RH_Context** * *context*, unsigned char *free_mem*)

Remove a request handler that hanles dynamic requests.

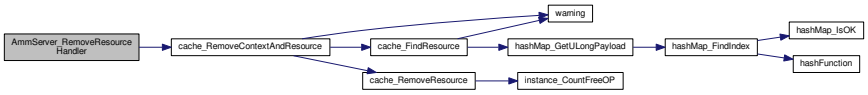
Parameters

An	AmmarServer Instance
An	AmmServer_RH_Context to be freed
Switch	to control freeing memory or not for this context (typically should be set to 1 except one knows what he is trying to do)

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.7.1.26 int AmmServer_ReplaceVarInMemoryFile (char * *page*, unsigned int *pageLength*, char * *var*, char * *value*)

Hot-Replace a variable inside a memory block , typically used to replace placeholders inside text files , like $$$$$$$$NAME$$$$$$$$, the value should be smaller or equal to the var beeing replaced.

Parameters

Pointer	to memory that contains the document
Size	of document
Variable	to be replaced
What	to replace it with

Return values

1=Ok,0=Failed	

Bug Value should not be bigger than variable otherwise things won't fit in the same memory block

6.7.1.27 int AmmServer_Running (struct **AmmServer_Instance** * *instance*)

Query if an instance of AmmarServer is initialized and running.

Parameters

An	AmmarServer Instance

Return values

1=Running,0=Stopped	

Here is the call graph for this function:

6.7.1.28 int AmmServer_SaveDynamicRequest (char * *filename,* struct **AmmServer_Instance** * *instance,* struct **AmmServer_DynamicRequest** * *rqst*)

Save Dynamic Request to file.

Parameters

Filename	to save the dynamic request
Instance	of an AmmarServer
Request	that we want to save to a file (see AmmServer_DynamicRequest)

Return values

1=Success,0=Failure	

Here is the call graph for this function:

6.7.1.29 int AmmServer_SelfCheck (struct AmmServer_Instance * *instance*)

Perform a sanity check on the instance of AmmarServer , this is mostly a dev debug tool and an entry point for code inside AmmServerlib.

Parameters

Ammar	Server Instance

Return values

1=Ok,0=Failed	

Bug Maybe remove AmmServer_SelfCheck

6.7.1.30 int AmmServer_SetIntSettingValue (struct AmmServer_Instance * *instance,* unsigned int *set_type,* int *set_value*)

Set an Integer inside the state of an instance , of course one can dive into the instance structure but this is a much more clean way to do this.

Parameters

An	AmmarServer Instance
An	ID about which integer info we want , see (AmmServSettings)
New	value to set

Return values

Value	of the integer we asked about

6.7.1.31 int AmmServer_SetStrSettingValue (struct AmmServer_Instance * *instance,* unsigned int *set_type,* char * *set_value*)

Set an string inside the state of an instance , of course one can dive into the instance structure but this is a much more clean way to do this.

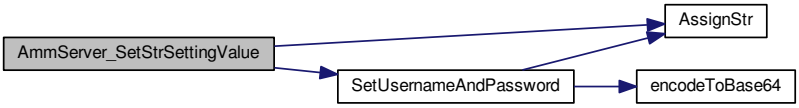
Parameters

An	AmmarServer Instance
An	ID about which integer info we want , see (AmmServStrSettings)
New	string value to set

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.7.1.32 int AmmServer_SignalCountAsBadClientBehaviour (struct AmmServer_Instance * *instance,* struct AmmServer_DynamicRequest * *rqst*)

Staged way to easily handle bad clients etc from the clients , currently a stub..!

Bug Client behaviours etc are not implemented yet

6.7.1.33 struct AmmServer_Instance* AmmServer_Start (char * *name,* char * *ip,* unsigned int *port,* char * *conf_file,* char * *web_root_path,* char * *templates_root_path*)

Start a Web Server , allocate memory , bind ports and return its instance..

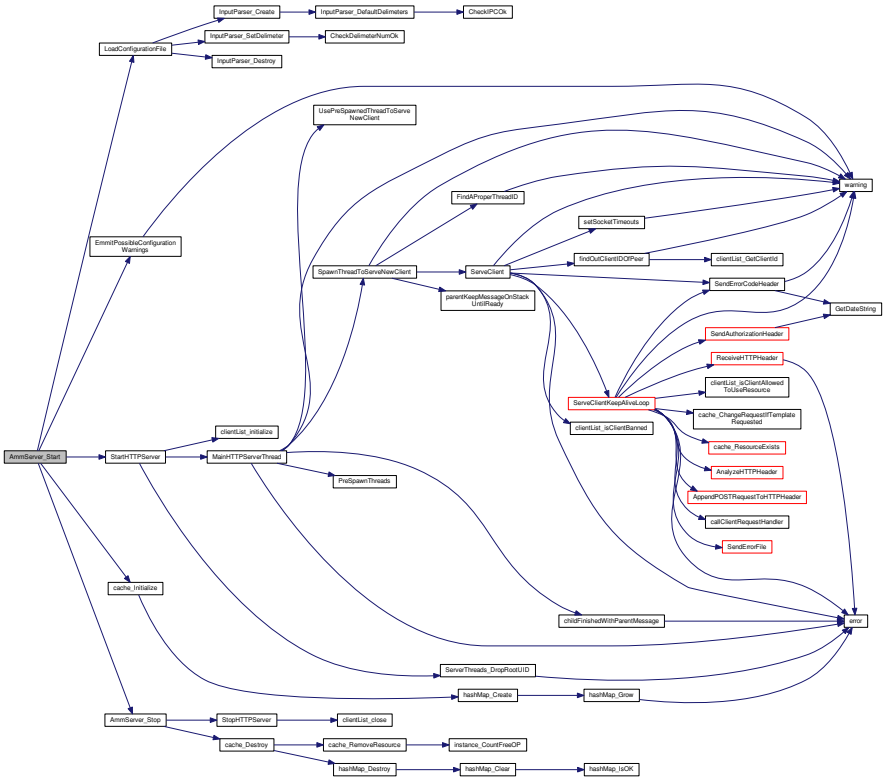
Parameters

String	containing the name of this Server
String	containing the IP to be binded (0.0.0.0 , for all interfaces)
Port	to use , ports under 1000 require superuser privileges
String	with the filename of a configuration file
String	with the root public_html directory , all directories that are childs of this dir could be visible
String	with the root directory for templates (custom 404 pages etc)

Return values

An	Ammar Server instance or 0=Failure

Here is the call graph for this function:



6.7.1.34 struct AmmServer_Instance* AmmServer_StartAdminInstance (char * *ip*, unsigned int *port*)

Planned functionality for a default http administrator panel per server per instance , currently not implemented correctly.

Parameters

IP	to bind the interface at
Port	to use

Return values

Value	of the integer we asked about

6.7.1.35 struct AmmServer_Instance* AmmServer_StartWithArgs (char * *name*, int *argc*, char ** *argv*, char * *ip*, unsigned int *port*, char * *conf_file*, char * *web_root_path*, char * *templates_root_path*)

Start a Web Server , allocate memory , bind ports and return its instance , also process arguments (argc and argv from int main(int argc, char *argv[])) ..

Parameters

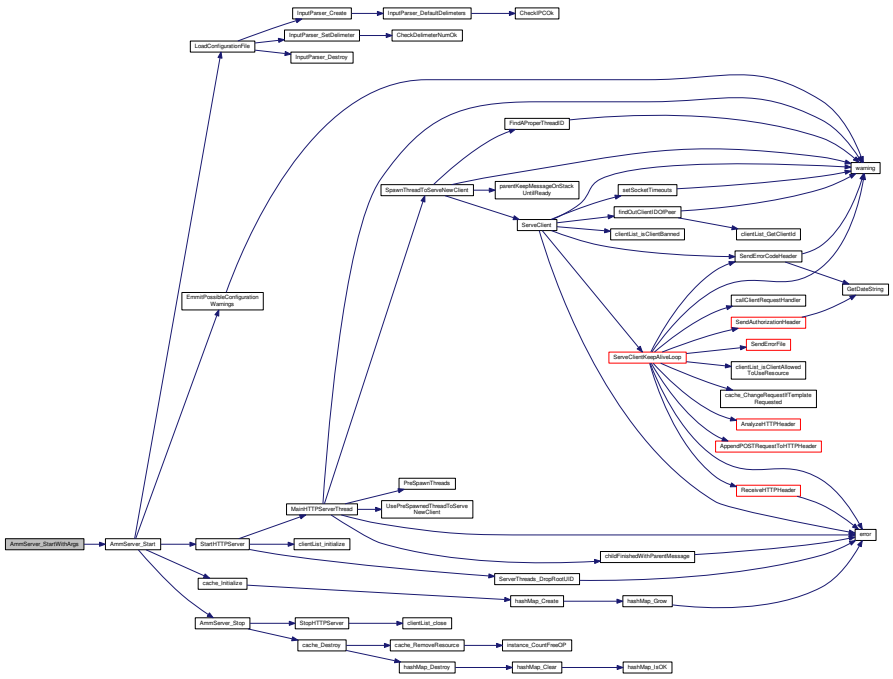
String	containing the name of this Server

argc,number	of arguments
argv,array	of strings
String	containing the IP to be binded (0.0.0.0 , for all interfaces)
Port	to use , ports under 1000 require superuser privileges
String	with the filename of a configuration file
String	with the root public_html directory , all directories that are childs of this dir could be visible
String	with the root directory for templates (custom 404 pages etc)

Return values

An	Ammar Server instance or 0=Failure

Here is the call graph for this function:



6.7.1.36 int AmmServer_Stop (struct AmmServer_Instance * *instance*)

Stop a Web Server , deallocate memory , free ports and free the server instance..

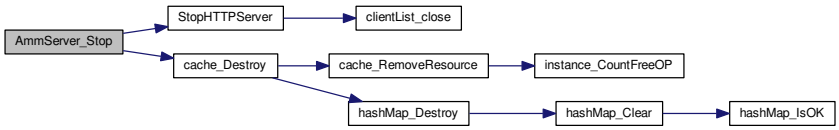
Parameters

An	AmmarServer Instance

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.7.1.37 unsigned int AmmServer_StringIsHTMLSafe (char * *str*)

Check if a string has html elements inside it , so if we append it to a web site we won't have html injected.

Parameters

Input	String

Return values

1=Safe,0=Unsafe	

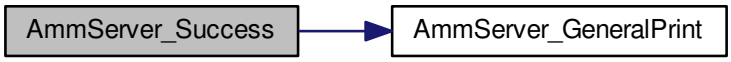
6.7.1.38 void AmmServer_Success (const char * *format,* ...)

Writes the C string pointed by format to stderr , as a success (Green) and logs it to the appropriate log If format includes format specifiers (subsequences beginning with %), the additional arguments following format are formatted and inserted in the resulting string replacing their respective specifiers.

Parameters

format,see	printf (http://www.cplusplus.com/reference/cstdio/printf/)
Arbitrary	number of other parameters that where defined in format

Here is the call graph for this function:



6.7.1.39 char* AmmServer_Version ( )

Returns a string with the version of AmmarServer , in case it returns NULL it means that we are linked to AmmarServerNULL which means a fake binary.

6.7.1.40 void AmmServer_Warning (const char * *format,* ...)

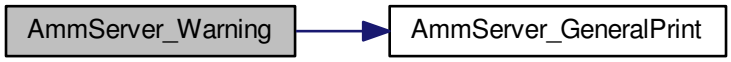
Writes the C string pointed by format to stderr , as a warning (Yellow) and logs it to the appropriate log If format includes format specifiers (subsequences beginning with %), the additional arguments following format are formatted

and inserted in the resulting string replacing their respective specifiers.

Parameters

format,see	printf (http://www.cplusplus.com/reference/cstdio/printf/)
Arbitrary	number of other parameters that where defined in format

Here is the call graph for this function:



6.7.1.41 int AmmServer_WriteFileFromMemory (char * *filename*, char * *memory*, unsigned int *memoryLength*)

Dump a memory block to a file.

Parameters

Output	Filename
Input	Pointer to memory
Size	of memory block

Return values

1=Ok,0=Failed	

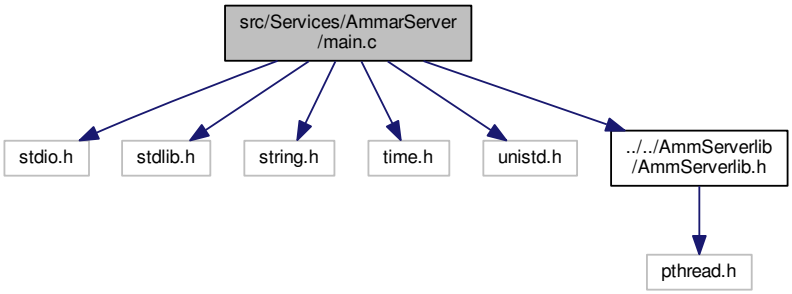
6.7.2 Variable Documentation

6.7.2.1 void(* TerminationCallback)()=0

6.8 src/Services/AmmarServer/main.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <unistd.h>
#include "../../AmmServerlib/AmmServerlib.h"
```

Include dependency graph for main.c:



Macros

- #define MAX_BINDING_PORT 65534
- #define ENABLE_PASSWORD_PROTECTION 0
- #define ENABLE_CHAT_BOX 0
- #define DEFAULT_BINDING_PORT 8080
- #define ADMIN_BINDING_PORT 8082
- #define ENABLE_ADMIN_PAGE 0
- #define MAX_SCRIPT_RESPONSE_SIZE 40960

Functions

- void * prepare_chatbox_content_callback (struct AmmServer_DynamicRequest *rqst)
- void * prepare_stats_content_callback (struct AmmServer_DynamicRequest *rqst)
- void * prepare_random_content_callback (struct AmmServer_DynamicRequest *rqst)
- void * prepare_form_content_callback (struct AmmServer_DynamicRequest *rqst)
- void * prepare_gps_content_callback (struct AmmServer_DynamicRequest *rqst)
- void * executeScriptFunction (struct AmmServer_DynamicRequest *rqst)
- void * request_override_callback (char *content)
- void init_dynamic_content ()
- void close_dynamic_content ()
- int main (int argc, char *argv[])

Variables

- char admin_root [MAX_FILE_PATH] ="admin_html/"
- char webserver_root [MAX_FILE_PATH] ="public_html/"
- char templates_root [MAX_FILE_PATH] ="public_html/templates/"
- char * executeScript =0
- struct AmmServer_Instance * default_server =0
- struct AmmServer_Instance * admin_server =0
- struct AmmServer_RequestOverride_Context GET_override ={{0}}
- struct AmmServer_RH_Context stats ={0}
- struct AmmServer_RH_Context form ={0}
- struct AmmServer_RH_Context chatbox ={0}
- struct AmmServer_RH_Context gps ={0}
- struct AmmServer_RH_Context random_chars ={0}
- struct AmmServer_RH_Context executeScriptRC ={0}

6.8.1 Macro Definition Documentation

6.8.1.1 #define ADMIN_BINDING_PORT 8082

6.8.1.2 #define DEFAULT_BINDING_PORT 8080

6.8.1.3 #define ENABLE_ADMIN_PAGE 0

6.8.1.4 #define ENABLE_CHAT_BOX 0

6.8.1.5 #define ENABLE_PASSWORD_PROTECTION 0

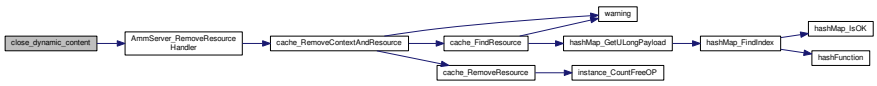
6.8.1.6 #define MAX_BINDING_PORT 65534

6.8.1.7 #define MAX_SCRIPT_RESPONSE_SIZE 40960

6.8.2 Function Documentation

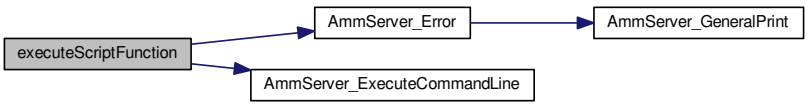
6.8.2.1 void close_dynamic_content ( )

Here is the call graph for this function:



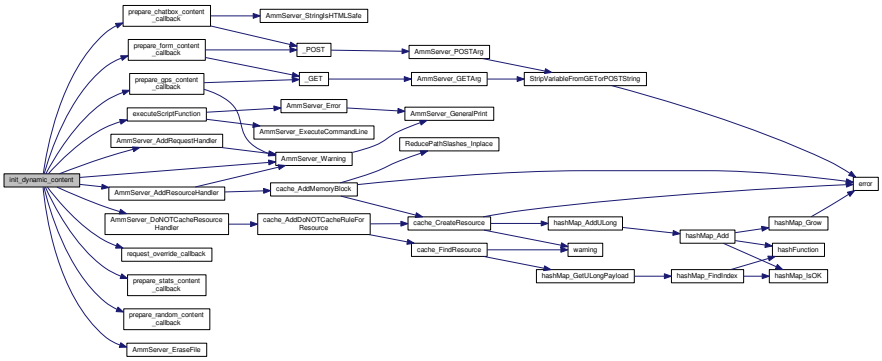
6.8.2.2 void* executeScriptFunction (struct **AmmServer_DynamicRequest** * *rqst*)

Here is the call graph for this function:



6.8.2.3 void init_dynamic_content ( )

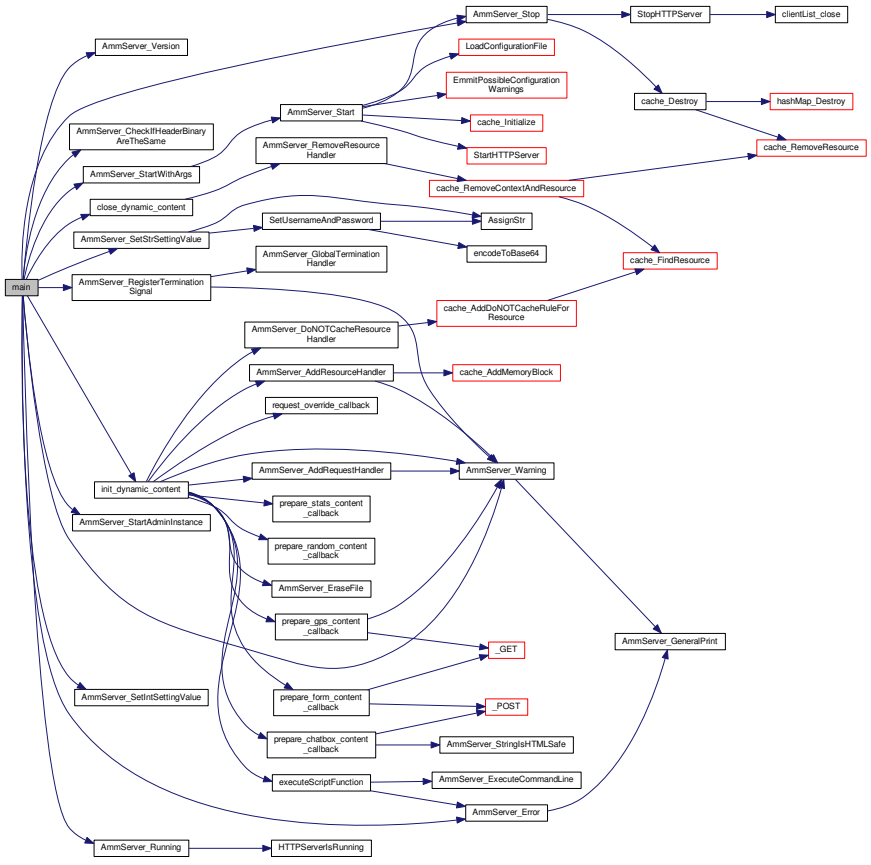
Here is the call graph for this function:



6.8.2.4 int main (int *argc*, char ∗ *argv[]*)

Dynamic content code ..! END ----------—

Here is the call graph for this function:

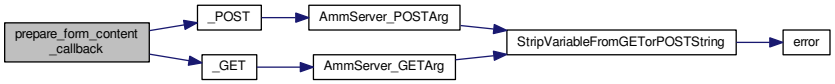


6.8.2.5 void* prepare_chatbox_content_callback (struct AmmServer_DynamicRequest * *rqst*)

Here is the call graph for this function:

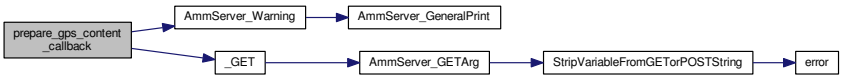
6.8.2.6 void* prepare_form_content_callback (struct AmmServer_DynamicRequest * *rqst*)

Here is the call graph for this function:



6.8.2.7 void* prepare_gps_content_callback (struct AmmServer_DynamicRequest * *rqst*)

Here is the call graph for this function:



6.8.2.8 void* prepare_random_content_callback (struct AmmServer_DynamicRequest * *rqst*)

6.8.2.9 void* prepare_stats_content_callback (struct AmmServer_DynamicRequest * *rqst*)

6.8.2.10 void* request_override_callback (char * *content*)

6.8.3 Variable Documentation

6.8.3.1 char admin_root[MAX_FILE_PATH] ="admin_html/"

6.8.3.2 struct AmmServer_Instance* admin_server =0

6.8.3.3 struct AmmServer_RH_Context chatbox ={0}

6.8.3.4 struct AmmServer_Instance* default_server =0

Dynamic content code ..! START!

6.8.3.5 char* executeScript =0

6.8.3.6 struct AmmServer_RH_Context executeScriptRC ={0}

6.8.3.7 struct AmmServer_RH_Context form ={0}

6.8.3.8 struct AmmServer_RequestOverride_Context GET_override ={{0}}

6.8.3.9 struct AmmServer_RH_Context gps ={0}

6.8.3.10 struct AmmServer_RH_Context random_chars ={0}

6.8.3.11 struct AmmServer_RH_Context stats ={0}

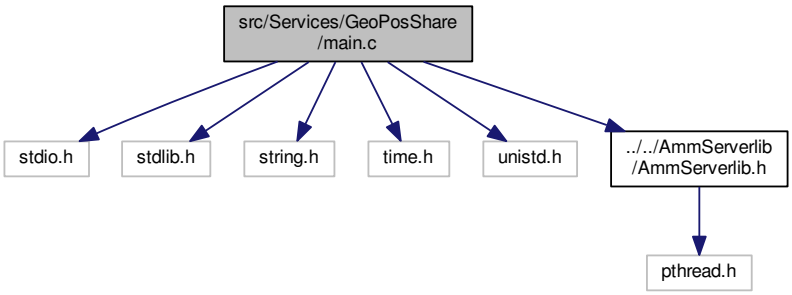
6.8.3.12 char templates_root[MAX_FILE_PATH] ="public_html/templates/"

6.8.3.13 char webserver_root[MAX_FILE_PATH] ="public_html/"

6.9 src/Services/GeoPosShare/main.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <unistd.h>
#include "../../AmmServerlib/AmmServerlib.h"
```

Include dependency graph for main.c:



Macros

- #define MAX_BINDING_PORT 65534
- #define DEFAULT_BINDING_PORT 8081
- #define ADMIN_BINDING_PORT 8082
- #define ENABLE_ADMIN_PAGE 0

Functions

- void * prepare_gps_content_callback (struct AmmServer_DynamicRequest *rqst)
- void * request_override_callback (char *content)

- void init_dynamic_content ()
- void close_dynamic_content ()
- int main (int argc, char ∗argv[])

Variables

- char admin_root [MAX_FILE_PATH] ="admin_html/"
- char webserver_root [MAX_FILE_PATH] ="public_html/"
- char templates_root [MAX_FILE_PATH] ="public_html/templates/"
- struct AmmServer_Instance ∗ default_server =0
- struct AmmServer_RequestOverride_Context GET_override ={{0}}
- struct AmmServer_RH_Context gps ={0}

6.9.1 Macro Definition Documentation

6.9.1.1 #define ADMIN_BINDING_PORT 8082

6.9.1.2 #define DEFAULT_BINDING_PORT 8081

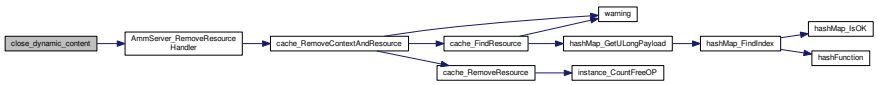
6.9.1.3 #define ENABLE_ADMIN_PAGE 0

6.9.1.4 #define MAX_BINDING_PORT 65534

6.9.2 Function Documentation

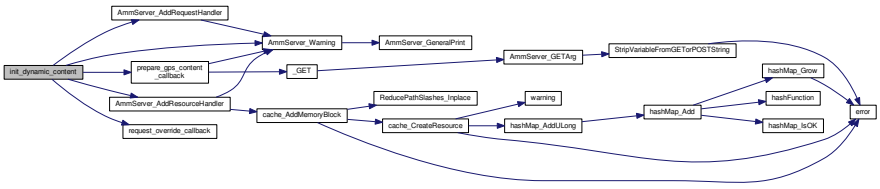
6.9.2.1 void close_dynamic_content ( )

Here is the call graph for this function:



6.9.2.2 void init_dynamic_content ( )

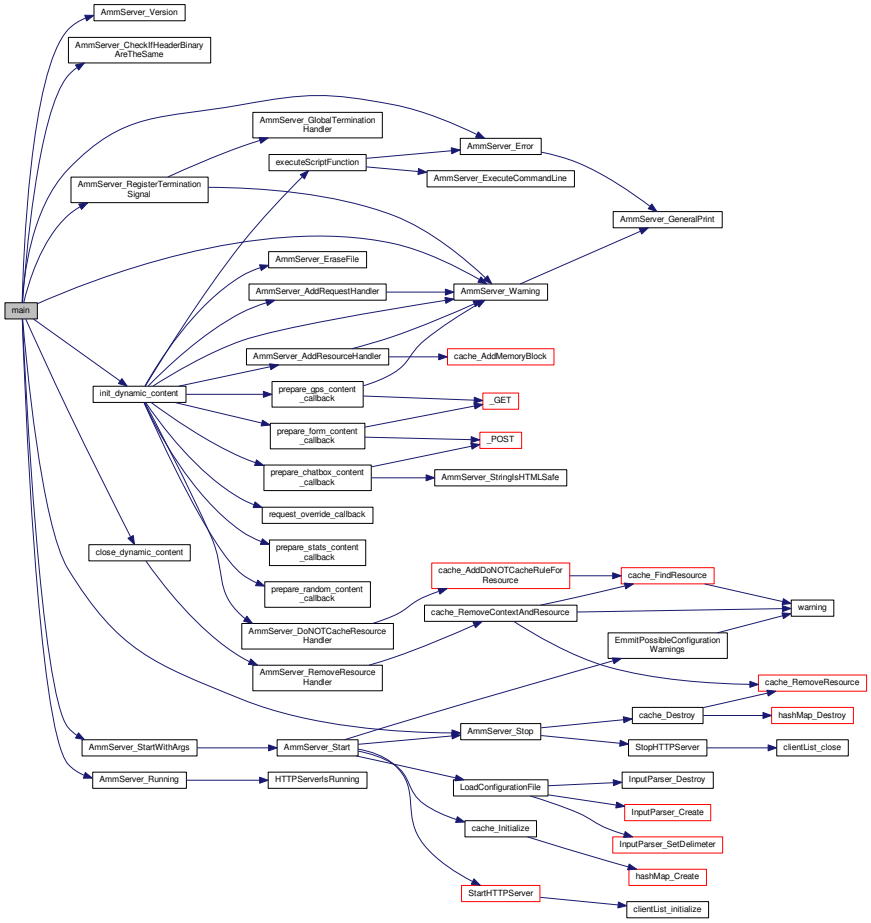
Here is the call graph for this function:



6.9.2.3 int main (int *argc,* char * *argv[]*)

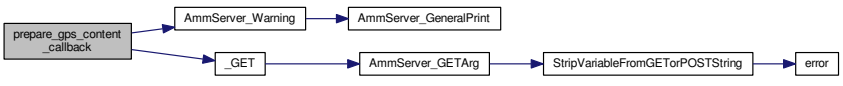
Dynamic content code ..! END ----------—

Here is the call graph for this function:



6.9.2.4 void* prepare_gps_content_callback (struct AmmServer_DynamicRequest * *rqst*)

Here is the call graph for this function:



6.9.2.5 void* request_override_callback (char * *content*)

6.9.3 Variable Documentation

6.9.3.1 char admin_root[MAX_FILE_PATH] ="admin_html/"

6.9.3.2 struct AmmServer_Instance* default_server =0

6.9.3.3 struct AmmServer_RequestOverride_Context GET_override ={{0}}

6.9.3.4 struct AmmServer_RH_Context gps ={0}

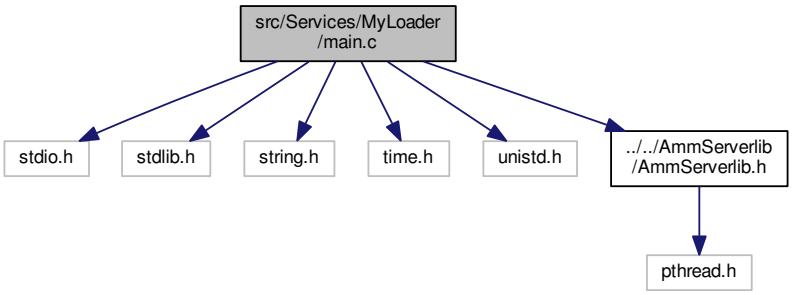
6.9.3.5 char templates_root[MAX_FILE_PATH] ="public_html/templates/"

6.9.3.6 char webserver_root[MAX_FILE_PATH] ="public_html/"

6.10 src/Services/MyLoader/main.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <unistd.h>
#include "../../AmmServerlib/AmmServerlib.h"
```

Include dependency graph for main.c:



Macros

- #define DEFAULT_BINDING_PORT 8081

Functions

- void * prepare_stats_content_callback (struct AmmServer_DynamicRequest *rqst)
- void request_override_callback (void *request)
- void * processUploadCallback (struct AmmServer_DynamicRequest *rqst)
- void init_dynamic_content ()
- void close_dynamic_content ()
- int main (int argc, char *argv[])

Variables

- char webserver_root [MAX_FILE_PATH] ="src/MyLoader/htmlData/"
- char templates_root [MAX_FILE_PATH] ="public_html/templates/"
- struct AmmServer_Instance * default_server =0

- struct AmmServer_RequestOverride_Context GET_override ={{0}}
- struct AmmServer_RH_Context uploadProcessor ={0}
- struct AmmServer_RH_Context stats ={0}

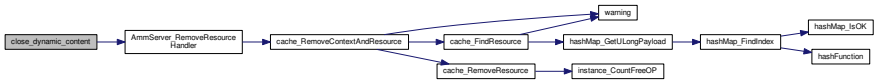
6.10.1 Macro Definition Documentation

6.10.1.1 #define DEFAULT_BINDING_PORT 8081

6.10.2 Function Documentation

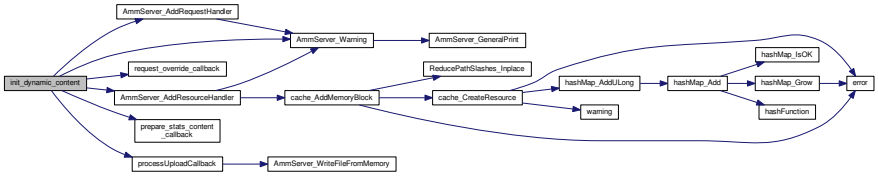
6.10.2.1 void close_dynamic_content ( )

Here is the call graph for this function:



6.10.2.2 void init_dynamic_content ( )

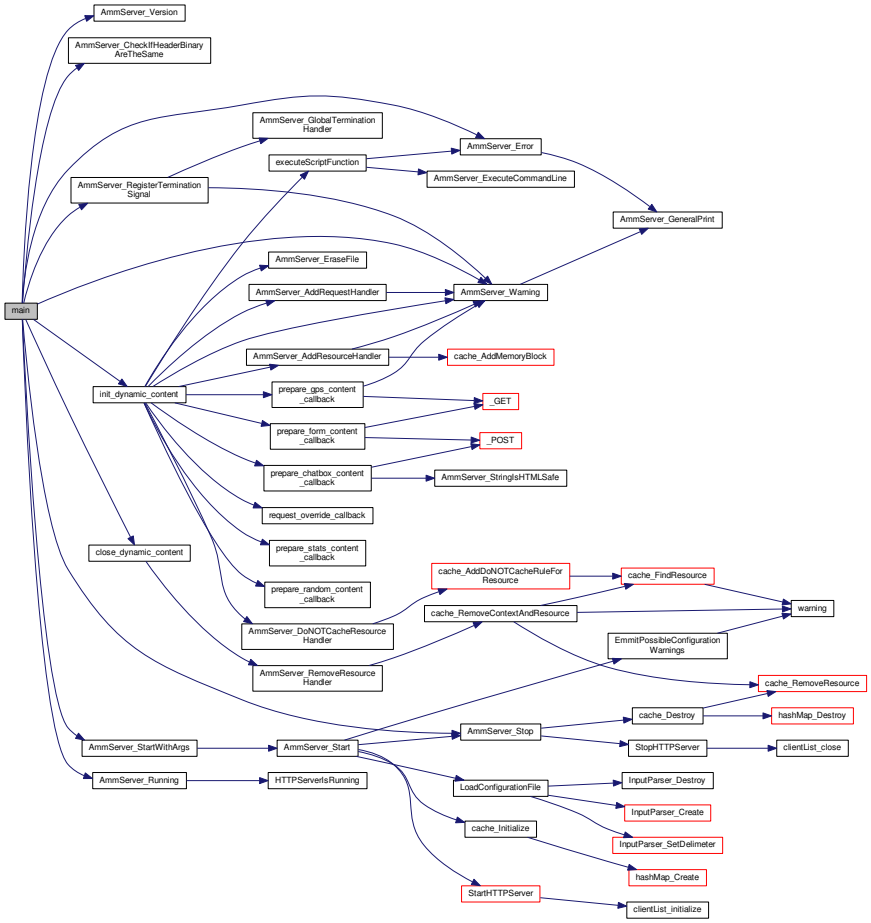
Here is the call graph for this function:



6.10.2.3 int main (int *argc*, char * *argv[]*)

Dynamic content code ..! END ----------—

Here is the call graph for this function:



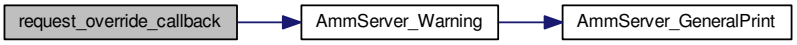
6.10.2.4 void* prepare_stats_content_callback (struct AmmServer_DynamicRequest * *rqst*)

6.10.2.5 void* processUploadCallback (struct AmmServer_DynamicRequest * *rqst*)

Here is the call graph for this function:

6.10.2.6 void request_override_callback (void * *request*)

Here is the call graph for this function:



6.10.3 Variable Documentation

6.10.3.1 struct AmmServer_Instance* default_server =0

6.10.3.2 struct AmmServer_RequestOverride_Context GET_override ={{0}}

6.10.3.3 struct AmmServer_RH_Context stats ={0}

6.10.3.4 char templates_root[MAX_FILE_PATH] ="public_html/templates/"

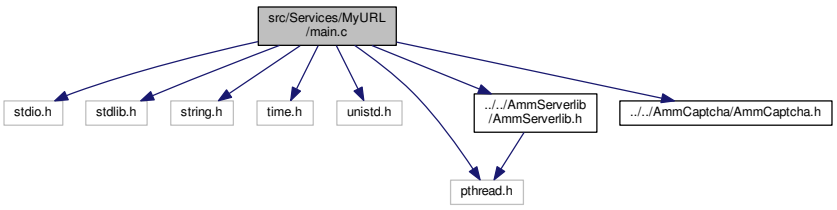
6.10.3.5 struct AmmServer_RH_Context uploadProcessor ={0}

6.10.3.6 char webserver_root[MAX_FILE_PATH] ="src/MyLoader/htmlData/"

6.11 src/Services/MyURL/main.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <unistd.h>
#include <pthread.h>
#include "../../AmmServerlib/AmmServerlib.h"
#include "../../AmmCaptcha/AmmCaptcha.h"
```

Include dependency graph for main.c:



Data Structures

- struct URLDB

Macros

- #define ENABLE_CAPTCHA_SYSTEM 1
- #define USE_BINARY_SEARCH 1
- #define MAX_BINDING_PORT 65534
- #define MAX_CAPTCHA_JPG_SIZE 10 * 1024
- #define DEFAULT_BINDING_PORT 8080
- #define DYNAMIC_PAGES_MEMORY_COMMITED 4096
- #define MAX_TO_SIZE 32
- #define MAX_LONG_URL_SIZE 2048
- #define MAX_LINKS 200000
- #define LINK_ALLOCATION_STEP 5000
- #define REGROUP_AFTER_X_UNSORTED_LINKS 1000

Functions

- int is_an_unsafe_str (char *input, unsigned int input_length)
- int Append2MyURLDBFile (char *filename, char *longURL, char *shortURL)
- unsigned long hashURL (char *str)
- unsigned int allocateLinksIfNeeded ()
- int isURLDBSorted ()
- int printURLDB ()
- int struct_cmp_urldb_items (const void *a, const void *b)
- unsigned int Find_longURLSerial (char *shortURL, int *found)
- unsigned int Find_longURL (char *shortURL, int *found)
- char * Get_longURL (char *shortURL)
- int ReWriteMyURLDBFile (char *filename, struct URLDB *links, unsigned int loaded_links)
- int ResortDB (char *db_file, struct URLDB *links, unsigned int loaded_links)
- unsigned long Add_MyURL (char *longURL, char *shortURL, int saveit)
- int LoadMyURLDBFile (char *filename)
- void * serve_error_url_page (struct AmmServer_DynamicRequest *rqst)
- void * serve_captcha_page (struct AmmServer_DynamicRequest *rqst)
- void * serve_create_url_page (struct AmmServer_DynamicRequest *rqst)
- void * serve_goto_url_page (struct AmmServer_DynamicRequest *rqst)
- void resolveRequest (void *request)
- void init_dynamic_content ()
- void close_dynamic_content ()
- int main (int argc, char *argv[])

Variables

- char webserver_root [MAX_FILE_PATH] ="public_html/"
- char templates_root [MAX_FILE_PATH] ="public_html/templates/"
- char service_filename_noslash [5] ="go"
- char service_filename [5] ="/go"
- char service_root [128] ="http://myurl.ammar.gr/go"
- char service_root_withoutfilename [128] ="http://myurl.ammar.gr/"
- char * default_failed = (char*)"http://myurl.ammar.gr/error.html"
- char db_file [128] ="myurl.db"
- pthread_mutex_t db_fileLock
- pthread_mutex_t db_addIDLock
- char indexPagePath [128] ="src/Services/MyURL/myurl.html"
- char * indexPage =0
- unsigned int indexPageLength =0

- struct AmmServer_Instance * myurl_server =0
- struct AmmServer_RequestOverride_Context requestResolver ={{0}}
- struct AmmServer_RH_Context error_url ={0}
- struct AmmServer_RH_Context create_url ={0}
- struct AmmServer_RH_Context goto_url ={0}
- struct AmmServer_RH_Context captcha_url ={0}
- unsigned int loaded_links =0
- unsigned int sorted_links =0
- unsigned int allocated_links =0
- struct URLDB * links =0

6.11.1 Macro Definition Documentation

6.11.1.1 #define DEFAULT_BINDING_PORT 8080

6.11.1.2 #define DYNAMIC_PAGES_MEMORY_COMMITED 4096

6.11.1.3 #define ENABLE_CAPTCHA_SYSTEM 1

6.11.1.4 #define LINK_ALLOCATION_STEP 5000

6.11.1.5 #define MAX_BINDING_PORT 65534

6.11.1.6 #define MAX_CAPTCHA_JPG_SIZE 10 * 1024

6.11.1.7 #define MAX_LINKS 200000

6.11.1.8 #define MAX_LONG_URL_SIZE 2048

6.11.1.9 #define MAX_TO_SIZE 32

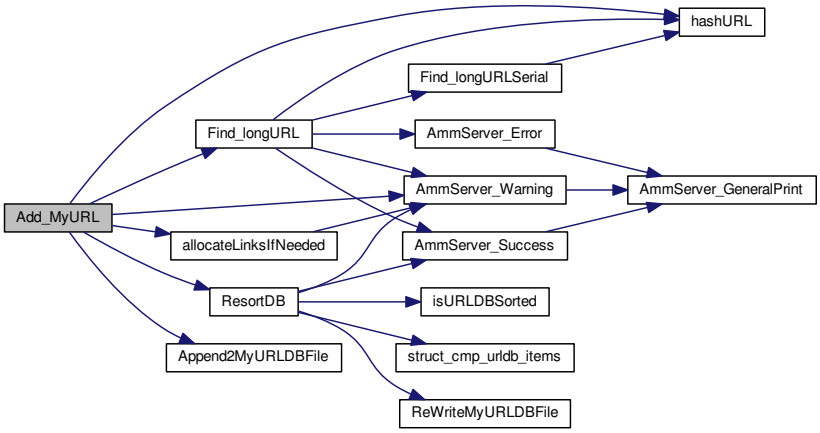
6.11.1.10 #define REGROUP_AFTER_X_UNSORTED_LINKS 1000

6.11.1.11 #define USE_BINARY_SEARCH 1

6.11.2 Function Documentation

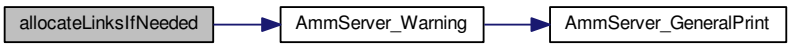
6.11.2.1 unsigned long Add_MyURL (char * *longURL*, char * *shortURL*, int *saveit*)

Here is the call graph for this function:



6.11.2.2 unsigned int allocateLinksIfNeeded ( )

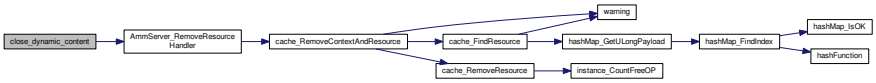
Here is the call graph for this function:



6.11.2.3 int Append2MyURLDBFile (char * *filename*, char * *longURL*, char * *shortURL*)

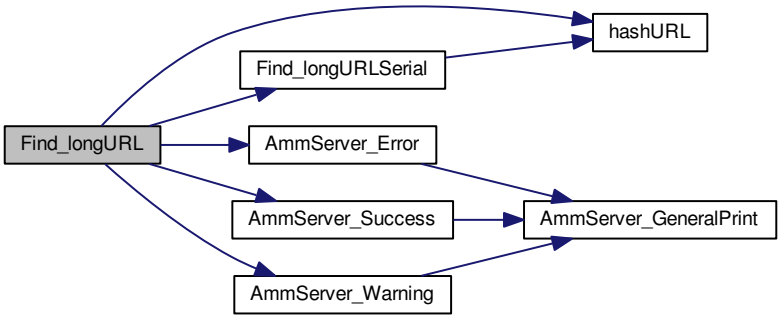
6.11.2.4 void close_dynamic_content ( )

Here is the call graph for this function:



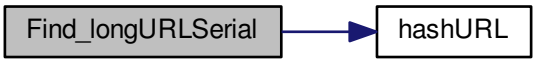
6.11.2.5 unsigned int Find_longURL (char * *shortURL*, int * *found*) [inline]

Here is the call graph for this function:



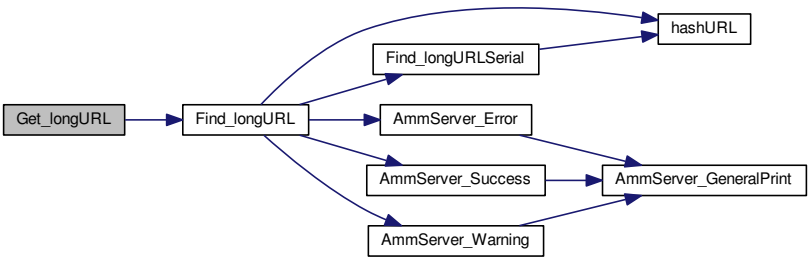
6.11.2.6 unsigned int Find_longURLSerial (char * *shortURL*, int * *found*) [inline]

Here is the call graph for this function:



6.11.2.7 char* Get_longURL (char * *shortURL*)

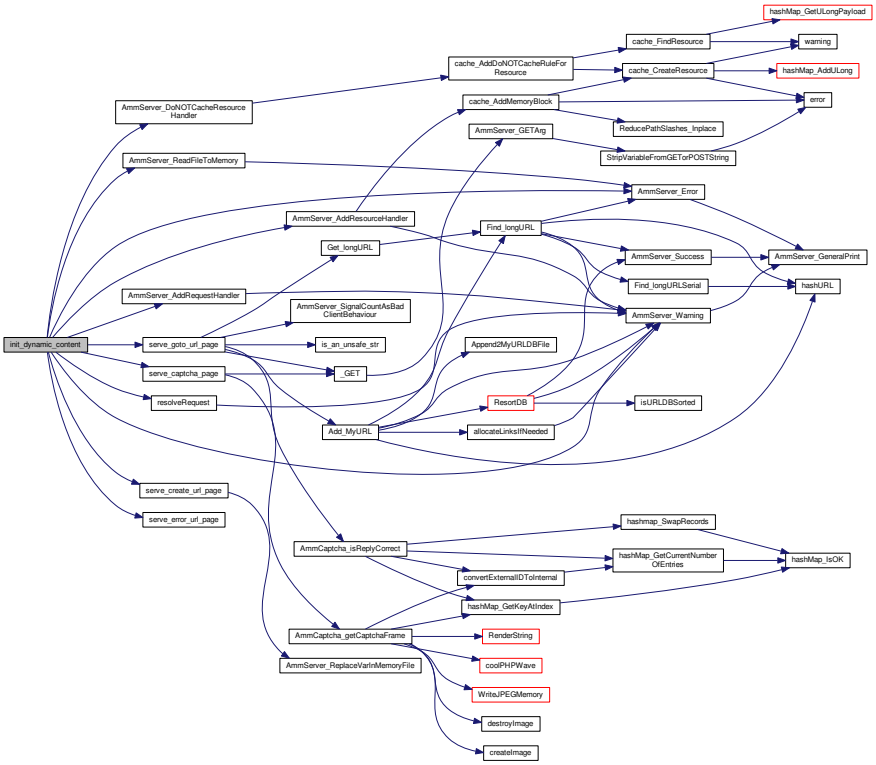
Here is the call graph for this function:



6.11.2.8 unsigned long hashURL (char * *str*)

6.11.2.9 void init_dynamic_content ( )

Here is the call graph for this function:

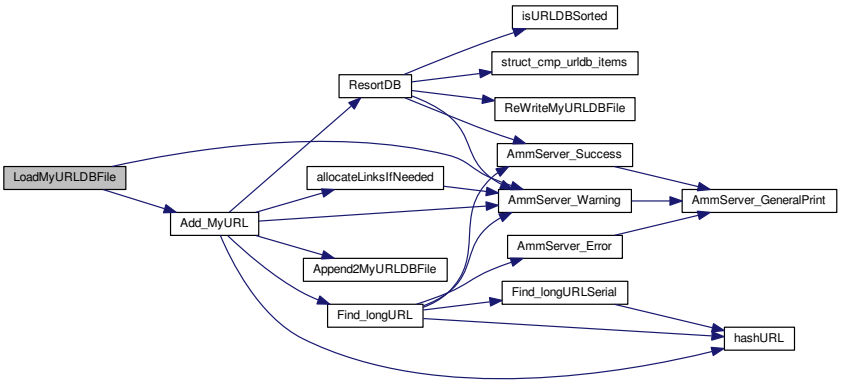


6.11.2.10 int is_an_unsafe_str (char * *input,* unsigned int *input_length*)

6.11.2.11 int isURLDBSorted ( )

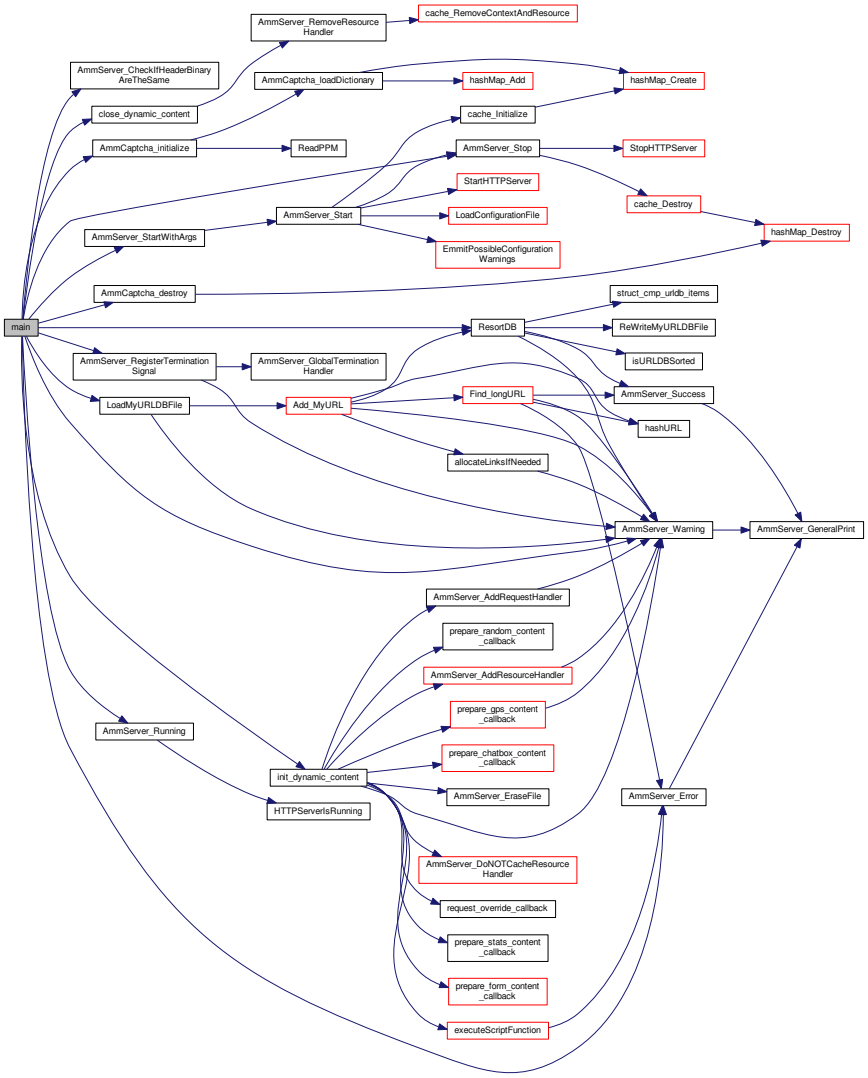
6.11.2.12 int LoadMyURLDBFile (char * *filename*)

Here is the call graph for this function:



6.11.2.13 int main (int *argc*, char * *argv[]*)

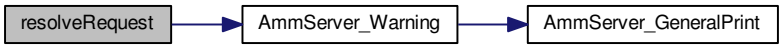
Here is the call graph for this function:



6.11.2.14 int printURLDB ( )

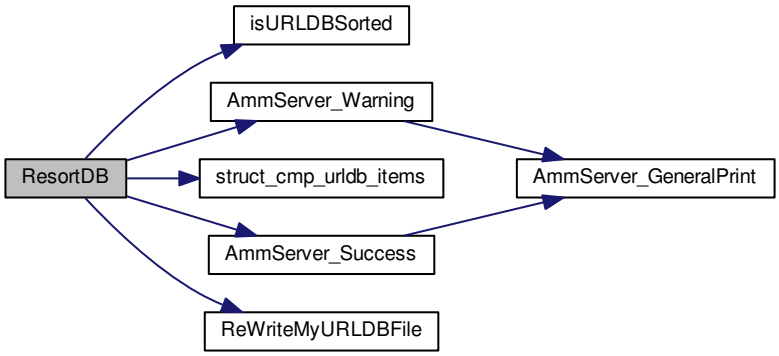
6.11.2.15 void resolveRequest (void * *request*)

Here is the call graph for this function:



6.11.2.16 int ResortDB (char * *db_file*, struct URLDB * *links*, unsigned int *loaded_links*)

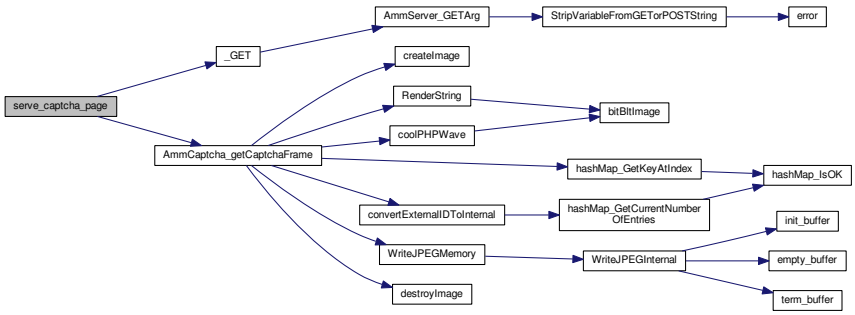
Here is the call graph for this function:



6.11.2.17 int ReWriteMyURLDBFile (char * *filename*, struct URLDB * *links*, unsigned int *loaded_links*)

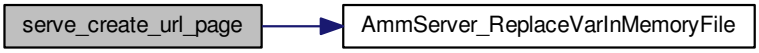
6.11.2.18 void* serve_captcha_page (struct AmmServer_DynamicRequest * *rqst*)

Here is the call graph for this function:



6.11.2.19 void* serve_create_url_page (struct AmmServer_DynamicRequest * *rqst*)

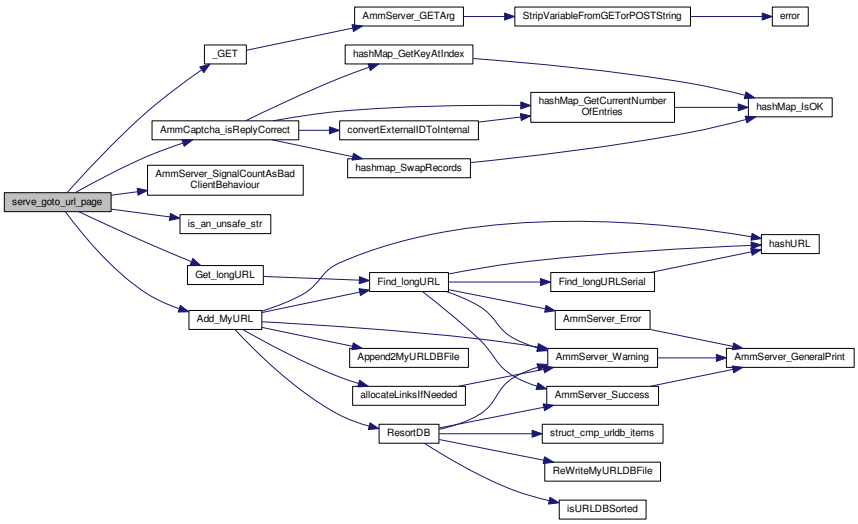
Here is the call graph for this function:



6.11.2.20 void* serve_error_url_page (struct AmmServer_DynamicRequest * *rqst*)

6.11.2.21 void* serve_goto_url_page (struct AmmServer_DynamicRequest * *rqst*)

Here is the call graph for this function:



6.11.2.22 int struct_cmp_urldb_items (const void * *a*, const void * *b*)

6.11.3 Variable Documentation

6.11.3.1 unsigned int allocated_links =0

6.11.3.2 struct AmmServer_RH_Context captcha_url ={0}

6.11.3.3 struct AmmServer_RH_Context create_url ={0}

6.11.3.4 pthread_mutex_t db_addIDLock

6.11.3.5 char db_file[128] ="myurl.db"

6.11.3.6 pthread_mutex_t db_fileLock

6.11.3.7 char* default_failed = (char*)"http://myurl.ammar.gr/error.html"

6.11.3.8 struct AmmServer_RH_Context error_url ={0}

6.11.3.9 struct AmmServer_RH_Context goto_url ={0}

6.11.3.10 char* indexPage =0

6.11.3.11 unsigned int indexPageLength =0

6.11.3.12 char indexPagePath[128] ="src/Services/MyURL/myurl.html"

6.11.3.13 struct URLDB* links =0

6.11.3.14 unsigned int loaded_links =0

6.11.3.15 struct AmmServer_Instance* myurl_server =0

6.11.3.16 struct AmmServer_RequestOverride_Context requestResolver ={{0}}

6.11.3.17 char service_filename[5] ="/go"

6.11.3.18 char service_filename_noslash[5] ="go"

6.11.3.19 char service_root[128] ="http://myurl.ammar.gr/go"

6.11.3.20 char service_root_withoutfilename[128] ="http://myurl.ammar.gr/"

6.11.3.21 unsigned int sorted_links =0

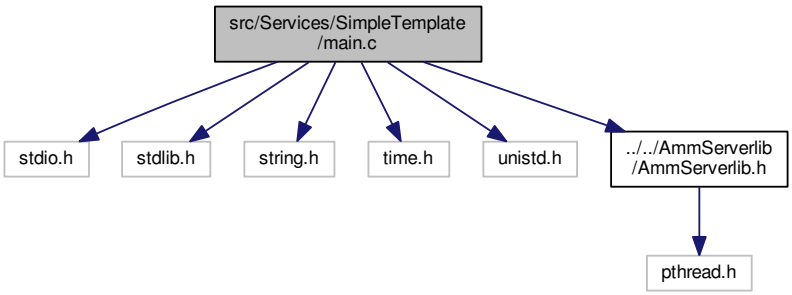
6.11.3.22 char templates_root[MAX_FILE_PATH] ="public_html/templates/"

6.11.3.23 char webserver_root[MAX_FILE_PATH] ="public_html/"

6.12 src/Services/SimpleTemplate/main.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <unistd.h>
#include "../../AmmServerlib/AmmServerlib.h"
```

Include dependency graph for main.c:



Macros

- #define DEFAULT_BINDING_PORT 8080

Functions

- void * prepare_stats_content_callback (struct AmmServer_DynamicRequest *rqst)

- void * prepare_random_content_callback (struct AmmServer_DynamicRequest *rqst)
- void request_override_callback (void *request)
- void init_dynamic_content ()
- void close_dynamic_content ()
- int main (int argc, char *argv[])

Variables

- char webserver_root [MAX_FILE_PATH] ="public_html/"
- char templates_root [MAX_FILE_PATH] ="public_html/templates/"
- struct AmmServer_Instance * default_server =0
- struct AmmServer_RequestOverride_Context GET_override ={{0}}
- struct AmmServer_RH_Context random_chars ={0}
- struct AmmServer_RH_Context stats ={0}

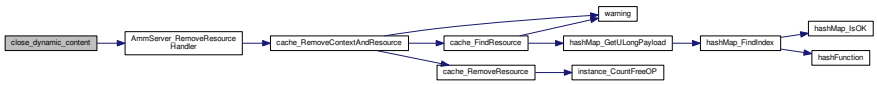
6.12.1 Macro Definition Documentation

6.12.1.1 #define DEFAULT_BINDING_PORT 8080

6.12.2 Function Documentation

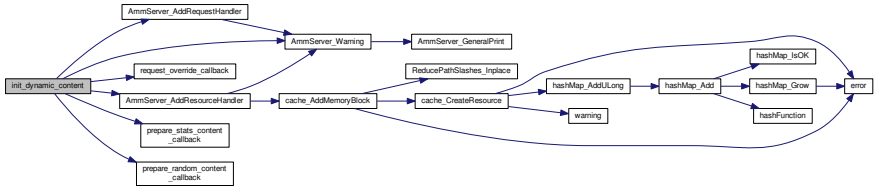
6.12.2.1 void close_dynamic_content ( )

Here is the call graph for this function:



6.12.2.2 void init_dynamic_content ( )

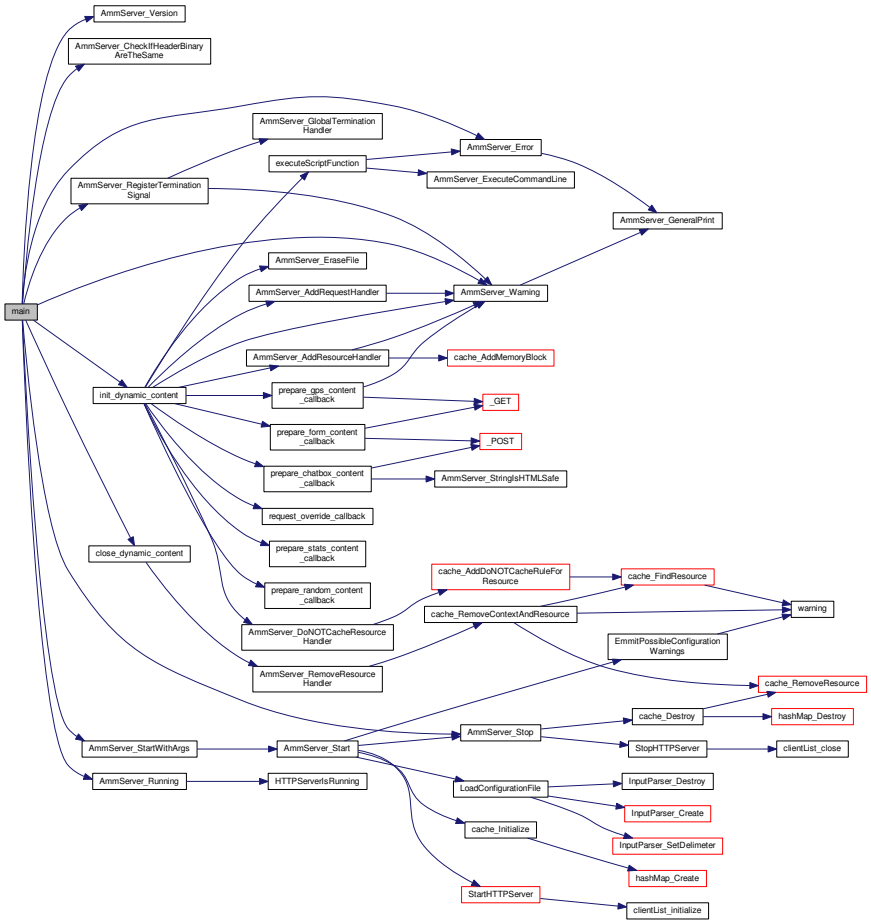
Here is the call graph for this function:



6.12.2.3 int main (int *argc*, char * *argv[]*)

Dynamic content code ..! END ----------—

Here is the call graph for this function:



6.12.2.4 void* prepare_random_content_callback (struct AmmServer_DynamicRequest * *rqst*)

6.12.2.5 void* prepare_stats_content_callback (struct AmmServer_DynamicRequest * *rqst*)

6.12.2.6 void request_override_callback (void * *request*)

6.12.3 Variable Documentation

6.12.3.1 struct AmmServer_Instance* default_server =0

Dynamic content code ..! START!

6.12.3.2 struct AmmServer_RequestOverride_Context GET_override ={{0}}

6.12.3.3 struct AmmServer_RH_Context random_chars ={0}

6.12.3.4 struct AmmServer_RH_Context stats ={0}

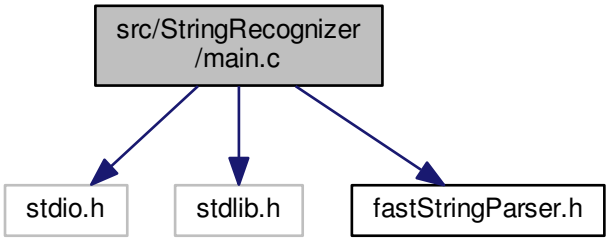
6.12.3.5 char templates_root[MAX_FILE_PATH] ="public_html/templates/"

6.12.3.6 char webserver_root[MAX_FILE_PATH] ="public_html/"

6.13 src/StringRecognizer/main.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include "fastStringParser.h"
```

Include dependency graph for main.c:



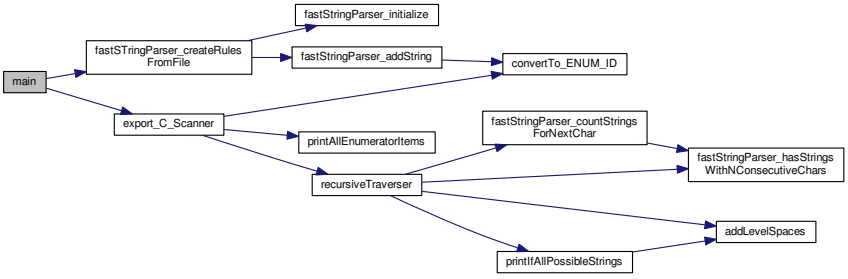
Functions

- int main (int argc, char ∗argv[])

6.13.1 Function Documentation

6.13.1.1 int main (int *argc,* char ∗ *argv[]*)

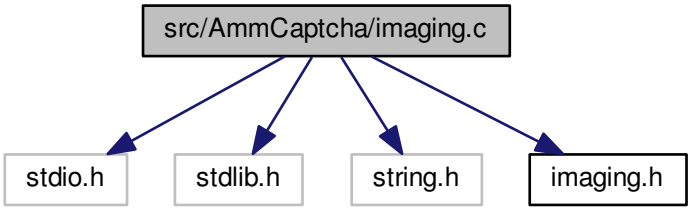
Here is the call graph for this function:



6.14 src/AmmCaptcha/imaging.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "imaging.h"
```

Include dependency graph for imaging.c:



Macros

- #define READ_CREATES_A_NEW_PIXEL_BUFFER 1
- #define PPMREADBUFLEN 256
- #define DISPLAY_DEBUG_INFO 0

Functions

- struct Image * createImage (unsigned int width, unsigned int height, unsigned int depth)
- struct Image * copyImage (struct Image *source)
- int destroyImage (struct Image *source)
- int bitBltImage (struct Image *target, unsigned int targetX, unsigned int targetY, struct Image *source, unsigned int sourceX, unsigned int sourceY, unsigned int width, unsigned int height)
- int bitBltImageRotated (struct Image *target, unsigned int targetCenterX, unsigned int targetCenterY, float rotation, struct Image *source, unsigned int sourceX, unsigned int sourceY, unsigned int width, unsigned int height)
- int ReadPPM (struct Image *pic, char *filename, char read_only_header)
- int WritePPM (struct Image *pic, char *filename)

6.14.1 Macro Definition Documentation

6.14.1.1 #define DISPLAY_DEBUG_INFO 0

6.14.1.2 #define PPMREADBUFLEN 256

6.14.1.3 #define READ_CREATES_A_NEW_PIXEL_BUFFER 1

6.14.2 Function Documentation

6.14.2.1 int bitBltImage (struct Image * *target,* unsigned int *targetX,* unsigned int *targetY,* struct Image * *source,* unsigned int *sourceX,* unsigned int *sourceY,* unsigned int *width,* unsigned int *height*)

6.14.2.2 int bitBltImageRotated (struct Image * *target,* unsigned int *targetCenterX,* unsigned int *targetCenterY,* float *rotation,* struct Image * *source,* unsigned int *sourceX,* unsigned int *sourceY,* unsigned int *width,* unsigned int *height*)

6.14.2.3 struct Image* copyImage (struct Image * *source*)

6.14.2.4 struct Image* createImage (unsigned int *width,* unsigned int *height,* unsigned int *depth*)

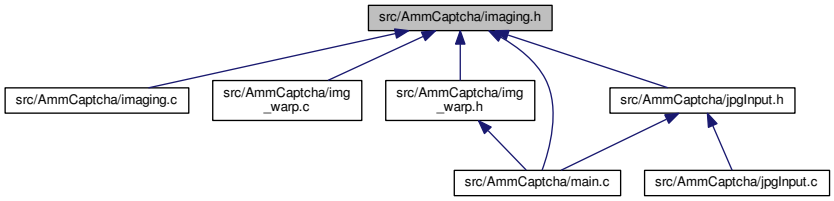
6.14.2.5 int destroyImage (struct Image * *source*)

6.14.2.6 int ReadPPM (struct Image * *pic,* char * *filename,* char *read_only_header*)

6.14.2.7 int WritePPM (struct Image * *pic,* char * *filename*)

6.15 src/AmmCaptcha/imaging.h File Reference

This graph shows which files directly or indirectly include this file:



Data Structures

- struct Image

Functions

- struct Image * createImage (unsigned int width, unsigned int height, unsigned int depth)
- struct Image * copyImage (struct Image *source)
- int destroyImage (struct Image *source)
- int bitBltImage (struct Image *target, unsigned int targetX, unsigned int targetY, struct Image *source, unsigned int sourceX, unsigned int sourceY, unsigned int width, unsigned int height)
- int ReadPPM (struct Image *pic, char *filename, char read_only_header)
- int WritePPM (struct Image *pic, char *filename)

6.15.1 Function Documentation

6.15.1.1 int bitBltImage (struct Image * *target,* unsigned int *targetX,* unsigned int *targetY,* struct Image * *source,* unsigned int *sourceX,* unsigned int *sourceY,* unsigned int *width,* unsigned int *height*)

6.15.1.2 struct Image* copyImage (struct Image * *source*)

6.15.1.3 struct Image* createImage (unsigned int *width,* unsigned int *height,* unsigned int *depth*)

6.15.1.4 int destroyImage (struct Image * *source*)

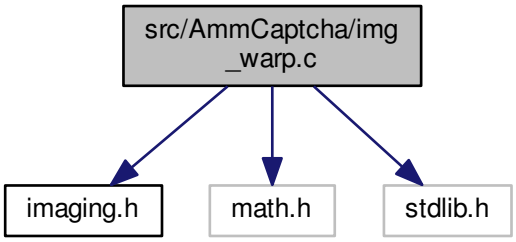
6.15.1.5 int ReadPPM (struct Image * *pic,* char * *filename,* char *read_only_header*)

6.15.1.6 int WritePPM (struct Image * *pic,* char * *filename*)

6.16 src/AmmCaptcha/img_warp.c File Reference

```
#include "imaging.h"
#include <math.h>
#include <stdlib.h>
```

Include dependency graph for img_warp.c:



Macros

- #define ABS(num1) ((num1) >=0 ? (num1) : (-1∗num1))
- #define ABSDIFF(num1, num2) ((num1-num2) >=0 ? (num1-num2) : (num2 - num1))

Functions

- int warpImage (struct Image ∗target, unsigned int posX, unsigned int posY, signed int warpDeltaX, signed int warpDeltaY)
- int coolPHPWave (struct Image ∗target, unsigned int periodX, unsigned int periodY, signed int amplitudeX, signed int amplitudeY)

6.16.1 Macro Definition Documentation

6.16.1.1 #define ABS(*num1*) ((num1) >=0 ? (num1) : (-1∗num1))

6.16.1.2 #define ABSDIFF(*num1, num2*) ((num1-num2) >=0 ? (num1-num2) : (num2 - num1))

6.16.2 Function Documentation

6.16.2.1 int coolPHPWave (struct Image ∗ *target,* unsigned int *periodX,* unsigned int *periodY,* signed int *amplitudeX,* signed int *amplitudeY*)

This is a C version of the PHP script from Jose Rodriguez , currently used as a swirling mechanism , it is GPLv3 as this project :)

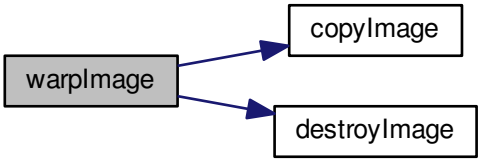
Author

Jose Rodriguez jose.rodriguez@exec.cl GPLv3 captcha 0.3

Here is the call graph for this function:

6.16.2.2 int warpImage (struct Image * *target*, unsigned int *posX*, unsigned int *posY*, signed int *warpDeltaX*, signed int *warpDeltaY*)

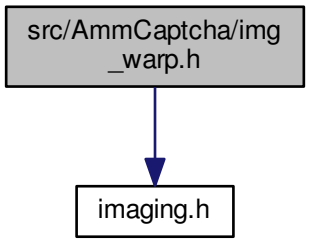
Here is the call graph for this function:



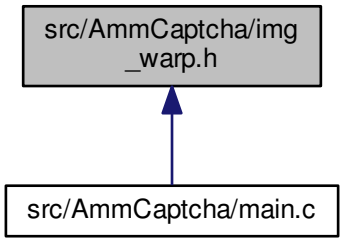
6.17 src/AmmCaptcha/img_warp.h File Reference

```
#include "imaging.h"
```

Include dependency graph for img_warp.h:



This graph shows which files directly or indirectly include this file:



Functions

- int warpImage (struct Image *target, unsigned int posX, unsigned int posY, signed int warpDeltaX, signed int warpDeltaY)
- int coolPHPWave (struct Image *target, unsigned int periodX, unsigned int periodY, signed int amplitudeX, signed int amplitudeY)

6.17.1 Function Documentation

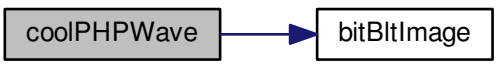
6.17.1.1 int coolPHPWave (struct Image * *target*, unsigned int *periodX*, unsigned int *periodY*, signed int *amplitudeX*, signed int *amplitudeY*)

This is a C version of the PHP script from Jose Rodriguez , currently used as a swirling mechanism , it is GPLv3 as this project :)

Author

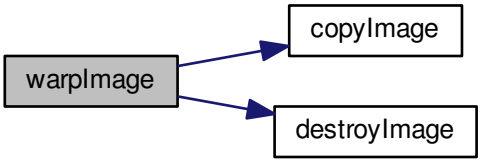
Jose Rodriguez jose.rodriguez@exec.cl GPLv3 captcha 0.3

Here is the call graph for this function:



6.17.1.2 int warpImage (struct Image * *target*, unsigned int *posX*, unsigned int *posY*, signed int *warpDeltaX*, signed int *warpDeltaY*)

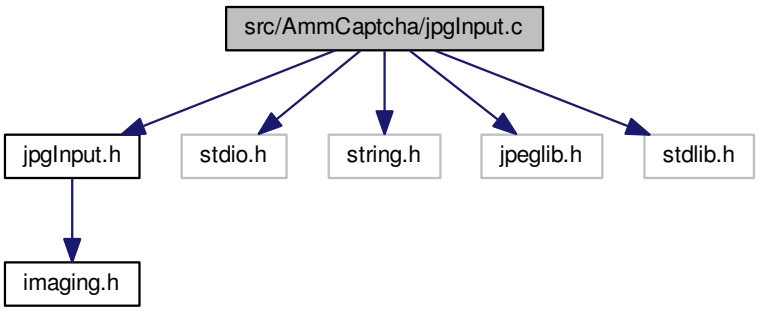
Here is the call graph for this function:



6.18 src/AmmCaptcha/jpgInput.c File Reference

```
#include "jpgInput.h"
#include <stdio.h>
#include <string.h>
#include <jpeglib.h>
#include <stdlib.h>
```

Include dependency graph for jpgInput.c:



Functions

- void init_buffer (struct jpeg_compress_struct *cinfo)
- int empty_buffer (struct jpeg_compress_struct *cinfo)
- void term_buffer (struct jpeg_compress_struct *cinfo)
- int fastJPGHeaderCheck (FILE *file)
- int ReadJPEG (char *filename, struct Image *pic, char read_only_header)
- int WriteJPEGInternal (char *filename, struct Image *pic, char *mem, unsigned long *mem_size)
- int WriteJPEGFile (struct Image *pic, char *filename)
- int WriteJPEGMemory (struct Image *pic, char *mem, unsigned long *mem_size)
- int jpegtest ()

6.18.1 Function Documentation

6.18.1.1 int empty_buffer (struct jpeg_compress_struct * *cinfo*)

6.18.1.2 int fastJPGHeaderCheck (FILE * *file*)

6.18.1.3 void init_buffer (struct jpeg_compress_struct * *cinfo*)

read_jpeg_file Reads from a jpeg file on disk specified by filename and saves into the raw_image buffer in an uncompressed format.

Returns

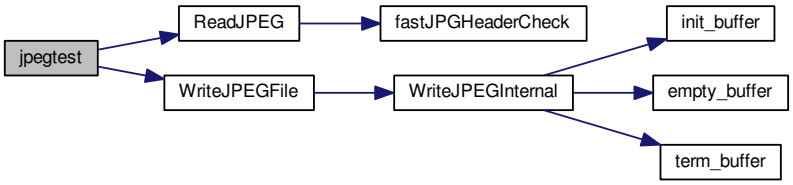
positive integer if successful, -1 otherwise

Parameters

**filename*	char string specifying the file name to read from

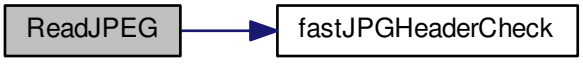
6.18.1.4 int jpegtest ( )

Here is the call graph for this function:



6.18.1.5 int ReadJPEG (char * *filename,* struct Image * *pic,* char *read_only_header*)

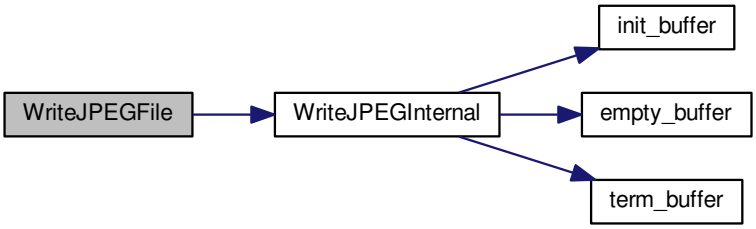
Here is the call graph for this function:



6.18.1.6 void term_buffer (struct jpeg_compress_struct * *cinfo*)

6.18.1.7 int WriteJPEGFile (struct Image * *pic*, char * *filename*)

Here is the call graph for this function:



6.18.1.8 int WriteJPEGInternal (char * *filename*, struct Image * *pic*, char * *mem*, unsigned long * *mem_size*)

write_jpeg_file Writes the raw image data stored in the raw_image buffer to a jpeg image with default compression and smoothing options in the file specified by *filename.

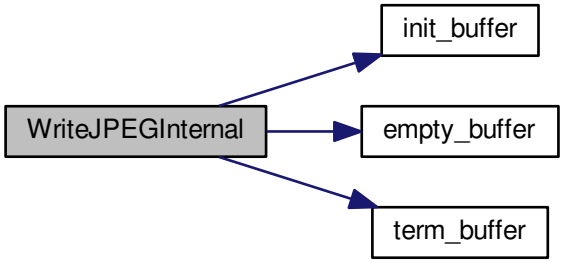
Returns

positive integer if successful, -1 otherwise

Parameters

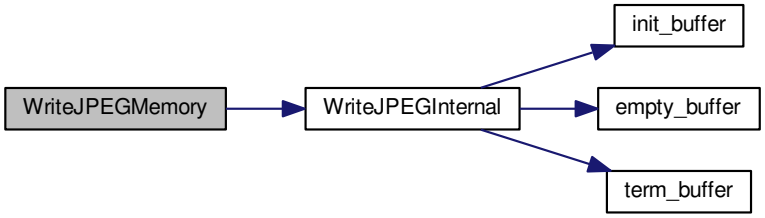
filename	char string specifying the file name to save to

Here is the call graph for this function:



6.18.1.9 int WriteJPEGMemory (struct Image * *pic,* char * *mem,* unsigned long * *mem_size*)

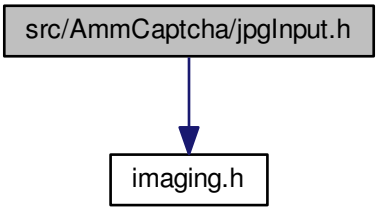
Here is the call graph for this function:



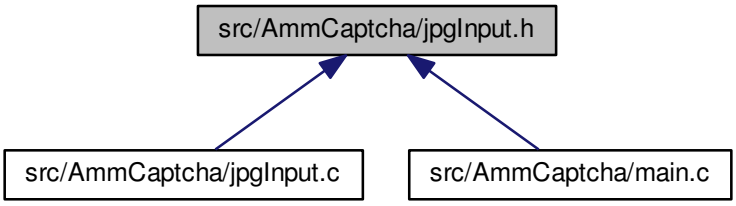
6.19 src/AmmCaptcha/jpgInput.h File Reference

```
#include "imaging.h"
```

Include dependency graph for jpgInput.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define USE_JPG_FILES 1

Functions

- int ReadJPEG (char *filename, struct Image *pic, char read_only_header)
- int WriteJPEGFile (struct Image *pic, char *filename)
- int WriteJPEGMemory (struct Image *pic, char *mem, unsigned long *mem_size)

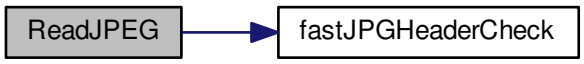
6.19.1 Macro Definition Documentation

6.19.1.1 #define USE_JPG_FILES 1

6.19.2 Function Documentation

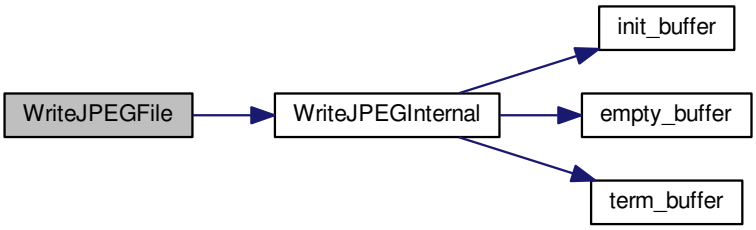
6.19.2.1 int ReadJPEG (char * *filename*, struct Image * *pic*, char *read_only_header*)

Here is the call graph for this function:



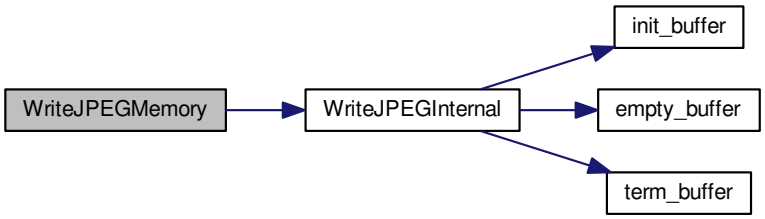
6.19.2.2 int WriteJPEGFile (struct Image * *pic*, char * *filename*)

Here is the call graph for this function:



6.19.2.3 int WriteJPEGMemory (struct Image * *pic,* char * *mem,* unsigned long * *mem_size*)

Here is the call graph for this function:

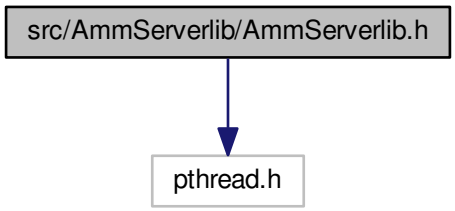


6.20 src/AmmServerlib/AmmServerlib.h File Reference

The Main Header for AmmarServer.

```
#include <pthread.h>
```

Include dependency graph for AmmServerlib.h:



This graph shows which files directly or indirectly include this file:

Data Structures

- struct HTTPHeader

  Each HTTP Request has a header , this is the internal structure that carries the information about the header of an HTTP request parsed and ready for easy for consumption by the various consumers of HTTP requests.

- struct AmmServer_RequestOverride_Context

  We can override/intercept connections before the very fundamental HTTP stage using a request override context and AmmServer_AddRequestHandler This is the structure that holds the information and what to be called back to populate the response.

- struct AmmServer_DynamicRequest

  When a call to a function that is a dynamic request is done this is the structure that holds the information.

- struct AmmServer_RH_Context

  We can override resources to respond with our own C function code , to do so a AmmServer_DynamicRequest must be populated using a AmmServer_AddResourceHandler.

- struct AmmServer_Instance_Settings

  Each Instance of AmmarServer has some basic settings , which are stored in AmmServer_Instance_Settings.

- struct AmmServer_Instance

  This holds all the information about an Ammar Server Instance , sockets , thread pools , cache , memory , settings etc , this is the central structure for holding context.

- struct HTTPTransaction

  Structure to keep data for an HTTP Transaction.

Macros

- #define AMMAR_SERVER_HTTP_HEADER_SPEC 125

  An enumerator that lists the types of requests , per HTTP spec , see http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html Of course not all of them are supported/used internally but they are listed in the same order to maintain spec compatibility.

- #define MAX_IP_STRING_SIZE 32
- #define MAX_QUERY 2048
- #define MAX_RESOURCE 2048
- #define MAX_FILE_PATH 1024
- #define POPEN_BUFFER_SIZE 256

  Size for popen replies.

- #define MAX_INSTANCE_NAME_STRING 128

Enumerations

- enum TypesOfRequests {
  NONE =0, HEAD, GET, POST,
  PUT, DELETE, TRACE, OPTIONS,
  CONNECT, PATCH, BAD }

  An enumerator that lists the types of requests , per HTTP spec , see http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html Of course not all of them are supported/used internally but they are listed in the same order to maintain spec compatibility.

- enum RHScenarios { SAME_PAGE_FOR_ALL_CLIENTS = 0, DIFFERENT_PAGE_FOR_EACH_CLIENT }

  Each Dynamic Resource Handler can have multiple profiles for optimizing performance/memory usage etc. For now there are 2 profiles/scenarios. The first one is where there is a global state that all clients should share The second one is where there is a different page for each client , which is more memory intensive since there are separate buffers etc for each request.

- enum AmmServInfos { AMMINF_ACTIVE_CLIENTS =0, AMMINF_ACTIVE_THREADS }

  Enumerator for calls AmmServer_GetInfo.

- enum AmmServSettings { AMMSET_PASSWORD_PROTECTION =0, AMMSET_TEST }

  Enumerator for calls AmmServer_GetIntSettingValue and AmmServer_SetIntSettingValue.

- enum AmmServStrSettings { AMMSET_USERNAME_STR =0, AMMSET_PASSWORD_STR, AMMSET_TESTSTR }

  Enumerator for calls AmmServer_GetStrSettingValue and AmmServer_SetStrSettingValue.

Functions

- char * AmmServer_Version ()

  Returns a string with the version of AmmarServer , in case it returns NULL it means that we are linked to AmmarServerNULL which means a fake binary.
- int AmmServer_CheckIfHeaderBinaryAreTheSame (int headerSpec)

  Internal Check to compare against changes of the header files.
- void AmmServer_Warning (const char *format,...)

  Writes the C string pointed by format to stderr , as a warning (Yellow) and logs it to the appropriate log If format includes format specifiers (subsequences beginning with %), the additional arguments following format are formatted and inserted in the resulting string replacing their respective specifiers.
- void AmmServer_Error (const char *format,...)

  Writes the C string pointed by format to stderr , as an error (Red) and logs it to the appropriate log If format includes format specifiers (subsequences beginning with %), the additional arguments following format are formatted and inserted in the resulting string replacing their respective specifiers.
- void AmmServer_Success (const char *format,...)

  Writes the C string pointed by format to stderr , as a success (Green) and logs it to the appropriate log If format includes format specifiers (subsequences beginning with %), the additional arguments following format are formatted and inserted in the resulting string replacing their respective specifiers.
- struct AmmServer_Instance * AmmServer_Start (char *name, char *ip, unsigned int port, char *conf_file, char *web_root_path, char *templates_root_path)

  Start a Web Server , allocate memory , bind ports and return its instance..
- struct AmmServer_Instance * AmmServer_StartWithArgs (char *name, int argc, char **argv, char *ip, unsigned int port, char *conf_file, char *web_root_path, char *templates_root_path)

  *Start a Web Server , allocate memory , bind ports and return its instance , also process arguments (argc and argv from int main(int argc, char *argv[])) ..*
- int AmmServer_Stop (struct AmmServer_Instance *instance)

  Stop a Web Server , deallocate memory , free ports and free the server instance..
- int AmmServer_Running (struct AmmServer_Instance *instance)

  Query if an instance of AmmarServer is initialized and running.
- int AmmServer_AddRequestHandler (struct AmmServer_Instance *instance, struct AmmServer_RequestOverride_Context *RequestOverrideContext, char *request_type, void *callback)

  Add a request handler to handle requests , before they get processed internally Calling this will bind a C function that will be called and produce output when someone asks for any resource using the specified method TODO : Improve this documenatation.
- int AmmServer_AddResourceHandler (struct AmmServer_Instance *instance, struct AmmServer_RH_Context *context, char *resource_name, char *web_root, unsigned int allocate_mem_bytes, unsigned int callback_every_x_msec, void *callback, unsigned int scenario)

  Add a request handler to handle dynamic requests , the core mechanic of AmmarServer Calling this will bind a C function that will be called and produce output when someone asks for a resource TODO : Improve this documenatation.
- int AmmServer_RemoveResourceHandler (struct AmmServer_Instance *instance, struct AmmServer_RH_Context *context, unsigned char free_mem)

  Remove a request handler that hanles dynamic requests.
- int AmmServer_GetInfo (struct AmmServer_Instance *instance, unsigned int info_type)

  Get an Integer out of the state of an instance , of course one can dive into the instance structure but this is a much more clean way to do this.
- int AmmServer_GetIntSettingValue (struct AmmServer_Instance *instance, unsigned int set_type)

  Get an Integer out of the state of an instance , of course one can dive into the instance structure but this is a much more clean way to do this.
- int AmmServer_SetIntSettingValue (struct AmmServer_Instance *instance, unsigned int set_type, int set_value)

  Set an Integer inside the state of an instance , of course one can dive into the instance structure but this is a much more clean way to do this.
- char * AmmServer_GetStrSettingValue (struct AmmServer_Instance *instance, unsigned int set_type)

Get a String out of the state of an instance , of course one can dive into the instance structure but this is a much more clean way to do this.

- int AmmServer_SetStrSettingValue (struct AmmServer_Instance *instance, unsigned int set_type, char *set_value)

  Set an string inside the state of an instance , of course one can dive into the instance structure but this is a much more clean way to do this.

- int AmmServer_POSTArg (struct AmmServer_Instance *instance, struct AmmServer_DynamicRequest *rqst, char *var_id_IN, char *var_value_OUT, unsigned int max_var_value_OUT)

  Get a POST argument.

- int AmmServer_GETArg (struct AmmServer_Instance *instance, struct AmmServer_DynamicRequest *rqst, char *var_id_IN, char *var_value_OUT, unsigned int max_var_value_OUT)

  Get a GET argument.

- int AmmServer_FILES (struct AmmServer_Instance *instance, struct AmmServer_DynamicRequest *rqst, char *var_id_IN, char *var_value_OUT, unsigned int max_var_value_OUT)

  Access a FILE submitted by a dynamic requested.

- int _POST (struct AmmServer_Instance *instance, struct AmmServer_DynamicRequest *rqst, char *var_id_IN, char *var_value_OUT, unsigned int max_var_value_OUT)

  Shorthand/Shortcut for AmmServer_POSTArg()

- int _GET (struct AmmServer_Instance *instance, struct AmmServer_DynamicRequest *rqst, char *var_id_IN, char *var_value_OUT, unsigned int max_var_value_OUT)

  Shorthand/Shortcut for AmmServer_GETArg()

- int _FILES (struct AmmServer_Instance *instance, struct AmmServer_DynamicRequest *rqst, char *var_id_IN, char *var_value_OUT, unsigned int max_var_value_OUT)

  Shorthand/Shortcut for AmmServer_FILES()

- int AmmServer_SignalCountAsBadClientBehaviour (struct AmmServer_Instance *instance, struct AmmServer_DynamicRequest *rqst)

  Staged way to easily handle bad clients etc from the clients , currently a stub..!

- int AmmServer_SaveDynamicRequest (char *filename, struct AmmServer_Instance *instance, struct AmmServer_DynamicRequest *rqst)

  Save Dynamic Request to file.

- int AmmServer_DoNOTCacheResourceHandler (struct AmmServer_Instance *instance, struct AmmServer_RH_Context *context)

  Set resource handler to no-cache mode , this means whoever asks for it will never get a cached response.

- int AmmServer_DoNOTCacheResource (struct AmmServer_Instance *instance, char *resource_name)

  Set resource to no-cache mode , this means whoever asks for it will never get a cached response.

- struct AmmServer_Instance * AmmServer_StartAdminInstance (char *ip, unsigned int port)

  Planned functionality for a default http administrator panel per server per instance , currently not implemented correctly.

- int AmmServer_SelfCheck (struct AmmServer_Instance *instance)

  Perform a sanity check on the instance of AmmarServer , this is mostly a dev debug tool and an entry point for code inside AmmServerlib.

- int AmmServer_ExecuteCommandLine (char *command, char *what2GetBack, unsigned int what2GetBackMaxSize)

  Execute a command and copy its output to the provided buffer.

- int AmmServer_ReplaceVarInMemoryFile (char *page, unsigned int pageLength, char *var, char *value)

  Hot-Replace a variable inside a memory block , typically used to replace placeholders inside text files , like $$$$$$$$-NAME$$$$$$$$, the value should be smaller or equal to the var beeing replaced.

- char * AmmServer_ReadFileToMemory (char *filename, unsigned int *length)

  Read a file and store it to a freshly allocated memory block.

- int AmmServer_WriteFileFromMemory (char *filename, char *memory, unsigned int memoryLength)

  Dump a memory block to a file.

- int AmmServer_RegisterTerminationSignal (void *callback)

Register a function to call a function that gracefully terminates a client when a SIGKILL or the time to stop the server comes.

- int AmmServer_DirectoryExists (char *filename)

  Check if directory Exists.
- int AmmServer_FileExists (char *filename)

  Check if file Exists.
- int AmmServer_EraseFile (char *filename)

  Erase a File.
- unsigned int AmmServer_StringIsHTMLSafe (char *str)

  Check if a string has html elements inside it , so if we append it to a web site we won't have html injected.

6.20.1 Detailed Description

The Main Header for AmmarServer. Any application that may want to interface with AmmarServer will probably want to link to libAmmarServer.a and include this header. It provides the entry point for setting up a web share and access to sub-modules on runtime.

Author

Ammar Qammaz (AmmarkoV)

Bug AmmarServer is not properly pentested yet

6.20.2 Macro Definition Documentation

6.20.2.1 #define AMMAR_SERVER_HTTP_HEADER_SPEC 125

An enumerator that lists the types of requests , per HTTP spec , see http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html Of course not all of them are supported/used internally but they are listed in the same order to maintain spec compatibility.

Bug A potential bug might arise if the specs of the header file are changed and someone is linking with an older version libAmmServer.a thats why this value exists

6.20.2.2 #define MAX_FILE_PATH 1024

6.20.2.3 #define MAX_INSTANCE_NAME_STRING 128

6.20.2.4 #define MAX_IP_STRING_SIZE 32

6.20.2.5 #define MAX_QUERY 2048

6.20.2.6 #define MAX_RESOURCE 2048

6.20.2.7 #define POPEN_BUFFER_SIZE 256

Size for popen replies.

6.20.3 Enumeration Type Documentation

6.20.3.1 enum AmmServInfos

Enumerator for calls AmmServer_GetInfo.

Enumerator

AMMINF_ACTIVE_CLIENTS
AMMINF_ACTIVE_THREADS

6.20.3.2 enum AmmServSettings

Enumerator for calls AmmServer_GetIntSettingValue and AmmServer_SetIntSettingValue.

Enumerator

AMMSET_PASSWORD_PROTECTION
AMMSET_TEST

6.20.3.3 enum AmmServStrSettings

Enumerator for calls AmmServer_GetStrSettingValue and AmmServer_SetStrSettingValue.

Enumerator

AMMSET_USERNAME_STR
AMMSET_PASSWORD_STR
AMMSET_TESTSTR

6.20.3.4 enum RHScenarios

Each Dynamic Resource Handler can have multiple profiles for optimizing performance/memory usage etc. For now there are 2 profiles/scenarios. The first one is where there is a global state that all clients should share The second one is where there is a different page for each client , which is more memory intensive since there are separate buffers etc for each request.

Enumerator

SAME_PAGE_FOR_ALL_CLIENTS
DIFFERENT_PAGE_FOR_EACH_CLIENT

6.20.3.5 enum TypesOfRequests

An enumerator that lists the types of requests , per HTTP spec , see http://www.w3.org/-Protocols/rfc2616/rfc2616-sec9.html Of course not all of them are supported/used internally but they are listed in the same order to maintain spec compatibility.

Enumerator

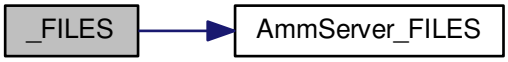
NONE
HEAD
GET
POST
PUT
DELETE
TRACE
OPTIONS
CONNECT
PATCH
BAD

6.20.4 Function Documentation

6.20.4.1 int _FILES (struct AmmServer_Instance * *instance*, struct AmmServer_DynamicRequest * *rqst*, char * *var_id_IN*, char * *var_value_OUT*, unsigned int *max_var_value_OUT*)

Shorthand/Shortcut for AmmServer_FILES()

Here is the call graph for this function:



6.20.4.2 int _GET (struct AmmServer_Instance * *instance*, struct AmmServer_DynamicRequest * *rqst*, char * *var_id_IN*, char * *var_value_OUT*, unsigned int *max_var_value_OUT*)

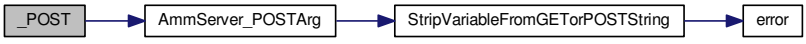
Shorthand/Shortcut for AmmServer_GETArg()

Here is the call graph for this function:

6.20.4.3 int _POST (struct AmmServer_Instance * *instance*, struct AmmServer_DynamicRequest * *rqst*, char * *var_id_IN*, char * *var_value_OUT*, unsigned int *max_var_value_OUT*)

Shorthand/Shortcut for AmmServer_POSTArg()

Here is the call graph for this function:



6.20.4.4 int AmmServer_AddRequestHandler (struct AmmServer_Instance * *instance*, struct AmmServer_RequestOverride_Context * *RequestOverrideContext*, char * *request_type*, void * *callback*)

Add a request handler to handle requests , before they get processed internally Calling this will bind a C function that will be called and produce output when someone asks for any resource using the specified method TODO : Improve this documenatation.

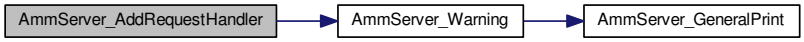
Parameters

An	AmmarServer Instance
A	AmmServer_RequestOverride_Context to be populated
Request	Type
Pointer	to function callback

Return values

1=Success,0=Fail	

Here is the call graph for this function:



6.20.4.5 int AmmServer_AddResourceHandler (struct AmmServer_Instance * *instance*, struct AmmServer_RH_Context * *context*, char * *resource_name*, char * *web_root*, unsigned int *allocate_mem_bytes*, unsigned int *callback_every_x_msec*, void * *callback*, unsigned int *scenario*)

Add a request handler to handle dynamic requests , the core mechanic of AmmarServer Calling this will bind a C function that will be called and produce output when someone asks for a resource TODO : Improve this documenatation.

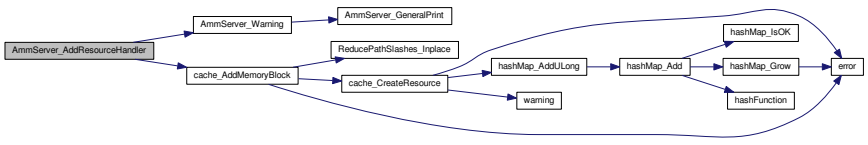
Parameters

An	AmmarServer Instance
An	AmmServer_RH_Context to be populated
Name	of resource that should get dynamic responses (i.e. "index.html")
Root	Path for the specific resource
Memory	chunk to allocate for responses , (this is the max response size)
Minimum	time between two calls of the function (0 = no minimum time)
Function	to be called and provides output when someone asks for resource
Scenario/Profile	of this resource (see RHScenarios)

Return values

1=Success,0=Fail	

Here is the call graph for this function:



6.20.4.6 int AmmServer_CheckIfHeaderBinaryAreTheSame (int *headerSpec*)

Internal Check to compare against changes of the header files.

Parameters

Header	(should be AMMAR_SERVER_HTTP_HEADER_SPEC)

Return values

1=Success,0=Failure	

6.20.4.7 int AmmServer_DirectoryExists (char * *filename*)

Check if directory Exists.

Parameters

Path	to directory

Return values

1=Exists,0=Does	not Exist

Here is the call graph for this function:

6.20.4.8 int AmmServer_DoNOTCacheResource (struct **AmmServer_Instance** * *instance,* char * *resource_name*)

Set resource to no-cache mode , this means whoever asks for it will never get a cached response.

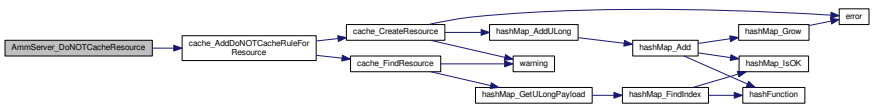
Parameters

Instance	of an AmmarServer
Resource	name that we want to always serve fresh

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.20.4.9 int AmmServer_DoNOTCacheResourceHandler (struct **AmmServer_Instance** * *instance,* struct **AmmServer_RH_Context** * *context*)

Set resource handler to no-cache mode , this means whoever asks for it will never get a cached response.

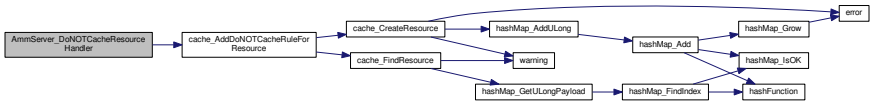
Parameters

Instance	of an AmmarServer
Resource	context that should always be served fresh (AmmServer_RH_Context)

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.20.4.10 int AmmServer_EraseFile (char * *filename*)

Erase a File.

Parameters

Path	to file

Return values

1=Success,0=Failure	

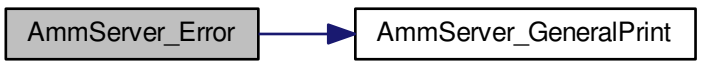
6.20.4.11 void AmmServer_Error (const char * *format,* ...)

Writes the C string pointed by format to stderr , as an error (Red) and logs it to the appropriate log If format includes format specifiers (subsequences beginning with %), the additional arguments following format are formatted and inserted in the resulting string replacing their respective specifiers.

Parameters

format,see	printf (`http://www.cplusplus.com/reference/cstdio/printf/`)
Arbitrary	number of other parameters that where defined in format

Here is the call graph for this function:



6.20.4.12 int AmmServer_ExecuteCommandLine (char * *command,* char * *what2GetBack,* unsigned int *what2GetBackMaxSize*)

Execute a command and copy its output to the provided buffer.

Parameters

Command	to execute
Allocated	memory to store the result
Size	of Allocated memory

Return values

1=Ok,0=Failed	

Bug Executing commands can be dangerous , always check and sanitize input before executing , Also be sure about the max size of output so that you don't lose a part of it , also make something like escapeshellcmd

6.20.4.13 int AmmServer_FileExists (char * *filename*)

Check if file Exists.

Parameters

Path	to file

Return values

1=Exists,0=Does	not Exist

Here is the call graph for this function:

6.20.4.14 int AmmServer_FILES (struct AmmServer_Instance * *instance,* struct AmmServer_DynamicRequest * *rqst,* char * *var_id_IN,* char * *var_value_OUT,* unsigned int *max_var_value_OUT*)

Access a FILE submitted by a dynamic requested.

Parameters

Instance	of an AmmarServer
Request	that contains the POST argument (see AmmServer_DynamicRequest)
Input	Name of argument we are looking for
Output	Pointer that will be copied with the value we were looking for
Maximum	Size for output Value

Return values

1=Success,0=Failure	

6.20.4.15 int AmmServer_GETArg (struct AmmServer_Instance * *instance,* struct AmmServer_DynamicRequest * *rqst,* char * *var_id_IN,* char * *var_value_OUT,* unsigned int *max_var_value_OUT*)

Get a GET argument.

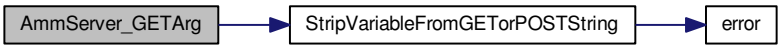
Parameters

Instance	of an AmmarServer
Request	that contains the POST argument (see AmmServer_DynamicRequest)
Input	Name of argument we are looking for
Output	Pointer that will be copied with the value we were looking for
Maximum	Size for output Value

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.20.4.16 int AmmServer_GetInfo (struct AmmServer_Instance * *instance,* unsigned int *info_type*)

Get an Integer out of the state of an instance , of course one can dive into the instance structure but this is a much more clean way to do this.

Parameters

An	AmmarServer Instance
An	ID about which info we want , see (AmmServInfos)

Return values

Value	of the integer we asked about

6.20.4.17 int AmmServer_GetIntSettingValue (struct AmmServer_Instance * *instance,* unsigned int *set_type*)

Get an Integer out of the state of an instance , of course one can dive into the instance structure but this is a much more clean way to do this.

Parameters

An	AmmarServer Instance
An	ID about which integer info we want , see (AmmServSettings)

Return values

Value	of the integer we asked about

6.20.4.18 char* AmmServer_GetStrSettingValue (struct AmmServer_Instance * *instance,* unsigned int *set_type*)

Get a String out of the state of an instance , of course one can dive into the instance structure but this is a much more clean way to do this.

Parameters

An	AmmarServer Instance
An	ID about which string info we want , see (AmmServStrSettings)

Return values

Value	of the string we asked about

6.20.4.19 int AmmServer_POSTArg (struct AmmServer_Instance * *instance,* struct AmmServer_DynamicRequest * *rqst,* char * *var_id_IN,* char * *var_value_OUT,* unsigned int *max_var_value_OUT*)

Get a POST argument.

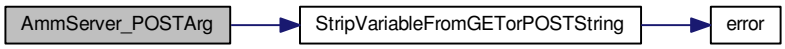
Parameters

Instance	of an AmmarServer
Request	that contains the POST argument (see AmmServer_DynamicRequest)
Input	Name of argument we are looking for
Output	Pointer that will be copied with the value we were looking for
Maximum	Size for output Value

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.20.4.20 char* AmmServer_ReadFileToMemory (char * *filename,* unsigned int * *length*)

Read a file and store it to a freshly allocated memory block.

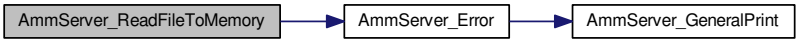
Parameters

Input	Filename
Output	Maximum Size

Return values

Pointer	to the new memory or 0=Failed

Here is the call graph for this function:



6.20.4.21 int AmmServer_RegisterTerminationSignal (void * *callback*)

Register a function to call a function that gracefully terminates a client when a SIGKILL or the time to stop the server comes.

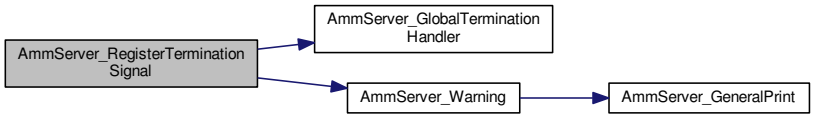
Parameters

Pointer	to function

Return values

1=Exists,0=Does	not Exist

Here is the call graph for this function:



6.20.4.22 int AmmServer_RemoveResourceHandler (struct AmmServer_Instance * *instance,* struct AmmServer_RH_Context * *context,* unsigned char *free_mem*)

Remove a request handler that hanles dynamic requests.

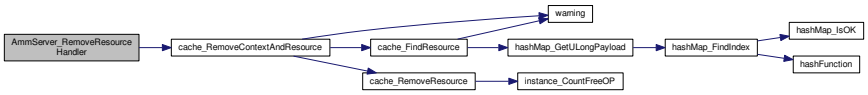
Parameters

An	AmmarServer Instance
An	AmmServer_RH_Context to be freed
Switch	to control freeing memory or not for this context (typically should be set to 1 except one knows what he is trying to do)

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.20.4.23 int AmmServer_ReplaceVarInMemoryFile (char * *page,* unsigned int *pageLength,* char * *var,* char * *value*)

Hot-Replace a variable inside a memory block , typically used to replace placeholders inside text files , like $$$$$$$$NAME$$$$$$$$, the value should be smaller or equal to the var beeing replaced.

Parameters

Pointer	to memory that contains the document
Size	of document
Variable	to be replaced

What	to replace it with

Return values

1=Ok,0=Failed	

Bug Value should not be bigger than variable otherwise things won't fit in the same memory block

6.20.4.24 int AmmServer_Running (struct AmmServer_Instance * *instance*)

Query if an instance of AmmarServer is initialized and running.

Parameters

An	AmmarServer Instance

Return values

1=Running,0=Stopped	

Here is the call graph for this function:

6.20.4.25 int AmmServer_SaveDynamicRequest (char * *filename,* struct AmmServer_Instance * *instance,* struct AmmServer_DynamicRequest * *rqst*)

Save Dynamic Request to file.

Parameters

Filename	to save the dynamic request
Instance	of an AmmarServer
Request	that we want to save to a file (see AmmServer_DynamicRequest)

Return values

1=Success,0=Failure	

Here is the call graph for this function:

6.20.4.26 int AmmServer_SelfCheck (struct AmmServer_Instance * *instance*)

Perform a sanity check on the instance of AmmarServer , this is mostly a dev debug tool and an entry point for code inside AmmServerlib.

Parameters

Ammar	Server Instance

Return values

1=Ok,0=Failed	

Bug Maybe remove AmmServer_SelfCheck

6.20.4.27 int AmmServer_SetIntSettingValue (struct AmmServer_Instance * *instance,* unsigned int *set_type,* int *set_value*)

Set an Integer inside the state of an instance , of course one can dive into the instance structure but this is a much more clean way to do this.

Parameters

An	AmmarServer Instance
An	ID about which integer info we want , see (AmmServSettings)
New	value to set

Return values

Value	of the integer we asked about

6.20.4.28 int AmmServer_SetStrSettingValue (struct AmmServer_Instance * *instance,* unsigned int *set_type,* char * *set_value*)

Set an string inside the state of an instance , of course one can dive into the instance structure but this is a much more clean way to do this.

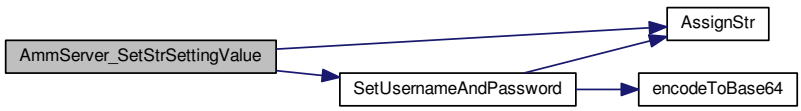
Parameters

An	AmmarServer Instance
An	ID about which integer info we want , see (AmmServStrSettings)
New	string value to set

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.20.4.29 int AmmServer_SignalCountAsBadClientBehaviour (struct AmmServer_Instance * *instance,* struct AmmServer_DynamicRequest * *rqst*)

Staged way to easily handle bad clients etc from the clients , currently a stub..!

Bug Client behaviours etc are not implemented yet

6.20.4.30 struct AmmServer_Instance* AmmServer_Start (char * *name,* char * *ip,* unsigned int *port,* char * *conf_file,* char * *web_root_path,* char * *templates_root_path*)

Start a Web Server , allocate memory , bind ports and return its instance..

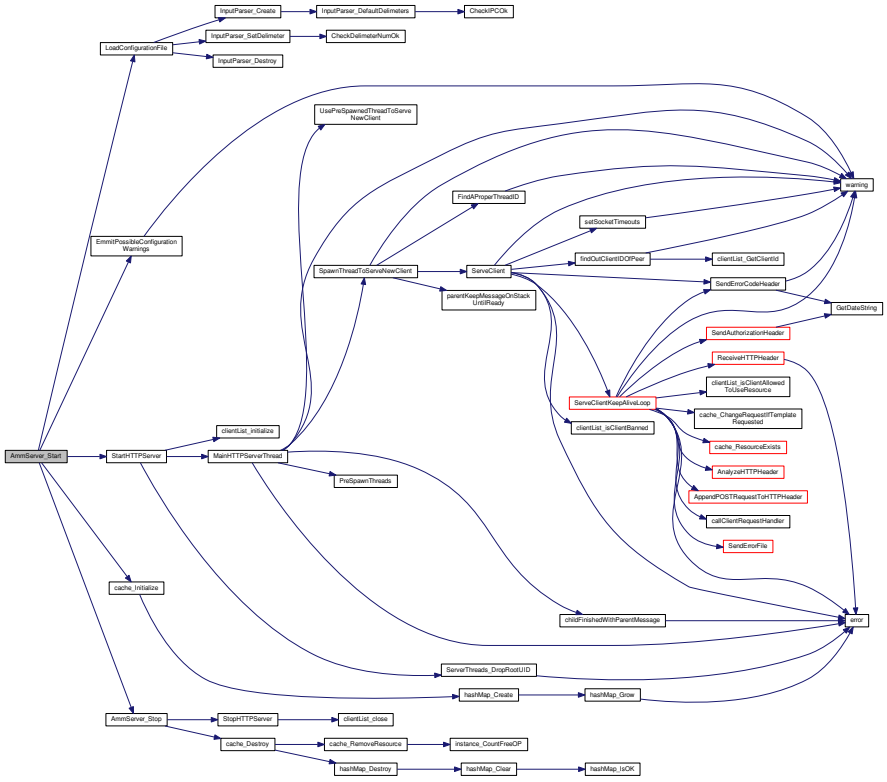
Parameters

String	containing the name of this Server
String	containing the IP to be binded (0.0.0.0 , for all interfaces)
Port	to use , ports under 1000 require superuser privileges
String	with the filename of a configuration file
String	with the root public_html directory , all directories that are childs of this dir could be visible
String	with the root directory for templates (custom 404 pages etc)

Return values

An	Ammar Server instance or 0=Failure

Here is the call graph for this function:



6.20.4.31 struct AmmServer_Instance* AmmServer_StartAdminInstance (char * *ip,* unsigned int *port*)

Planned functionality for a default http administrator panel per server per instance , currently not implemented correctly.

Parameters

IP	to bind the interface at
Port	to use

Return values

Value	of the integer we asked about

6.20.4.32 struct AmmServer_Instance* AmmServer_StartWithArgs (char * *name*, int *argc*, char ** *argv*, char * *ip*, unsigned int *port*, char * *conf_file*, char * *web_root_path*, char * *templates_root_path*)

Start a Web Server , allocate memory , bind ports and return its instance , also process arguments (argc and argv from int main(int argc, char *argv[])) ..

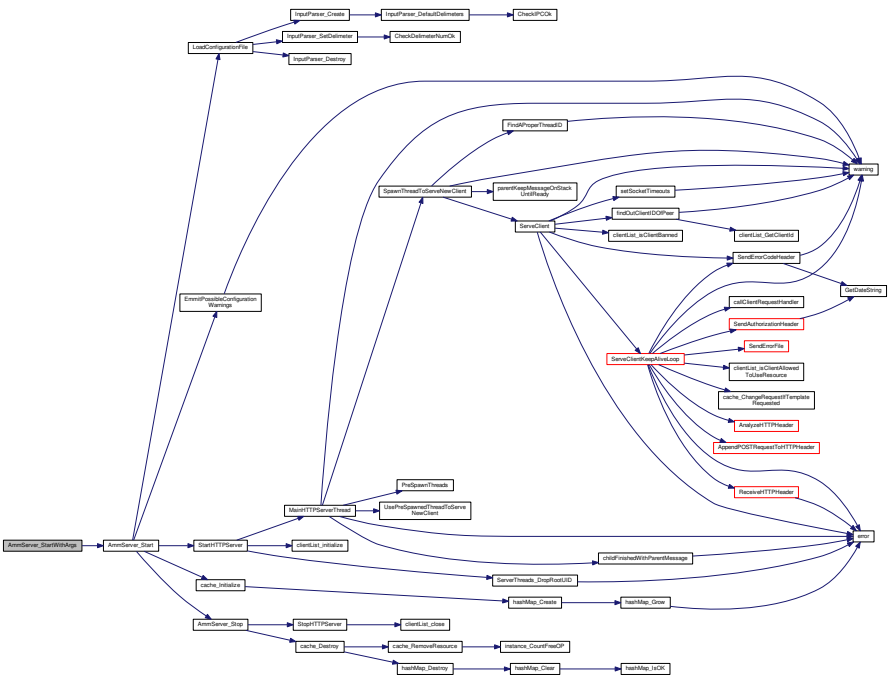
Parameters

String	containing the name of this Server
argc,number	of arguments
argv,array	of strings
String	containing the IP to be binded (0.0.0.0 , for all interfaces)
Port	to use , ports under 1000 require superuser privileges
String	with the filename of a configuration file
String	with the root public_html directory , all directories that are childs of this dir could be visible
String	with the root directory for templates (custom 404 pages etc)

Return values

An	Ammar Server instance or 0=Failure

Here is the call graph for this function:



6.20.4.33 int AmmServer_Stop (struct AmmServer_Instance * *instance*)

Stop a Web Server , deallocate memory , free ports and free the server instance..

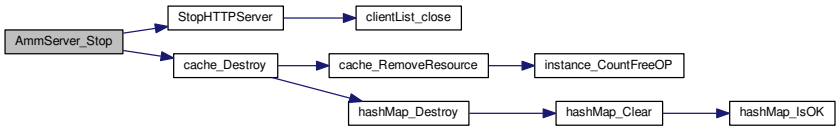
Parameters

An	AmmarServer Instance

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.20.4.34 unsigned int AmmServer_StringIsHTMLSafe (char * *str*)

Check if a string has html elements inside it , so if we append it to a web site we won't have html injected.

Parameters

Input	String

Return values

1=Safe,0=Unsafe	

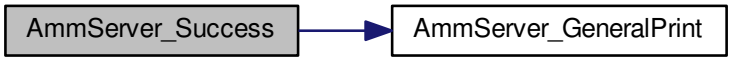
6.20.4.35 void AmmServer_Success (const char * *format,* ...)

Writes the C string pointed by format to stderr , as a success (Green) and logs it to the appropriate log If format includes format specifiers (subsequences beginning with %), the additional arguments following format are formatted and inserted in the resulting string replacing their respective specifiers.

Parameters

format,see	printf (http://www.cplusplus.com/reference/cstdio/printf/)
Arbitrary	number of other parameters that where defined in format

Here is the call graph for this function:



6.20.4.36 char* AmmServer_Version ( )

Returns a string with the version of AmmarServer , in case it returns NULL it means that we are linked to AmmarServerNULL which means a fake binary.

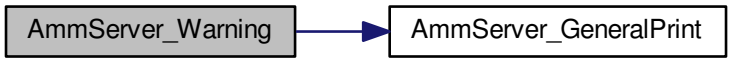
6.20.4.37 void AmmServer_Warning (const char * *format,* ...)

Writes the C string pointed by format to stderr , as a warning (Yellow) and logs it to the appropriate log If format includes format specifiers (subsequences beginning with %), the additional arguments following format are formatted and inserted in the resulting string replacing their respective specifiers.

Parameters

format,see	printf (http://www.cplusplus.com/reference/cstdio/printf/)
Arbitrary	number of other parameters that where defined in format

Here is the call graph for this function:



6.20.4.38 int AmmServer_WriteFileFromMemory (char * *filename,* char * *memory,* unsigned int *memoryLength*)

Dump a memory block to a file.

Parameters

Output	Filename
Input	Pointer to memory
Size	of memory block

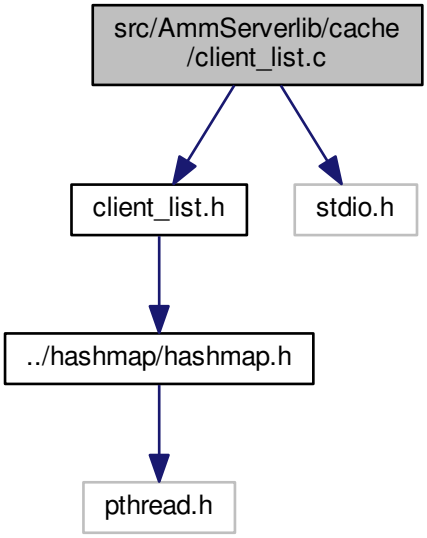
Return values

1=Ok,0=Failed	

6.21 src/AmmServerlib/cache/client_list.c File Reference

```
#include "client_list.h"
#include <stdio.h>
```

Include dependency graph for client_list.c:



Macros

- #define COMPILE_WITH_CLIENT_LIST 1

Functions

- unsigned int clientList_GetClientId (struct clientListContext *clientList, char *ip)

  Get the internal index id of an IP.
- int clientList_isClientBanned (struct clientListContext *clientList, clientID client_id)

  Check if client ID is banned , therefore we should deny all service to him.
- int clientList_isClientAllowedToUseResource (struct clientListContext *clientList, clientID client_id, char *resource)

  Ask if the client is allowed to use resource.
- int clientList_signalClientStoppedUsingResource (struct clientListContext *clientList, clientID client_id, char *resource)

  Signal that resource has stopped beeing used for internal statistics.
- struct clientListContext * clientList_initialize ()

  Create , allocate and return a client list.
- int clientList_close (struct clientListContext *clientList)

  Close and destroy client list.

6.21.1 Macro Definition Documentation

6.21.1.1 #define COMPILE_WITH_CLIENT_LIST 1

6.21.2 Function Documentation

6.21.2.1 int clientList_close (struct clientListContext * *clientList*)

Close and destroy client list.

Parameters

ClientList	to destroy

Return values

1=Success,0=Failure	

6.21.2.2 unsigned int clientList_GetClientId (struct clientListContext * *clientList,* char * *ip*)

Get the internal index id of an IP.

Parameters

ClientList	
String	containing the IP of the client we want to query

Return values

ID	of client we searched for

6.21.2.3 struct clientListContext* clientList_initialize ( )

Create , allocate and return a client list.

Return values

Pointer	to a freshly allocated client list or 0=Failure

6.21.2.4 int clientList_isClientAllowedToUseResource (struct clientListContext * *clientList,* clientID *client_id,* char * *resource*)

Ask if the client is allowed to use resource.

Parameters

ClientList	
ClientID	we are talking about
String	of the resource

Return values

1=Allowed,0=Denied	

6.21.2.5 int clientList_isClientBanned (struct clientListContext * *clientList,* clientID *client_id*)

Check if client ID is banned , therefore we should deny all service to him.

Parameters

ClientList	
ClientID	we are asking about

Return values

1=Banned,0=OK	

6.21.2.6 int clientList_signalClientStoppedUsingResource (struct clientListContext * *clientList,* clientID *client_id,* char * *resource*)

Signal that resource has stopped beeing used for internal statistics.

Parameters

ClientList	
ClientID	we are talking about
String	of the resource

Return values

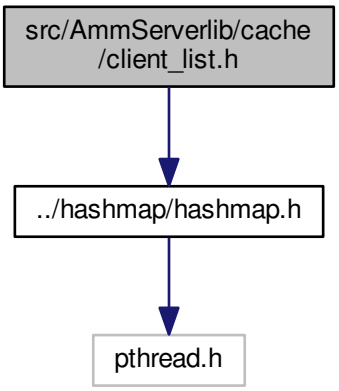
1=Ok,0=Failed	

6.22 src/AmmServerlib/cache/client_list.h File Reference

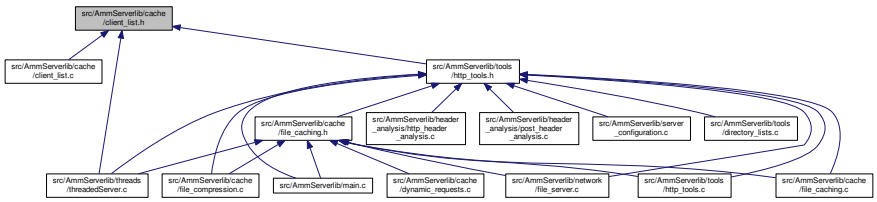
Client list for IPs that should also serve as a banlist manage QoS etc.

```
#include "../hashmap/hashmap.h"
```

Include dependency graph for client_list.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct clientListContext

  The client list is just a hashmap (see hashmap.h)

Typedefs

- typedef unsigned int clientID

  Typedef to make clientID stand out.

Functions

- unsigned int clientList_GetClientId (struct clientListContext ∗clientList, char ∗ip)

  Get the internal index id of an IP.
- int clientList_isClientBanned (struct clientListContext ∗clientList, clientID client_id)

  Check if client ID is banned , therefore we should deny all service to him.
- int clientList_isClientAllowedToUseResource (struct clientListContext ∗clientList, clientID client_id, char ∗resource)

  Ask if the client is allowed to use resource.
- int clientList_signalClientStoppedUsingResource (struct clientListContext ∗clientList, clientID client_id, char ∗resource)

  Signal that resource has stopped beeing used for internal statistics.
- struct clientListContext ∗ clientList_initialize ()

  Create , allocate and return a client list.
- int clientList_close (struct clientListContext ∗clientList)

  Close and destroy client list.

6.22.1 Detailed Description

Client list for IPs that should also serve as a banlist manage QoS etc.

Author

Ammar Qammaz (AmmarkoV)

Bug Client Lists are a stub and not implemented yet

6.22.2 Typedef Documentation

6.22.2.1 typedef unsigned int clientID

Typedef to make clientID stand out.

6.22.3 Function Documentation

6.22.3.1 int clientList_close (struct clientListContext ∗ *clientList*)

Close and destroy client list.

Parameters

ClientList	to destroy

Return values

1=Success,0=Failure	

6.22.3.2 unsigned int clientList_GetClientId (struct clientListContext * *clientList,* char * *ip*)

Get the internal index id of an IP.

Parameters

ClientList	
String	containing the IP of the client we want to query

Return values

ID	of client we searched for

6.22.3.3 struct clientListContext* clientList_initialize ( )

Create , allocate and return a client list.

Return values

Pointer	to a freshly allocated client list or 0=Failure

6.22.3.4 int clientList_isClientAllowedToUseResource (struct clientListContext * *clientList,* clientID *client_id,* char * *resource*)

Ask if the client is allowed to use resource.

Parameters

ClientList	
ClientID	we are talking about
String	of the resource

Return values

1=Allowed,0=Denied	

6.22.3.5 int clientList_isClientBanned (struct clientListContext * *clientList,* clientID *client_id*)

Check if client ID is banned , therefore we should deny all service to him.

Parameters

ClientList	
ClientID	we are asking about

Return values

1=Banned,0=OK	

6.22.3.6 int clientList_signalClientStoppedUsingResource (struct clientListContext * *clientList,* clientID *client_id,* char * *resource*)

Signal that resource has stopped beeing used for internal statistics.

Parameters

ClientList	
ClientID	we are talking about
String	of the resource

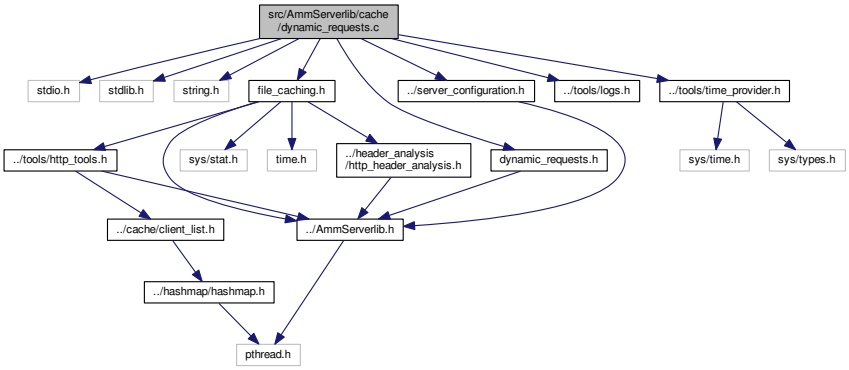
Return values

1=Ok,0=Failed	

6.23 src/AmmServerlib/cache/dynamic_requests.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "dynamic_requests.h"
#include "file_caching.h"
#include "../server_configuration.h"
#include "../tools/logs.h"
#include "../tools/time_provider.h"
```

Include dependency graph for dynamic_requests.c:



Functions

- int dynamicRequest_ContentAvailiable (struct AmmServer_Instance *instance, unsigned int index)

  Ask if dynamic content is available for this cache index.

- char * dynamicRequest_serveContent (struct AmmServer_Instance *instance, struct HTTPHeader *request, struct AmmServer_RH_Context *shared_context, unsigned int index, unsigned long *memSize, unsigned char *compressionSupported, unsigned char *freeContentAfterUsingIt)

  Handles and serves a dynamic request.

- int callClientRequestHandler (struct AmmServer_Instance *instance, struct HTTPHeader *output)

  Execute callback function associated with dynamic content , providing it with the http header it needs to output data to.
- int saveDynamicRequest (char *filename, struct AmmServer_Instance *instance, struct AmmServer_DynamicRequest *rqst)

  Save Dynamic request to a file (for debugging it)

6.23.1 Function Documentation

6.23.1.1 int callClientRequestHandler (struct AmmServer_Instance * *instance,* struct HTTPHeader * *output*)

Execute callback function associated with dynamic content , providing it with the http header it needs to output data to.

Parameters

An	AmmarServer Instance
HTTPHeader	containing the output of the request

Return values

1=Ok,0=Failed	

6.23.1.2 int dynamicRequest_ContentAvailiable (struct AmmServer_Instance * *instance,* unsigned int *index*)

Ask if dynamic content is available for this cache index.

Parameters

An	AmmarServer Instance
Index	of cache we want to ask for

Return values

1=Availiable,0=Not	Availiable

6.23.1.3 char* dynamicRequest_serveContent (struct AmmServer_Instance * *instance,* struct HTTPHeader * *request,* struct AmmServer_RH_Context * *shared_context,* unsigned int *index,* unsigned long * *memSize,* unsigned char * *compressionSupported,* unsigned char * *freeContentAfterUsingIt*)

Handles and serves a dynamic request.

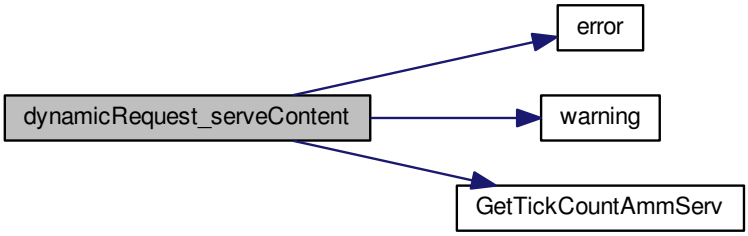
Parameters

An	AmmarServer Instance
HTTPHeader	containing the request done
Resource	Context for specific dynamic Request
Index	of cache item , containing this dynamic request
Memory	Size allocated by the new dynamic request
Outputs	if compression was supported (and used) by client
Outputs	if client wants to free buffer on it's own or it should be handled automatically

Return values

Pointer	To New Content or ,0=Failed

Here is the call graph for this function:



6.23.1.4 int saveDynamicRequest (char * *filename,* struct **AmmServer_Instance** * *instance,* struct **AmmServer_DynamicRequest** * *rqst*)

Save Dynamic request to a file (for debugging it)

Parameters

ClientList	
ClientID	we are talking about
String	of the resource

Return values

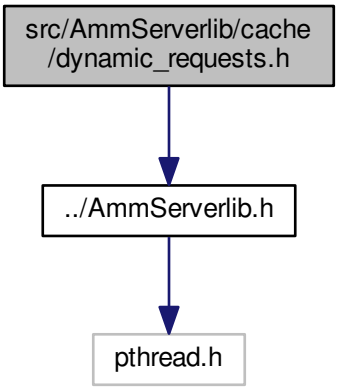
1=Ok,0=Failed	

6.24 src/AmmServerlib/cache/dynamic_requests.h File Reference

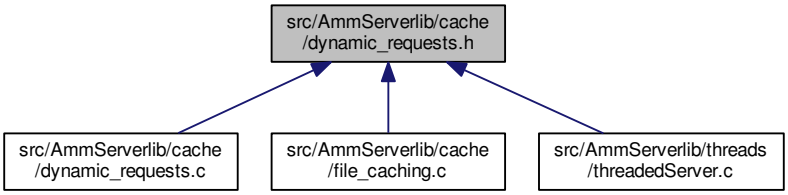
Dynamic request handler , one of the most important parts of this library.

```
#include "../AmmServerlib.h"
```

Include dependency graph for dynamic_requests.h:



This graph shows which files directly or indirectly include this file:



Functions

- int dynamicRequest_ContentAvailiable (struct AmmServer_Instance *instance, unsigned int index)

  Ask if dynamic content is available for this cache index.

- char * dynamicRequest_serveContent (struct AmmServer_Instance *instance, struct HTTPHeader *request, struct AmmServer_RH_Context *shared_context, unsigned int index, unsigned long *memSize, unsigned char *compressionSupported, unsigned char *freeContentAfterUsingIt)

  Handles and serves a dynamic request.

- int callClientRequestHandler (struct AmmServer_Instance *instance, struct HTTPHeader *output)

  Execute callback function associated with dynamic content , providing it with the http header it needs to output data to.

- int saveDynamicRequest (char *filename, struct AmmServer_Instance *instance, struct AmmServer_-DynamicRequest *rqst)

  Save Dynamic request to a file (for debugging it)

6.24.1 Detailed Description

Dynamic request handler , one of the most important parts of this library.

Author

Ammar Qammaz (AmmarkoV)

Bug Compression should be improved

6.24.2 Function Documentation

6.24.2.1 int callClientRequestHandler (struct AmmServer_Instance * *instance,* struct HTTPHeader * *output*)

Execute callback function associated with dynamic content , providing it with the http header it needs to output data to.

Parameters

An	AmmarServer Instance
HTTPHeader	containing the output of the request

Return values

1=Ok,0=Failed	

6.24.2.2 int dynamicRequest_ContentAvailiable (struct AmmServer_Instance * *instance,* unsigned int *index*)

Ask if dynamic content is available for this cache index.

Parameters

An	AmmarServer Instance
Index	of cache we want to ask for

Return values

1=Availiable,0=Not	Availiable

6.24.2.3 char* dynamicRequest_serveContent (struct AmmServer_Instance * *instance,* struct HTTPHeader * *request,* struct AmmServer_RH_Context * *shared_context,* unsigned int *index,* unsigned long * *memSize,* unsigned char * *compressionSupported,* unsigned char * *freeContentAfterUsingIt*)

Handles and serves a dynamic request.

Parameters

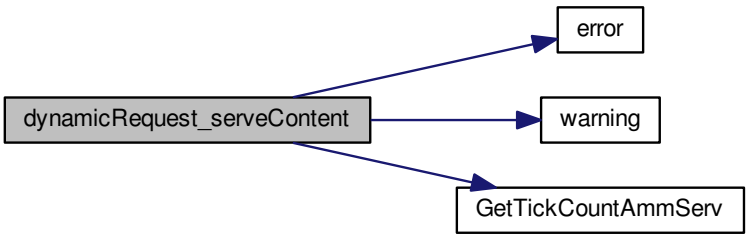
An	AmmarServer Instance
HTTPHeader	containing the request done
Resource	Context for specific dynamic Request
Index	of cache item , containing this dynamic request
Memory	Size allocated by the new dynamic request
Outputs	if compression was supported (and used) by client

Outputs	if client wants to free buffer on it's own or it should be handled automatically

Return values

Pointer	To New Content or ,0=Failed

Here is the call graph for this function:



6.24.2.4 int saveDynamicRequest (char * *filename*, struct AmmServer_Instance * *instance*, struct AmmServer_DynamicRequest * *rqst*)

Save Dynamic request to a file (for debugging it)

Parameters

ClientList	
ClientID	we are talking about
String	of the resource

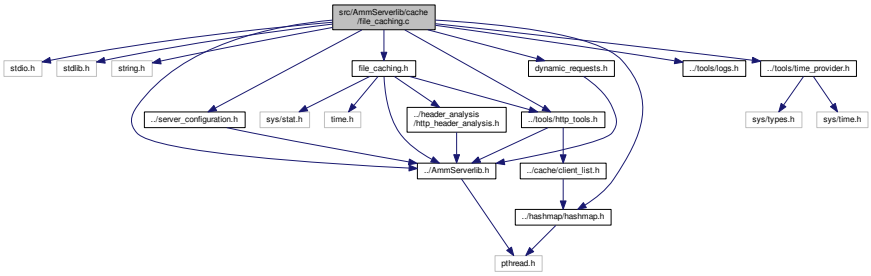
Return values

1=Ok,0=Failed	

6.25 src/AmmServerlib/cache/file_caching.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "../AmmServerlib.h"
#include "../server_configuration.h"
#include "file_caching.h"
#include "../tools/logs.h"
#include "../tools/http_tools.h"
#include "../tools/time_provider.h"
#include "../hashmap/hashmap.h"
#include "dynamic_requests.h"
```

Include dependency graph for file_caching.c:



Functions

- int cache_ChangeRequestIfTemplateRequested (struct AmmServer_Instance *instance, char *request, char *templates_root)

  The role of request caching is to intercept incoming requests and if they are referring to an internal resource using the TemplatesInternalURI URI we want to redirect the request to our templates folder ..! If the request was indeed a change request returns 1 else 0.
- int freeMallocIfNeeded (char *mem, unsigned char free_is_needed)

  Tool to check if a malloc'ed chunk of memory should be freed.
- int cache_Initialize (struct AmmServer_Instance *instance, unsigned int max_seperate_items, unsigned int max_total_allocation_MB, unsigned int max_allocation_per_entry_MB)

  Allocate and create a new empty cache.
- int cache_Destroy (struct AmmServer_Instance *instance)

  Deallocate and destroy the cache of an AmmarServer instance.
- unsigned int cache_FindResource (struct AmmServer_Instance *instance, char *resource, unsigned int *index)

  Query for a resource , and return its index.
- int cache_CreateResource (struct AmmServer_Instance *instance, char *resource, unsigned int *index)
- int cache_DestroyResource (unsigned int *index)
- int cache_LoadResourceFromDisk (struct AmmServer_Instance *instance, char *filename, unsigned int *index)
- int cache_AddFile (struct AmmServer_Instance *instance, char *filename, unsigned int *index, struct stat *last_modification)

  Add a filesystem file to cache.
- int cache_AddMemoryBlock (struct AmmServer_Instance *instance, struct AmmServer_RH_Context *context)

  Add a memory block to cache.
- int cache_AddDoNOTCacheRuleForResource (struct AmmServer_Instance *instance, char *filename)

  Create a rule for specific resource so that it will always be served fresh and not cached.
- int cache_RemoveResource (struct AmmServer_Instance *instance, unsigned int index)
- int cache_RemoveContextAndResource (struct AmmServer_Instance *instance, struct AmmServer_RH_-Context *context, unsigned char free_mem)

  Destroy Cache entry and resource context.
- unsigned long cache_GetHashOfResource (struct AmmServer_Instance *instance, unsigned int index)

  Get Hash Value of resource (to be used for E-Tags `http://en.wikipedia.org/wiki/HTTP_ETag` *or whatever other reason)*
- int cache_ResourceExists (struct AmmServer_Instance *instance, char *verified_filename, unsigned int *index)

  Query for a resource , and return its index.

- char * cache_GetResource (struct AmmServer_Instance *instance, struct HTTPHeader *request, unsigned int resourceCacheID, char *verified_filename, unsigned int *index, unsigned long *filesize, struct stat *last_-modification, unsigned char *compressionSupported, unsigned char *freeContentAfterUsingIt)

Get a resource to be served to a client . This call will try to find if it is already used , if it exists on disk , if it is a dynamic request etc , and return the specified buffer.

6.25.1 Function Documentation

6.25.1.1 int cache_AddDoNOTCacheRuleForResource (struct AmmServer_Instance * *instance,* char * *filename*)

Create a rule for specific resource so that it will always be served fresh and not cached.

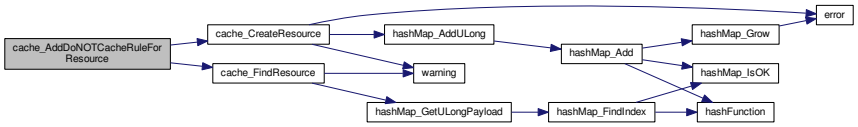
Parameters

An	AmmarServer Instance
Resource	filename

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.25.1.2 int cache_AddFile (struct AmmServer_Instance * *instance,* char * *filename,* unsigned int * *index,* struct stat * *last_modification*)

Add a filesystem file to cache.

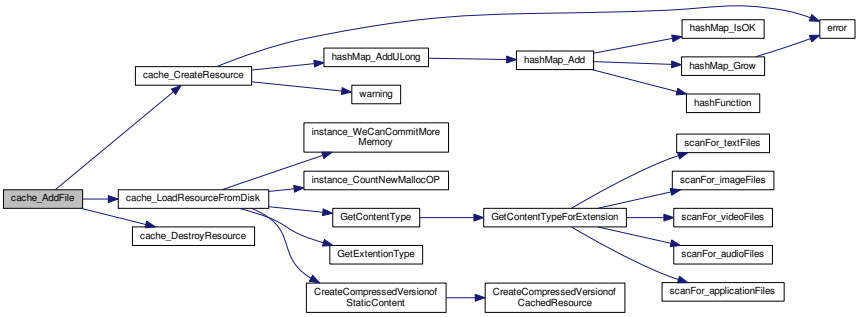
Parameters

An	AmmarServer Instance
Filename	pointing to the file to be added to cache
Output	index number of cache item
Output	time of last modification

Return values

1=Found,0=Failed	

Here is the call graph for this function:



6.25.1.3 int cache_AddMemoryBlock (struct AmmServer_Instance * *instance,* struct AmmServer_RH_Context * *context*)

Add a memory block to cache.

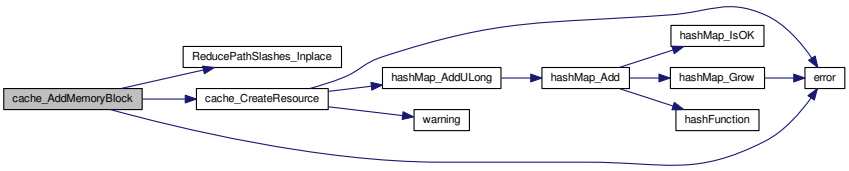
Parameters

An	AmmarServer Instance
Dynamic	Request to be added

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.25.1.4 int cache_ChangeRequestIfTemplateRequested (struct AmmServer_Instance * *instance,* char * *request,* char * *templates_root*)

The role of request caching is to intercept incoming requests and if they are referring to an internal resource using the TemplatesInternalURI URI we want to redirect the request to our templates folder ..! If the request was indeed a change request returns 1 else 0.

Parameters

An	AmmarServer Instance
String	of Request

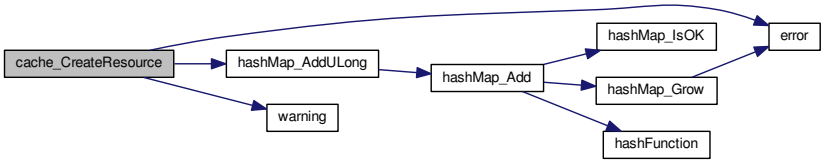
Filename	pointing to directory that contains templates

Return values

1=If	request was for a template and it got changed ,0= not changed request

6.25.1.5 int cache_CreateResource (struct AmmServer_Instance * *instance*, char * *resource*, unsigned int * *index*)

Here is the call graph for this function:



6.25.1.6 int cache_Destroy (struct AmmServer_Instance * *instance*)

Deallocate and destroy the cache of an AmmarServer instance.

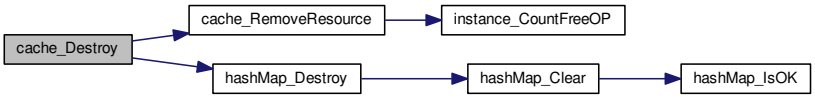
Parameters

An	AmmarServer Instance

Return values

1=Ok,0=Failed	

Here is the call graph for this function:



6.25.1.7 int cache_DestroyResource (unsigned int * *index*)

6.25.1.8 unsigned int cache_FindResource (struct AmmServer_Instance * *instance*, char * *resource*, unsigned int * *index*)

Query for a resource , and return its index.

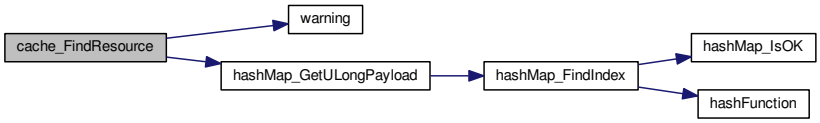
Parameters

An	AmmarServer Instance
Resource	we are searching for
Output	Index number

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.25.1.9 unsigned long cache_GetHashOfResource (struct AmmServer_Instance * *instance,* unsigned int *index*)

Get Hash Value of resource (to be used for E-Tags http://en.wikipedia.org/wiki/HTTP_ETag or whatever other reason)

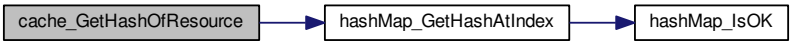
Parameters

An	AmmarServer Instance
Index	to the cache item requested

Return values

Hash	Value for Index specified , 0 = failure

Here is the call graph for this function:



6.25.1.10 char* cache_GetResource (struct AmmServer_Instance * *instance,* struct HTTPHeader * *request,* unsigned int *resourceCacheID,* char * *verified_filename,* unsigned int * *index,* unsigned long * *filesize,* struct stat * *last_modification,* unsigned char * *compressionSupported,* unsigned char * *freeContentAfterUsingIt*)

Get a resource to be served to a client . This call will try to find if it is already used , if it exists on disk , if it is a dynamic request etc , and return the specified buffer.

Parameters

An	AmmarServer Instance

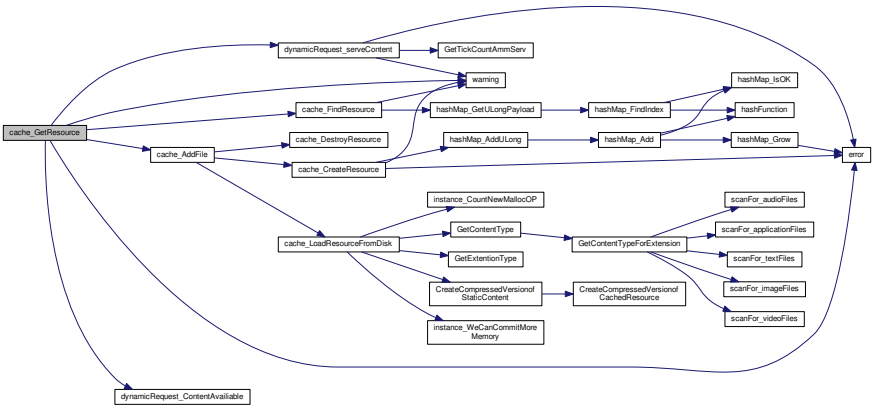
HTTPHeader	of request we are trying to service with the resource
cacheID	for resource
Filename	of the resource , this should be verified so that it doesn't access the whole filesystem but only subdirectories of the root public_html dir , and we consider this safe
Output	Index number of cache item we requested
Output	FileSize of cache item we requested
Output	last modification time of cache item we requested
Output	flag about whether the buffer returned is compressed or not
Output	flag about whether it is safe to automatically free the resource after using it , or there is an automatic handling of memory for the specific item

Bug If verified_filename , is not really verified (i.e. outside of the public_html root directory , this function could pose a security problem , since it will just blindly open and serve the filename given to it)

Return values

1=Ok,0=Failed	

Here is the call graph for this function:



6.25.1.11 int cache_Initialize (struct AmmServer_Instance * *instance,* unsigned int *max_seperate_items,* unsigned int *max_total_allocation_MB,* unsigned int *max_allocation_per_entry_MB*)

Allocate and create a new empty cache.

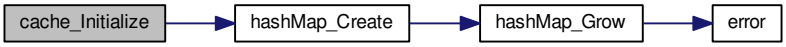
Parameters

An	AmmarServer Instance
Maximum	Number of separate items
Maximum	memory usage (Megabytes) for this entire cache
Maximum	memory usage (Megabytes) for a specific entry of the cache

Return values

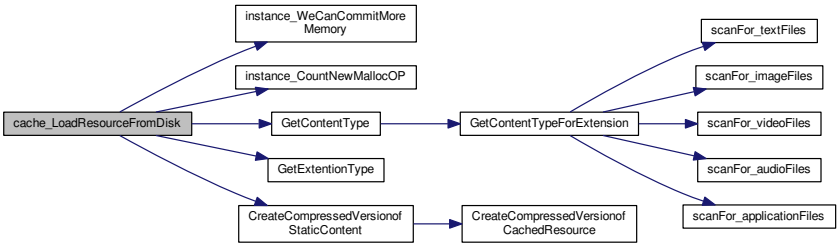
1=Ok,0=Failed	

Here is the call graph for this function:



6.25.1.12 int cache_LoadResourceFromDisk (struct AmmServer_Instance * *instance,* char * *filename,* unsigned int * *index*)

Here is the call graph for this function:



6.25.1.13 int cache_RemoveContextAndResource (struct AmmServer_Instance * *instance,* struct AmmServer_RH_Context * *context,* unsigned char *free_mem*)

Destroy Cache entry and resource context.

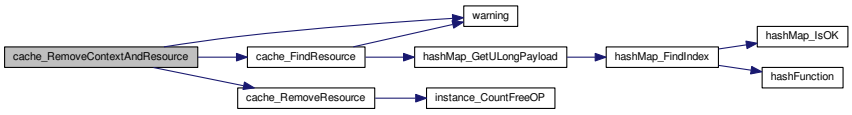
Parameters

An	AmmarServer Instance
Resource	Context to be removed
Flag	controlling whether memory should be freed

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.25.1.14 int cache_RemoveResource (struct AmmServer_Instance * *instance,* unsigned int *index*)

Here is the call graph for this function:

6.25.1.15 int cache_ResourceExists (struct AmmServer_Instance * *instance,* char * *verified_filename,* unsigned int * *index*)

Query for a resource , and return its index.

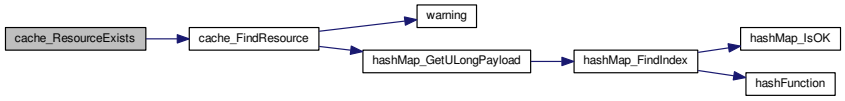
Parameters

An	AmmarServer Instance
Resource	we are searching for
Output	Index number

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.25.1.16 int freeMallocIfNeeded (char * *mem,* unsigned char *free_is_needed*)

Tool to check if a malloc'ed chunk of memory should be freed.

Parameters

Pointer	to memory
Flag	that signals if the pointer should be freed or not

Return values

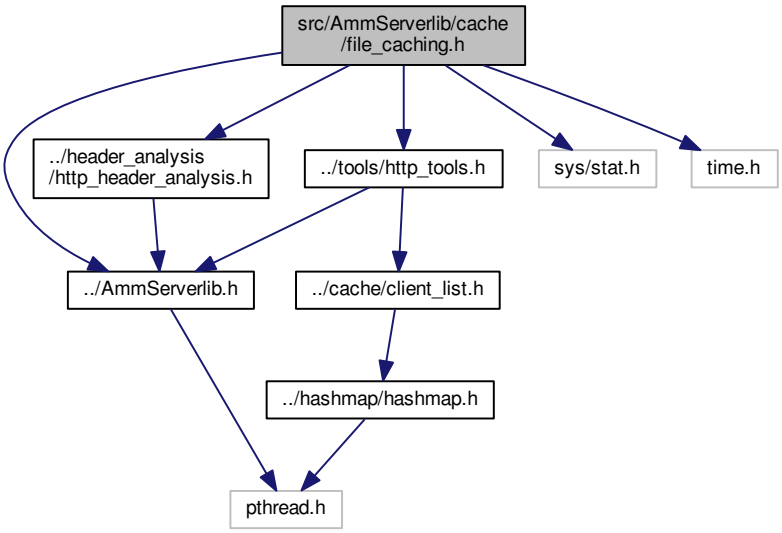
1=Ok,0=Failed	

6.26 src/AmmServerlib/cache/file_caching.h File Reference

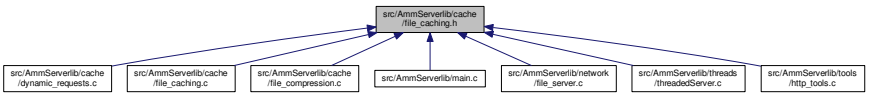
Central cache of AmmarServer , it reads/indexes and swaps resources asked by clients for fast performance.

```
#include "../AmmServerlib.h"
#include "../header_analysis/http_header_analysis.h"
#include "../tools/http_tools.h"
#include <sys/stat.h>
#include <time.h>
```

Include dependency graph for file_caching.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct timestamp

  Timestamp for a cache item entry.
- struct cache_item

  A cache item and all it's contents.

Functions

- int freeMallocIfNeeded (char *mem, unsigned char free_is_needed)

  Tool to check if a malloc'ed chunk of memory should be freed.
- int cache_ChangeRequestIfTemplateRequested (struct AmmServer_Instance *instance, char *request, char *templates_root)

  The role of request caching is to intercept incoming requests and if they are referring to an internal resource using the TemplatesInternalURI URI we want to redirect the request to our templates folder ..! If the request was indeed a change request returns 1 else 0.

- int cache_AddFile (struct AmmServer_Instance *instance, char *filename, unsigned int *index, struct stat *last_modification)

  Add a filesystem file to cache.
- int cache_AddMemoryBlock (struct AmmServer_Instance *instance, struct AmmServer_RH_Context *context)

  Add a memory block to cache.
- int cache_AddDoNOTCacheRuleForResource (struct AmmServer_Instance *instance, char *filename)

  Create a rule for specific resource so that it will always be served fresh and not cached.
- int cache_RemoveContextAndResource (struct AmmServer_Instance *instance, struct AmmServer_RH_Context *context, unsigned char free_mem)

  Destroy Cache entry and resource context.
- unsigned long cache_GetHashOfResource (struct AmmServer_Instance *instance, unsigned int index)

  Get Hash Value of resource (to be used for E-Tags http://en.wikipedia.org/wiki/HTTP_ETag *or whatever other reason)*
- unsigned int cache_FindResource (struct AmmServer_Instance *instance, char *resource, unsigned int *index)

  Query for a resource , and return its index.
- int cache_ResourceExists (struct AmmServer_Instance *instance, char *verified_filename, unsigned int *index)

  Query for a resource , and return its index.
- int cache_Initialize (struct AmmServer_Instance *instance, unsigned int max_seperate_items, unsigned int max_total_allocation_MB, unsigned int max_allocation_per_entry_MB)

  Allocate and create a new empty cache.
- int cache_Destroy (struct AmmServer_Instance *instance)

  Deallocate and destroy the cache of an AmmarServer instance.
- char * cache_GetResource (struct AmmServer_Instance *instance, struct HTTPHeader *request, unsigned int resourceCacheID, char *verified_filename, unsigned int *index, unsigned long *filesize, struct stat *last_modification, unsigned char *compressionSupported, unsigned char *freeContentAfterUsingIt)

  Get a resource to be served to a client . This call will try to find if it is already used , if it exists on disk , if it is a dynamic request etc , and return the specified buffer.

6.26.1 Detailed Description

Central cache of AmmarServer , it reads/indexes and swaps resources asked by clients for fast performance.

Author

Ammar Qammaz (AmmarkoV)

Bug File caching relies on hashmap for storing data , so it relies on optimizations done there for seek time optimization , other than that there needs to be a clean-up and code quality improvement

6.26.2 Function Documentation

6.26.2.1 int cache_AddDoNOTCacheRuleForResource (struct AmmServer_Instance * *instance,* char * *filename*)

Create a rule for specific resource so that it will always be served fresh and not cached.

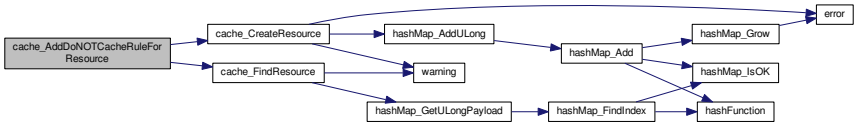
Parameters

An	AmmarServer Instance
Resource	filename

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.26.2.2 int cache_AddFile (struct AmmServer_Instance * *instance*, char * *filename*, unsigned int * *index*, struct stat * *last_modification*)

Add a filesystem file to cache.

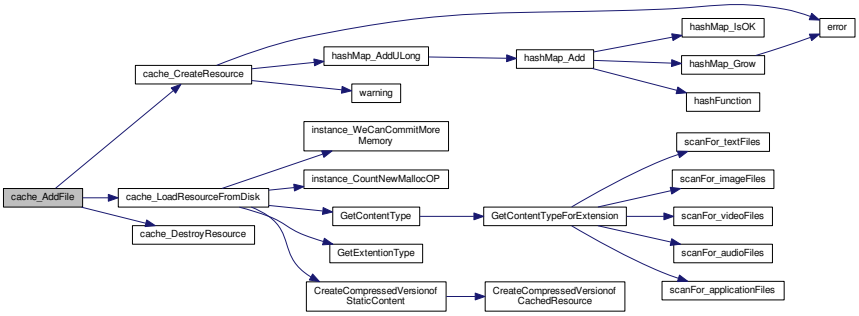
Parameters

An	AmmarServer Instance
Filename	pointing to the file to be added to cache
Output	index number of cache item
Output	time of last modification

Return values

1=Found,0=Failed	

Here is the call graph for this function:



6.26.2.3 int cache_AddMemoryBlock (struct AmmServer_Instance * *instance*, struct AmmServer_RH_Context * *context*)

Add a memory block to cache.

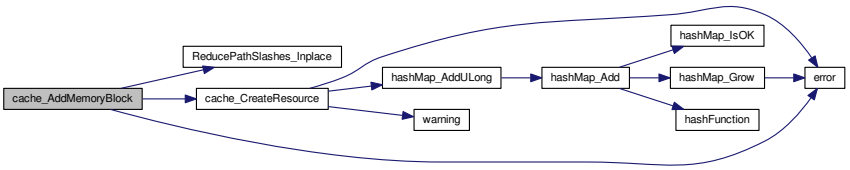
Parameters

An	AmmarServer Instance
Dynamic	Request to be added

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.26.2.4 int cache_ChangeRequestIfTemplateRequested (struct AmmServer_Instance * *instance*, char * *request*, char * *templates_root*)

The role of request caching is to intercept incoming requests and if they are referring to an internal resource using the TemplatesInternalURI URI we want to redirect the request to our templates folder ..! If the request was indeed a change request returns 1 else 0.

Parameters

An	AmmarServer Instance
String	of Request
Filename	pointing to directory that contains templates

Return values

1=If	request was for a template and it got changed ,0= not changed request

6.26.2.5 int cache_Destroy (struct AmmServer_Instance * *instance*)

Deallocate and destroy the cache of an AmmarServer instance.

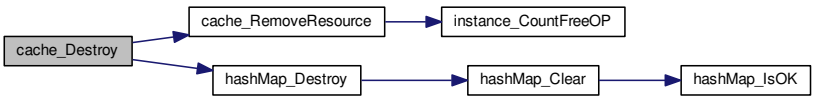
Parameters

An	AmmarServer Instance

Return values

1=Ok,0=Failed	

Here is the call graph for this function:



6.26.2.6 unsigned int cache_FindResource (struct AmmServer_Instance * *instance*, char * *resource*, unsigned int * *index*)

Query for a resource , and return its index.

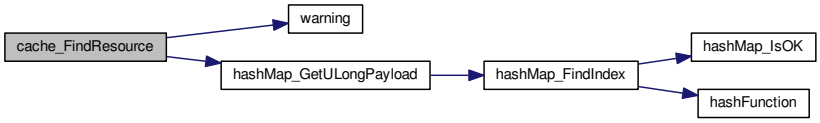
Parameters

An	AmmarServer Instance
Resource	we are searching for
Output	Index number

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.26.2.7 unsigned long cache_GetHashOfResource (struct AmmServer_Instance * *instance*, unsigned int *index*)

Get Hash Value of resource (to be used for E-Tags http://en.wikipedia.org/wiki/HTTP_ETag or whatever other reason)

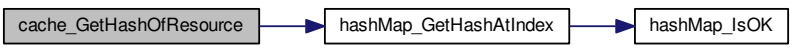
Parameters

An	AmmarServer Instance
Index	to the cache item requested

Return values

Hash	Value for Index specified , 0 = failure

Here is the call graph for this function:



6.26.2.8 char* cache_GetResource (struct AmmServer_Instance * *instance*, struct HTTPHeader * *request*, unsigned int *resourceCacheID*, char * *verified_filename*, unsigned int * *index*, unsigned long * *filesize*, struct stat * *last_modification*, unsigned char * *compressionSupported*, unsigned char * *freeContentAfterUsingIt*)

Get a resource to be served to a client . This call will try to find if it is already used , if it exists on disk , if it is a dynamic request etc , and return the specified buffer.

Parameters

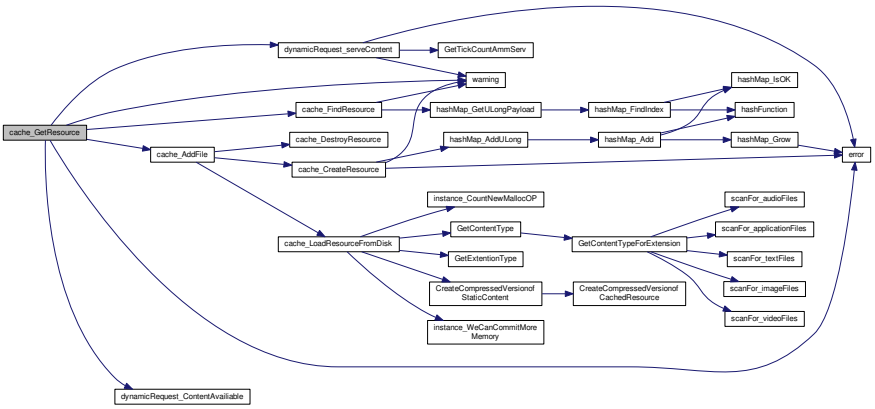
An	AmmarServer Instance
HTTPHeader	of request we are trying to service with the resource
cacheID	for resource
Filename	of the resource , this should be verified so that it doesn't access the whole filesystem but only subdirectories of the root public_html dir , and we consider this safe
Output	Index number of cache item we requested
Output	FileSize of cache item we requested
Output	last modification time of cache item we requested
Output	flag about whether the buffer returned is compressed or not
Output	flag about whether it is safe to automatically free the resource after using it , or there is an automatic handling of memory for the specific item

Bug If verified_filename , is not really verified (i.e. outside of the public_html root directory , this function could pose a security problem , since it will just blindly open and serve the filename given to it)

Return values

1=Ok,0=Failed	

Here is the call graph for this function:



6.26.2.9 int cache_Initialize (struct AmmServer_Instance * *instance*, unsigned int *max_seperate_items*, unsigned int *max_total_allocation_MB*, unsigned int *max_allocation_per_entry_MB*)

Allocate and create a new empty cache.

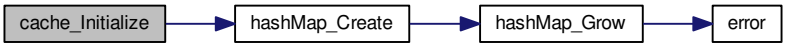
Parameters

An	AmmarServer Instance
Maximum	Number of separate items
Maximum	memory usage (Megabytes) for this entire cache
Maximum	memory usage (Megabytes) for a specific entry of the cache

Return values

1=Ok,0=Failed	

Here is the call graph for this function:



6.26.2.10 int cache_RemoveContextAndResource (struct AmmServer_Instance * *instance,* struct AmmServer_RH_Context * *context,* unsigned char *free_mem*)

Destroy Cache entry and resource context.

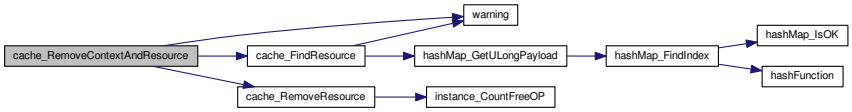
Parameters

An	AmmarServer Instance
Resource	Context to be removed
Flag	controlling whether memory should be freed

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.26.2.11 int cache_ResourceExists (struct AmmServer_Instance * *instance,* char * *verified_filename,* unsigned int * *index*)

Query for a resource , and return its index.

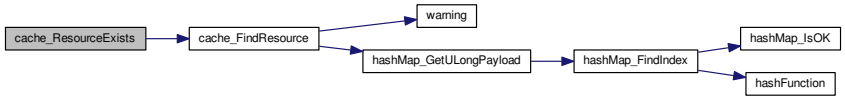
Parameters

An	AmmarServer Instance
Resource	we are searching for
Output	Index number

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.26.2.12 int freeMallocIfNeeded (char * *mem*, unsigned char *free_is_needed*)

Tool to check if a malloc'ed chunk of memory should be freed.

Parameters

Pointer	to memory
Flag	that signals if the pointer should be freed or not

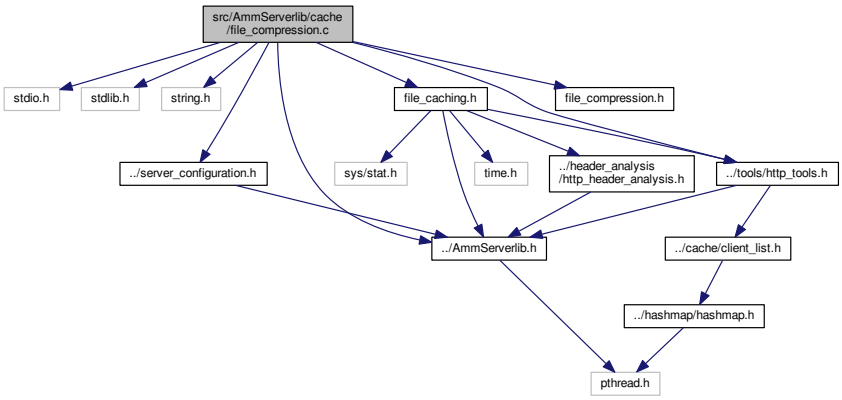
Return values

1=Ok,0=Failed	

6.27 src/AmmServerlib/cache/file_compression.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "../AmmServerlib.h"
#include "../server_configuration.h"
#include "../tools/http_tools.h"
#include "file_caching.h"
#include "file_compression.h"
```

Include dependency graph for file_compression.c:



Functions

- int CreateCompressedVersionofCachedResource (struct AmmServer_Instance *instance, unsigned int index, int compression_level)
- int CreateCompressedVersionofDynamicContent (struct AmmServer_Instance *instance, unsigned int index)

  Create compressed version of dynamic content , cache item.
- int CreateCompressedVersionofStaticContent (struct AmmServer_Instance *instance, unsigned int index)

  Create compressed version of static content , cache item.
- int CreateCompressedVersionofStaticContentPreloading (struct AmmServer_Instance *instance, unsigned int index)

  Create compressed version of static content which is preloaded , cache item.

6.27.1 Function Documentation

6.27.1.1 int CreateCompressedVersionofCachedResource (struct AmmServer_Instance * *instance,* unsigned int *index,* int *compression_level*) [inline]

6.27.1.2 int CreateCompressedVersionofDynamicContent (struct AmmServer_Instance * *instance,* unsigned int *index*)

Create compressed version of dynamic content , cache item.

Parameters

An	AmmarServer Instance
Index	of cache item

Return values

1=Success,0=Failure	

Here is the call graph for this function:

6.27.1.3 int CreateCompressedVersionofStaticContent (struct AmmServer_Instance * *instance,* unsigned int *index*)

Create compressed version of static content , cache item.

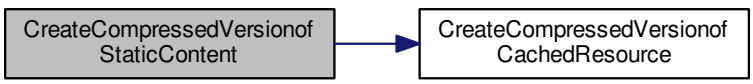
Parameters

An	AmmarServer Instance
Index	of cache item

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.27.1.4 int CreateCompressedVersionofStaticContentPreloading (struct AmmServer_Instance * *instance,* unsigned int *index*)

Create compressed version of static content which is preloaded , cache item.

Parameters

An	AmmarServer Instance
Index	of cache item

Return values

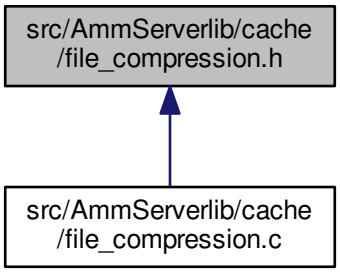
1=Success,0=Failure	

Here is the call graph for this function:

6.28 src/AmmServerlib/cache/file_compression.h File Reference

A tool that compresses memory blocks for better bandwidth usage on the expense of computing power.

This graph shows which files directly or indirectly include this file:



Functions

- int CreateCompressedVersionofDynamicContent (struct AmmServer_Instance *instance, unsigned int index)

  Create compressed version of dynamic content , cache item.
- int CreateCompressedVersionofStaticContent (struct AmmServer_Instance *instance, unsigned int index)

  Create compressed version of static content , cache item.
- int CreateCompressedVersionofStaticContentPreloading (struct AmmServer_Instance *instance, unsigned int index)

  Create compressed version of static content which is preloaded , cache item.

6.28.1 Detailed Description

A tool that compresses memory blocks for better bandwidth usage on the expense of computing power.

Author

Ammar Qammaz (AmmarkoV)

Bug Compression should be improved

6.28.2 Function Documentation

6.28.2.1 int CreateCompressedVersionofDynamicContent (struct AmmServer_Instance * *instance,* unsigned int *index*)

Create compressed version of dynamic content , cache item.

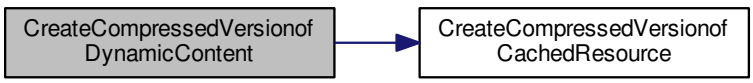
Parameters

An	AmmarServer Instance
Index	of cache item

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.28.2.2 int CreateCompressedVersionofStaticContent (struct AmmServer_Instance * *instance,* unsigned int *index*)

Create compressed version of static content , cache item.

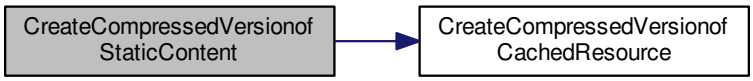
Parameters

An	AmmarServer Instance
Index	of cache item

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.28.2.3 int CreateCompressedVersionofStaticContentPreloading (struct AmmServer_Instance * *instance,* unsigned int *index*)

Create compressed version of static content which is preloaded , cache item.

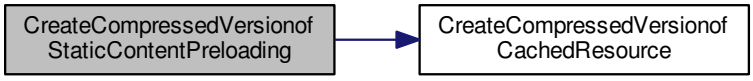
Parameters

An	AmmarServer Instance
Index	of cache item

Return values

1=Success,0=Failure	

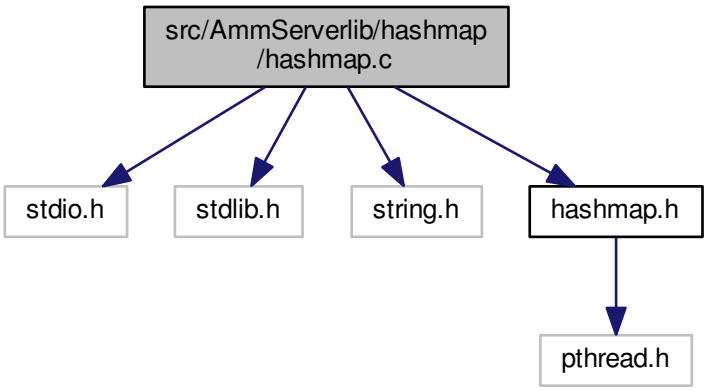
Here is the call graph for this function:



6.29 src/AmmServerlib/hashmap/hashmap.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "hashmap.h"
```

Include dependency graph for hashmap.c:



Functions

- unsigned long hashFunction (char ∗str)

  The function that converts a string to a number so that it will be easier to be searched.
- int hashMap_Grow (struct hashMap ∗hm, unsigned int growthSize)
- struct hashMap ∗ hashMap_Create (unsigned int initialEntries, unsigned int entryAllocationStep, void ∗clearItemFunction)

  Create and allocate a hash map.
- int hashMap_IsOK (struct hashMap ∗hm)
- int hashMap_GetCurrentNumberOfEntries (struct hashMap ∗hm)

  Get the current number of entries of hash map.
- int hashMap_GetMaxNumberOfEntries (struct hashMap ∗hm)

  Get the maximum number of entries of hash map.
- int hashMap_IsSorted (struct hashMap ∗hm)
- void hashMap_Clear (struct hashMap ∗hm)

  Clear all entries of hash map.
- void hashMap_Destroy (struct hashMap ∗hm)

  Destroy and deallocate a hash map.
- int cmpHashTableItems (const void ∗a, const void ∗b)
- int hashMap_Sort (struct hashMap ∗hm)

  Sort hash map.
- int hashMap_Add (struct hashMap ∗hm, char ∗key, void ∗val, unsigned int valLength)

  Add a new key to hash map.
- int hashMap_AddULong (struct hashMap ∗hm, char ∗key, unsigned long val)

  Add a new key (integer) to hash map.
- int hashMap_FindIndex (struct hashMap ∗hm, char ∗key, unsigned long ∗index)

  Find index of a key.
- int hashmap_SwapRecords (struct hashMap ∗hm, unsigned int index1, unsigned int index2)

  Swap two records.
- char ∗ hashMap_GetKeyAtIndex (struct hashMap ∗hm, unsigned int index)

  Return key value for index.

- unsigned long hashMap_GetHashAtIndex (struct hashMap *hm, unsigned int index)

  Return key hash for index.

- int hashMap_GetPayload (struct hashMap *hm, char *key, void *payload)

  Return payload for specified key.

- int hashMap_GetULongPayload (struct hashMap *hm, char *key, unsigned long *payload)

  Return numerical payload for specified key.

- int hashMap_ContainsKey (struct hashMap *hm, char *key)

  Check if hashmap contains a key.

- int hashMap_ContainsValue (struct hashMap *hm, void *val)

  Check if hashmap contains a value.

- int hashMap_SaveToFile (struct hashMap *hm, char *filename)

  Save hash map to a file.

- int hashMap_LoadToFile (struct hashMap *hm, char *filename)

  Load hash map from a file.

6.29.1 Function Documentation

6.29.1.1 int cmpHashTableItems (const void * *a*, const void * *b*)

6.29.1.2 unsigned long hashFunction (char * *str*)

The function that converts a string to a number so that it will be easier to be searched.

djb2 This algorithm (k=33) was first reported by dan bernstein many years ago in comp.lang.c. another version of this algorithm (now favored by bernstein) uses xor: hash(i) = hash(i - 1) * 33 ^ str[i]; the magic of number 33 (why it works better than many other constants, prime or not) has never been adequately explained. Needless to say , this is our hash function..!

6.29.1.3 int hashMap_Add (struct **hashMap** * *hm*, char * *key*, void * *val*, unsigned int *valLength*)

Add a new key to hash map.

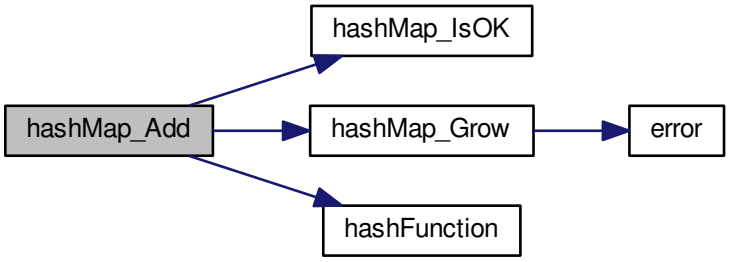
Parameters

HashMap	
String	with the key index
String	with the value of this record
Length	of the value

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.29.1.4 int hashMap_AddULong (struct hashMap * *hm*, char * *key*, unsigned long *val*)

Add a new key (integer) to hash map.

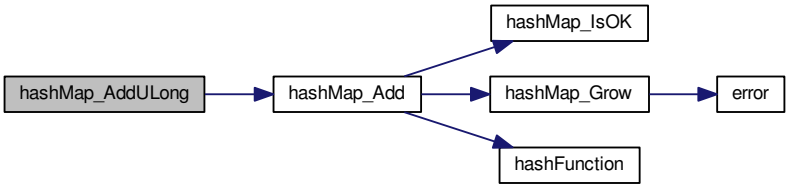
Parameters

HashMap	
String	with the key index
Number	value of this record

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.29.1.5 void hashMap_Clear (struct hashMap * *hm*)

Clear all entries of hash map.

Parameters

HashMap	

Return values

No	return value

Here is the call graph for this function:

6.29.1.6 int hashMap_ContainsKey (struct hashMap * *hm*, char * *key*)

Check if hashmap contains a key.

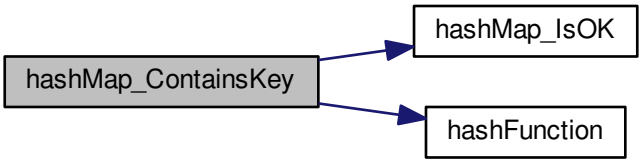
Parameters

HashMap	
String	of key

Return values

1=Exists,0=Does	not Exist

Here is the call graph for this function:



6.29.1.7 int hashMap_ContainsValue (struct hashMap * *hm*, void * *val*)

Check if hashmap contains a value.

Parameters

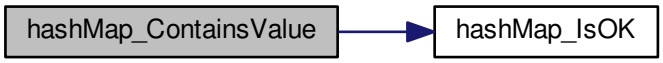
HashMap	

Value	to check for

Return values

1=Exists,0=Does	not Exist

Here is the call graph for this function:



6.29.1.8 struct hashMap* hashMap_Create (unsigned int *initialEntries,* unsigned int *entryAllocationStep,* void * *clearItemFunction*)

Create and allocate a hash map.

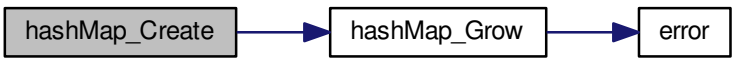
Parameters

Number	of initial entry space
Allocation	step for new allocations
Pointer	to a function that clears an item

Return values

Hashmap	Structure or , 0=Failure

Here is the call graph for this function:



6.29.1.9 void hashMap_Destroy (struct hashMap * *hm*)

Destroy and deallocate a hash map.

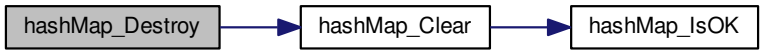
Parameters

HashMap	

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.29.1.10 int hashMap_FindIndex (struct **hashMap** * *hm*, char * *key*, unsigned long * *index*)

Find index of a key.

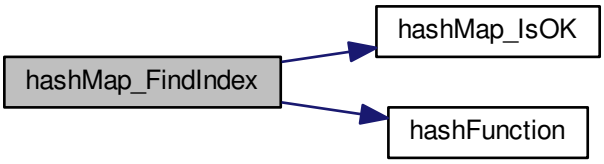
Parameters

HashMap	
Input	String with the key index to find
Output	index of the record that holds the data we were searching for

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.29.1.11 int hashMap_GetCurrentNumberOfEntries (struct **hashMap** * *hm*)

Get the current number of entries of hash map.

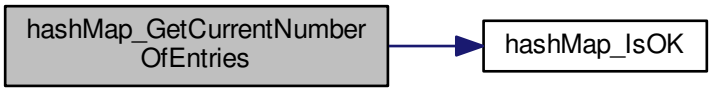
Parameters

HashMap	

Return values

Number	of entries

Here is the call graph for this function:



6.29.1.12 unsigned long hashMap_GetHashAtIndex (struct **hashMap** * *hm,* unsigned int *index*)

Return key hash for index.

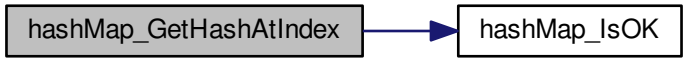
Parameters

HashMap	
Index	number

Return values

Hash	of key , or 0 for no key

Here is the call graph for this function:



6.29.1.13 char* hashMap_GetKeyAtIndex (struct **hashMap** * *hm,* unsigned int *index*)

Return key value for index.

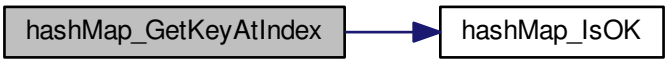
Parameters

HashMap	
Index	number

Return values

String	of key , or 0 for no key

Here is the call graph for this function:



6.29.1.14 int hashMap_GetMaxNumberOfEntries (struct **hashMap** * *hm*)

Get the maximum number of entries of hash map.

Parameters

HashMap	

Return values

Maximum	Number of entries

Here is the call graph for this function:

6.29.1.15 int hashMap_GetPayload (struct **hashMap** * *hm*, char * *key*, void * *payload*)

Return payload for specified key.

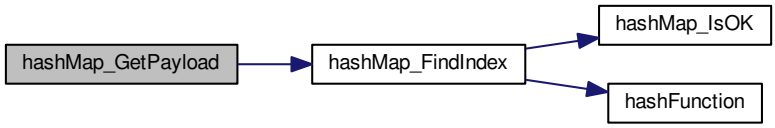
Parameters

HashMap	
Input	String of key
Output	Pointer of payload

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.29.1.16 int hashMap_GetULongPayload (struct hashMap * *hm,* char * *key,* unsigned long * *payload*)

Return numerical payload for specified key.

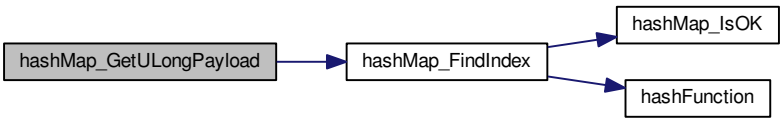
Parameters

HashMap	
Input	String of key
Output	Pointer of payload

Return values

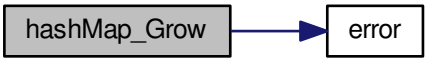
1=Success,0=Failure	

Here is the call graph for this function:



6.29.1.17 int hashMap_Grow (struct hashMap * *hm,* unsigned int *growthSize*)

Here is the call graph for this function:



6.29.1.18 int hashMap_IsOK (struct hashMap * *hm*)

6.29.1.19 int hashMap_IsSorted (struct hashMap * hm)

Here is the call graph for this function:

6.29.1.20 int hashMap_LoadToFile (struct hashMap * hm, char * filename)

Load hash map from a file.

Parameters

HashMap	structure
Filename	to save to

Return values

1=Success,0=Fail	

Here is the call graph for this function:

6.29.1.21 int hashMap_SaveToFile (struct hashMap * hm, char * filename)

Save hash map to a file.

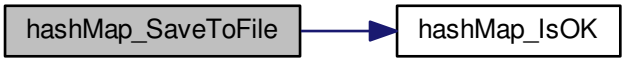
Parameters

HashMap	structure
Filename	to save to

Return values

1=Success,0=Fail	

Here is the call graph for this function:



6.29.1.22 int hashMap_Sort (struct hashMap * hm)

Sort hash map.

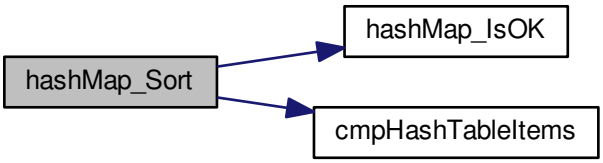
Parameters

HashMap	

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.29.1.23 int hashmap_SwapRecords (struct hashMap * hm, unsigned int index1, unsigned int index2)

Swap two records.

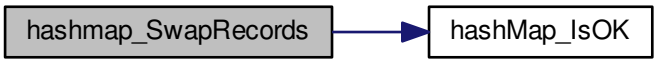
Parameters

HashMap	
Index	1 to be swapped
Index	2 to be swapped

Return values

1=Success,0=Failure	

Here is the call graph for this function:

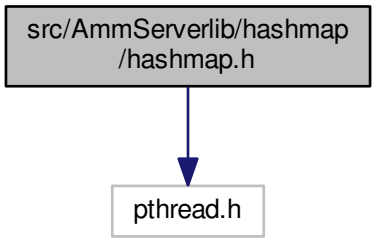


6.30 src/AmmServerlib/hashmap/hashmap.h File Reference

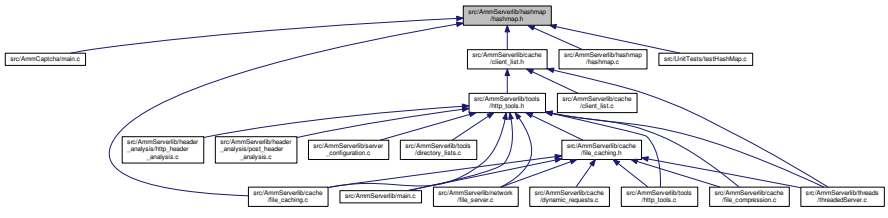
A uniform and clean way to create hashmaps in C and query them.

```
#include <pthread.h>
```

Include dependency graph for hashmap.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct hashMapEntry

  An entry on the hash map flattened out for ease of use.

- struct hashMap

  The central structure for the hash map.

Functions

- unsigned long hashFunction (char *str)

  The function that converts a string to a number so that it will be easier to be searched.
- struct hashMap * hashMap_Create (unsigned int initialEntries, unsigned int entryAllocationStep, void *clear-ItemFunction)

  Create and allocate a hash map.
- void hashMap_Destroy (struct hashMap *hm)

  Destroy and deallocate a hash map.
- int hashMap_Sort (struct hashMap *hm)

  Sort hash map.
- int hashMap_Add (struct hashMap *hm, char *key, void *val, unsigned int valLength)

  Add a new key to hash map.
- int hashMap_AddULong (struct hashMap *hm, char *key, unsigned long val)

  Add a new key (integer) to hash map.
- int hashMap_FindIndex (struct hashMap *hm, char *key, unsigned long *index)

  Find index of a key.
- int hashmap_SwapRecords (struct hashMap *hm, unsigned int index1, unsigned int index2)

  Swap two records.
- char * hashMap_GetKeyAtIndex (struct hashMap *hm, unsigned int index)

  Return key value for index.
- unsigned long hashMap_GetHashAtIndex (struct hashMap *hm, unsigned int index)

  Return key hash for index.
- int hashMap_GetPayload (struct hashMap *hm, char *key, void *payload)

  Return payload for specified key.
- int hashMap_GetULongPayload (struct hashMap *hm, char *key, unsigned long *payload)

  Return numerical payload for specified key.
- void hashMap_Clear (struct hashMap *hm)

  Clear all entries of hash map.
- int hashMap_ContainsKey (struct hashMap *hm, char *key)

  Check if hashmap contains a key.
- int hashMap_ContainsValue (struct hashMap *hm, void *val)

  Check if hashmap contains a value.
- int hashMap_GetMaxNumberOfEntries (struct hashMap *hm)

  Get the maximum number of entries of hash map.
- int hashMap_GetCurrentNumberOfEntries (struct hashMap *hm)

  Get the current number of entries of hash map.
- int hashMap_LoadToFile (struct hashMap *hm, char *filename)

  Load hash map from a file.
- int hashMap_SaveToFile (struct hashMap *hm, char *filename)

  Save hash map to a file.

6.30.1 Detailed Description

A uniform and clean way to create hashmaps in C and query them.

Author

Ammar Qammaz (AmmarkoV)

Bug This hashmap implementation uses serial searches for now , and needs a lot of work

6.30.2 Function Documentation

6.30.2.1 unsigned long hashFunction (char * *str*)

The function that converts a string to a number so that it will be easier to be searched.

djb2 This algorithm (k=33) was first reported by dan bernstein many years ago in comp.lang.c. another version of this algorithm (now favored by bernstein) uses xor: hash(i) = hash(i - 1) * 33 ^ str[i]; the magic of number 33 (why it works better than many other constants, prime or not) has never been adequately explained. Needless to say , this is our hash function..!

6.30.2.2 int hashMap_Add (struct **hashMap** * *hm,* char * *key,* void * *val,* unsigned int *valLength*)

Add a new key to hash map.

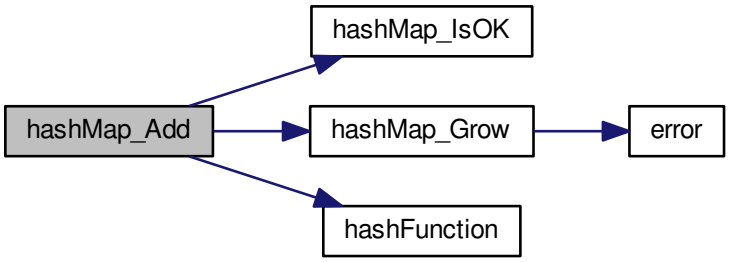
Parameters

HashMap	
String	with the key index
String	with the value of this record
Length	of the value

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.30.2.3 int hashMap_AddULong (struct **hashMap** * *hm,* char * *key,* unsigned long *val*)

Add a new key (integer) to hash map.

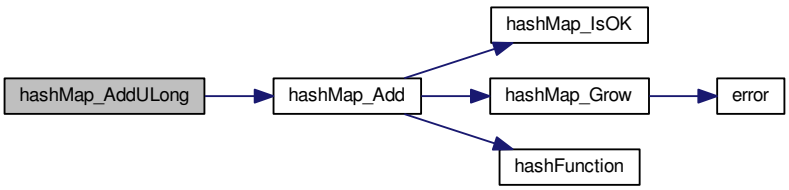
Parameters

HashMap	
String	with the key index
Number	value of this record

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.30.2.4 void hashMap_Clear (struct **hashMap** * *hm*)

Clear all entries of hash map.

Parameters

HashMap	

Return values

No	return value

Here is the call graph for this function:

6.30.2.5 int hashMap_ContainsKey (struct **hashMap** * *hm*, char * *key*)

Check if hashmap contains a key.

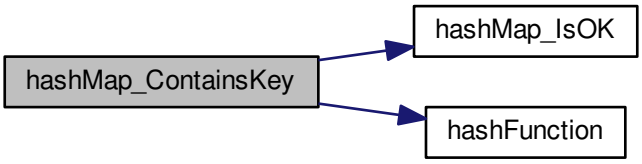
Parameters

HashMap	
String	of key

Return values

1=Exists,0=Does	not Exist

Here is the call graph for this function:



6.30.2.6 int hashMap_ContainsValue (struct hashMap * *hm,* void * *val*)

Check if hashmap contains a value.

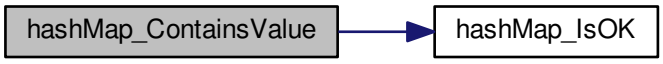
Parameters

HashMap	
Value	to check for

Return values

1=Exists,0=Does	not Exist

Here is the call graph for this function:



6.30.2.7 struct hashMap* hashMap_Create (unsigned int *initialEntries,* unsigned int *entryAllocationStep,* void * *clearItemFunction*)

Create and allocate a hash map.

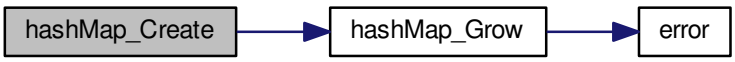
Parameters

Number	of initial entry space
Allocation	step for new allocations
Pointer	to a function that clears an item

Return values

Hashmap	Structure or , 0=Failure

Here is the call graph for this function:



6.30.2.8 void hashMap_Destroy (struct **hashMap** * *hm*)

Destroy and deallocate a hash map.

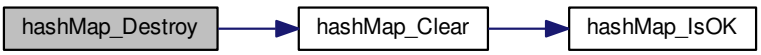
Parameters

HashMap	

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.30.2.9 int hashMap_FindIndex (struct **hashMap** * *hm,* char * *key,* unsigned long * *index*)

Find index of a key.

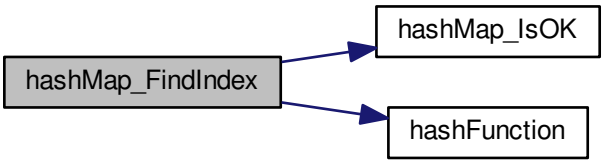
Parameters

HashMap	
Input	String with the key index to find
Output	index of the record that holds the data we were searching for

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.30.2.10 int hashMap_GetCurrentNumberOfEntries (struct **hashMap** * *hm*)

Get the current number of entries of hash map.

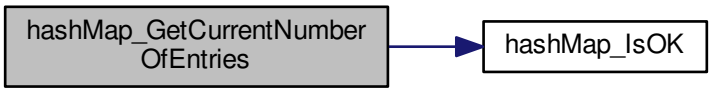
Parameters

HashMap	

Return values

Number	of entries

Here is the call graph for this function:



6.30.2.11 unsigned long hashMap_GetHashAtIndex (struct **hashMap** * *hm*, unsigned int *index*)

Return key hash for index.

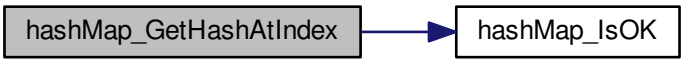
Parameters

HashMap	
Index	number

Return values

Hash	of key , or 0 for no key

Here is the call graph for this function:



6.30.2.12 char* hashMap_GetKeyAtIndex (struct hashMap * *hm*, unsigned int *index*)

Return key value for index.

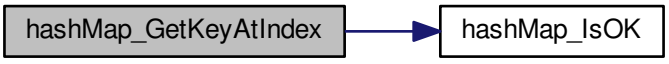
Parameters

HashMap	
Index	number

Return values

String	of key , or 0 for no key

Here is the call graph for this function:



6.30.2.13 int hashMap_GetMaxNumberOfEntries (struct hashMap * *hm*)

Get the maximum number of entries of hash map.

Parameters

HashMap	

Return values

Maximum	Number of entries

Here is the call graph for this function:

6.30.2.14 int hashMap_GetPayload (struct hashMap * hm, char * key, void * payload)

Return payload for specified key.

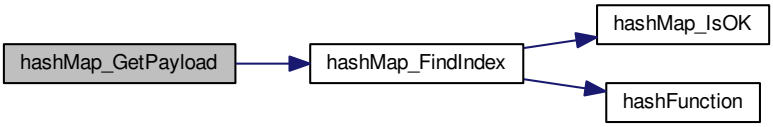
Parameters

HashMap	
Input	String of key
Output	Pointer of payload

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.30.2.15 int hashMap_GetULongPayload (struct hashMap * hm, char * key, unsigned long * payload)

Return numerical payload for specified key.

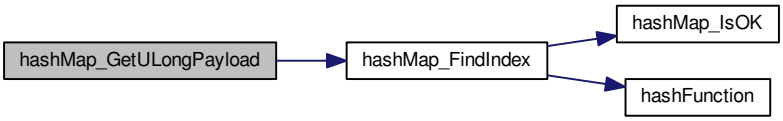
Parameters

HashMap	
Input	String of key
Output	Pointer of payload

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.30.2.16 int hashMap_LoadToFile (struct hashMap * *hm*, char * *filename*)

Load hash map from a file.

Parameters

HashMap	structure
Filename	to save to

Return values

1=Success,0=Fail	

Here is the call graph for this function:

6.30.2.17 int hashMap_SaveToFile (struct hashMap * *hm*, char * *filename*)

Save hash map to a file.

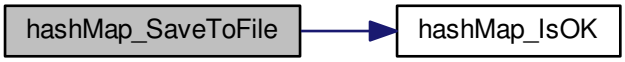
Parameters

HashMap	structure
Filename	to save to

Return values

1=Success,0=Fail	

Here is the call graph for this function:



6.30.2.18 int hashMap_Sort (struct **hashMap** * *hm*)

Sort hash map.

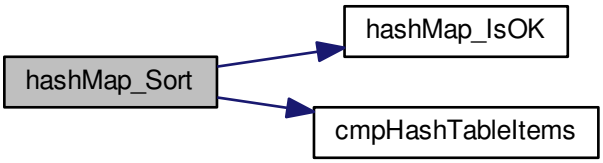
Parameters

HashMap	

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.30.2.19 int hashmap_SwapRecords (struct **hashMap** * *hm,* unsigned int *index1,* unsigned int *index2*)

Swap two records.

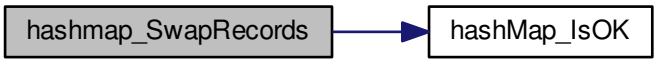
Parameters

HashMap	
Index	1 to be swapped
Index	2 to be swapped

Return values

1=Success,0=Failure	

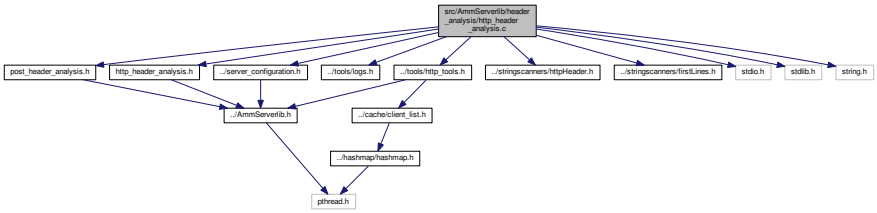
Here is the call graph for this function:



6.31 src/AmmServerlib/header_analysis/http_header_analysis.c File Reference

```
#include "http_header_analysis.h"
#include "post_header_analysis.h"
#include "../tools/http_tools.h"
#include "../tools/logs.h"
#include "../server_configuration.h"
#include "../stringscanners/httpHeader.h"
#include "../stringscanners/firstLines.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

Include dependency graph for http_header_analysis.c:



Macros

- #define CR 13
- #define LF 10

Functions

- char * ReceiveHTTPHeader (struct AmmServer_Instance *instance, int clientSock, unsigned long *headerLength)

  Receive an HTTP Header from a socket and prepare it for further processing.
- int AppendPOSTRequestToHTTPHeader (struct HTTPTransaction *transaction)

  POST requests also have a payload appended that we consider part of the whole "header" so we need to keep on reading it..!
- int FreeHTTPHeader (struct HTTPHeader *output)

  Deallocate memory occupied by an HTTP Header.
- int HTTPHeaderComplete (char *request, unsigned int request_length)

Ask if a header is complete inside an incoming request , detected by four consecutive bytes CR LF CR LF that mark the end of a header.

- int HTTPHeaderIsPOST (char *request, unsigned int requestLength)

  Ask if a header is a POST request, detected by the first four consecutive bytes being P O S T.
- int ProcessFirstHTTPLine (struct HTTPHeader *output, char *request, unsigned int request_length, char *webserver_root)
- int ProcessAuthorizationHTTPLine (struct AmmServer_Instance *instance, struct HTTPHeader *output, char *request, unsigned int request_length, unsigned int *payload_pos)
- int ProcessRangeHTTPLine (char *request, unsigned int requestLength, unsigned long *rangeStart, unsigned long *rangeEnd)
- int AnalyzeHTTPLineRequest (struct AmmServer_Instance *instance, struct HTTPHeader *output, char *request, unsigned int request_length, unsigned int lines_gathered, char *webserver_root)
- int AnalyzeHTTPHeader (struct AmmServer_Instance *instance, struct HTTPTransaction *transaction)

  Analyze HTTP header (after it has been accumulated into memory)

6.31.1 Macro Definition Documentation

6.31.1.1 #define CR 13

6.31.1.2 #define LF 10

6.31.2 Function Documentation

6.31.2.1 int AnalyzeHTTPHeader (struct AmmServer_Instance * *instance,* struct HTTPTransaction * *transaction*)

Analyze HTTP header (after it has been accumulated into memory)

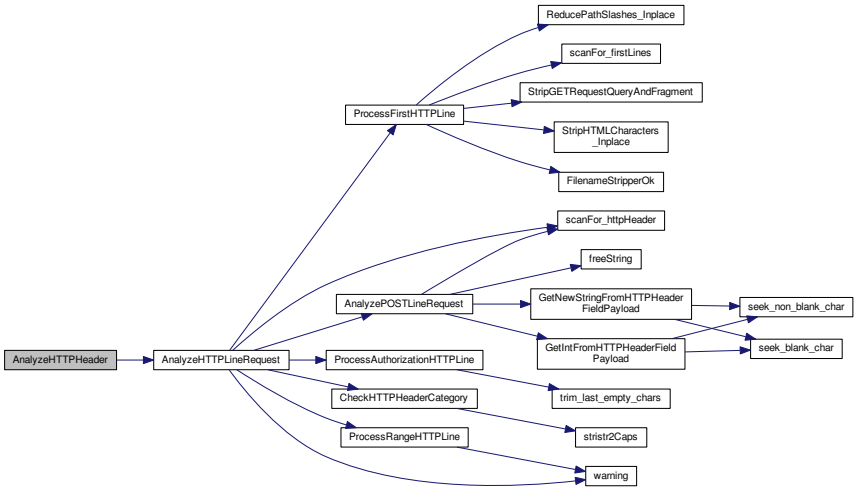
Parameters

An	AmmarServer Instance
HTTP-Transaction	we are talking about

Return values

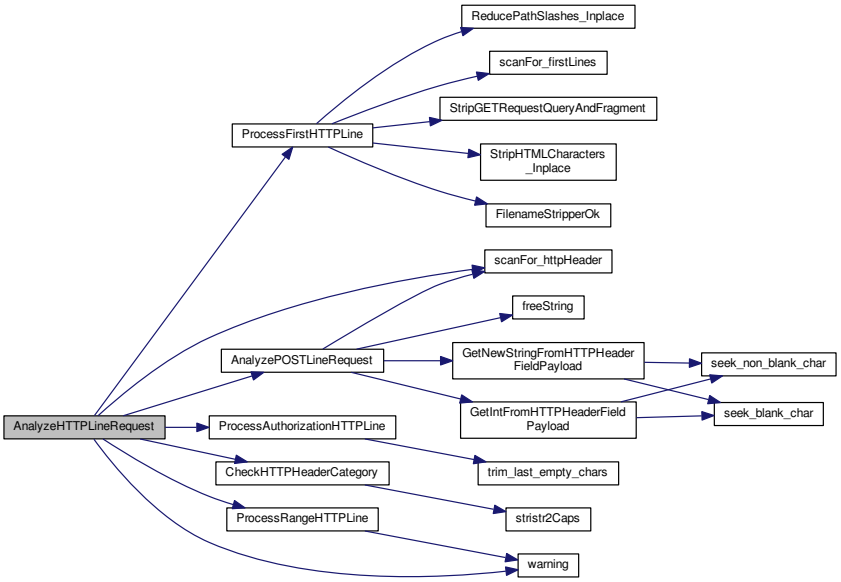
1=Success,0=Failure	

Here is the call graph for this function:



6.31.2.2 int AnalyzeHTTPLineRequest (struct AmmServer_Instance * *instance*, struct HTTPHeader * *output*, char * *request*, unsigned int *request_length*, unsigned int *lines_gathered*, char * *webserver_root*)

Here is the call graph for this function:



6.31.2.3 int AppendPOSTRequestToHTTPHeader (struct HTTPTransaction * *transaction*)

POST requests also have a payload appended that we consider part of the whole "header" so we need to keep on reading it..!

Parameters

HTTP-Transaction	

Return values

1=Success,0=Failure	

Here is the call graph for this function:

6.31.2.4 int FreeHTTPHeader (struct HTTPHeader * *output*)

Deallocate memory occupied by an HTTP Header.

Parameters

HTTPHeader	to be deallocated

Return values

1=Success,0=Failure	

6.31.2.5 int HTTPHeaderComplete (char * *request*, unsigned int *request_length*)

Ask if a header is complete inside an incoming request , detected by four consecutive bytes CR LF CR LF that mark the end of a header.

Parameters

Pointer	to incoming request (streaming) string
Length	of incoming string

Return values

1=Complete,0=Incomplete	

6.31.2.6 int HTTPHeaderIsPOST (char * *request*, unsigned int *requestLength*)

Ask if a header is a POST request, detected by the first four consecutive bytes being P O S T.

Parameters

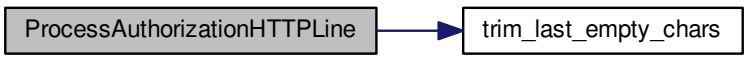
Pointer	to incoming request (streaming) string
Length	of incoming string

Return values

1=POST,0=Not	POST request

6.31.2.7 int ProcessAuthorizationHTTPLine (struct AmmServer_Instance * *instance*, struct HTTPHeader * *output*, char * *request*, unsigned int *request_length*, unsigned int * *payload_pos*) [inline]

Here is the call graph for this function:

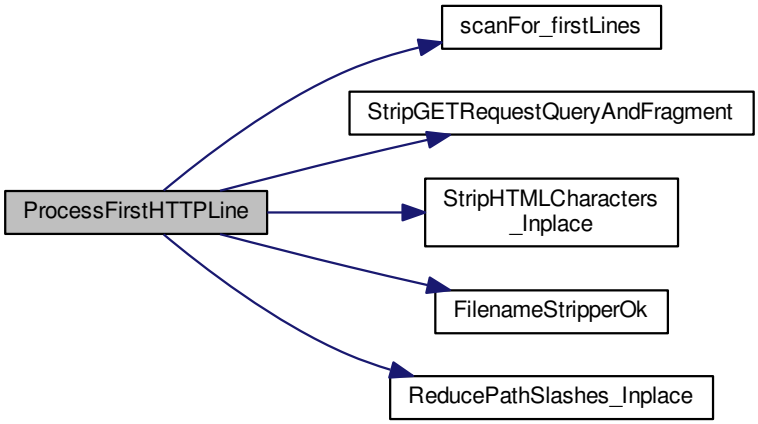


6.31.2.8 int ProcessFirstHTTPLine (struct HTTPHeader * *output*, char * *request*, unsigned int *request_length*, char * *webserver_root*)

Input String Verification from client Since this string will be passed to fopen this can be dangerous so we perform some security checks with FilenameStripperOk to make sure no escape characters subdirs out of public_html (via public_html/../etc) and overflows may happen..! Most of the functions are implemented in http_tools! The results

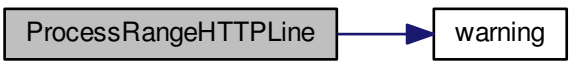
are then copied to output->resource and output->verified_local_resource which contain the resource requested as the client stated it and as we verified for local filesystem..!

Here is the call graph for this function:



6.31.2.9 int ProcessRangeHTTPLine (char * *request*, unsigned int *requestLength*, unsigned long * *rangeStart*, unsigned long * *rangeEnd*) [inline]

Here is the call graph for this function:



6.31.2.10 char* ReceiveHTTPHeader (struct AmmServer_Instance * *instance*, int *clientSock*, unsigned long * *headerLength*)

Receive an HTTP Header from a socket and prepare it for further processing.

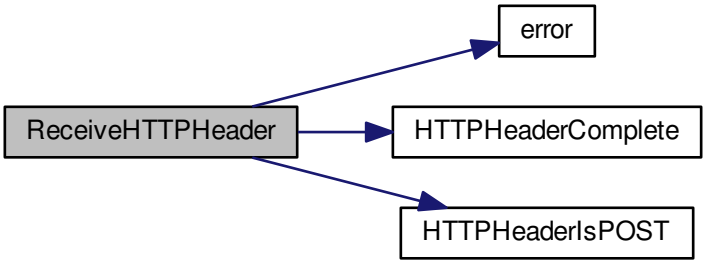
Parameters

An	AmmarServer Instance
Socket	that we should recv to get the http header
Output	length of incoming header

Return values

Pointer	to memory containing HTTPHeader,0=Failure

Here is the call graph for this function:

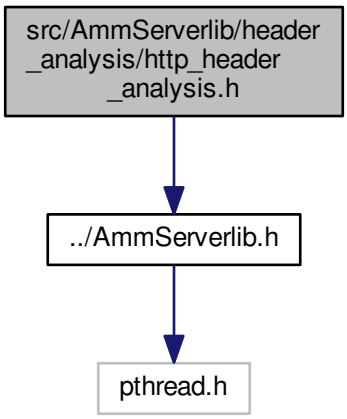


6.32 src/AmmServerlib/header_analysis/http_header_analysis.h File Reference

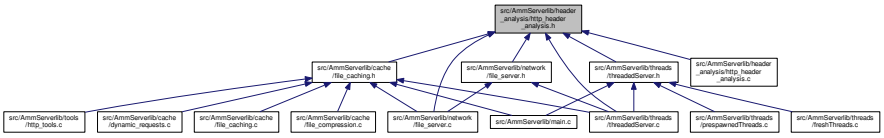
Tools to process HTTP requests.

```
#include "../AmmServerlib.h"
```

Include dependency graph for http_header_analysis.h:



This graph shows which files directly or indirectly include this file:



Functions

- char * ReceiveHTTPHeader (struct AmmServer_Instance *instance, int clientSock, unsigned long *headerLength)

  Receive an HTTP Header from a socket and prepare it for further processing.
- int AppendPOSTRequestToHTTPHeader (struct HTTPTransaction *transaction)

  POST requests also have a payload appended that we consider part of the whole "header" so we need to keep on reading it..!
- int FreeHTTPHeader (struct HTTPHeader *output)

  Deallocate memory occupied by an HTTP Header.
- int HTTPHeaderComplete (char *request, unsigned int request_length)

  Ask if a header is complete inside an incoming request , detected by four consecutive bytes CR LF CR LF that mark the end of a header.
- int HTTPHeaderIsPOST (char *request, unsigned int requestLength)

  Ask if a header is a POST request, detected by the first four consecutive bytes being P O S T.
- int AnalyzeHTTPHeader (struct AmmServer_Instance *instance, struct HTTPTransaction *transaction)

  Analyze HTTP header (after it has been accumulated into memory)

6.32.1 Detailed Description

Tools to process HTTP requests.

Author

Ammar Qammaz (AmmarkoV)

Bug HTTP header analysis can be improved (code style etc) although the recent use of stringscanners has greatly improved it and reduced lines of code

6.32.2 Function Documentation

6.32.2.1 int AnalyzeHTTPHeader (struct AmmServer_Instance * *instance*, struct HTTPTransaction * *transaction*)

Analyze HTTP header (after it has been accumulated into memory)

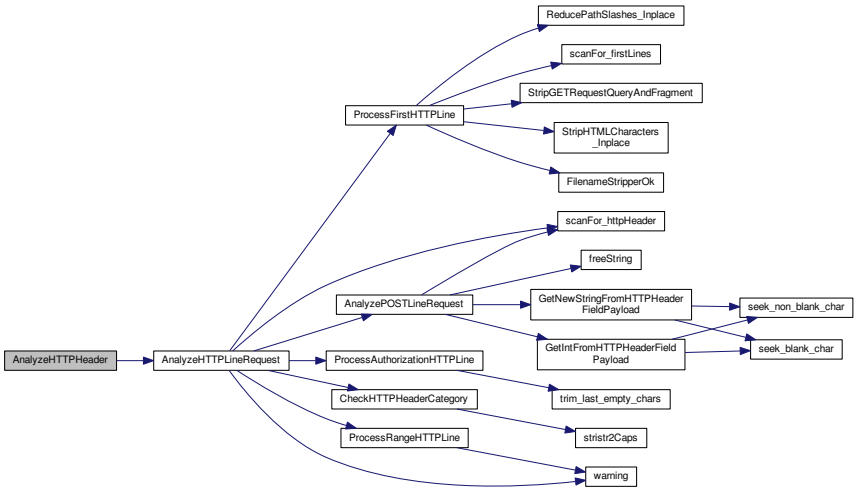
Parameters

An	AmmarServer Instance
HTTP-Transaction	we are talking about

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.32.2.2 int AppendPOSTRequestToHTTPHeader (struct **HTTPTransaction** * *transaction*)

POST requests also have a payload appended that we consider part of the whole "header" so we need to keep on reading it..!

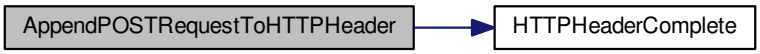
Parameters

HTTP-Transaction	

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.32.2.3 int FreeHTTPHeader (struct **HTTPHeader** * *output*)

Deallocate memory occupied by an HTTP Header.

Parameters

HTTPHeader	to be deallocated

Return values

1=Success,0=Failure	

6.32.2.4 int HTTPHeaderComplete (char * *request,* unsigned int *request_length*)

Ask if a header is complete inside an incoming request , detected by four consecutive bytes CR LF CR LF that mark the end of a header.

Parameters

Pointer	to incoming request (streaming) string
Length	of incoming string

Return values

1=Complete,0=Incomplete	

6.32.2.5 int HTTPHeaderIsPOST (char * *request,* unsigned int *requestLength*)

Ask if a header is a POST request, detected by the first four consecutive bytes being P O S T.

Parameters

Pointer	to incoming request (streaming) string
Length	of incoming string

Return values

1=POST,0=Not	POST request

6.32.2.6 char* ReceiveHTTPHeader (struct AmmServer_Instance * *instance,* int *clientSock,* unsigned long * *headerLength*)

Receive an HTTP Header from a socket and prepare it for further processing.

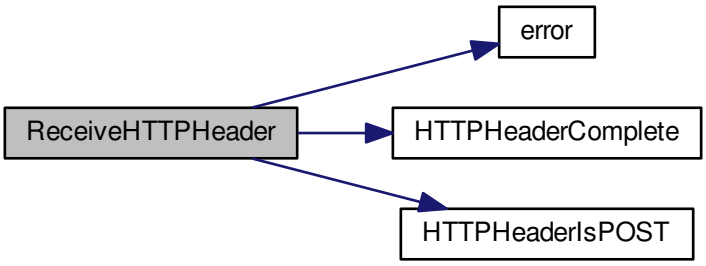
Parameters

An	AmmarServer Instance
Socket	that we should recv to get the http header
Output	length of incoming header

Return values

Pointer	to memory containing HTTPHeader,0=Failure

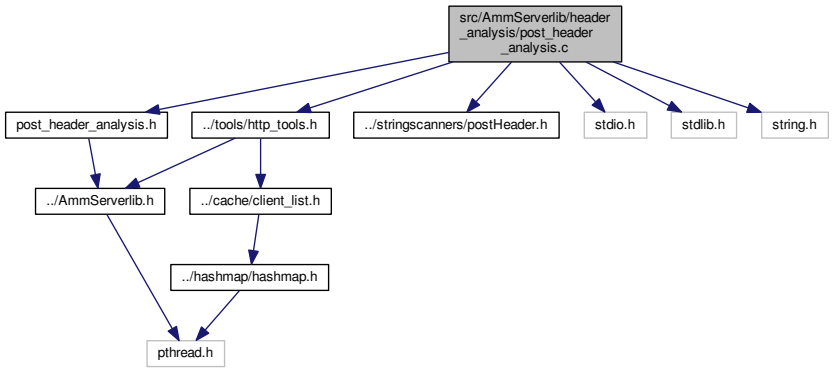
Here is the call graph for this function:



6.33 src/AmmServerlib/header_analysis/post_header_analysis.c File Reference

```
#include "post_header_analysis.h"
#include "../tools/http_tools.h"
#include "../stringscanners/postHeader.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

Include dependency graph for post_header_analysis.c:



Functions

- int AnalyzePOSTLineRequest (struct AmmServer_Instance *instance, struct HTTPHeader *output, char *request, unsigned int request_length, unsigned int lines_gathered, char *webserver_root)

  Analyze a POST request line by line filling in the structures that define it.

6.33.1 Function Documentation

6.33.1.1 int AnalyzePOSTLineRequest (struct AmmServer_Instance * *instance*, struct HTTPHeader * *output*, char * *request*, unsigned int *request_length*, unsigned int *lines_gathered*, char * *webserver_root*)

Analyze a POST request line by line filling in the structures that define it.

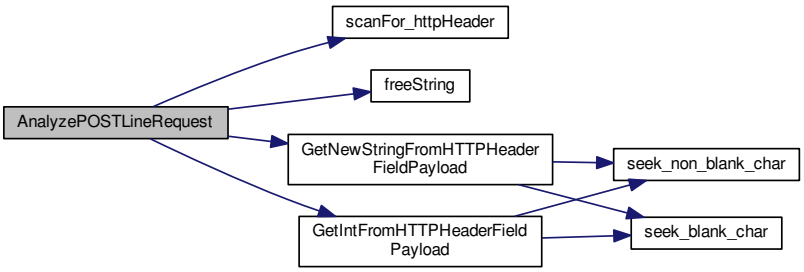
Parameters

An	AmmarServer Instance
Output	HTTPHeader with information
Memory	block with the incoming request
Length	of Memory block of the incoming request
Current	line we are at
Filename	of web server root directory (public_html)

Return values

1=Success,0=Failure	

Here is the call graph for this function:

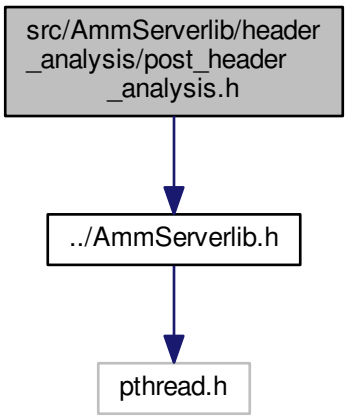


6.34 src/AmmServerlib/header_analysis/post_header_analysis.h File Reference

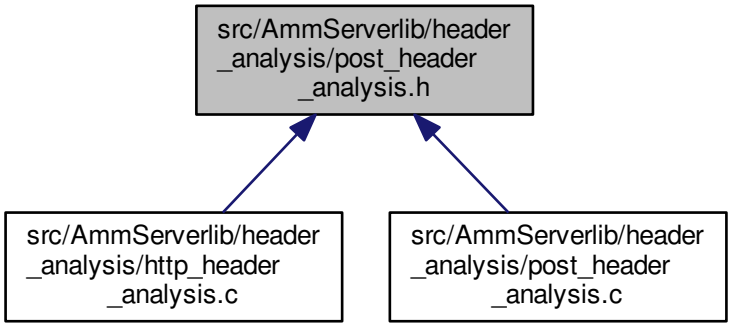
Tools to process POST requests.

```
#include "../AmmServerlib.h"
```

Include dependency graph for post_header_analysis.h:



This graph shows which files directly or indirectly include this file:



Functions

- int AnalyzePOSTLineRequest (struct AmmServer_Instance *instance, struct HTTPHeader *output, char *request, unsigned int request_length, unsigned int lines_gathered, char *webserver_root)

  Analyze a POST request line by line filling in the structures that define it.

6.34.1 Detailed Description

Tools to process POST requests.

Author

Ammar Qammaz (AmmarkoV)

Bug POST header analysis is not fully implemented yet

6.34.2 Function Documentation

6.34.2.1 int AnalyzePOSTLineRequest (struct AmmServer_Instance * *instance,* struct HTTPHeader * *output,* char * *request,* unsigned int *request_length,* unsigned int *lines_gathered,* char * *webserver_root*)

Analyze a POST request line by line filling in the structures that define it.

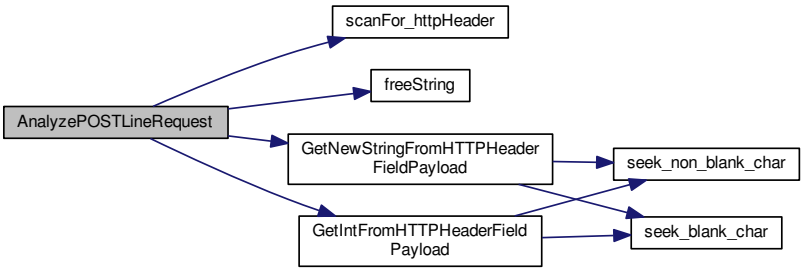
Parameters

An	AmmarServer Instance
Output	HTTPHeader with information
Memory	block with the incoming request
Length	of Memory block of the incoming request
Current	line we are at
Filename	of web server root directory (public_html)

Return values

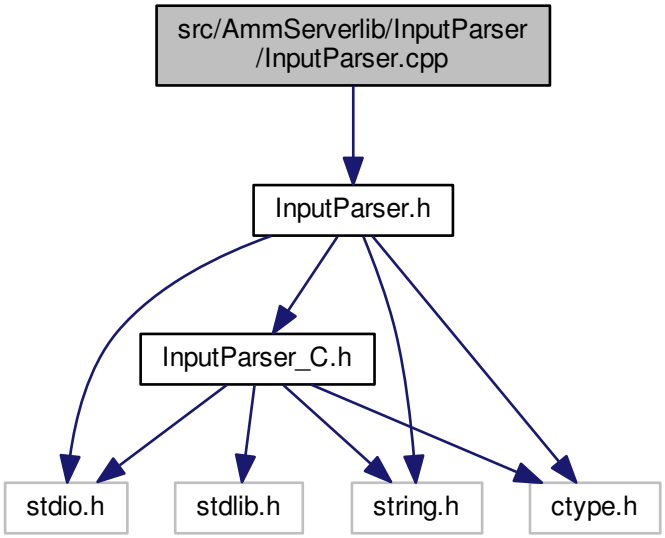
1=Success,0=Failure	

Here is the call graph for this function:



6.35 src/AmmServerlib/InputParser/InputParser.cpp File Reference

`#include "InputParser.h"`
Include dependency graph for InputParser.cpp:



Functions

- const char * Version ()

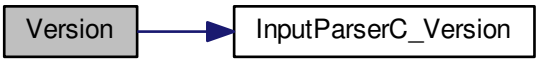
Variables

- const char * ver =" VERSION 1.30 - 7/1/10 \0"

6.35.1 Function Documentation

6.35.1.1 const char* Version ( )

Here is the call graph for this function:



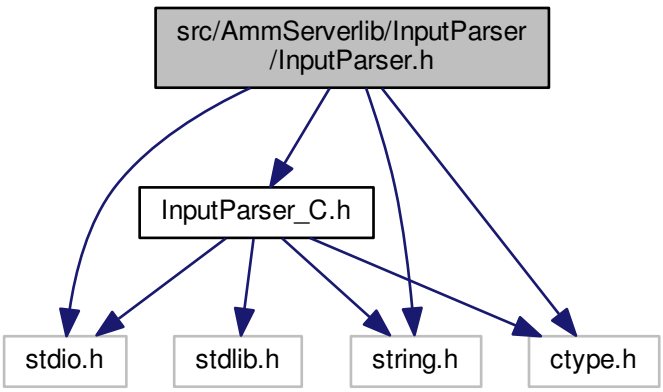
6.35.2 Variable Documentation

6.35.2.1 const char* ver =" VERSION 1.30 - 7/1/10 \0"

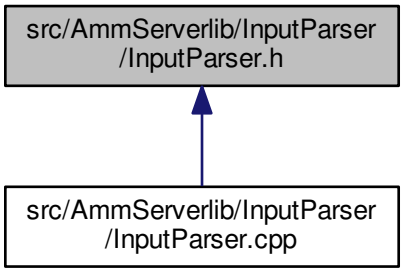
6.36 src/AmmServerlib/InputParser/InputParser.h File Reference

```
#include "InputParser_C.h"
#include <stdio.h>
#include <string.h>
#include <ctype.h>
```

Include dependency graph for InputParser.h:



This graph shows which files directly or indirectly include this file:



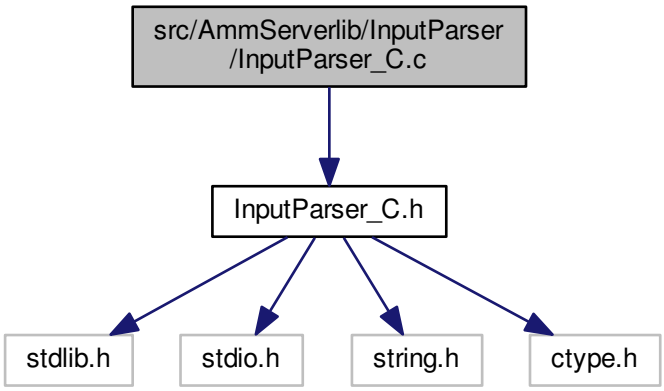
Data Structures

- class InputParser

6.37 src/AmmServerlib/InputParser/InputParser_C.c File Reference

```
#include "InputParser_C.h"
```

Include dependency graph for InputParser_C.c:



Macros

- #define WARN_ABOUT_INCORRECTLY_ALLOCATED_STACK_STRINGS 1

Functions

- char * InputParserC_Version ()
- int InputParser_ClearNonCharacters (char *inpt, unsigned int length)
- int InputParser_TrimCharactersStart (char *inpt, unsigned int length, char what2trim)
- int InputParser_TrimCharactersEnd (char *inpt, unsigned int length, char what2trim)
- int InputParser_TrimCharacters (char *inpt, unsigned int length, char what2trim)
- signed int Str2Int_internal (char *inpt, unsigned int start_from, unsigned int length)
- unsigned char CheckIPCOk (struct InputParserC *ipc)
- void InputParser_DefaultDelimeters (struct InputParserC *ipc)
- struct InputParserC * InputParser_Create (unsigned int max_string_count, unsigned int max_delimiter_count)
- void InputParser_Destroy (struct InputParserC *ipc)
- unsigned char CheckDelimeterNumOk (struct InputParserC *ipc, int num)
- void InputParser_SetDelimeter (struct InputParserC *ipc, int num, char tmp)
- char InputParser_GetDelimeter (struct InputParserC *ipc, int num)
- unsigned char InputParser_SelfCheck (struct InputParserC *ipc)
- unsigned char CheckWordNumOk (struct InputParserC *ipc, unsigned int num)
- unsigned int InputParser_GetWord (struct InputParserC *ipc, unsigned int num, char *wheretostore, unsigned int storagesize)
- unsigned char InputParser_WordCompareNoCase (struct InputParserC *ipc, unsigned int num, char *word, unsigned int wordsize)
- unsigned char InputParser_WordCompareNoCaseAuto (struct InputParserC *ipc, unsigned int num, char *word)
- unsigned char InputParser_WordCompare (struct InputParserC *ipc, unsigned int num, char *word, unsigned int wordsize)
- unsigned char InputParser_WordCompareAuto (struct InputParserC *ipc, unsigned int num, char *word)
- unsigned int InputParser_GetUpcaseWord (struct InputParserC *ipc, unsigned int num, char *wheretostore, unsigned int storagesize)
- unsigned int InputParser_GetLowercaseWord (struct InputParserC *ipc, unsigned int num, char *wheretostore, unsigned int storagesize)
- char InputParser_GetWordChar (struct InputParserC *ipc, unsigned int num, unsigned int pos)
- signed int InputParser_GetWordInt (struct InputParserC *ipc, unsigned int num)
- float InputParser_GetWordFloat (struct InputParserC *ipc, unsigned int num)
- unsigned int InputParser_GetWordLength (struct InputParserC *ipc, unsigned int num)
- int InputParser_SeperateWords (struct InputParserC *ipc, char *inpt, char keepcopy)
- int InputParser_SeperateWordsCC (struct InputParserC *ipc, const char *inpt, char keepcopy)
- int InputParser_SeperateWordsUC (struct InputParserC *ipc, unsigned char *inpt, char keepcopy)

Variables

- int warningsAboutIncorrectlyAllocatedStackIssued = 0
- char _ipc_ver [] =" 0.356 written from scratch - 8/2/10 \0"

6.37.1 Macro Definition Documentation

6.37.1.1 #define WARN_ABOUT_INCORRECTLY_ALLOCATED_STACK_STRINGS 1

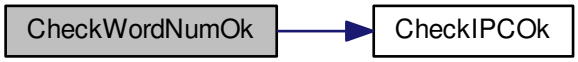
6.37.2 Function Documentation

6.37.2.1 unsigned char CheckDelimeterNumOk (struct InputParserC * *ipc*, int *num*) [inline]

6.37.2.2 unsigned char CheckIPCOk (struct InputParserC * *ipc*) [inline]

6.37.2.3 unsigned char CheckWordNumOk (struct InputParserC * *ipc,* unsigned int *num*) [inline]

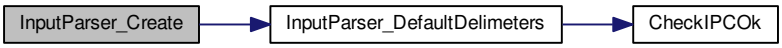
Here is the call graph for this function:



6.37.2.4 int InputParser_ClearNonCharacters (char * *inpt,* unsigned int *length*)

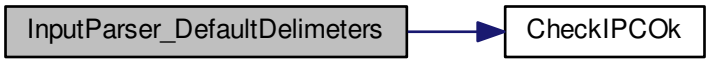
6.37.2.5 struct InputParserC* InputParser_Create (unsigned int *max_string_count,* unsigned int *max_delimiter_count*)

Here is the call graph for this function:



6.37.2.6 void InputParser_DefaultDelimeters (struct InputParserC * *ipc*)

Here is the call graph for this function:



6.37.2.7 void InputParser_Destroy (struct InputParserC * *ipc*)

6.37.2.8 char InputParser_GetDelimeter (struct InputParserC * *ipc*, int *num*)

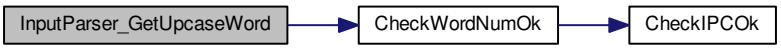
Here is the call graph for this function:

6.37.2.9 unsigned int InputParser_GetLowercaseWord (struct InputParserC * *ipc*, unsigned int *num*, char * *wheretostore*, unsigned int *storagesize*)

Here is the call graph for this function:

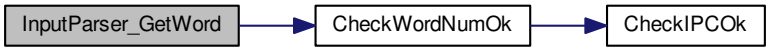
6.37.2.10 unsigned int InputParser_GetUpcaseWord (struct InputParserC * *ipc*, unsigned int *num*, char * *wheretostore*, unsigned int *storagesize*)

Here is the call graph for this function:



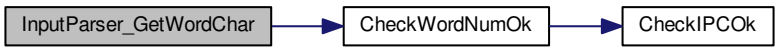
6.37.2.11 unsigned int InputParser_GetWord (struct InputParserC * *ipc*, unsigned int *num*, char * *wheretostore*, unsigned int *storagesize*)

Here is the call graph for this function:



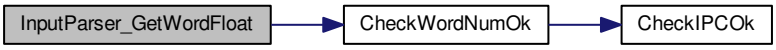
6.37.2.12 char InputParser_GetWordChar (struct InputParserC * *ipc,* unsigned int *num,* unsigned int *pos*)

Here is the call graph for this function:



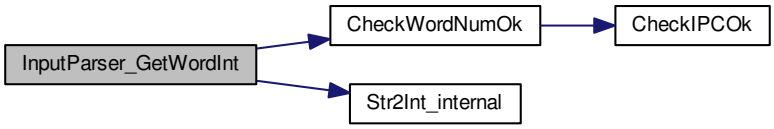
6.37.2.13 float InputParser_GetWordFloat (struct InputParserC * *ipc,* unsigned int *num*)

Here is the call graph for this function:



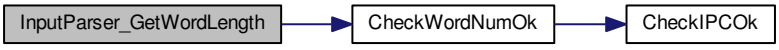
6.37.2.14 signed int InputParser_GetWordInt (struct InputParserC * *ipc,* unsigned int *num*)

Here is the call graph for this function:



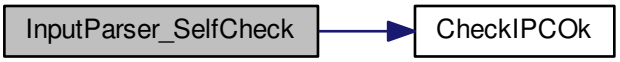
6.37.2.15 unsigned int InputParser_GetWordLength (struct InputParserC * *ipc,* unsigned int *num*)

Here is the call graph for this function:



6.37.2.16 unsigned char InputParser_SelfCheck (struct InputParserC * *ipc*)

Here is the call graph for this function:

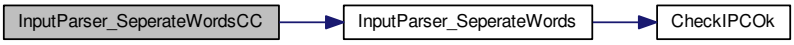


6.37.2.17 int InputParser_SeperateWords (struct InputParserC * *ipc*, char * *inpt*, char *keepcopy*)

Here is the call graph for this function:

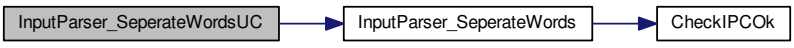
6.37.2.18 int InputParser_SeperateWordsCC (struct InputParserC * *ipc*, const char * *inpt*, char *keepcopy*)

Here is the call graph for this function:



6.37.2.19 int InputParser_SeperateWordsUC (struct InputParserC * *ipc*, unsigned char * *inpt*, char *keepcopy*)

Here is the call graph for this function:

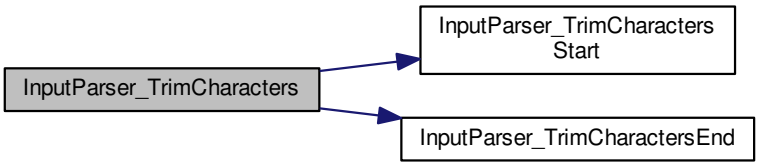


6.37.2.20 void InputParser_SetDelimeter (struct InputParserC * *ipc*, int *num*, char *tmp*)

Here is the call graph for this function:

6.37.2.21 int InputParser_TrimCharacters (char * *inpt*, unsigned int *length*, char *what2trim*)

Here is the call graph for this function:

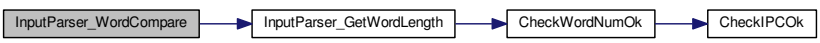


6.37.2.22 int InputParser_TrimCharactersEnd (char * *inpt*, unsigned int *length*, char *what2trim*)

6.37.2.23 int InputParser_TrimCharactersStart (char * *inpt*, unsigned int *length*, char *what2trim*)

6.37.2.24 unsigned char InputParser_WordCompare (struct InputParserC * *ipc*, unsigned int *num*, char * *word*, unsigned int *wordsize*)

Here is the call graph for this function:

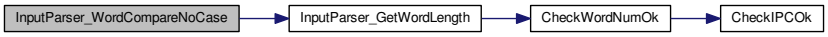


6.37.2.25 unsigned char InputParser_WordCompareAuto (struct InputParserC * *ipc,* unsigned int *num,* char * *word*)

Here is the call graph for this function:

6.37.2.26 unsigned char InputParser_WordCompareNoCase (struct InputParserC * *ipc,* unsigned int *num,* char * *word,* unsigned int *wordsize*)

Here is the call graph for this function:



6.37.2.27 unsigned char InputParser_WordCompareNoCaseAuto (struct InputParserC * *ipc,* unsigned int *num,* char * *word*)

Here is the call graph for this function:

6.37.2.28 char* InputParserC_Version ( )

6.37.2.29 signed int Str2Int_internal (char * *inpt,* unsigned int *start_from,* unsigned int *length*) [inline]

6.37.3 Variable Documentation

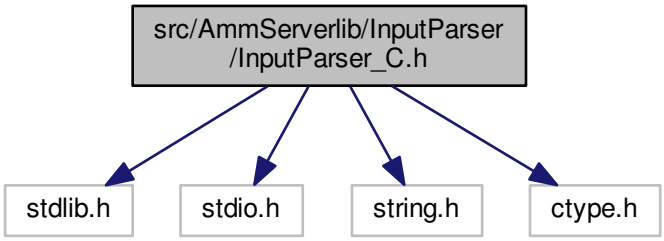
6.37.3.1 char _ipc_ver[ ] =" 0.356 written from scratch - 8/2/10 \0"

6.37.3.2 int warningsAboutIncorrectlyAllocatedStackIssued = 0

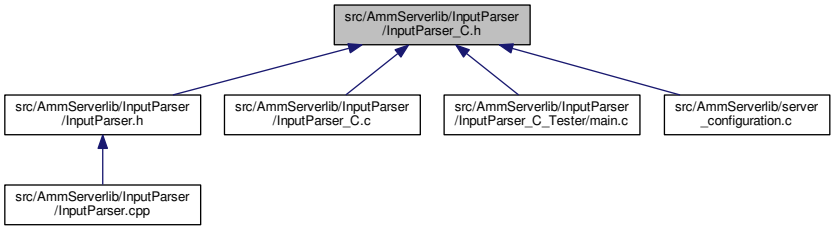
6.38 src/AmmServerlib/InputParser/InputParser_C.h File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <ctype.h>
```

Include dependency graph for InputParser_C.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct tokens
- struct guard_byte
- struct InputParserC

Macros

- #define DELIM_MAX_MAX 6
- #define CONTAINERS_MAX 1
- #define MAX_COMPLICITY 4
- #define MAX_MEMORY 256
- #define MAX_STRING 2048

Functions

- char * InputParserC_Version ()
- int InputParser_ClearNonCharacters (char *inpt, unsigned int length)
- int InputParser_TrimCharactersStart (char *inpt, unsigned int length, char what2trim)
- int InputParser_TrimCharactersEnd (char *inpt, unsigned int length, char what2trim)
- int InputParser_TrimCharacters (char *inpt, unsigned int length, char what2trim)
- void InputParser_DefaultDelimeters (struct InputParserC *ipc)

- void InputParser_SetDelimeter (struct InputParserC *ipc, int num, char tmp)
- char InputParser_GetDelimeter (struct InputParserC *ipc, int num)
- struct InputParserC * InputParser_Create (unsigned int max_string_count, unsigned int max_delimiter_count)
- void InputParser_Destroy (struct InputParserC *ipc)
- unsigned char InputParser_SelfCheck (struct InputParserC *ipc)
- unsigned char CheckWordNumOk (struct InputParserC *ipc, unsigned int num)
- char InputParser_GetWordChar (struct InputParserC *ipc, unsigned int num, unsigned int pos)
- unsigned char InputParser_WordCompareNoCase (struct InputParserC *ipc, unsigned int num, char *word, unsigned int wordsize)
- unsigned char InputParser_WordCompareNoCaseAuto (struct InputParserC *ipc, unsigned int num, char *word)
- unsigned char InputParser_WordCompare (struct InputParserC *ipc, unsigned int num, char *word, unsigned int wordsize)
- unsigned char InputParser_WordCompareAuto (struct InputParserC *ipc, unsigned int num, char *word)
- unsigned int InputParser_GetWord (struct InputParserC *ipc, unsigned int num, char *wheretostore, unsigned int storagesize)
- unsigned int InputParser_GetUpcaseWord (struct InputParserC *ipc, unsigned int num, char *wheretostore, unsigned int storagesize)
- unsigned int InputParser_GetLowercaseWord (struct InputParserC *ipc, unsigned int num, char *wheretostore, unsigned int storagesize)
- signed int InputParser_GetWordInt (struct InputParserC *ipc, unsigned int num)
- float InputParser_GetWordFloat (struct InputParserC *ipc, unsigned int num)
- unsigned int InputParser_GetWordLength (struct InputParserC *ipc, unsigned int num)
- int InputParser_SeperateWords (struct InputParserC *ipc, char *inpt, char keepcopy)
- int InputParser_SeperateWordsCC (struct InputParserC *ipc, const char *inpt, char keepcopy)
- int InputParser_SeperateWordsUC (struct InputParserC *ipc, unsigned char *inpt, char keepcopy)

6.38.1 Macro Definition Documentation

6.38.1.1 #define CONTAINERS_MAX 1

6.38.1.2 #define DELIM_MAX_MAX 6

6.38.1.3 #define MAX_COMPLICITY 4

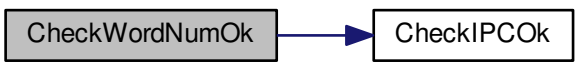
6.38.1.4 #define MAX_MEMORY 256

6.38.1.5 #define MAX_STRING 2048

6.38.2 Function Documentation

6.38.2.1 unsigned char CheckWordNumOk (struct InputParserC * *ipc,* unsigned int *num*) [inline]

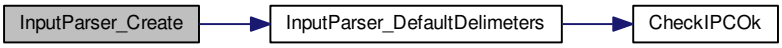
Here is the call graph for this function:



6.38.2.2 int InputParser_ClearNonCharacters (char * *inpt,* unsigned int *length*)

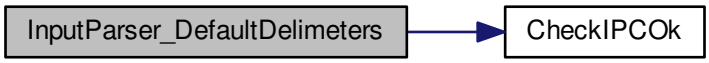
6.38.2.3 struct InputParserC* InputParser_Create (unsigned int *max_string_count,* unsigned int *max_delimiter_count*)

Here is the call graph for this function:



6.38.2.4 void InputParser_DefaultDelimeters (struct InputParserC * *ipc*)

Here is the call graph for this function:



6.38.2.5 void InputParser_Destroy (struct InputParserC * *ipc*)

6.38.2.6 char InputParser_GetDelimeter (struct InputParserC * *ipc,* int *num*)

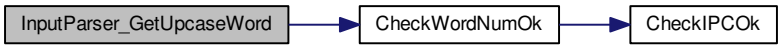
Here is the call graph for this function:

6.38.2.7 unsigned int InputParser_GetLowercaseWord (struct **InputParserC** * *ipc*, unsigned int *num*, char * *wheretostore*, unsigned int *storagesize*)

Here is the call graph for this function:

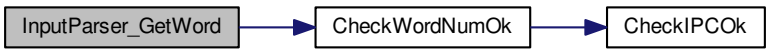
6.38.2.8 unsigned int InputParser_GetUpcaseWord (struct **InputParserC** * *ipc*, unsigned int *num*, char * *wheretostore*, unsigned int *storagesize*)

Here is the call graph for this function:



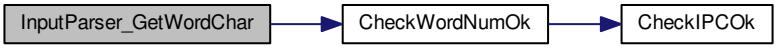
6.38.2.9 unsigned int InputParser_GetWord (struct **InputParserC** * *ipc*, unsigned int *num*, char * *wheretostore*, unsigned int *storagesize*)

Here is the call graph for this function:



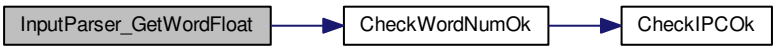
6.38.2.10 char InputParser_GetWordChar (struct **InputParserC** * *ipc*, unsigned int *num*, unsigned int *pos*)

Here is the call graph for this function:



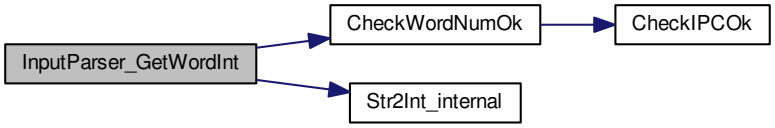
6.38.2.11 float InputParser_GetWordFloat (struct InputParserC * *ipc,* unsigned int *num*)

Here is the call graph for this function:



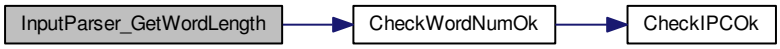
6.38.2.12 signed int InputParser_GetWordInt (struct InputParserC * *ipc,* unsigned int *num*)

Here is the call graph for this function:



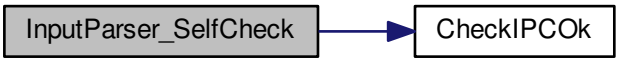
6.38.2.13 unsigned int InputParser_GetWordLength (struct InputParserC * *ipc,* unsigned int *num*)

Here is the call graph for this function:



6.38.2.14 unsigned char InputParser_SelfCheck (struct InputParserC * *ipc*)

Here is the call graph for this function:

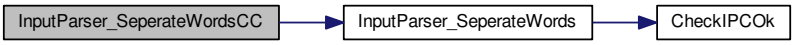


6.38.2.15 int InputParser_SeperateWords (struct InputParserC * *ipc*, char * *inpt*, char *keepcopy*)

Here is the call graph for this function:

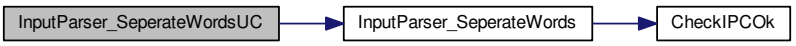
6.38.2.16 int InputParser_SeperateWordsCC (struct InputParserC * *ipc*, const char * *inpt*, char *keepcopy*)

Here is the call graph for this function:



6.38.2.17 int InputParser_SeperateWordsUC (struct InputParserC * *ipc*, unsigned char * *inpt*, char *keepcopy*)

Here is the call graph for this function:

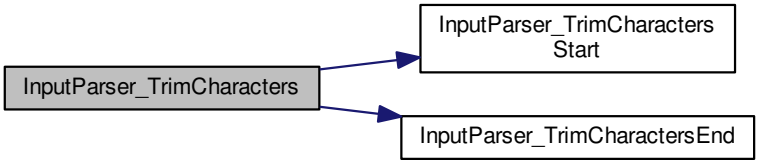


6.38.2.18 void InputParser_SetDelimeter (struct InputParserC * *ipc*, int *num*, char *tmp*)

Here is the call graph for this function:

6.38.2.19 int InputParser_TrimCharacters (char * *inpt*, unsigned int *length*, char *what2trim*)

Here is the call graph for this function:



6.38.2.20 int InputParser_TrimCharactersEnd (char * *inpt*, unsigned int *length*, char *what2trim*)

6.38.2.21 int InputParser_TrimCharactersStart (char * *inpt*, unsigned int *length*, char *what2trim*)

6.38.2.22 unsigned char InputParser_WordCompare (struct **InputParserC** * *ipc*, unsigned int *num*, char * *word*, unsigned int *wordsize*)

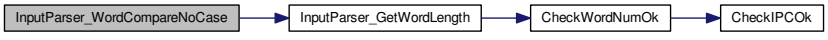
Here is the call graph for this function:

6.38.2.23 unsigned char InputParser_WordCompareAuto (struct **InputParserC** * *ipc*, unsigned int *num*, char * *word*)

Here is the call graph for this function:

6.38.2.24 unsigned char InputParser_WordCompareNoCase (struct **InputParserC** * *ipc*, unsigned int *num*, char * *word*, unsigned int *wordsize*)

Here is the call graph for this function:



6.38.2.25 unsigned char InputParser_WordCompareNoCaseAuto (struct InputParserC * *ipc,* unsigned int *num,* char * *word*)

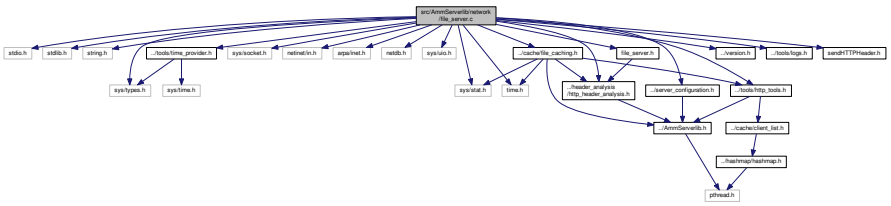
Here is the call graph for this function:

6.38.2.26 char* InputParserC_Version ( )

6.39 src/AmmServerlib/network/file_server.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <sys/uio.h>
#include <sys/stat.h>
#include <time.h>
#include "../version.h"
#include "file_server.h"
#include "../cache/file_caching.h"
#include "../header_analysis/http_header_analysis.h"
#include "../server_configuration.h"
#include "../tools/http_tools.h"
#include "../tools/time_provider.h"
#include "../tools/logs.h"
#include "sendHTTPHeader.h"
```

Include dependency graph for file_server.c:



Functions

- int SendPart (int clientsock, char *message, unsigned int message_size)
- int TransmitFileToSocketInternal (FILE *pFile, int clientsock, unsigned long bytesToSendStart)
- int TransmitFileToSocket (int clientsock, char *verified_filename, unsigned long start_at_byte, unsigned long end_at_byte)
- unsigned long SendFile (struct AmmServer_Instance *instance, struct HTTPTransaction *transaction, char *verified_filename_pending_copy, unsigned int force_error_code)

Send a File to a client.

- unsigned long SendErrorFile (struct AmmServer_Instance *instance, struct HTTPTransaction *transaction, unsigned int errorCode)

  Send an Error "file" response to a client , this is just a wrapper for a SendFile call with a force_error_code set.

- unsigned long SendMemoryBlockAsFile (char *filename, int clientsock, char *mem, unsigned long mem_block)

  Send a memory block to a client as a file.

Variables

- unsigned int files_open = 0

6.39.1 Function Documentation

6.39.1.1 unsigned long SendErrorFile (struct AmmServer_Instance * *instance,* struct HTTPTransaction * *transaction,* unsigned int *errorCode*)

Send an Error "file" response to a client , this is just a wrapper for a SendFile call with a force_error_code set.

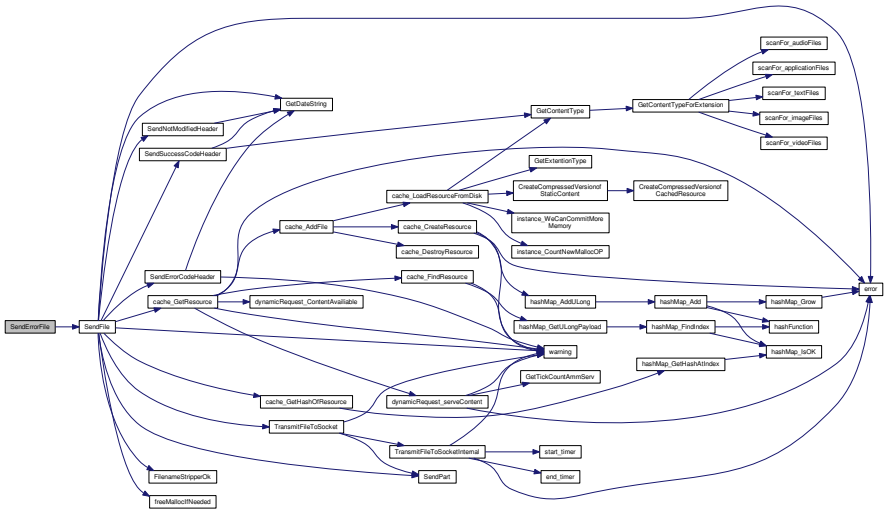
Parameters

An	AmmarServer Instance
HTTP-Transaction	this send file is part of
Error	Code to send

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.39.1.2 unsigned long SendFile (struct AmmServer_Instance * *instance,* struct HTTPTransaction * *transaction,* char * *verified_filename_pending_copy,* unsigned int *force_error_code*)

Send a File to a client.

Parameters

An	AmmarServer Instance
HTTP-Transaction	this send file is part of
Filename	that has been verified but has not been copied to the http checked for safety
Force	SendFile to fail with a specific error code (0 = dont force error)

Return values

1=Success,0=Failure	

Start sending the header first..! Due to error messages also having body payloads they are also handled here , creating clutter in the code but this way there is no need to write the same thing twice..! !

PRELIMINARY HEADER SENDING START ----------

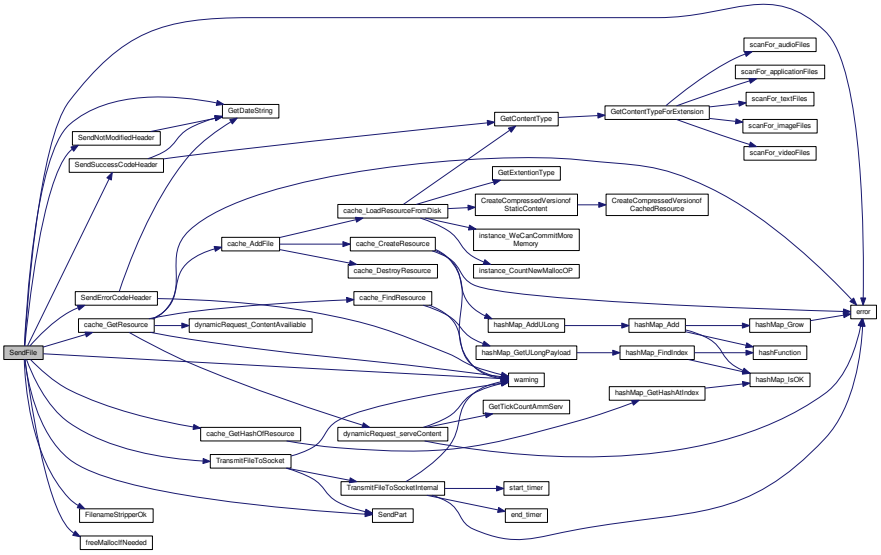
TODO Reorganize this : THIS SHOULD NOT BE SENT YET , SINCE WE MAY WANT TO EMMIT A 304 Not Modified Header if content is unmodified..!

PRELIMINARY HEADER SEND END ----------

Serve cached file !

Serve file by reading it from disk !

Here is the call graph for this function:



6.39.1.3 unsigned long SendMemoryBlockAsFile (char * *filename,* int *clientsock,* char * *mem,* unsigned long *mem_block*)

Send a memory block to a client as a file.

Parameters

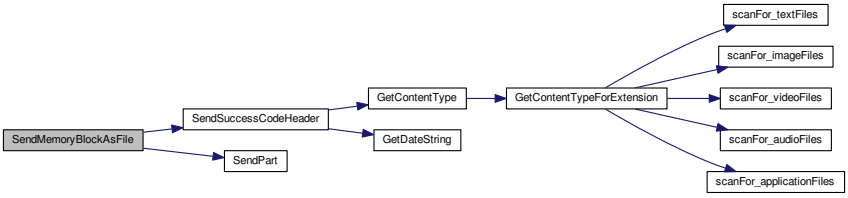
Filename	to pretend that we are sending for
Socket	we want to write to
Pointer	to memory that holds what we want to send to the client

Length	of memory block we want to send

Return values

1=Success,0=Failure	

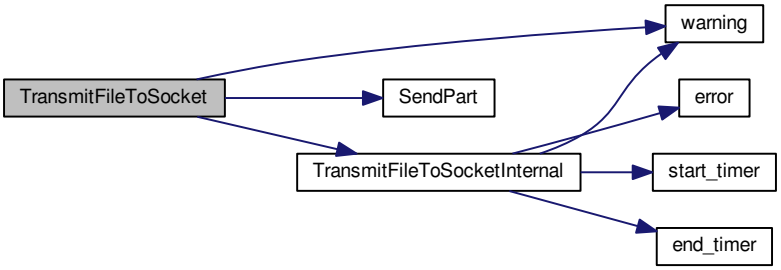
Here is the call graph for this function:



6.39.1.4 int SendPart (int *clientsock,* char * *message,* unsigned int *message_size*)

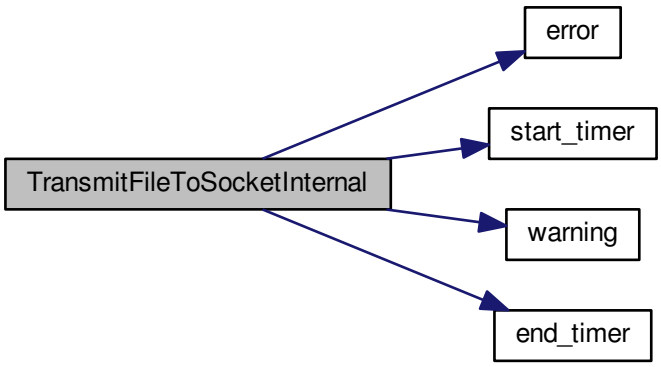
6.39.1.5 int TransmitFileToSocket (int *clientsock,* char * *verified_filename,* unsigned long *start_at_byte,* unsigned long *end_at_byte*)

Here is the call graph for this function:



6.39.1.6 int TransmitFileToSocketInternal (FILE * *pFile,* int *clientsock,* unsigned long *bytesToSendStart*) [inline]

Here is the call graph for this function:



6.39.2 Variable Documentation

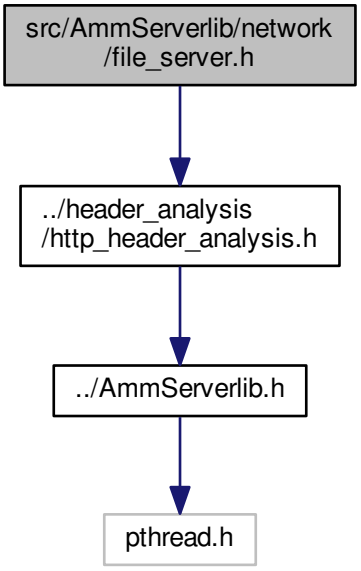
6.39.2.1 unsigned int files_open = 0

6.40 src/AmmServerlib/network/file_server.h File Reference

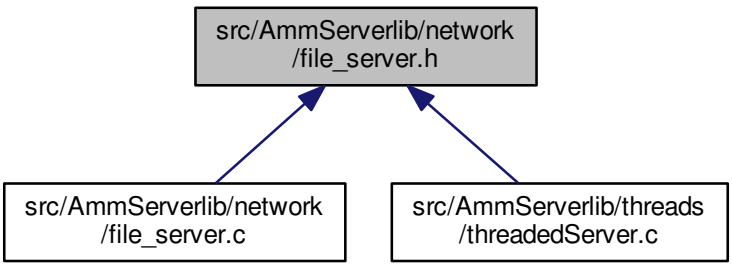
Basic file server functionality of AmmarServer.

```
#include "../header_analysis/http_header_analysis.h"
```

Include dependency graph for file_server.h:



This graph shows which files directly or indirectly include this file:



Functions

- unsigned long SendFile (struct AmmServer_Instance *instance, struct HTTPTransaction *transaction, char *verified_filename_pending_copy, unsigned int force_error_code)

  Send a File to a client.

- unsigned long SendErrorFile (struct AmmServer_Instance *instance, struct HTTPTransaction *transaction, unsigned int errorCode)

  Send an Error "file" response to a client , this is just a wrapper for a SendFile call with a force_error_code set.

- unsigned long SendMemoryBlockAsFile (char *filename, int clientsock, char *mem, unsigned long mem_block)

  Send a memory block to a client as a file.

6.40.1 Detailed Description

Basic file server functionality of AmmarServer.

Author

Ammar Qammaz (AmmarkoV)

6.40.2 Function Documentation

6.40.2.1 unsigned long SendErrorFile (struct AmmServer_Instance * *instance*, struct HTTPTransaction * *transaction*, unsigned int *errorCode*)

Send an Error "file" response to a client , this is just a wrapper for a SendFile call with a force_error_code set.

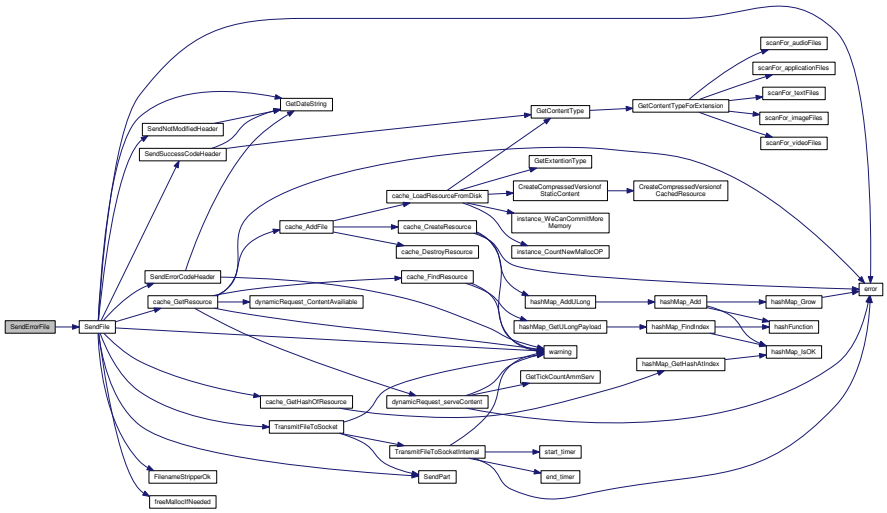
Parameters

An	AmmarServer Instance
HTTP-Transaction	this send file is part of
Error	Code to send

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.40.2.2 unsigned long SendFile (struct AmmServer_Instance * *instance*, struct HTTPTransaction * *transaction*, char * *verified_filename_pending_copy*, unsigned int *force_error_code*)

Send a File to a client.

Parameters

An	AmmarServer Instance
HTTP-Transaction	this send file is part of
Filename	that has been verified but has not been copied to the http checked for safety
Force	SendFile to fail with a specific error code (0 = dont force error)

Return values

1=Success,0=Failure	

Start sending the header first..! Due to error messages also having body payloads they are also handled here , creating clutter in the code but this way there is no need to write the same thing twice..! !

PRELIMINARY HEADER SENDING START ----------

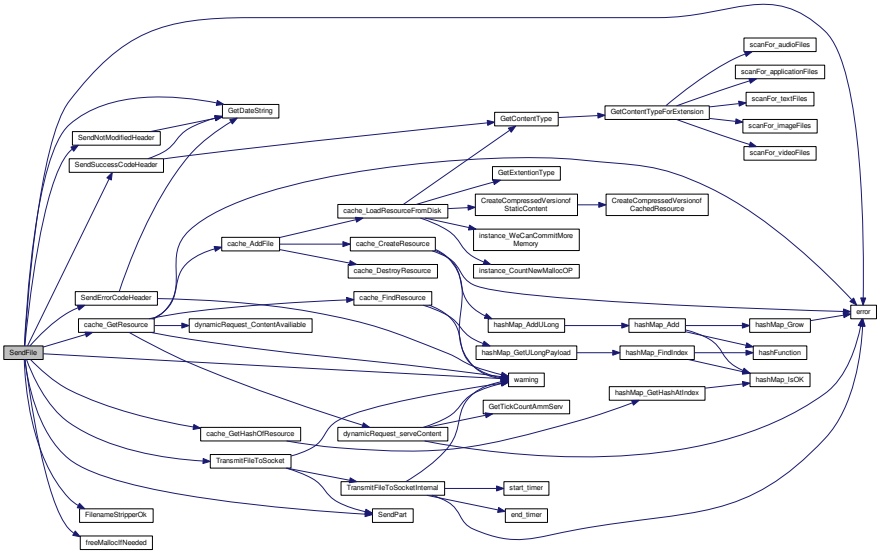
TODO Reorganize this : THIS SHOULD NOT BE SENT YET , SINCE WE MAY WANT TO EMMIT A 304 Not Modified Header if content is unmodified..!

PRELIMINARY HEADER SEND END ----------

Serve cached file !

Serve file by reading it from disk !

Here is the call graph for this function:



6.40.2.3 unsigned long SendMemoryBlockAsFile (char * *filename,* int *clientsock,* char * *mem,* unsigned long *mem_block*)

Send a memory block to a client as a file.

Parameters

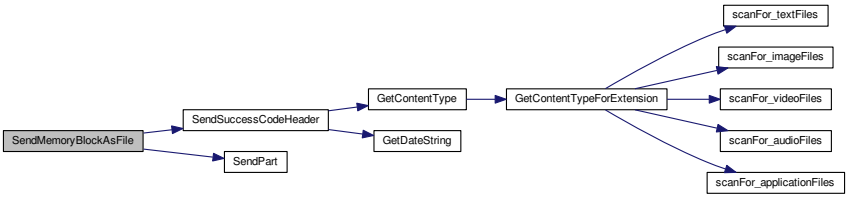
Filename	to pretend that we are sending for
Socket	we want to write to
Pointer	to memory that holds what we want to send to the client

Length	of memory block we want to send

Return values

1=Success,0=Failure	

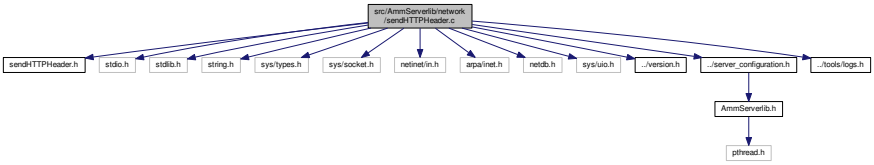
Here is the call graph for this function:



6.41 src/AmmServerlib/network/sendHTTPHeader.c File Reference

```
#include "sendHTTPHeader.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <sys/uio.h>
#include "../version.h"
#include "../server_configuration.h"
#include "../tools/logs.h"
```

Include dependency graph for sendHTTPHeader.c:



Functions

- unsigned long SendErrorCodeHeader (int clientsock, unsigned int error_code, char ∗verified_filename, char ∗templates_root)

  Send an Error Code header.
- unsigned long SendSuccessCodeHeader (int clientsock, int success_code, char ∗verified_filename)

  Send a Success header , meaning that what was asked for will follow.
- unsigned long SendNotModifiedHeader (int clientsock)

  Send a 304 Not Modified response.
- unsigned long SendAuthorizationHeader (int clientsock, char ∗message, char ∗verified_filename)

  Send a 401 Not Authorized response.

6.41.1 Function Documentation

6.41.1.1 unsigned long SendAuthorizationHeader (int *clientsock,* char * *message,* char * *verified_filename*)

Send a 401 Not Authorized response.

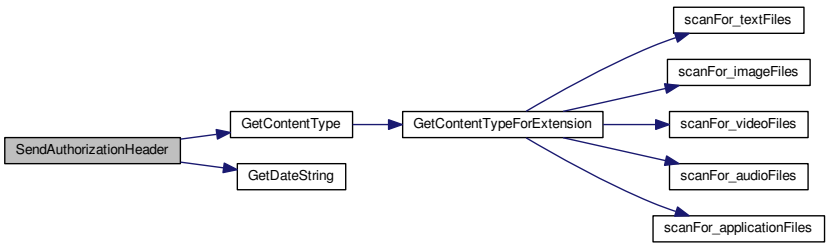
Parameters

Socket	to send to
String	with message to be sent
Verified	Filename of file asked to be transmitted

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.41.1.2 unsigned long SendErrorCodeHeader (int *clientsock,* unsigned int *error_code,* char * *verified_filename,* char * *templates_root*)

Send an Error Code header.

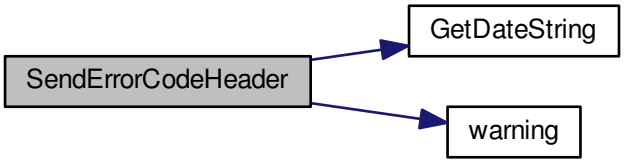
Parameters

Socket	to send to
ErrorCode	to be transmitted to client
Verified	Filename of file to transmit (appended with error code)
Filename	to directory when error template files are stored

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.41.1.3 unsigned long SendNotModifiedHeader (int *clientsock*)

Send a 304 Not Modified response.

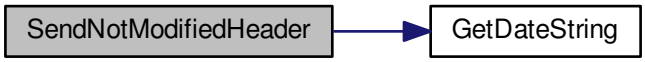
Parameters

Socket	to send to

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.41.1.4 unsigned long SendSuccessCodeHeader (int *clientsock,* int *success_code,* char * *verified_filename*)

Send a Success header , meaning that what was asked for will follow.

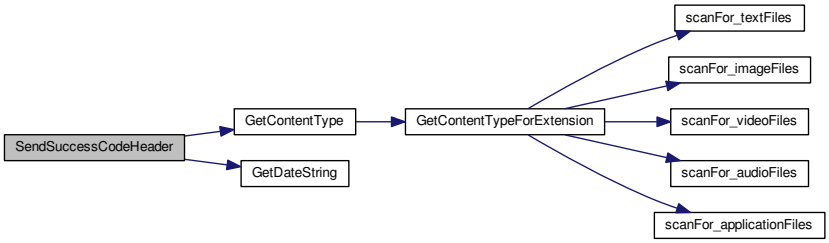
Parameters

Socket	to send to
Success	code (typically 200 ok)
Verified	Filename of file to transmit

Return values

1=Success,0=Failure	

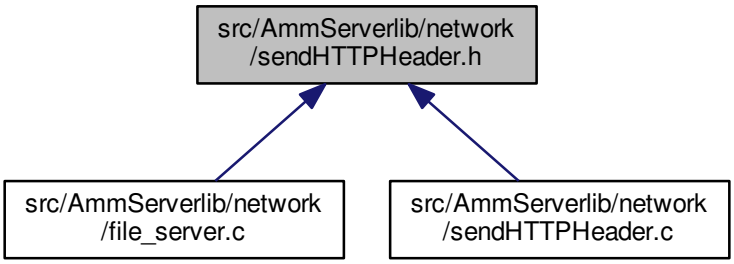
Here is the call graph for this function:



6.42 src/AmmServerlib/network/sendHTTPHeader.h File Reference

Small code segments that transmit HTTP responses.

This graph shows which files directly or indirectly include this file:



Functions

- unsigned long SendErrorCodeHeader (int clientsock, unsigned int error_code, char ∗verified_filename, char ∗templates_root)

  Send an Error Code header.
- unsigned long SendSuccessCodeHeader (int clientsock, int success_code, char ∗verified_filename)

  Send a Success header , meaning that what was asked for will follow.
- unsigned long SendNotModifiedHeader (int clientsock)

  Send a 304 Not Modified response.
- unsigned long SendAuthorizationHeader (int clientsock, char ∗message, char ∗verified_filename)

  Send a 401 Not Authorized response.

6.42.1 Detailed Description

Small code segments that transmit HTTP responses.

Author

Ammar Qammaz (AmmarkoV)

6.42.2 Function Documentation

6.42.2.1 unsigned long SendAuthorizationHeader (int *clientsock,* char * *message,* char * *verified_filename*)

Send a 401 Not Authorized response.

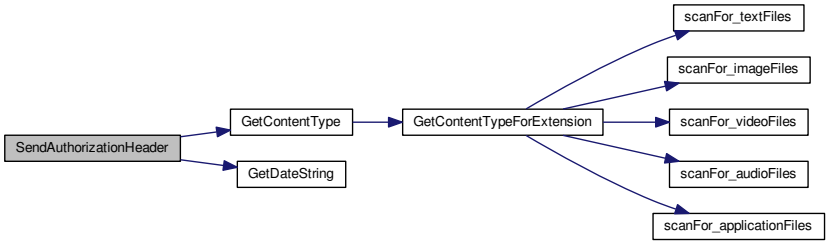
Parameters

Socket	to send to
String	with message to be sent
Verified	Filename of file asked to be transmitted

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.42.2.2 unsigned long SendErrorCodeHeader (int *clientsock,* unsigned int *error_code,* char * *verified_filename,* char * *templates_root*)

Send an Error Code header.

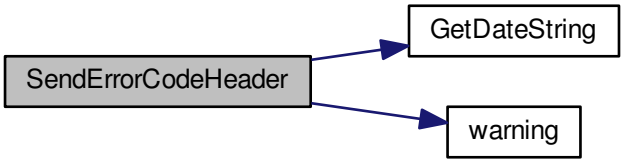
Parameters

Socket	to send to
ErrorCode	to be transmitted to client
Verified	Filename of file to transmit (appended with error code)
Filename	to directory when error template files are stored

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.42.2.3 unsigned long SendNotModifiedHeader (int *clientsock*)

Send a 304 Not Modified response.

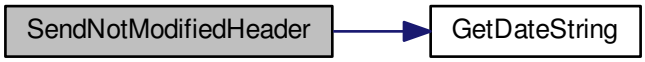
Parameters

Socket	to send to

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.42.2.4 unsigned long SendSuccessCodeHeader (int *clientsock,* int *success_code,* char * *verified_filename*)

Send a Success header , meaning that what was asked for will follow.

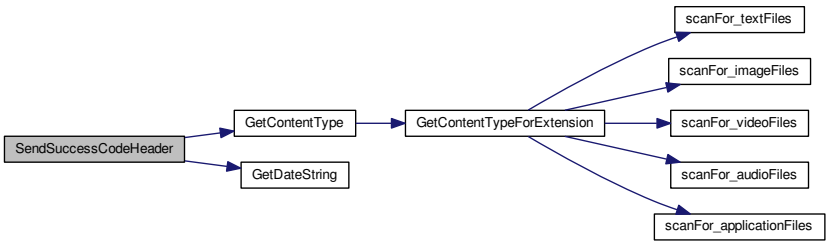
Parameters

Socket	to send to
Success	code (typically 200 ok)
Verified	Filename of file to transmit

Return values

1=Success,0=Failure	

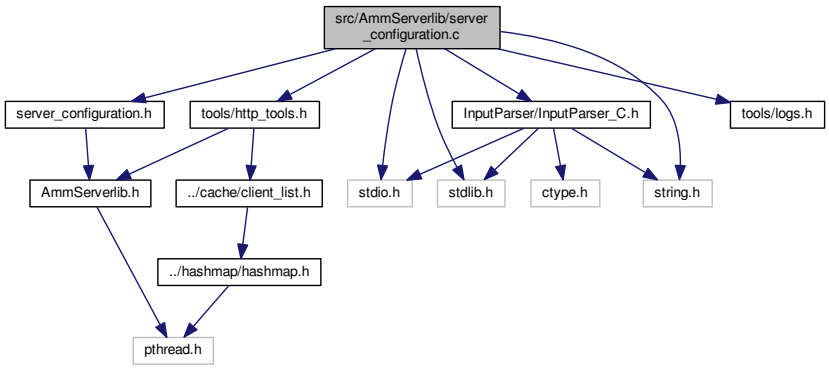
Here is the call graph for this function:



6.43 src/AmmServerlib/server_configuration.c File Reference

```
#include "server_configuration.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "tools/http_tools.h"
#include "tools/logs.h"
#include "InputParser/InputParser_C.h"
```

Include dependency graph for server_configuration.c:



Functions

- int instance_WeCanCommitMoreMemory (struct AmmServer_Instance *instance, unsigned long additional_mem_to_malloc_in_bytes)

  Check if we can commit more memory on an AmmarServer instance.
- int instance_CountNewMallocOP (struct AmmServer_Instance *instance, unsigned long additional_mem_to_malloc_in_bytes)

  Register a new memory Allocation to instance memory counters.
- int instance_CountFreeOP (struct AmmServer_Instance *instance, unsigned long additional_mem_to_malloc_in_bytes)

  Register a new memory free operation to instance memory counters.
- int EmmitPossibleConfigurationWarnings (struct AmmServer_Instance *instance)

- int LoadConfigurationFile (struct AmmServer_Instance *instance, char *conf_file)
- int AssignStr (char **dest, char *source)
- int SetUsernameAndPassword (struct AmmServer_Instance *instance, char *username, char *password)

  Set a username and password for clients to access specific webserver instance.

Variables

- unsigned int GLOBAL_KILL_SERVER_SWITCH = 0

  Setting this to 1 will signal that all instances of AmmarServer need to die at once.
- char USERNAME_UID_FOR_DAEMON [MAX_FILE_PATH] =DEFAULT_USERNAME_UID_FOR_DAEMON

  Default Username that initially gets set to DEFAULT_USERNAME_UID_FOR_DAEMON but can be changed through a configuration file.
- int CHANGE_TO_UID =NON_ROOT_UID_IF_USER_FAILS
- signed int CHANGE_PRIORITY =0

  Value that gets set from configuration files , and if it is non-zero it will trigger a priority change (change nice value)
- int varSocketTimeoutREAD_seconds = DEFAULT_SOCKET_READ_TIMEOUT_SECS
- int varSocketTimeoutWRITE_seconds =DEFAULT_SOCKET_WRITE_TIMEOUT_SECS
- unsigned char CACHING_ENABLED =1

  If caching is disabled server becomes a very simple file server , dynamic requests are also disabled.
- int MAX_SEPERATE_CACHE_ITEMS = 1024

  Maximum Number of separate items in cache (per instance of AmmarServer)
- int MAX_CACHE_SIZE_IN_MB = 128

  Maximum memory usage (Megabytes) for the entire cache (per instance of AmmarServer)
- int MAX_CACHE_SIZE_FOR_EACH_FILE_IN_MB = 3

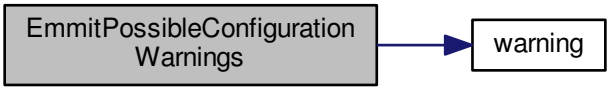
  Maximum memory usage (Megabytes) for a specific entry of the cache (per instance of AmmarServer)
- int AccessLogEnable =0
- char AccessLog [MAX_FILE_PATH] ="access.log"
- int ErrorLogEnable =0
- char ErrorLog [MAX_FILE_PATH] ="error.log"
- char TemplatesInternalURI [MAX_RESOURCE] = TEMPLATE_INTERNAL_URI

6.43.1 Function Documentation

6.43.1.1 int AssignStr (char ** *dest*, char * *source*)

6.43.1.2 int EmmitPossibleConfigurationWarnings (struct AmmServer_Instance * *instance*)

Here is the call graph for this function:



6.43.1.3 int instance_CountFreeOP (struct AmmServer_Instance * *instance*, unsigned long *additional_mem_to_malloc_in_bytes*)

Register a new memory free operation to instance memory counters.

Parameters

An	AmmarServer instance
Memory	that was freed

Return values

1=Success,0=Failure	

6.43.1.4 int instance_CountNewMallocOP (struct AmmServer_Instance * *instance*, unsigned long *additional_mem_to_malloc_in_bytes*)

Register a new memory Allocation to instance memory counters.

Parameters

An	AmmarServer instance
Memory	that was additionally allocated

Return values

1=Success,0=Failure	

6.43.1.5 int instance_WeCanCommitMoreMemory (struct AmmServer_Instance * *instance*, unsigned long *additional_mem_to_malloc_in_bytes*)

Check if we can commit more memory on an AmmarServer instance.

Parameters

An	AmmarServer instance
Memory	to additionally allocate

Return values

1=Ok,0=Don'tAllocate	

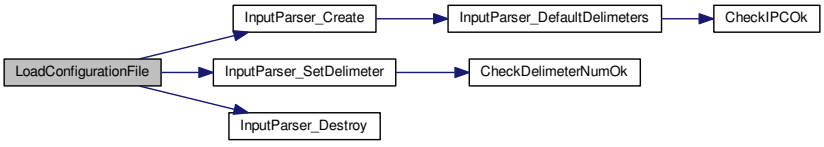
6.43.1.6 int LoadConfigurationFile (struct AmmServer_Instance * *instance*, char * *conf_file*)

Parameters

Load	a configuration file

Bug LoadConfigurationFiles etc is not ready yet , although it relies on InputParser and should be easy to implement , there are just things missing still and that's why I postpone implementing it

Here is the call graph for this function:



6.43.1.7 int SetUsernameAndPassword (struct AmmServer_Instance * *instance,* char * *username,* char * *password*)

Set a username and password for clients to access specific webserver instance.

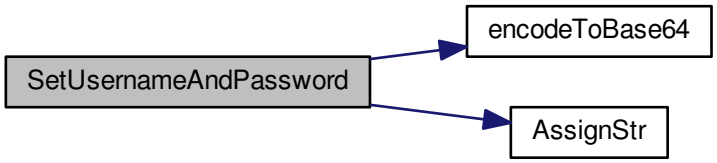
Parameters

An	AmmarServer instance
String	with new username
String	with new password

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.43.2 Variable Documentation

6.43.2.1 char AccessLog[MAX_FILE_PATH] ="access.log"

6.43.2.2 int AccessLogEnable =0

6.43.2.3 unsigned char CACHING_ENABLED =1

If caching is disabled server becomes a very simple file server , dynamic requests are also disabled.

6.43.2.4 signed int CHANGE_PRIORITY =0

Value that gets set from configuration files , and if it is non-zero it will trigger a priority change (change nice value)

6.43.2.5 int CHANGE_TO_UID =NON_ROOT_UID_IF_USER_FAILS

6.43.2.6 char ErrorLog[MAX_FILE_PATH] ="error.log"

6.43.2.7 int ErrorLogEnable =0

6.43.2.8 unsigned int GLOBAL_KILL_SERVER_SWITCH = 0

Setting this to 1 will signal that all instances of AmmarServer need to die at once.

6.43.2.9 int MAX_CACHE_SIZE_FOR_EACH_FILE_IN_MB = 3

Maximum memory usage (Megabytes) for a specific entry of the cache (per instance of AmmarServer)

6.43.2.10 int MAX_CACHE_SIZE_IN_MB = 128

Maximum memory usage (Megabytes) for the entire cache (per instance of AmmarServer)

6.43.2.11 int MAX_SEPERATE_CACHE_ITEMS = 1024

Maximum Number of separate items in cache (per instance of AmmarServer)

6.43.2.12 char TemplatesInternalURI[MAX_RESOURCE] = TEMPLATE_INTERNAL_URI

6.43.2.13 char USERNAME_UID_FOR_DAEMON[MAX_FILE_PATH] =DEFAULT_USERNAME_UID_FOR_DAEMON

Default Username that initially gets set to DEFAULT_USERNAME_UID_FOR_DAEMON but can be changed through a configuration file.

6.43.2.14 int varSocketTimeoutREAD_seconds = DEFAULT_SOCKET_READ_TIMEOUT_SECS

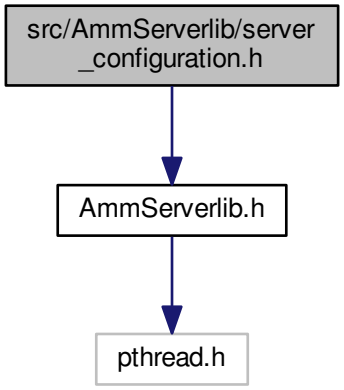
6.43.2.15 int varSocketTimeoutWRITE_seconds =DEFAULT_SOCKET_WRITE_TIMEOUT_SECS

6.44 src/AmmServerlib/server_configuration.h File Reference

The Main Header for the settings used by AmmarServer.

```
#include "AmmServerlib.h"
```

Include dependency graph for server_configuration.h:



This graph shows which files directly or indirectly include this file:

Macros

- #define THREAD_SLEEP_TIME_WHEN_OUR_PRESPAWNED_THREAD_IS_NEXT 700

  Next prespawned thread , should be vigilant and ready to serve so it has a shorter delay than the other prespawned threads (0.7ms max delay seems like a good value)
- #define THREAD_SLEEP_TIME_FOR_PRESPAWNED_THREADS 25000

  Sleep time for threads that are prespawned until they check for potential new work , the lowest the value here , the shortest the wait time for clients , but this causes higher CPU usage (for idle tasks) and ultimately more power consumption A good default time is 25000 , (25ms)
- #define CALCULATE_TIME_FOR_UPLOADS 1

  Calculate (And output) transmission speed for files broadcast by AmmarServer.
- #define COMPILE_WITH_CLIENT_LIST 0

  Precompiler switch that controls baking in (or not) the client list capabilities , currently disabled since client lists are not yet implemented.
- #define MAX_CLIENTS_LISTENING_FOR 5000

  Maximum Target of concurrent clients being listened at the same time C10K tests require this to be 10000 (`http://en.wikipedia.org/wiki/C10k_problem` *)*
- #define MAX_CLIENT_THREADS 3000

  Maximum Number of concurrent threads being created at the same time , depending on the size of the listen pool this can be smaller than the MAX_CLIENTS_LISTENING_FOR and connections will be queued and served sequentially.
- #define MAX_CLIENT_PRESPAWNED_THREADS 0

  Prespawned theads reduce overall latency but they increase CPU load , 0 disables them.
- #define MAX_CLIENTS_PER_IP 3

  Maximum connections per IP , this is a little dangerous since multiple PC's can have a single gateway , but it is a good heuristic to better share resources.
- #define MAX_HTTP_REQUEST_HEADER_LINES 1024

  An incoming header should not have more than X numbers of lines.
- #define MAX_RESOURCE_SLASHES 15

  Max slashes in a Resource (i.e. `http://xxx.xxx.xxx.xxx/test/resource` *has 4 slashes.*
- #define MAX_CONFIGURATION_FILE_LINE_SIZE 512

  Maximum line length in configuration file.
- #define MAX_CONTENT_TYPE 128

  Maximum length of a content type record.
- #define MAX_FILE_READ_BLOCK_KB 1024

  Length of blocks allocated , read and sent in order to transmit a file to a client , bigger values read faster from the disk and possibly better utilize bandwidth in the expense of memory consumption.
- #define MAX_HTTP_REQUEST_HEADER 4096

  Maximum size of an incoming HTTP Header.
- #define HTTP_POST_GROWTH_STEP_REQUEST_HEADER 512/∗KB∗/∗1024

  Maximum size of an incoming HTTP Header allocation step.
- #define MAX_HTTP_POST_REQUEST_HEADER 4/∗MB∗/∗1024∗1024

  Maximum size of an incoming POST Header , since it carries files this should be big enough (say 4 MB)
- #define MAX_HTTP_REQUEST_HEADER_REPLY 1024

  Maximum size of an http header reply.
- #define INITIAL_DIRECTORY_LIST_RESPONSE_BODY 64/∗KB∗/∗1024

  Controls initial allocated size for a directory listing.
- #define GROWSTEP_DIRECTORY_LIST_RESPONSE_BODY 16/∗KB∗/∗1024

  Controls allocation step for when we run out of space for a directory listing.
- #define MAX_DIRECTORY_LIST_RESPONSE_BODY 128/∗KB∗/∗1024

  Maximum space allocated for a directory listing.
- #define REALLOC_TO_SAVE_MORE_THAN_THIS_NUMBER_BYTES 4096

When we compress a file we may have a buffer allocated for 16KB and the compressed size might be 1.6KB (if we get an impressive 1:10 ratio) If that's the case we could do a system call to free memory and allocate a 1.6KB chunk of memory thus being economic in memory requirements.

- #define ENABLE_AUTOMATIC_CONFIGURATION_LOADING 1

  If this enabled and we haven't specified a configuration file we will try to open an ammarServer.conf.
- #define ENABLE_POST 1

  Enable POST request handling , switching this to 0 will completely deny them reducing attack surface.
- #define ENABLE_COMPRESSION 0

  Enable Compression using ZLib , this increases CPU usage , code surface , requires the zlib library to be linked , but on the other hand conserves bandwidth and memory.
- #define ENABLE_DYNAMIC_CONTENT_COMPRESSION 0

  Enable Compression for dynamic content , this can be tuned per dynamic resource , but this is a global switch for all nodes This generally doesnt seem like a very good idea unless you have a dynamic html file of 20KB+ with very rare changes to compensate for the overhead.
- #define ENABLE_DROPPING_ROOT_UID_IF_ROOT 1

  In order to bind ports under 1000 , a process needs to have Super user UID , after we bind the port we really don't want to have our process running as a super user , it is a serious security liability This should always be 1.
- #define ENABLE_DROPPING_UID_ALWAYS 0

  If this is enabled we will always change our UID no matter if we are a super user or not (if this is disabled only super user processes will get the UID change)
- #define DEFAULT_USERNAME_UID_FOR_DAEMON "www-data"

  Default Username to change to if we are running from root.
- #define NON_ROOT_UID_IF_USER_FAILS 1500

  Non Root UID to change to.
- #define ENABLE_INTERNAL_RESOURCES_RESOLVE 1

  Resolve internal resources to redirect them to point templates (this should always be 1 , although its implementation is a little dodgy right now)
- #define ENABLE_DIRECTORY_LISTING 1

  Enable directory listing , if this is disabled attack surface gets significantly reduced.
- #define EPOCH_YEAR_IN_TM_YEAR 1900

  TM structures carry the year after 1900 (see http://www.cplusplus.com/reference/ctime/tm/ *) so this is encoded here as a reminder.*
- #define DEFAULT_SOCKET_READ_TIMEOUT_SECS 4

  Default timeout value before which a socket blocking on a read call should be considered dead.
- #define DEFAULT_SOCKET_WRITE_TIMEOUT_SECS 4

  Default timeout value before which a socket blocking on a write call should be considered dead.
- #define TEMPLATE_INTERNAL_URI "_asvres_/"

  String that corresponds to the template directory (for directory_lists)

Functions

- int instance_WeCanCommitMoreMemory (struct AmmServer_Instance *instance, unsigned long additional_mem_to_malloc_in_bytes)

  Check if we can commit more memory on an AmmarServer instance.
- int instance_CountNewMallocOP (struct AmmServer_Instance *instance, unsigned long additional_mem_to_malloc_in_bytes)

  Register a new memory Allocation to instance memory counters.
- int instance_CountFreeOP (struct AmmServer_Instance *instance, unsigned long additional_mem_to_malloc_in_bytes)

  Register a new memory free operation to instance memory counters.
- int EmmitPossibleConfigurationWarnings ()

  Internal check of server configuration and possible error messages in impossible situations.
- int LoadConfigurationFile (struct AmmServer_Instance *instance, char *conf_file)

- int AssignStr (char **dest, char *source)
- int SetUsernameAndPassword (struct AmmServer_Instance *instance, char *username, char *password)

  Set a username and password for clients to access specific webserver instance.

Variables

- unsigned int GLOBAL_KILL_SERVER_SWITCH

  Setting this to 1 will signal that all instances of AmmarServer need to die at once.
- char USERNAME_UID_FOR_DAEMON [MAX_FILE_PATH]

  Default Username that initially gets set to DEFAULT_USERNAME_UID_FOR_DAEMON but can be changed through a configuration file.
- int CHANGE_TO_UID
- int CHANGE_PRIORITY

  Value that gets set from configuration files , and if it is non-zero it will trigger a priority change (change nice value)
- int varSocketTimeoutREAD_seconds
- int varSocketTimeoutWRITE_seconds
- unsigned char CACHING_ENABLED

  If caching is disabled server becomes a very simple file server , dynamic requests are also disabled.
- int MAX_SEPERATE_CACHE_ITEMS

  Maximum Number of separate items in cache (per instance of AmmarServer)
- int MAX_CACHE_SIZE_IN_MB

  Maximum memory usage (Megabytes) for the entire cache (per instance of AmmarServer)
- int MAX_CACHE_SIZE_FOR_EACH_FILE_IN_MB

  Maximum memory usage (Megabytes) for a specific entry of the cache (per instance of AmmarServer)
- int AccessLogEnable
- char AccessLog [MAX_FILE_PATH]
- int ErrorLogEnable
- char ErrorLog [MAX_FILE_PATH]
- char TemplatesInternalURI [MAX_RESOURCE]

6.44.1 Detailed Description

The Main Header for the settings used by AmmarServer. Take extra care when changing something here , since its impact is global

Author

Ammar Qammaz (AmmarkoV)

Bug Server configuration at some point should be ported from defines to a per instance configuration file , some of these defines will always remain since they control global allocations

6.44.2 Macro Definition Documentation

6.44.2.1 #define CALCULATE_TIME_FOR_UPLOADS 1

Calculate (And output) transmission speed for files broadcast by AmmarServer.

6.44.2.2 #define COMPILE_WITH_CLIENT_LIST 0

Precompiler switch that controls baking in (or not) the client list capabilities , currently disabled since client lists are not yet implemented.

6.44.2.3 #define DEFAULT_SOCKET_READ_TIMEOUT_SECS 4

Default timeout value before which a socket blocking on a read call should be considered dead.

6.44.2.4 #define DEFAULT_SOCKET_WRITE_TIMEOUT_SECS 4

Default timeout value before which a socket blocking on a write call should be considered dead.

6.44.2.5 #define DEFAULT_USERNAME_UID_FOR_DAEMON "www-data"

Default Username to change to if we are running from root.

6.44.2.6 #define ENABLE_AUTOMATIC_CONFIGURATION_LOADING 1

If this enabled and we haven't specified a configuration file we will try to open an ammarServer.conf.

6.44.2.7 #define ENABLE_COMPRESSION 0

Enable Compression using ZLib , this increases CPU usage , code surface , requires the zlib library to be linked , but on the other hand conserves bandwidth and memory.

6.44.2.8 #define ENABLE_DIRECTORY_LISTING 1

Enable directory listing , if this is disabled attack surface gets significantly reduced.

6.44.2.9 #define ENABLE_DROPPING_ROOT_UID_IF_ROOT 1

In order to bind ports under 1000 , a process needs to have Super user UID , after we bind the port we *really* don't want to have our process running as a super user , it is a serious security liability This should always be 1.

6.44.2.10 #define ENABLE_DROPPING_UID_ALWAYS 0

If this is enabled we will always change our UID no matter if we are a super user or not (if this is disabled only super user processes will get the UID change)

6.44.2.11 #define ENABLE_DYNAMIC_CONTENT_COMPRESSION 0

Enable Compression for dynamic content , this can be tuned per dynamic resource , but this is a global switch for all nodes This generally doesnt seem like a very good idea unless you have a dynamic html file of 20KB+ with very rare changes to compensate for the overhead.

6.44.2.12 #define ENABLE_INTERNAL_RESOURCES_RESOLVE 1

Resolve internal resources to redirect them to point templates (this should always be 1 , although its implementation is a little dodgy right now)

6.44.2.13 #define ENABLE_POST 1

Enable POST request handling , switching this to 0 will completely deny them reducing attack surface.

6.44.2.14 #define EPOCH_YEAR_IN_TM_YEAR 1900

TM structures carry the year after 1900 (see http://www.cplusplus.com/reference/ctime/tm/) so this is encoded here as a reminder.

6.44.2.15 #define GROWSTEP_DIRECTORY_LIST_RESPONSE_BODY 16/*KB*/*1024

Controls allocation step for when we run out of space for a directory listing.

6.44.2.16 #define HTTP_POST_GROWTH_STEP_REQUEST_HEADER 512/*KB*/*1024

Maximum size of an incoming HTTP Header allocation step.

6.44.2.17 #define INITIAL_DIRECTORY_LIST_RESPONSE_BODY 64/*KB*/*1024

Controls initial allocated size for a directory listing.

6.44.2.18 #define MAX_CLIENT_PRESPAWNED_THREADS 0

Prespawned theads reduce overall latency but they increase CPU load , 0 disables them.

6.44.2.19 #define MAX_CLIENT_THREADS 3000

Maximum Number of concurrent threads being created at the same time , depending on the size of the listen pool this can be smaller than the MAX_CLIENTS_LISTENING_FOR and connections will be queued and served sequentially.

6.44.2.20 #define MAX_CLIENTS_LISTENING_FOR 5000

Maximum Target of concurrent clients being listened at the same time C10K tests require this to be 10000 (http://en.wikipedia.org/wiki/C10k_problem)

6.44.2.21 #define MAX_CLIENTS_PER_IP 3

Maximum connections per IP , this is a little dangerous since multiple PC's can have a single gateway , but it is a good heuristic to better share resources.

Bug MAX_CLIENTS_PER_IP is not used if there is no client list declared

6.44.2.22 #define MAX_CONFIGURATION_FILE_LINE_SIZE 512

Maximum line length in configuration file.

6.44.2.23 #define MAX_CONTENT_TYPE 128

Maximum length of a content type record.

6.44.2.24 #define MAX_DIRECTORY_LIST_RESPONSE_BODY 128/*KB*/*1024

Maximum space allocated for a directory listing.

6.44.2.25 #define MAX_FILE_READ_BLOCK_KB 1024

Length of blocks allocated , read and sent in order to transmit a file to a client , bigger values read faster from the disk and possibly better utilize bandwidth in the expense of memory consumption.

6.44.2.26 #define MAX_HTTP_POST_REQUEST_HEADER 4/*MB*/*1024*1024

Maximum size of an incoming POST Header , since it carries files this should be big enough (say 4 MB)

6.44.2.27 #define MAX_HTTP_REQUEST_HEADER 4096

Maximum size of an incoming HTTP Header.

6.44.2.28 #define MAX_HTTP_REQUEST_HEADER_LINES 1024

An incoming header should not have more than X numbers of lines.

6.44.2.29 #define MAX_HTTP_REQUEST_HEADER_REPLY 1024

Maximum size of an http header reply.

6.44.2.30 #define MAX_RESOURCE_SLASHES 15

Max slashes in a Resource (i.e. `http://xxx.xxx.xxx.xxx/test/resource` has 4 slashes.

6.44.2.31 #define NON_ROOT_UID_IF_USER_FAILS 1500

Non Root UID to change to.

6.44.2.32 #define REALLOC_TO_SAVE_MORE_THAN_THIS_NUMBER_BYTES 4096

When we compress a file we may have a buffer allocated for 16KB and the compressed size might be 1.6KB (if we get an impressive 1:10 ratio) If that's the case we could do a system call to free memory and allocate a 1.6KB chunk of memory thus being economic in memory requirements.

6.44.2.33 #define TEMPLATE_INTERNAL_URI "_asvres_/"

String that corresponds to the template directory (for directory_lists)

Bug Please note that the file server has limits for filenames so this should not be very long *asvres*/filename.jpg is OK a filename like *asvres*/filenamemplampla.jpg will return a 404

6.44.2.34 #define THREAD_SLEEP_TIME_FOR_PRESPAWNED_THREADS 25000

Sleep time for threads that are prespawned until they check for potential new work , the lowest the value here , the shortest the wait time for clients , but this causes higher CPU usage (for idle tasks) and ultimately more power consumption A good default time is 25000 , (25ms)

6.44.2.35 #define THREAD_SLEEP_TIME_WHEN_OUR_PRESPAWNED_THREAD_IS_NEXT 700

Next prespawned thread , should be vigilant and ready to serve so it has a shorter delay than the other prespawned threads (0.7ms max delay seems like a good value)

6.44.3 Function Documentation

6.44.3.1 int AssignStr (char ** *dest,* char * *source*)

6.44.3.2 int EmmitPossibleConfigurationWarnings ( )

Internal check of server configuration and possible error messages in impossible situations.

Return values

1=Success,0=Failure	

6.44.3.3 int instance_CountFreeOP (struct AmmServer_Instance * *instance,* unsigned long *additional_mem_to_malloc_in_bytes*)

Register a new memory free operation to instance memory counters.

Parameters

An	AmmarServer instance
Memory	that was freed

Return values

1=Success,0=Failure	

6.44.3.4 int instance_CountNewMallocOP (struct AmmServer_Instance * *instance,* unsigned long *additional_mem_to_malloc_in_bytes*)

Register a new memory Allocation to instance memory counters.

Parameters

An	AmmarServer instance
Memory	that was additionally allocated

Return values

1=Success,0=Failure	

6.44.3.5 int instance_WeCanCommitMoreMemory (struct AmmServer_Instance * *instance,* unsigned long *additional_mem_to_malloc_in_bytes*)

Check if we can commit more memory on an AmmarServer instance.

Parameters

An	AmmarServer instance

Memory	to additionally allocate

Return values

1=Ok,0=Don'tAllocate	

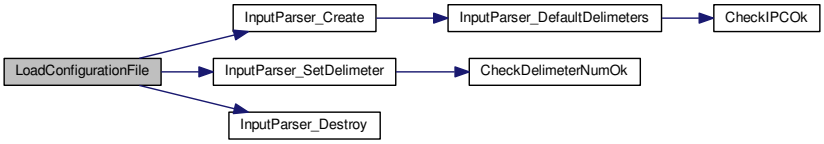
6.44.3.6 int LoadConfigurationFile (struct AmmServer_Instance * *instance*, char * *conf_file*)

Parameters

Load	a configuration file

Bug LoadConfigurationFiles etc is not ready yet , although it relies on InputParser and should be easy to implement , there are just things missing still and that's why I postpone implementing it

Here is the call graph for this function:



6.44.3.7 int SetUsernameAndPassword (struct AmmServer_Instance * *instance*, char * *username*, char * *password*)

Set a username and password for clients to access specific webserver instance.

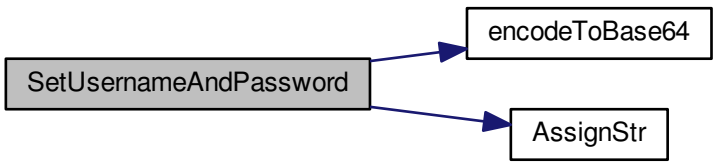
Parameters

An	AmmarServer instance
String	with new username
String	with new password

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.44.4 Variable Documentation

6.44.4.1 char AccessLog[MAX_FILE_PATH]

6.44.4.2 int AccessLogEnable

6.44.4.3 unsigned char CACHING_ENABLED

If caching is disabled server becomes a very simple file server , dynamic requests are also disabled.

6.44.4.4 int CHANGE_PRIORITY

Value that gets set from configuration files , and if it is non-zero it will trigger a priority change (change nice value)

6.44.4.5 int CHANGE_TO_UID

6.44.4.6 char ErrorLog[MAX_FILE_PATH]

6.44.4.7 int ErrorLogEnable

6.44.4.8 unsigned int GLOBAL_KILL_SERVER_SWITCH

Setting this to 1 will signal that all instances of AmmarServer need to die at once.

6.44.4.9 int MAX_CACHE_SIZE_FOR_EACH_FILE_IN_MB

Maximum memory usage (Megabytes) for a specific entry of the cache (per instance of AmmarServer)

6.44.4.10 int MAX_CACHE_SIZE_IN_MB

Maximum memory usage (Megabytes) for the entire cache (per instance of AmmarServer)

6.44.4.11 int MAX_SEPERATE_CACHE_ITEMS

Maximum Number of separate items in cache (per instance of AmmarServer)

6.44.4.12 char TemplatesInternalURI[MAX_RESOURCE]

6.44.4.13 char USERNAME_UID_FOR_DAEMON[MAX_FILE_PATH]

Default Username that initially gets set to DEFAULT_USERNAME_UID_FOR_DAEMON but can be changed through a configuration file.

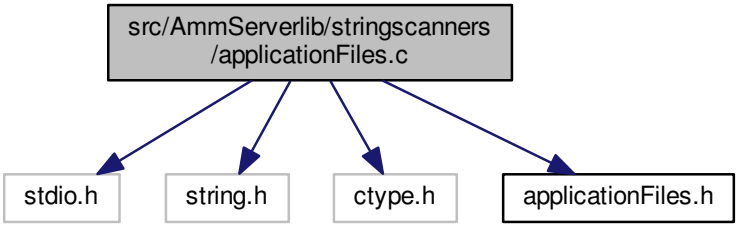
6.44.4.14 int varSocketTimeoutREAD_seconds

6.44.4.15 int varSocketTimeoutWRITE_seconds

6.45 src/AmmServerlib/stringscanners/applicationFiles.c File Reference

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include "applicationFiles.h"
```

Include dependency graph for applicationFiles.c:



Functions

- int scanFor_applicationFiles (char *str, unsigned int strLength)

  Scan a string for one of the words of the applicationFiles word set.

6.45.1 Function Documentation

6.45.1.1 int scanFor_applicationFiles (char * *str,* unsigned int *strLength*)

Scan a string for one of the words of the applicationFiles word set.

Parameters

Input	String , to be scanned
Length	of Input String

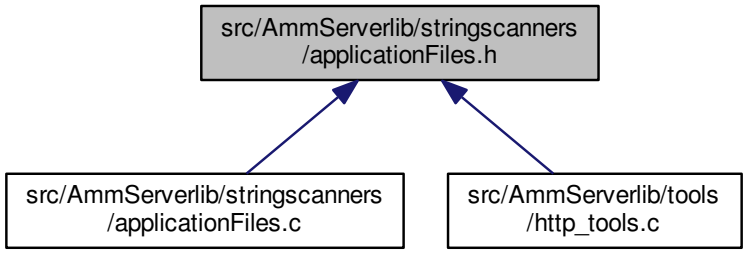
Return values

See	above enumerator

6.46 src/AmmServerlib/stringscanners/applicationFiles.h File Reference

A tool that scans for a string in a very fast and robust way.

This graph shows which files directly or indirectly include this file:



Enumerations

- enum {
  APPLICATIONFILES_EMPTY =0, APPLICATIONFILES_EXE, APPLICATIONFILES_DLL, APPLICATIONFILES_SCR,
  APPLICATIONFILES_CPL, APPLICATIONFILES_SWF, APPLICATIONFILES_PDF, APPLICATIONFILES_END_OF_ITEMS }

  Enumerator for the IDs of applicationFiles so we can know what the result was.

Functions

- int scanFor_applicationFiles (char *str, unsigned int strLength)

  Scan a string for one of the words of the applicationFiles word set.

6.46.1 Detailed Description

A tool that scans for a string in a very fast and robust way.

Author

Ammar Qammaz (AmmarkoV)

6.46.2 Enumeration Type Documentation

6.46.2.1 anonymous enum

Enumerator for the IDs of applicationFiles so we can know what the result was.

Enumerator

APPLICATIONFILES_EMPTY

APPLICATIONFILES_EXE

APPLICATIONFILES_DLL

APPLICATIONFILES_SCR

APPLICATIONFILES_CPL

APPLICATIONFILES_SWF

APPLICATIONFILES_PDF

APPLICATIONFILES_END_OF_ITEMS

6.46.3 Function Documentation

6.46.3.1 int scanFor_applicationFiles (char * *str,* unsigned int *strLength*)

Scan a string for one of the words of the applicationFiles word set.

Parameters

Input	String , to be scanned
Length	of Input String

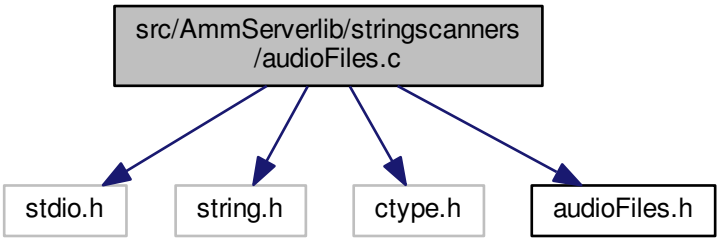
Return values

See	above enumerator

6.47 src/AmmServerlib/stringscanners/audioFiles.c File Reference

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include "audioFiles.h"
```

Include dependency graph for audioFiles.c:



Functions

- int scanFor_audioFiles (char *str, unsigned int strLength)

  Scan a string for one of the words of the audioFiles word set.

6.47.1 Function Documentation

6.47.1.1 int scanFor_audioFiles (char * *str,* unsigned int *strLength*)

Scan a string for one of the words of the audioFiles word set.

Parameters

Input	String , to be scanned
Length	of Input String

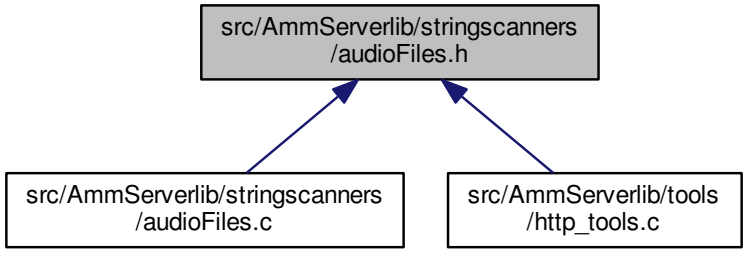
Return values

See	above enumerator

6.48 src/AmmServerlib/stringscanners/audioFiles.h File Reference

A tool that scans for a string in a very fast and robust way.

This graph shows which files directly or indirectly include this file:



Enumerations

- enum {
  AUDIOFILES_EMPTY =0, AUDIOFILES_MP3, AUDIOFILES_WAV, AUDIOFILES_MID,
  AUDIOFILES_OGG, AUDIOFILES_VOC, AUDIOFILES_AU, AUDIOFILES_END_OF_ITEMS }

  Enumerator for the IDs of audioFiles so we can know what the result was.

Functions

- int scanFor_audioFiles (char ∗str, unsigned int strLength)

  Scan a string for one of the words of the audioFiles word set.

6.48.1 Detailed Description

A tool that scans for a string in a very fast and robust way.

Author

Ammar Qammaz (AmmarkoV)

6.48.2 Enumeration Type Documentation

6.48.2.1 anonymous enum

Enumerator for the IDs of audioFiles so we can know what the result was.

Enumerator

AUDIOFILES_EMPTY

AUDIOFILES_MP3

AUDIOFILES_WAV

AUDIOFILES_MID

AUDIOFILES_OGG

AUDIOFILES_VOC

AUDIOFILES_AU

AUDIOFILES_END_OF_ITEMS

6.48.3 Function Documentation

6.48.3.1 int scanFor_audioFiles (char * *str,* unsigned int *strLength*)

Scan a string for one of the words of the audioFiles word set.

Parameters

Input	String , to be scanned
Length	of Input String

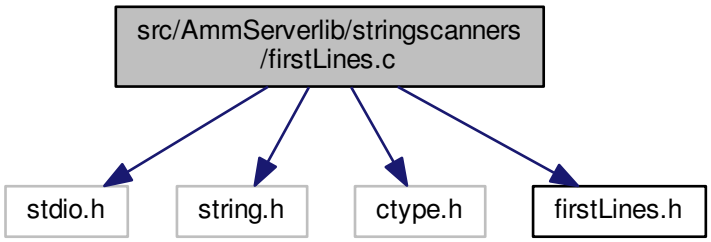
Return values

See	above enumerator

6.49 src/AmmServerlib/stringscanners/firstLines.c File Reference

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include "firstLines.h"
```

Include dependency graph for firstLines.c:



Functions

- int scanFor_firstLines (char *str, unsigned int strLength)

  Scan a string for one of the words of the firstLines word set.

6.49.1 Function Documentation

6.49.1.1 int scanFor_firstLines (char * *str*, unsigned int *strLength*)

Scan a string for one of the words of the firstLines word set.

Parameters

Input	String , to be scanned
Length	of Input String

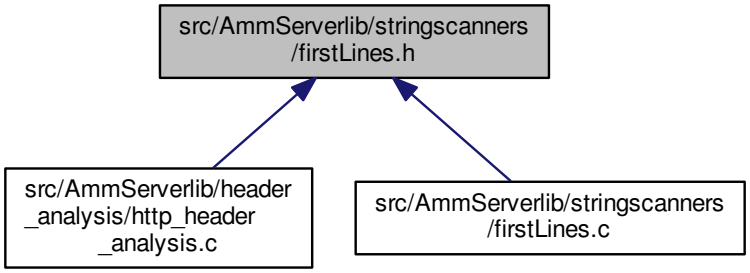
Return values

See	above enumerator

6.50 src/AmmServerlib/stringscanners/firstLines.h File Reference

A tool that scans for a string in a very fast and robust way.

This graph shows which files directly or indirectly include this file:



Enumerations

- enum {
  FIRSTLINES_EMPTY =0, FIRSTLINES_GET, FIRSTLINES_HEAD, FIRSTLINES_POST,
  FIRSTLINES_PUT, FIRSTLINES_DELETE, FIRSTLINES_TRACE, FIRSTLINES_OPTIONS,
  FIRSTLINES_CONNECT, FIRSTLINES_PATCH, FIRSTLINES_END_OF_ITEMS }

  Enumerator for the IDs of firstLines so we can know what the result was.

Functions

- int scanFor_firstLines (char *str, unsigned int strLength)

  Scan a string for one of the words of the firstLines word set.

6.50.1 Detailed Description

A tool that scans for a string in a very fast and robust way.

Author

Ammar Qammaz (AmmarkoV)

6.50.2 Enumeration Type Documentation

6.50.2.1 anonymous enum

Enumerator for the IDs of firstLines so we can know what the result was.

Enumerator

FIRSTLINES_EMPTY

FIRSTLINES_GET

FIRSTLINES_HEAD

FIRSTLINES_POST

FIRSTLINES_PUT

FIRSTLINES_DELETE

FIRSTLINES_TRACE

FIRSTLINES_OPTIONS

FIRSTLINES_CONNECT

FIRSTLINES_PATCH

FIRSTLINES_END_OF_ITEMS

6.50.3 Function Documentation

6.50.3.1 int scanFor_firstLines (char * *str,* unsigned int *strLength*)

Scan a string for one of the words of the firstLines word set.

Parameters

Input	String , to be scanned
Length	of Input String

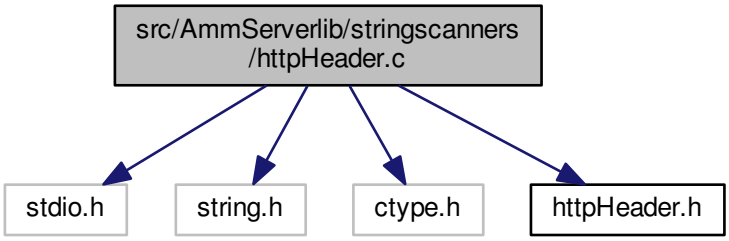
Return values

See	above enumerator

6.51 src/AmmServerlib/stringscanners/httpHeader.c File Reference

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include "httpHeader.h"
```

Include dependency graph for httpHeader.c:



Functions

- int scanFor_httpHeader (char *str, unsigned int strLength)

  Scan a string for one of the words of the httpHeader word set.

6.51.1 Function Documentation

6.51.1.1 int scanFor_httpHeader (char * *str,* unsigned int *strLength*)

Scan a string for one of the words of the httpHeader word set.

Parameters

Input	String , to be scanned
Length	of Input String

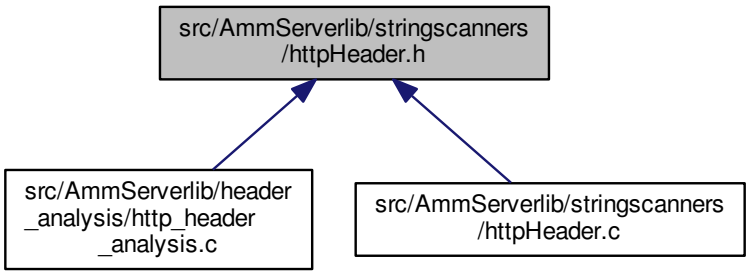
Return values

See	above enumerator

6.52 src/AmmServerlib/stringscanners/httpHeader.h File Reference

A tool that scans for a string in a very fast and robust way.

This graph shows which files directly or indirectly include this file:



Enumerations

- enum {
  HTTPHEADER_EMPTY =0, HTTPHEADER_AUTHORIZATION, HTTPHEADER_ACCEPT_ENCODING, HTTPHEADER_COOKIE,
  HTTPHEADER_CONNECTION, HTTPHEADER_HOST, HTTPHEADER_IF_NONE_MATCH, HTTPHEADER_IF_MODIFIED_SINCE,
  HTTPHEADER_RANGE, HTTPHEADER_REFERRER, HTTPHEADER_REFERER, HTTPHEADER_USER_AGENT,
  HTTPHEADER_END_OF_ITEMS }

  Enumerator for the IDs of httpHeader so we can know what the result was.

Functions

- int scanFor_httpHeader (char *str, unsigned int strLength)

  Scan a string for one of the words of the httpHeader word set.

6.52.1 Detailed Description

A tool that scans for a string in a very fast and robust way.

Author

Ammar Qammaz (AmmarkoV)

6.52.2 Enumeration Type Documentation

6.52.2.1 anonymous enum

Enumerator for the IDs of httpHeader so we can know what the result was.

Enumerator

HTTPHEADER_EMPTY

HTTPHEADER_AUTHORIZATION

HTTPHEADER_ACCEPT_ENCODING

HTTPHEADER_COOKIE
HTTPHEADER_CONNECTION
HTTPHEADER_HOST
HTTPHEADER_IF_NONE_MATCH
HTTPHEADER_IF_MODIFIED_SINCE
HTTPHEADER_RANGE
HTTPHEADER_REFERRER
HTTPHEADER_REFERER
HTTPHEADER_USER_AGENT
HTTPHEADER_END_OF_ITEMS

6.52.3 Function Documentation

6.52.3.1 int scanFor_httpHeader (char * *str,* unsigned int *strLength*)

Scan a string for one of the words of the httpHeader word set.

Parameters

Input	String , to be scanned
Length	of Input String

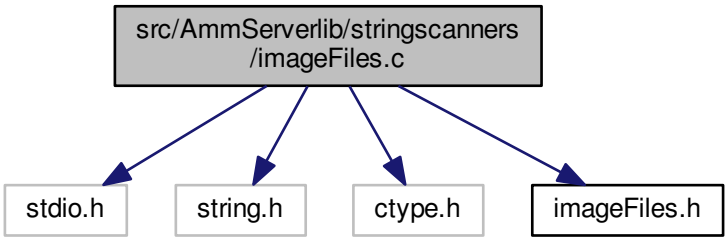
Return values

See	above enumerator

6.53 src/AmmServerlib/stringscanners/imageFiles.c File Reference

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include "imageFiles.h"
```

Include dependency graph for imageFiles.c:



Functions

- int scanFor_imageFiles (char *str, unsigned int strLength)

  Scan a string for one of the words of the imageFiles word set.

6.53.1 Function Documentation

6.53.1.1 int scanFor_imageFiles (char * *str*, unsigned int *strLength*)

Scan a string for one of the words of the imageFiles word set.

Parameters

Input	String , to be scanned
Length	of Input String

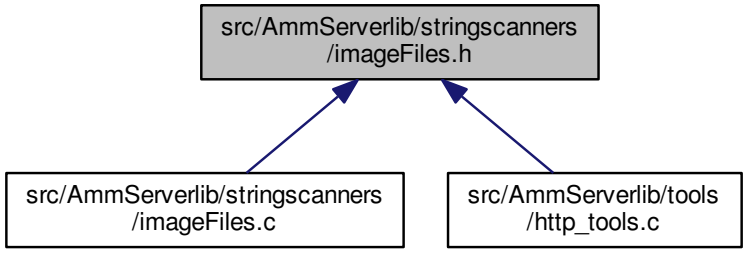
Return values

See	above enumerator

6.54 src/AmmServerlib/stringscanners/imageFiles.h File Reference

A tool that scans for a string in a very fast and robust way.

This graph shows which files directly or indirectly include this file:



Enumerations

- enum {
  IMAGEFILES_EMPTY =0, IMAGEFILES_GIF, IMAGEFILES_PNG, IMAGEFILES_JPG,
  IMAGEFILES_JPEG, IMAGEFILES_WEBP, IMAGEFILES_BMP, IMAGEFILES_TIFF,
  IMAGEFILES_DIB, IMAGEFILES_RLE, IMAGEFILES_J2C, IMAGEFILES_ICO,
  IMAGEFILES_PPM, IMAGEFILES_PNM, IMAGEFILES_RAW, IMAGEFILES_SVG,
  IMAGEFILES_END_OF_ITEMS }

  Enumerator for the IDs of imageFiles so we can know what the result was.

Functions

- int scanFor_imageFiles (char *str, unsigned int strLength)

  Scan a string for one of the words of the imageFiles word set.

6.54.1 Detailed Description

A tool that scans for a string in a very fast and robust way.

Author

Ammar Qammaz (AmmarkoV)

6.54.2 Enumeration Type Documentation

6.54.2.1 anonymous enum

Enumerator for the IDs of imageFiles so we can know what the result was.

Enumerator

IMAGEFILES_EMPTY

IMAGEFILES_GIF

IMAGEFILES_PNG

IMAGEFILES_JPG

IMAGEFILES_JPEG

IMAGEFILES_WEBP

IMAGEFILES_BMP

IMAGEFILES_TIFF

IMAGEFILES_DIB

IMAGEFILES_RLE

IMAGEFILES_J2C

IMAGEFILES_ICO

IMAGEFILES_PPM

IMAGEFILES_PNM

IMAGEFILES_RAW

IMAGEFILES_SVG

IMAGEFILES_END_OF_ITEMS

6.54.3 Function Documentation

6.54.3.1 int scanFor_imageFiles (char * *str,* unsigned int *strLength*)

Scan a string for one of the words of the imageFiles word set.

Parameters

Input	String , to be scanned
Length	of Input String

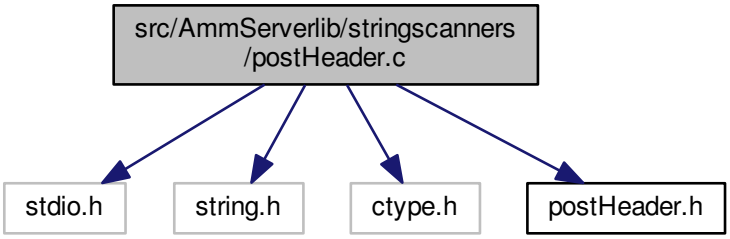
Return values

See	above enumerator

6.55 src/AmmServerlib/stringscanners/postHeader.c File Reference

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include "postHeader.h"
```

Include dependency graph for postHeader.c:



Functions

- int scanFor_postHeader (char *str, unsigned int strLength)

  Scan a string for one of the words of the postHeader word set.

6.55.1 Function Documentation

6.55.1.1 int scanFor_postHeader (char * *str,* unsigned int *strLength*)

Scan a string for one of the words of the postHeader word set.

Parameters

Input	String , to be scanned
Length	of Input String

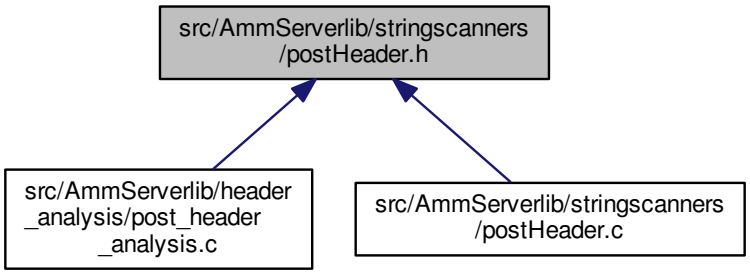
Return values

See	above enumerator

6.56 src/AmmServerlib/stringscanners/postHeader.h File Reference

A tool that scans for a string in a very fast and robust way.

This graph shows which files directly or indirectly include this file:



Enumerations

- enum {
  POSTHEADER_EMPTY =0, POSTHEADER_CONTENT_TYPE, POSTHEADER_CONTENT_DISPOSITIO-
  N, POSTHEADER_CONTENT_LENGTH,
  POSTHEADER_END_OF_ITEMS }

  Enumerator for the IDs of postHeader so we can know what the result was.

Functions

- int scanFor_postHeader (char ∗str, unsigned int strLength)

  Scan a string for one of the words of the postHeader word set.

6.56.1 Detailed Description

A tool that scans for a string in a very fast and robust way.

Author

Ammar Qammaz (AmmarkoV)

6.56.2 Enumeration Type Documentation

6.56.2.1 anonymous enum

Enumerator for the IDs of postHeader so we can know what the result was.

Enumerator

POSTHEADER_EMPTY

POSTHEADER_CONTENT_TYPE

POSTHEADER_CONTENT_DISPOSITION

POSTHEADER_CONTENT_LENGTH

POSTHEADER_END_OF_ITEMS

6.56.3 Function Documentation

6.56.3.1 int scanFor_postHeader (char * *str,* unsigned int *strLength*)

Scan a string for one of the words of the postHeader word set.

Parameters

Input	String , to be scanned
Length	of Input String

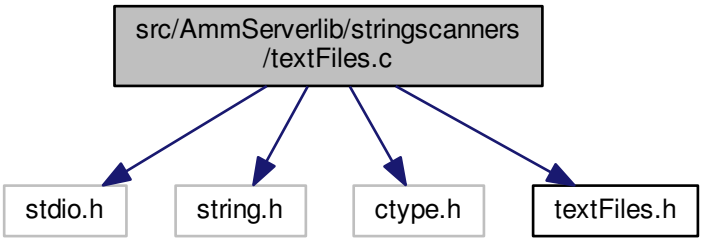
Return values

See	above enumerator

6.57 src/AmmServerlib/stringscanners/textFiles.c File Reference

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include "textFiles.h"
```

Include dependency graph for textFiles.c:



Functions

- int scanFor_textFiles (char *str, unsigned int strLength)

  Scan a string for one of the words of the textFiles word set.

6.57.1 Function Documentation

6.57.1.1 int scanFor_textFiles (char * *str,* unsigned int *strLength*)

Scan a string for one of the words of the textFiles word set.

Parameters

Input	String , to be scanned
Length	of Input String

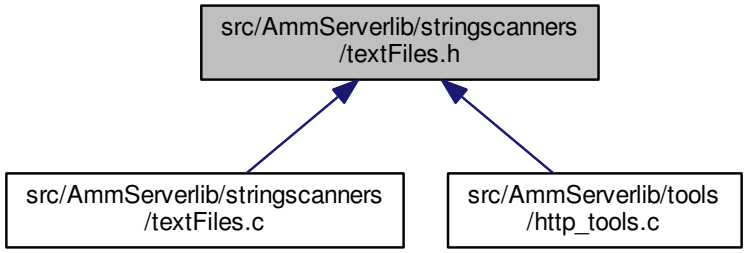
Return values

See	above enumerator

6.58 src/AmmServerlib/stringscanners/textFiles.h File Reference

A tool that scans for a string in a very fast and robust way.

This graph shows which files directly or indirectly include this file:



Enumerations

- enum {
  TEXTFILES_EMPTY =0, TEXTFILES_HTML, TEXTFILES_HTM, TEXTFILES_CSS,
  TEXTFILES_TXT, TEXTFILES_DOC, TEXTFILES_RTF, TEXTFILES_ODF,
  TEXTFILES_ODT, TEXTFILES_END_OF_ITEMS }

  Enumerator for the IDs of textFiles so we can know what the result was.

Functions

- int scanFor_textFiles (char *str, unsigned int strLength)

  Scan a string for one of the words of the textFiles word set.

6.58.1 Detailed Description

A tool that scans for a string in a very fast and robust way.

Author

Ammar Qammaz (AmmarkoV)

6.58.2 Enumeration Type Documentation

6.58.2.1 anonymous enum

Enumerator for the IDs of textFiles so we can know what the result was.

Enumerator

TEXTFILES_EMPTY

TEXTFILES_HTML

TEXTFILES_HTM

TEXTFILES_CSS

TEXTFILES_TXT

TEXTFILES_DOC

TEXTFILES_RTF

TEXTFILES_ODF

TEXTFILES_ODT

TEXTFILES_END_OF_ITEMS

6.58.3 Function Documentation

6.58.3.1 int scanFor_textFiles (char * *str,* unsigned int *strLength*)

Scan a string for one of the words of the textFiles word set.

Parameters

Input	String , to be scanned
Length	of Input String

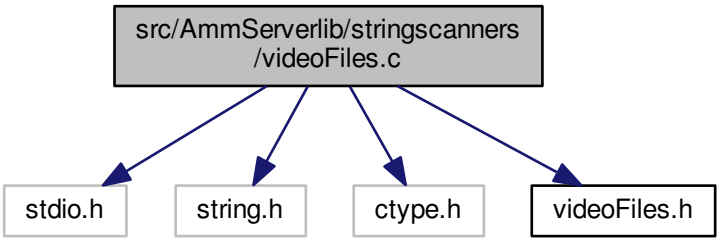
Return values

See	above enumerator

6.59 src/AmmServerlib/stringscanners/videoFiles.c File Reference

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include "videoFiles.h"
```

Include dependency graph for videoFiles.c:



Functions

- int scanFor_videoFiles (char *str, unsigned int strLength)

  Scan a string for one of the words of the videoFiles word set.

6.59.1 Function Documentation

6.59.1.1 int scanFor_videoFiles (char * *str,* unsigned int *strLength*)

Scan a string for one of the words of the videoFiles word set.

Parameters

Input	String , to be scanned
Length	of Input String

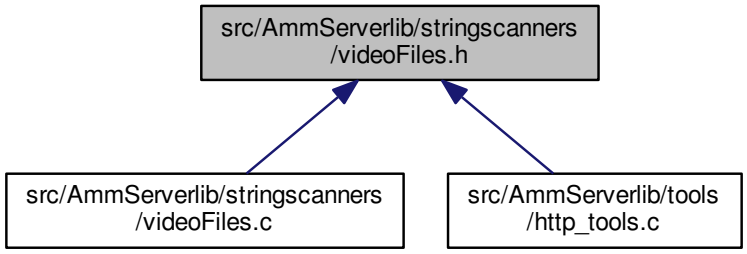
Return values

See	above enumerator

6.60 src/AmmServerlib/stringscanners/videoFiles.h File Reference

A tool that scans for a string in a very fast and robust way.

This graph shows which files directly or indirectly include this file:



Enumerations

- enum {
  VIDEOFILES_EMPTY =0, VIDEOFILES_AVI, VIDEOFILES_MPEG4, VIDEOFILES_MPEG,
  VIDEOFILES_MP4, VIDEOFILES_WEBM, VIDEOFILES_MKV, VIDEOFILES_3GP,
  VIDEOFILES_H263, VIDEOFILES_H264, VIDEOFILES_FLV, VIDEOFILES_END_OF_ITEMS }

  Enumerator for the IDs of videoFiles so we can know what the result was.

Functions

- int scanFor_videoFiles (char ∗str, unsigned int strLength)

  Scan a string for one of the words of the videoFiles word set.

6.60.1 Detailed Description

A tool that scans for a string in a very fast and robust way.

Author

Ammar Qammaz (AmmarkoV)

6.60.2 Enumeration Type Documentation

6.60.2.1 anonymous enum

Enumerator for the IDs of videoFiles so we can know what the result was.

Enumerator

VIDEOFILES_EMPTY

VIDEOFILES_AVI

VIDEOFILES_MPEG4

VIDEOFILES_MPEG

VIDEOFILES_MP4

VIDEOFILES_WEBM

VIDEOFILES_MKV

VIDEOFILES_3GP

VIDEOFILES_H263

VIDEOFILES_H264

VIDEOFILES_FLV

VIDEOFILES_END_OF_ITEMS

6.60.3 Function Documentation

6.60.3.1 int scanFor_videoFiles (char * *str,* unsigned int *strLength*)

Scan a string for one of the words of the videoFiles word set.

Parameters

Input	String , to be scanned
Length	of Input String

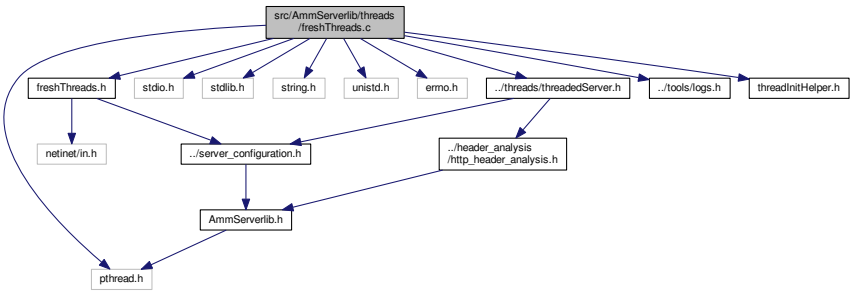
Return values

See	above enumerator

6.61 src/AmmServerlib/threads/freshThreads.c File Reference

```
#include "freshThreads.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <pthread.h>
#include <errno.h>
#include "../threads/threadedServer.h"
#include "../tools/logs.h"
#include "threadInitHelper.h"
```

Include dependency graph for freshThreads.c:



Macros

- #define WEIRD_THING_THAT_WORKS 1
- #define MAX_TRIES_TO_FIND_A_THREAD_ID 5

Functions

- unsigned int FindAProperThreadID (struct AmmServer_Instance ∗instance, int ∗success)
- int SpawnThreadToServeNewClient (struct AmmServer_Instance ∗instance, int clientsock, struct sockaddr_in client, unsigned int clientlen, char ∗webserver_root, char ∗templates_root)

  Create a new Thread that will serve the incoming client socket connection.

6.61.1 Macro Definition Documentation

6.61.1.1 #define MAX_TRIES_TO_FIND_A_THREAD_ID 5

6.61.1.2 #define WEIRD_THING_THAT_WORKS 1

6.61.2 Function Documentation

6.61.2.1 unsigned int FindAProperThreadID (struct AmmServer_Instance ∗ *instance,* int ∗ *success*)

Here is the call graph for this function:

6.61.2.2 int SpawnThreadToServeNewClient (struct AmmServer_Instance ∗ *instance,* int *clientsock,* struct sockaddr_in *client,* unsigned int *clientlen,* char ∗ *webserver_root,* char ∗ *templates_root*)

Create a new Thread that will serve the incoming client socket connection.

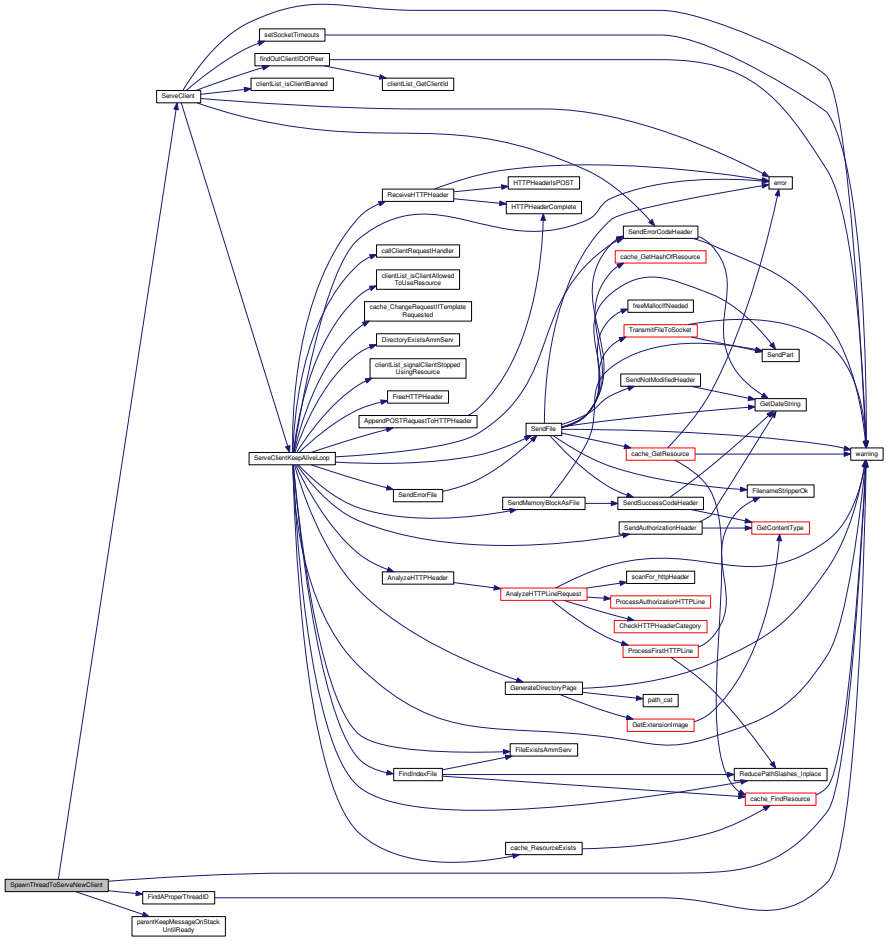
Parameters

An	AmmarServer Instance
Client	socket to be read
Client	socket to be read (sockaddr_in)
Length	of client
Filename	of root directory for this connection (public_html)
Filename	of template directory for this connection (for 404.html etc)

Return values

1=Success,0=Fail	

Here is the call graph for this function:

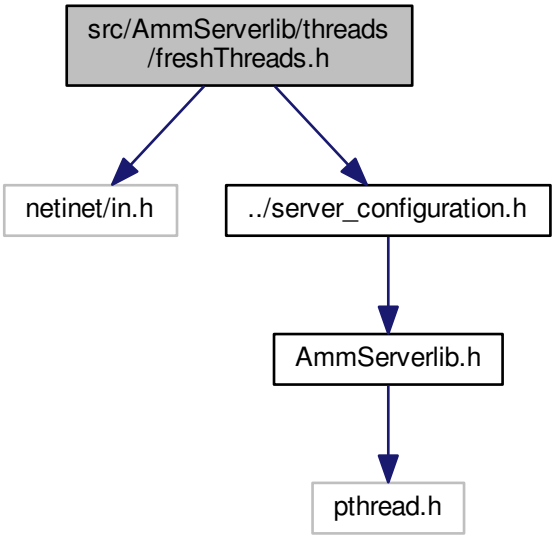


6.62 src/AmmServerlib/threads/freshThreads.h File Reference

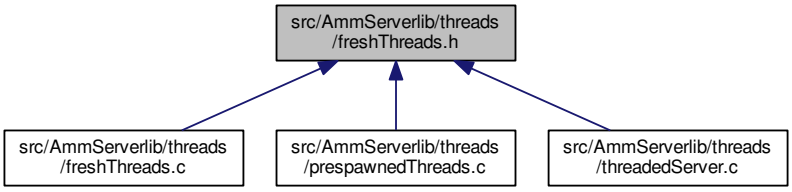
Creating new threads to serve clients , we only have one call that generates a thread that serves a client connection.

```
#include <netinet/in.h>
#include "../server_configuration.h"
```

Include dependency graph for freshThreads.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct PassToHTTPThread

  A structure that holds information to be passed from the main thread to the new (fresh) thread.

Functions

- int SpawnThreadToServeNewClient (struct AmmServer_Instance *instance, int clientsock, struct sockaddr_in client, unsigned int clientlen, char *webserver_root, char *templates_root)

  Create a new Thread that will serve the incoming client socket connection.

6.62.1 Detailed Description

Creating new threads to serve clients , we only have one call that generates a thread that serves a client connection.

Author

Ammar Qammaz (AmmarkoV)

6.62.2 Function Documentation

6.62.2.1 int SpawnThreadToServeNewClient (struct AmmServer_Instance * *instance*, int *clientsock*, struct sockaddr_in *client*, unsigned int *clientlen*, char * *webserver_root*, char * *templates_root*)

Create a new Thread that will serve the incoming client socket connection.

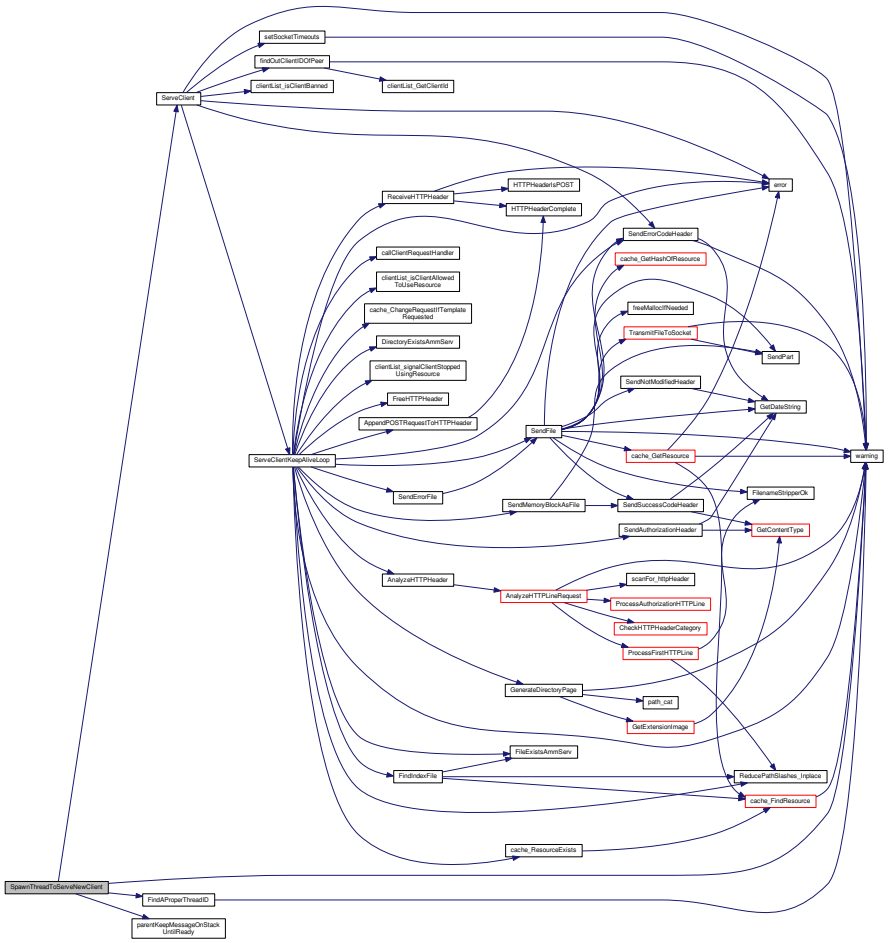
Parameters

An	AmmarServer Instance
Client	socket to be read
Client	socket to be read (sockaddr_in)
Length	of client
Filename	of root directory for this connection (public_html)
Filename	of template directory for this connection (for 404.html etc)

Return values

1=Success,0=Fail	

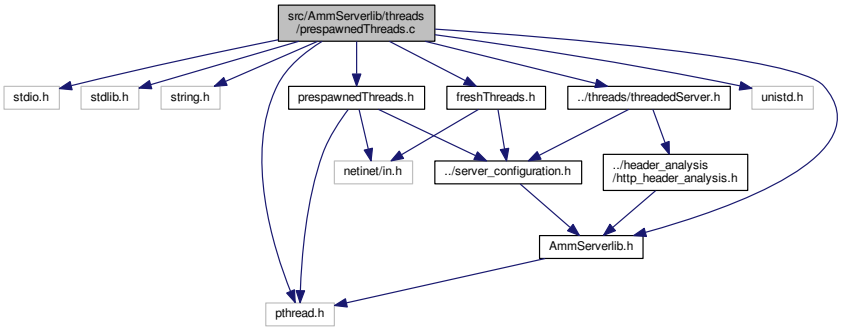
Here is the call graph for this function:



6.63 src/AmmServerlib/threads/prespawnedThreads.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "prespawnedThreads.h"
#include "freshThreads.h"
#include <pthread.h>
#include <unistd.h>
#include "../threads/threadedServer.h"
#include "../AmmServerlib.h"
```

Include dependency graph for prespawnedThreads.c:



Data Structures

- struct PassToPreSpawnedThread

Functions

- void * PreSpawnedThread (void *ptr)
- void PreSpawnThreads (struct AmmServer_Instance *instance)

  Create an initial pool of PreSpawned Threads , before handling any connections to be ready when a connection arrives.

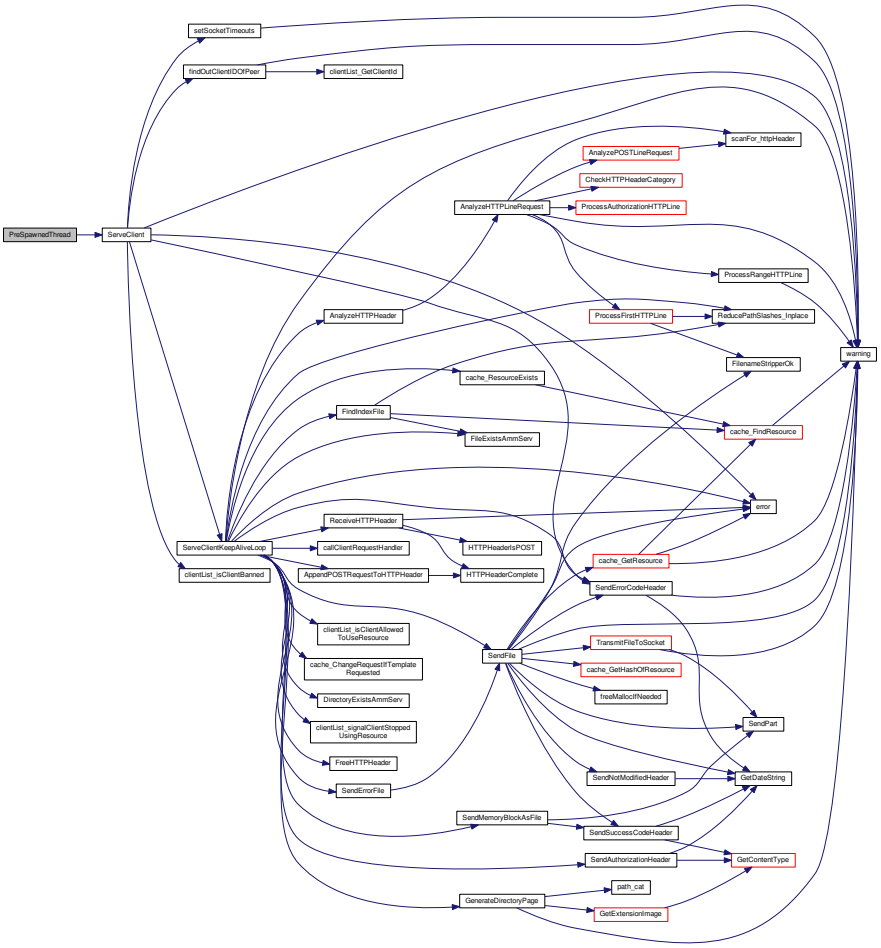
- int UsePreSpawnedThreadToServeNewClient (struct AmmServer_Instance *instance, int clientsock, struct sockaddr_in client, unsigned int clientlen, char *webserver_root, char *templates_root)

  Use a PreSpawned Thread that will serve the incoming client socket connection.

6.63.1 Function Documentation

6.63.1.1 void* PreSpawnedThread (void * *ptr*)

Here is the call graph for this function:



6.63.1.2 void PreSpawnThreads (struct AmmServer_Instance * *instance*)

Create an initial pool of PreSpawned Threads , before handling any connections to be ready when a connection arrives.

Parameters

An	AmmarServer Instance

Return values

1=Success,0=Fail	

6.63.1.3 int UsePreSpawnedThreadToServeNewClient (struct AmmServer_Instance * *instance,* int *clientsock,* struct sockaddr_in *client,* unsigned int *clientlen,* char * *webserver_root,* char * *templates_root*)

Use a PreSpawned Thread that will serve the incoming client socket connection.

Parameters

An	AmmarServer Instance
Client	socket to be read
Client	socket to be read (sockaddr_in)
Length	of client
Filename	of root directory for this connection (public_html)
Filename	of template directory for this connection (for 404.html etc)

Return values

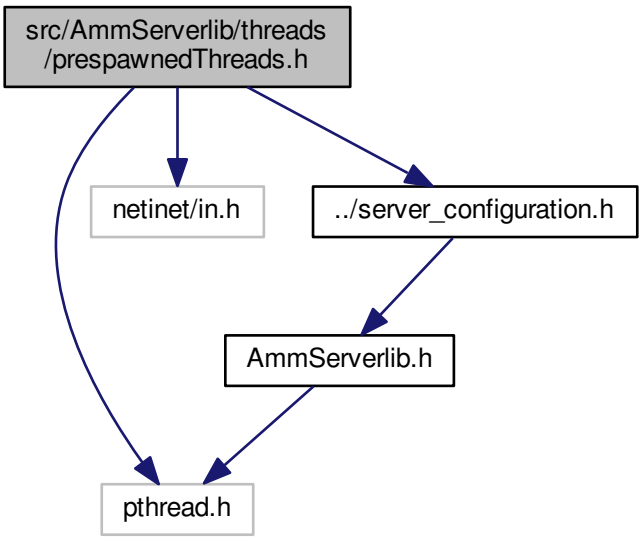
1=Success,0=Fail	

6.64 src/AmmServerlib/threads/prespawnedThreads.h File Reference

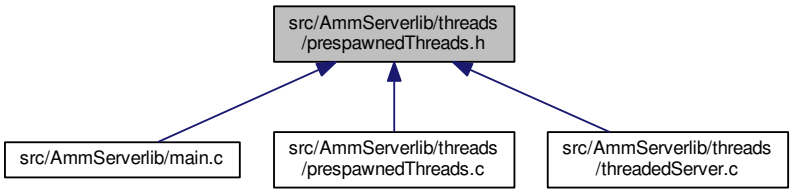
Using already created threads to serve clients , we have a pool of threads that can be used to serve connections.

```
#include <pthread.h>
#include <netinet/in.h>
#include "../server_configuration.h"
```

Include dependency graph for prespawnedThreads.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct PreSpawnedThread

  A structure that holds information to be passed from the main thread to the new (prespawned) thread.

Functions

- void PreSpawnThreads (struct AmmServer_Instance *instance)

  Create an initial pool of PreSpawned Threads , before handling any connections to be ready when a connection arrives.
- int UsePreSpawnedThreadToServeNewClient (struct AmmServer_Instance *instance, int clientsock, struct sockaddr_in client, unsigned int clientlen, char *webserver_root, char *templates_root)

  Use a PreSpawned Thread that will serve the incoming client socket connection.

6.64.1 Detailed Description

Using already created threads to serve clients , we have a pool of threads that can be used to serve connections.

Author

Ammar Qammaz (AmmarkoV)

Bug Prespawned threads have race conditions ?

6.64.2 Function Documentation

6.64.2.1 void PreSpawnThreads (struct AmmServer_Instance * *instance*)

Create an initial pool of PreSpawned Threads , before handling any connections to be ready when a connection arrives.

Parameters

An	AmmarServer Instance

Return values

1=Success,0=Fail	

6.64.2.2 int UsePreSpawnedThreadToServeNewClient (struct AmmServer_Instance * *instance,* int *clientsock,* struct sockaddr_in *client,* unsigned int *clientlen,* char * *webserver_root,* char * *templates_root*)

Use a PreSpawned Thread that will serve the incoming client socket connection.

Parameters

An	AmmarServer Instance
Client	socket to be read
Client	socket to be read (sockaddr_in)
Length	of client
Filename	of root directory for this connection (public_html)
Filename	of template directory for this connection (for 404.html etc)

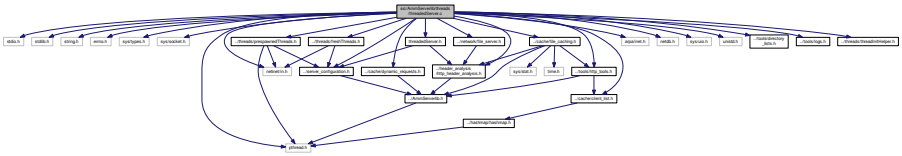
Return values

1=Success,0=Fail	

6.65 src/AmmServerlib/threads/threadedServer.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <sys/uio.h>
#include <unistd.h>
#include <pthread.h>
#include "threadedServer.h"
#include "../tools/directory_lists.h"
#include "../network/file_server.h"
#include "../header_analysis/http_header_analysis.h"
#include "../tools/http_tools.h"
#include "../tools/logs.h"
#include "../cache/file_caching.h"
#include "../server_configuration.h"
#include "../threads/freshThreads.h"
#include "../threads/prespawnedThreads.h"
#include "../threads/threadInitHelper.h"
#include "../cache/client_list.h"
#include "../cache/dynamic_requests.h"
```

Include dependency graph for threadedServer.c:



Functions

- int HTTPServerIsRunning (struct AmmServer_Instance *instance)

  Ask if the HTTP server is running.

- int ServeClientKeepAliveLoop (struct AmmServer_Instance *instance, struct HTTPTransaction *transaction)
- void * ServeClient (void *ptr)

  Main Call to Serve a client , this will in turn pick a prespawned thread or create a new one.

- void * MainHTTPServerThread (void *ptr)
- int StartHTTPServer (struct AmmServer_Instance *instance, char *ip, unsigned int port, char *root_path, char *templates_path)

  Start HTTP server.

- int StopHTTPServer (struct AmmServer_Instance *instance)

  Stop a running HTTP server , unbind ports , deallocate structures etc.

6.65.1 Function Documentation

6.65.1.1 int HTTPServerIsRunning (struct AmmServer_Instance * *instance*)

Ask if the HTTP server is running.

Parameters

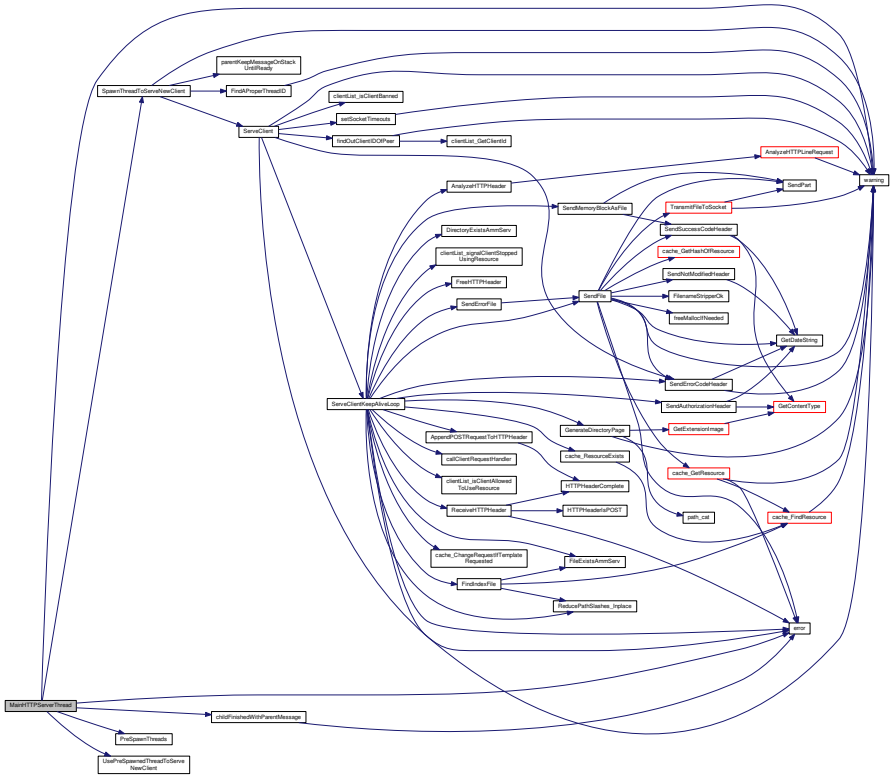
An	AmmarServer Instance

Return values

1=Success,0=Failure	

6.65.1.2 void* MainHTTPServerThread (void * *ptr*)

Here is the call graph for this function:



6.65.1.3 void* ServeClient (void * *ptr*)

Main Call to Serve a client , this will in turn pick a prespawned thread or create a new one.

Parameters

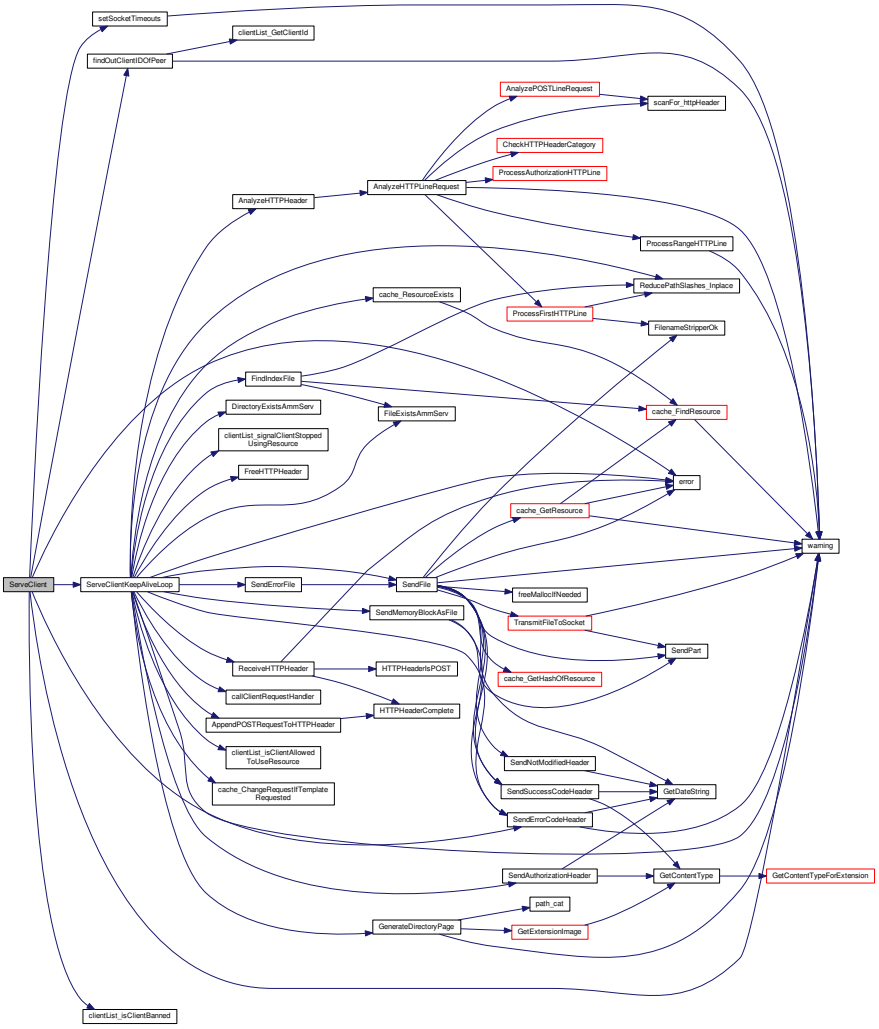
PassToHTTP-Thread	with information to pass to the new thread (prespawned or not)

Return values

This	function returns 0

START OF CLIENT IS NOT ON IP-BANNED-LIST!

Here is the call graph for this function:

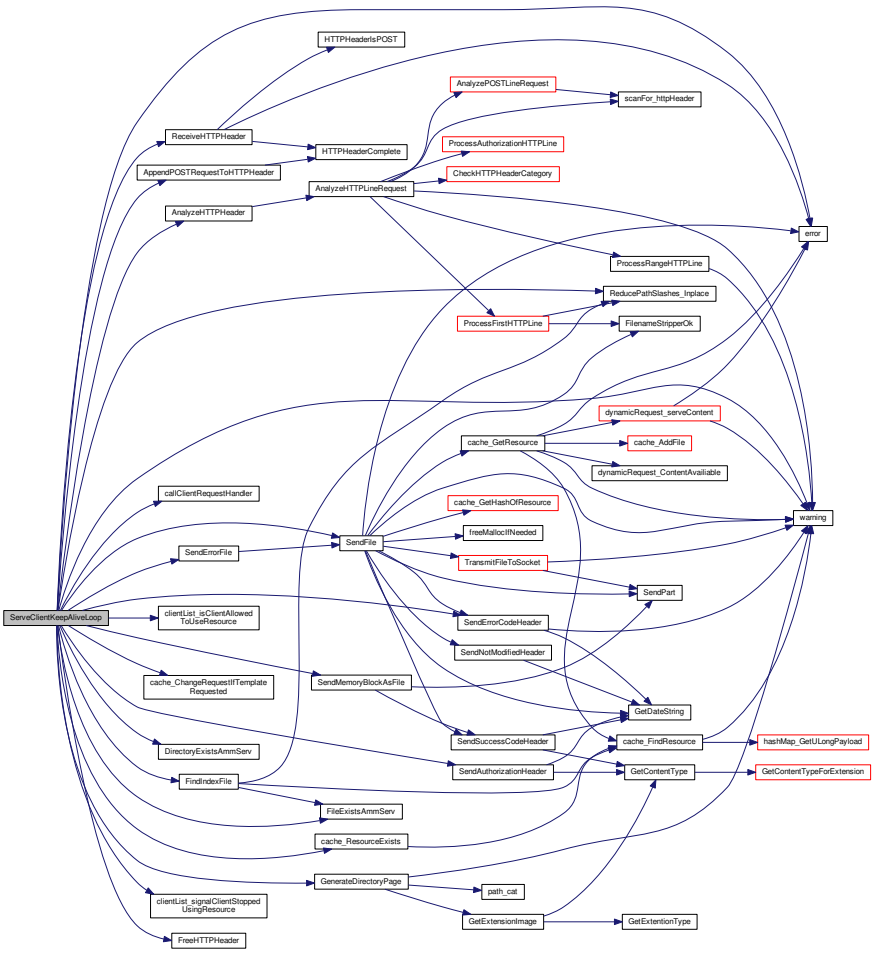


6.65.1.4 int ServeClientKeepAliveLoop (struct AmmServer_Instance * *instance,* struct HTTPTransaction * *transaction*) [inline]

PART 1 : Sense what we want to serve , and set the flags resource_is_a_directory , resource_is_a_file , generate_directory_list accordingly..!

PART 2 : The flags resource_is_a_directory , resource_is_a_file , generate_directory_list have been set to the correct (:P) value so all we have to do now is serve the correct repsonse..!

Here is the call graph for this function:



6.65.1.5 int StartHTTPServer (struct AmmServer_Instance * *instance*, char * *ip*, unsigned int *port*, char * *root_path*, char * *templates_path*)

Start HTTP server.

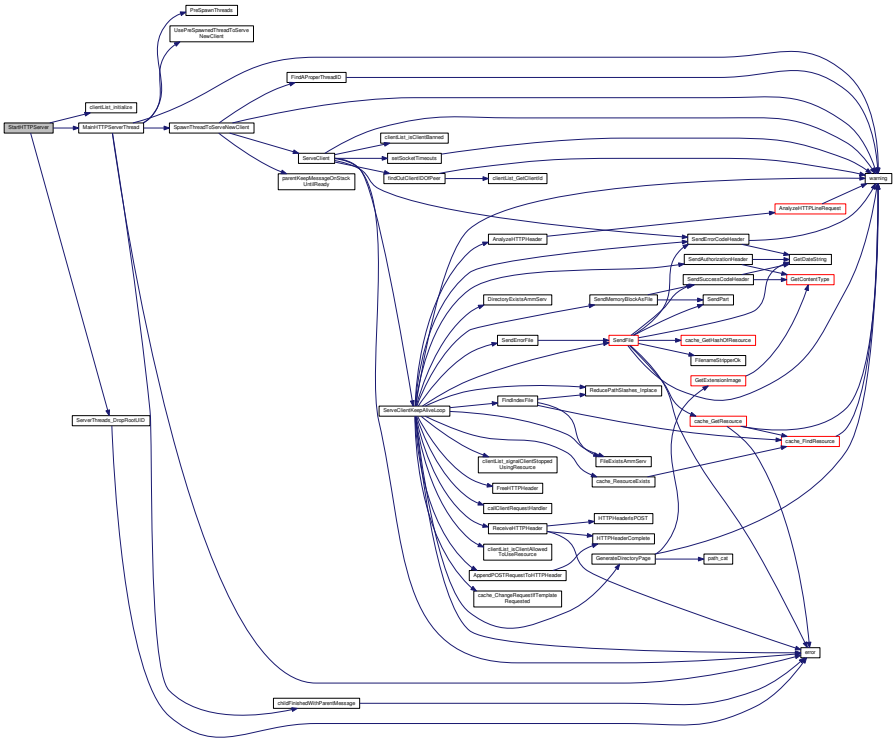
Parameters

An	AmmarServer Instance
String	with the binding IP for the new server
Port	for binding the new server , ports under 1000 require super user privileges
Filename	to root path for this webserver (public_html)
Filename	to root path for templates (404.html etc)

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.65.1.6 int StopHTTPServer (struct AmmServer_Instance * *instance*)

Stop a running HTTP server , unbind ports , deallocate structures etc.

Parameters

An	AmmarServer Instance

Bug Stop web server should be improved , to make sure it unbinds the closing socket

Return values

1=Success,0=Failure	

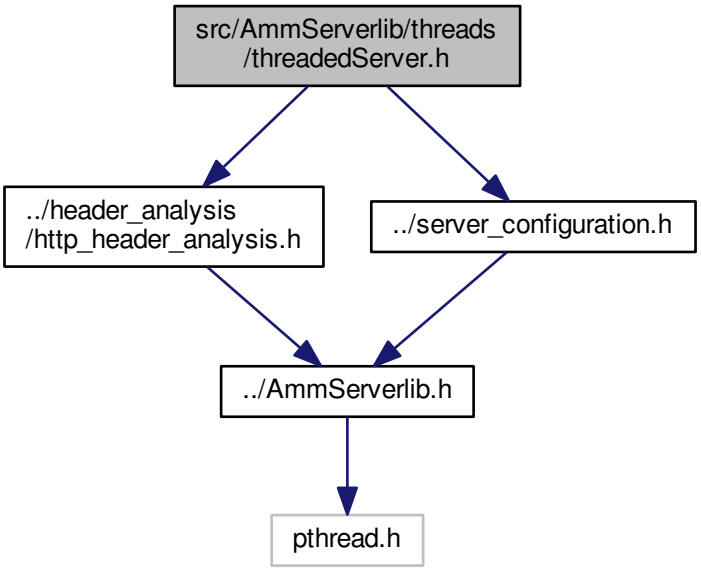
Here is the call graph for this function:

6.66 src/AmmServerlib/threads/threadedServer.h File Reference

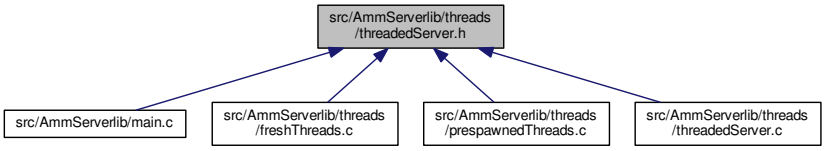
Creating new threads to serve clients , we only have one call that generates a thread that serves a client connection.

```
#include "../header_analysis/http_header_analysis.h"
#include "../server_configuration.h"
```

Include dependency graph for threadedServer.h:



This graph shows which files directly or indirectly include this file:



Functions

- void * ServeClient (void *ptr)

  Main Call to Serve a client , this will in turn pick a prespawned thread or create a new one.
- int StartHTTPServer (struct AmmServer_Instance *instance, char *ip, unsigned int port, char *root_path, char *templates_path)

  Start HTTP server.
- int StopHTTPServer (struct AmmServer_Instance *instance)

  Stop a running HTTP server , unbind ports , deallocate structures etc.
- int HTTPServerIsRunning (struct AmmServer_Instance *instance)

Ask if the HTTP server is running.

6.66.1 Detailed Description

Creating new threads to serve clients , we only have one call that generates a thread that serves a client connection.

Author

Ammar Qammaz (AmmarkoV)

6.66.2 Function Documentation

6.66.2.1 int HTTPServerIsRunning (struct AmmServer_Instance * *instance*)

Ask if the HTTP server is running.

Parameters

An	AmmarServer Instance

Return values

1=Success,0=Failure	

6.66.2.2 void* ServeClient (void * *ptr*)

Main Call to Serve a client , this will in turn pick a prespawned thread or create a new one.

Parameters

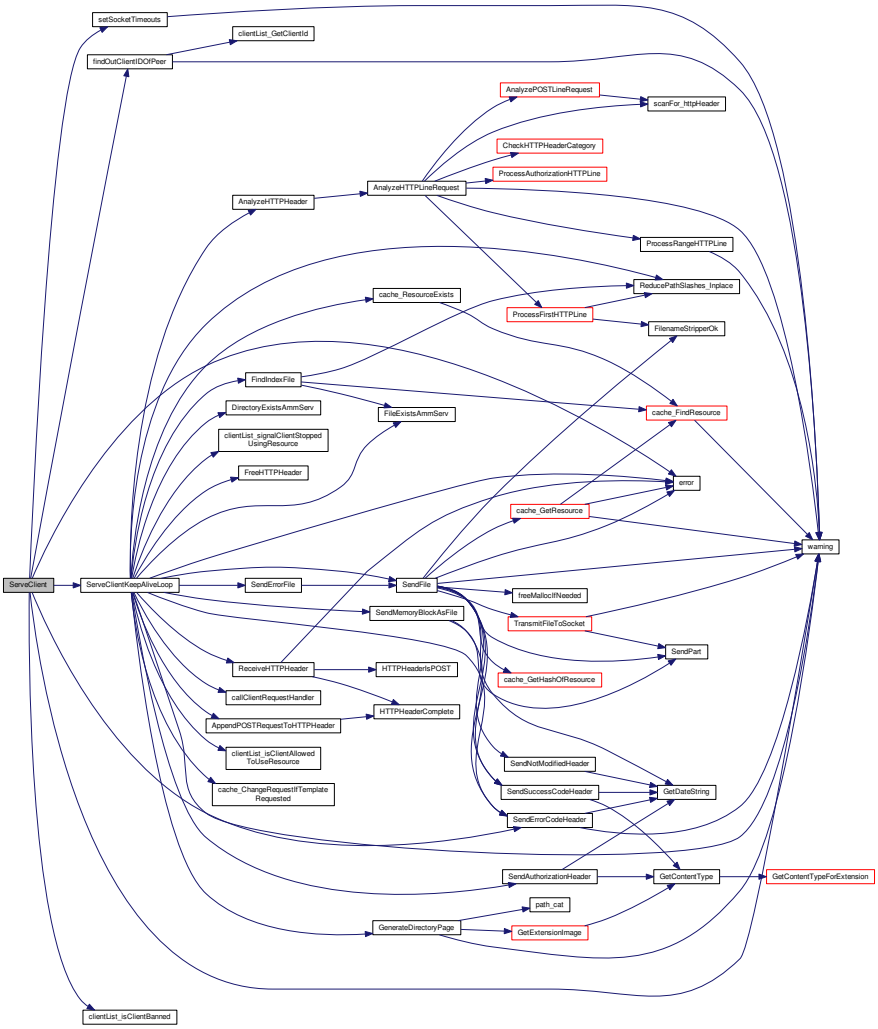
PassToHTTP-Thread	with information to pass to the new thread (prespawned or not)

Return values

This	function returns 0

START OF CLIENT IS NOT ON IP-BANNED-LIST!

Here is the call graph for this function:



6.66.2.3 int StartHTTPServer (struct AmmServer_Instance * *instance*, char * *ip*, unsigned int *port*, char * *root_path*, char * *templates_path*)

Start HTTP server.

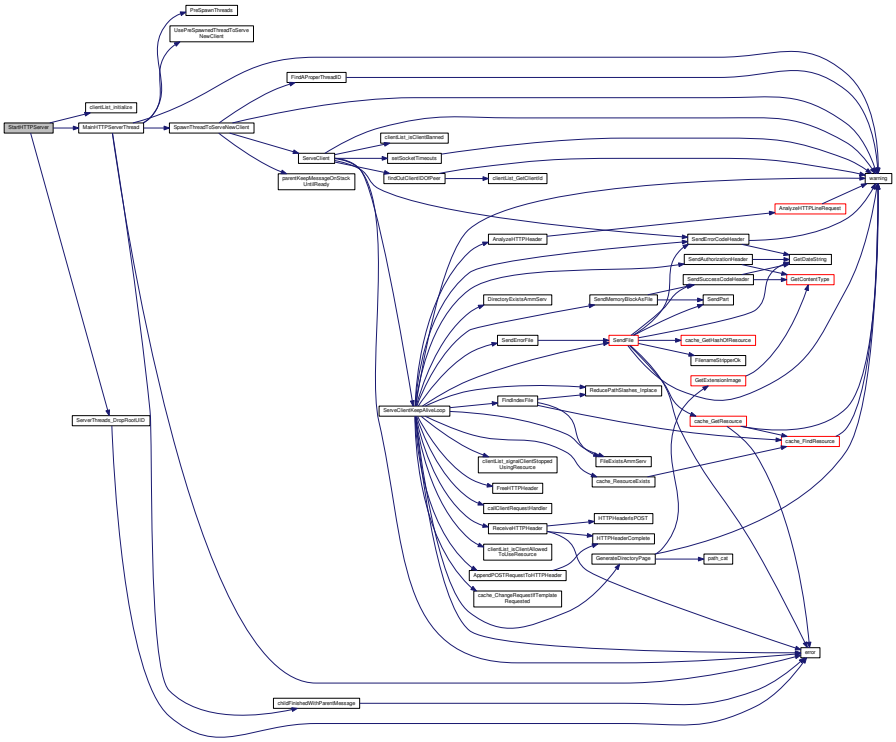
Parameters

An	AmmarServer Instance
String	with the binding IP for the new server
Port	for binding the new server , ports under 1000 require super user privileges
Filename	to root path for this webserver (public_html)
Filename	to root path for templates (404.html etc)

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.66.2.4 int StopHTTPServer (struct AmmServer_Instance * *instance*)

Stop a running HTTP server , unbind ports , deallocate structures etc.

Parameters

An	AmmarServer Instance

Bug Stop web server should be improved , to make sure it unbinds the closing socket

Return values

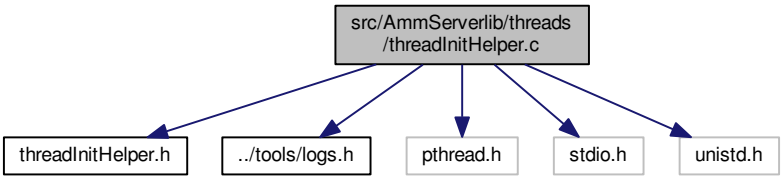
1=Success,0=Failure	

Here is the call graph for this function:

6.67 src/AmmServerlib/threads/threadInitHelper.c File Reference

```
#include "threadInitHelper.h"
#include "../tools/logs.h"
#include <pthread.h>
#include <stdio.h>
#include <unistd.h>
```

Include dependency graph for threadInitHelper.c:



Macros

- #define SLEEP_FOR_N_NANOSECONDS_WAITING_STACK_MESSAGE 10

Functions

- int parentKeepMessageOnStackUntilReadyOrTimeout (volatile int *childSwitch, unsigned int maxWaitTime)

  A call to help waiting for a message to be consumed by a child thread , or until a timeout , this should be called by the parent thread.
- int parentKeepMessageOnStackUntilReady (volatile int *childSwitch)

  A call to help waiting for a message to be consumed by a child thread , this should be called by the parent thread.
- void childFinishedWithParentMessage (volatile int *childSwitch)

  A call to help waiting for a message to be consumed by a child thread , this should be called by the child thread when it finished copying the message.

6.67.1 Macro Definition Documentation

6.67.1.1 #define SLEEP_FOR_N_NANOSECONDS_WAITING_STACK_MESSAGE 10

6.67.2 Function Documentation

6.67.2.1 void childFinishedWithParentMessage (volatile int * *childSwitch*)

A call to help waiting for a message to be consumed by a child thread , this should be called by the child thread when it finished copying the message.

Parameters

Pointer	to the switch signaling that the child has read the message , so that the parent will keep it in his stack

Return values

1=Success,0=Fail	

Here is the call graph for this function:

6.67.2.2 int parentKeepMessageOnStackUntilReady (volatile int * *childSwitch*)

A call to help waiting for a message to be consumed by a child thread , this should be called by the parent thread.

Parameters

Pointer	to the switch signaling that the child has read the message , so that the parent will keep it in his stack

Return values

1=Success,0=Fail	

6.67.2.3 int parentKeepMessageOnStackUntilReadyOrTimeout (volatile int * *childSwitch,* unsigned int *maxWaitTime*)

A call to help waiting for a message to be consumed by a child thread , or until a timeout , this should be called by the parent thread.

Parameters

Pointer	to the switch signaling that the child has read the message , so that the parent will keep it in his stack
Maximum	time to wait before terminating the wait to ensure no dead-locks

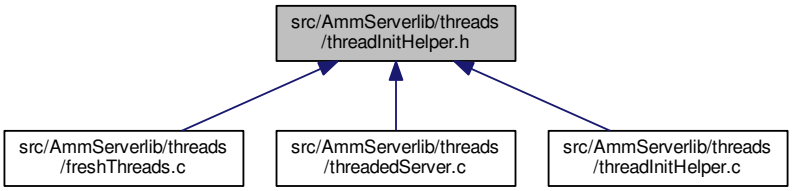
Return values

1=Success,0=Fail	

6.68 src/AmmServerlib/threads/threadInitHelper.h File Reference

Helper Functions to help with passing messages around ..

This graph shows which files directly or indirectly include this file:



Functions

- int parentKeepMessageOnStackUntilReadyOrTimeout (volatile int ∗childSwitch, unsigned int maxWaitTime)

  A call to help waiting for a message to be consumed by a child thread , or until a timeout , this should be called by the parent thread.

- int parentKeepMessageOnStackUntilReady (volatile int ∗childSwitch)

  A call to help waiting for a message to be consumed by a child thread , this should be called by the parent thread.

- void childFinishedWithParentMessage (volatile int ∗childSwitch)

  A call to help waiting for a message to be consumed by a child thread , this should be called by the child thread when it finished copying the message.

6.68.1 Detailed Description

Helper Functions to help with passing messages around ..

Author

Ammar Qammaz (AmmarkoV)

6.68.2 Function Documentation

6.68.2.1 void childFinishedWithParentMessage (volatile int ∗ *childSwitch*)

A call to help waiting for a message to be consumed by a child thread , this should be called by the child thread when it finished copying the message.

Parameters

Pointer	to the switch signaling that the child has read the message , so that the parent will keep it in his stack

Return values

1=Success,0=Fail	

Here is the call graph for this function:

6.68.2.2 int parentKeepMessageOnStackUntilReady (volatile int * *childSwitch*)

A call to help waiting for a message to be consumed by a child thread , this should be called by the parent thread.

Parameters

Pointer	to the switch signaling that the child has read the message , so that the parent will keep it in his stack

Return values

1=Success,0=Fail	

6.68.2.3 int parentKeepMessageOnStackUntilReadyOrTimeout (volatile int * *childSwitch,* unsigned int *maxWaitTime*)

A call to help waiting for a message to be consumed by a child thread , or until a timeout , this should be called by the parent thread.

Parameters

Pointer	to the switch signaling that the child has read the message , so that the parent will keep it in his stack
Maximum	time to wait before terminating the wait to ensure no dead-locks

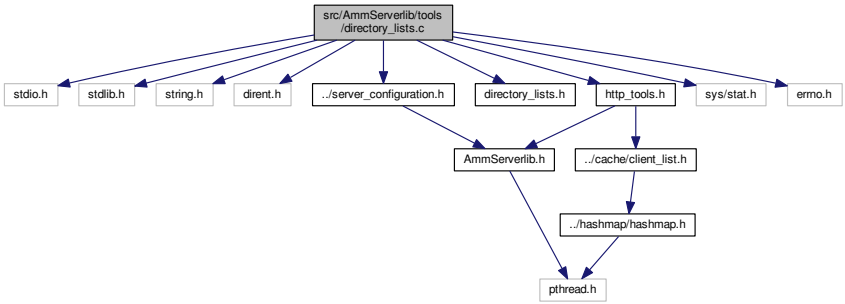
Return values

1=Success,0=Fail	

6.69 src/AmmServerlib/tools/directory_lists.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dirent.h>
#include "../server_configuration.h"
#include "directory_lists.h"
#include "http_tools.h"
#include <sys/stat.h>
#include <errno.h>
```

Include dependency graph for directory_lists.c:



Macros

- #define starting "<html>\ <head>\ <meta http-equiv=\"Content-Type\" content=\"text/html; charset=UTF-8\" />\ <title>AmmarServer Directory listing</title>\ </head>\ <body>\ <h1>AmmarServer Directory Listing</h1>\ <hr>\ <!-- TODO : Add some clientside javascript search and sorting capabilities! :P -->\ <table>\ <tr>\ <td></td>\ <td>Filename</td><td>Byte Size</td><td>Modification Date</td>\ </tr>\n"
- #define tag_pre_image "<tr><td><img src=\"/"
- #define tag_after_image "\">"

Functions

- char * path_cat (const char *str1, char *str2)
- char * GenerateDirectoryPage (char *system_path, char *client_path, unsigned long *memoryUsed)

  Return a memory buffer containing the contents o a directory listing.

6.69.1 Macro Definition Documentation

6.69.1.1 #define starting "<html>\ <head>\ <meta http-equiv=\"Content-Type\" content=\"text/html; charset=UTF-8\" />\ <title>AmmarServer Directory listing</title>\ </head>\ <body>\ <h1>AmmarServer Directory Listing</h1>\ <hr>\ <!-- TODO : Add some clientside javascript search and sorting capabilities! :P -->\ <table>\ <tr>\ <td></td>\ <td>Filename</td><td>Byte Size</td><td>Modification Date</td>\ </tr>\n"

6.69.1.2 #define tag_after_image "\">"

6.69.1.3 #define tag_pre_image "<tr><td><img src=\"/"

6.69.2 Function Documentation

6.69.2.1 char* GenerateDirectoryPage (char * *system_path*, char * *client_path*, unsigned long * *memoryUsed*)

Return a memory buffer containing the contents o a directory listing.

Parameters

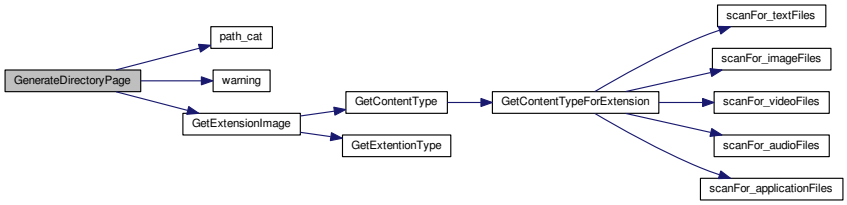
System	path to list
Client	path (relative to root directory of client etc)
Input	size of memory tou allocate and Output size of memory used

Return values

Pointer	to memory that contains directory listing ,0=Failure

Bug GenerateDirectoryPage does not handle memory correctly , code is in very bad shape

Here is the call graph for this function:

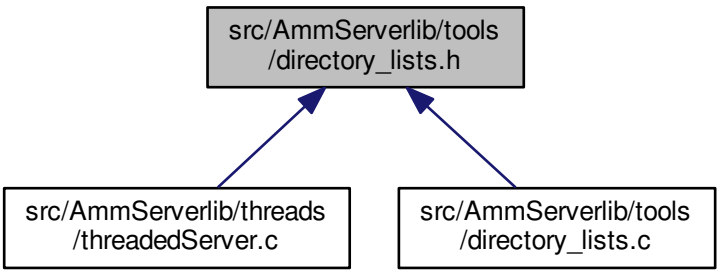


6.69.2.2 char* path_cat (const char * *str1*, char * *str2*)

6.70 src/AmmServerlib/tools/directory_lists.h File Reference

Basic file server functionality of AmmarServer.

This graph shows which files directly or indirectly include this file:



Functions

- char * GenerateDirectoryPage (char *system_path, char *client_path, unsigned long *memoryUsed)

  Return a memory buffer containing the contents o a directory listing.

6.70.1 Detailed Description

Basic file server functionality of AmmarServer.

Author

Ammar Qammaz (AmmarkoV)

6.70.2 Function Documentation

6.70.2.1 char* GenerateDirectoryPage (char * *system_path,* char * *client_path,* unsigned long * *memoryUsed*)

Return a memory buffer containing the contents o a directory listing.

Parameters

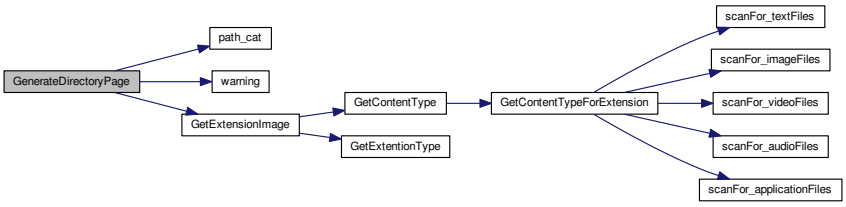
System	path to list
Client	path (relative to root directory of client etc)
Input	size of memory tou allocate and Output size of memory used

Return values

Pointer	to memory that contains directory listing ,0=Failure

Bug GenerateDirectoryPage does not handle memory correctly , code is in very bad shape

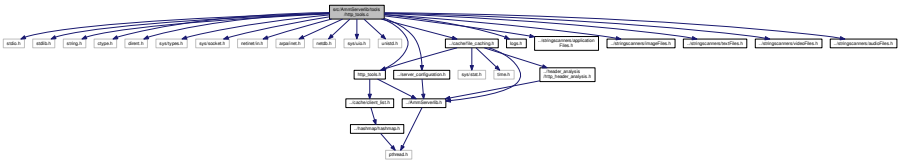
Here is the call graph for this function:



6.71 src/AmmServerlib/tools/http_tools.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <dirent.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <sys/uio.h>
#include <unistd.h>
#include "http_tools.h"
#include "logs.h"
#include "../server_configuration.h"
#include "../cache/file_caching.h"
#include "../stringscanners/applicationFiles.h"
#include "../stringscanners/imageFiles.h"
#include "../stringscanners/textFiles.h"
#include "../stringscanners/videoFiles.h"
#include "../stringscanners/audioFiles.h"
```

Include dependency graph for http_tools.c:



Functions

- unsigned int ServerThreads_DropRootUID ()

  Drop Root UID , if we have one (and according to server_configuration.h)
- char FileExistsAmmServ (char ∗filename)

  Check if file Exists.
- char DirectoryExistsAmmServ (char ∗dirpath)

  Check if directory Exists.
- int GetContentTypeForExtension (char ∗theextension, char ∗content_type, unsigned int contentTypeLength)
- int GetExtentionType (char ∗theextension)

  Convert an Extension Type to a contentTypeEnumerator.
- void convertToUpperCase (char ∗sPtr)
- int GetContentType (char ∗filename, char ∗contentType, unsigned int contentTypeLength)

  Convert a filename to a contentType.
- int GetExtensionImage (char ∗filename, char ∗theimagepath, unsigned int theimagepath_length)

  Return template image for specific content type (for directory listings etc)
- int ReducePathSlashes_Inplace (char ∗filename)

  Filenames may contain ///// with an arbitrary number of slashes , we convert them to a single slash ,.
- int StripGETRequestQueryAndFragment (char ∗filename, char ∗query, unsigned int max_query_length)
- int StripVariableFromGETorPOSTString (char ∗input, char ∗var_id, char ∗var_val, unsigned int var_val_length)
- int StripHTMLCharacters_Inplace (char ∗filename, int enable_security)

HTML characters should be converted to plain c byte chars after we get them , this poses some security threats since this might allow "weird" bytes to get set that in conjunction with an overflow somewhere else might trick the server into executing ,.

- int FilenameStripperOk (char *filename)

  Strip filename and security check it.
- int strToUpcase (char *strTarget, char *strSource, unsigned int strLength)
- int stristr (char *str1CAPS, unsigned int str1_length, char *str2CAPS, unsigned int str2_length, unsigned int *pos_found)
- int stristr2Caps (char *str1, unsigned int str1_length, char *str2CAPS, unsigned int str2_length, unsigned int *pos_found)
- int trim_last_empty_chars (char *input, unsigned int input_length)
- int seek_non_blank_char (char *input, char *input_end)
- int seek_blank_char (char *input, char *input_end)
- unsigned int GetIntFromHTTPHeaderFieldPayload (char *request, unsigned int request_length)
- char * GetNewStringFromHTTPHeaderFieldPayload (char *request, unsigned int request_length)
- int encodeToBase64 (char *src, unsigned s_len, char *dst, unsigned d_len)

  Convert a string to base64 , required for the authorization tokens.
- int CheckHTTPHeaderCategoryAllCaps (char *lineCAPS, unsigned int line_length, char *potential_strCAPS, unsigned int *payload_start)
- int CheckHTTPHeaderCategory (char *line, unsigned int line_length, char *potential_strCAPS, unsigned int *payload_start)
- int FindIndexFile (struct AmmServer_Instance *instance, char *webserver_root, char *directory, char *indexfile)
- char * RequestHTTPWebPage (char *hostname, unsigned int port, char *filename, unsigned int max_content)

  A very basic http client for testing connections and maybe in the future make AmmarServers communicate with each other.
- int freeString (char **str)

  Free C string and set it to 0.
- int setSocketTimeouts (int clientSock)

  Enforce socket timeouts declared in server_configuration.h and configuration files to socket.
- clientID findOutClientIDOfPeer (struct AmmServer_Instance *instance, int clientSock)

  Tool that resolve a client socket to its IP , then uses it to try to clientList_GetClientId and returns the id number.

6.71.1 Function Documentation

6.71.1.1 int CheckHTTPHeaderCategory (char * *line*, unsigned int *line_length*, char * *potential_strCAPS*, unsigned int * *payload_start*)

Here is the call graph for this function:

6.71.1.2 int CheckHTTPHeaderCategoryAllCaps (char * *lineCAPS,* unsigned int *line_length,* char * *potential_strCAPS,* unsigned int * *payload_start*)

Here is the call graph for this function:

6.71.1.3 void convertToUpperCase (char * *sPtr*)

6.71.1.4 char DirectoryExistsAmmServ (char * *dirpath*)

Check if directory Exists.

Parameters

Path	to directory

Return values

1=Exists,0=Does	not Exist

6.71.1.5 int encodeToBase64 (char * *src,* unsigned *s_len,* char * *dst,* unsigned *d_len*)

Convert a string to base64 , required for the authorization tokens.

Parameters

Input	string
Input	string length
Output	string
Input	Maximum output length

Return values

1=Success,0=Failure	

6.71.1.6 char FileExistsAmmServ (char * *filename*)

Check if file Exists.

Parameters

Path	to file

Return values

1=Exists,0=Does	not Exist

6.71.1.7 int FilenameStripperOk (char * *filename*)

Strip filename and security check it.

Parameters

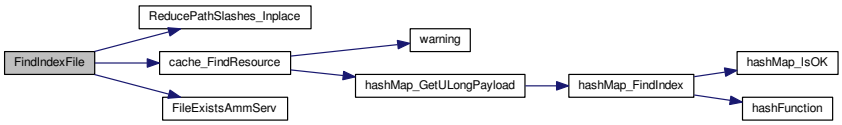
Pointer	to string pointer to be analyzed

Return values

1=Ok,0=Failed	

6.71.1.8 int FindIndexFile (struct AmmServer_Instance * *instance,* char * *webserver_root,* char * *directory,* char * *indexfile*)

Here is the call graph for this function:



6.71.1.9 clientID findOutClientIDOfPeer (struct AmmServer_Instance * *instance,* int *clientSock*)

Tool that resolve a client socket to its IP , then uses it to try to clientList_GetClientId and returns the id number.

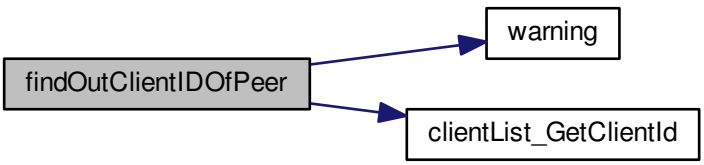
Parameters

An	AmmarServer instance
client	socket

Return values

ClientID	or ,0=Failure

Here is the call graph for this function:



6.71.1.10 int freeString (char ** *str*)

Free C string and set it to 0.

Parameters

Pointer	to string pointer to be freed

Return values

1=Success,0=Failure	

6.71.1.11 int GetContentType (char * *filename,* char * *contentType,* unsigned int *contentTypeLength*)

Convert a filename to a contentType.

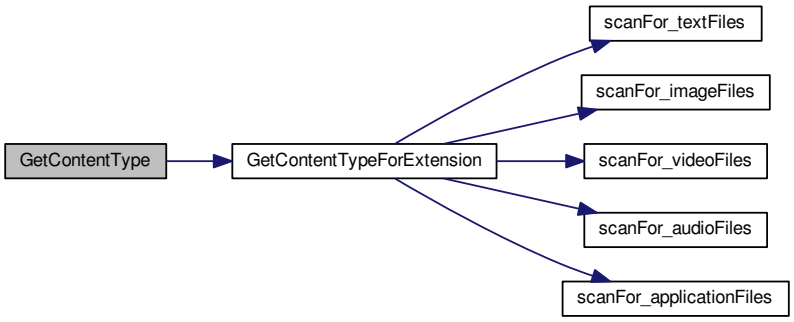
Parameters

String	with the filename we want to examine
Output	String with the contentType
Output	contentType length

Return values

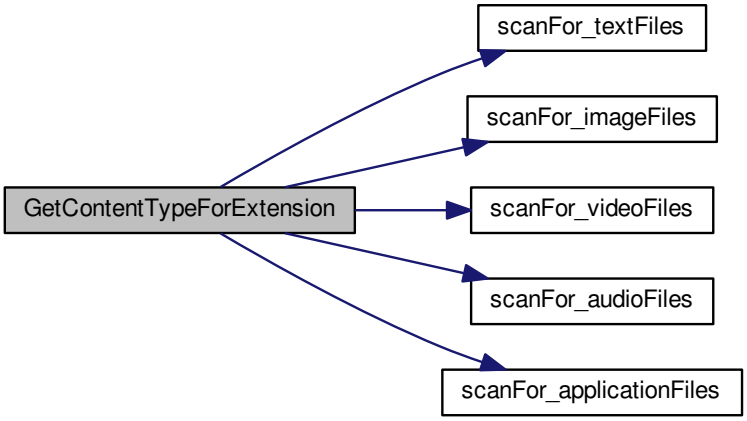
contentTypeEnumerator	

Here is the call graph for this function:



6.71.1.12 int GetContentTypeForExtension (char * *theextension*, char * *content_type*, unsigned int *contentTypeLength*)

Here is the call graph for this function:



6.71.1.13 int GetExtensionImage (char * *filename*, char * *theimagepath*, unsigned int *theimagepath_length*)

Return template image for specific content type (for directory listings etc)

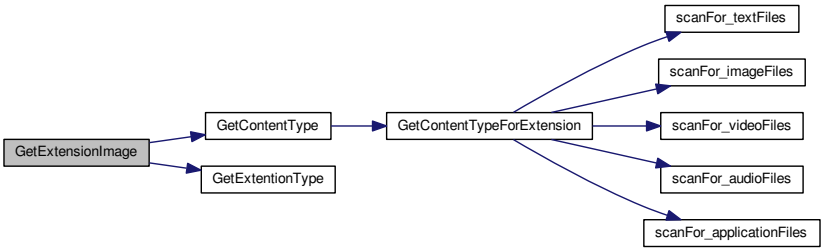
Parameters

Filename	of file
Path	to Image
Length	of path to Image

Return values

1=Exists,0=Does	not Exist

Here is the call graph for this function:



6.71.1.14 int GetExtentionType (char * *theextension*)

Convert an Extension Type to a contentTypeEnumerator.

Parameters

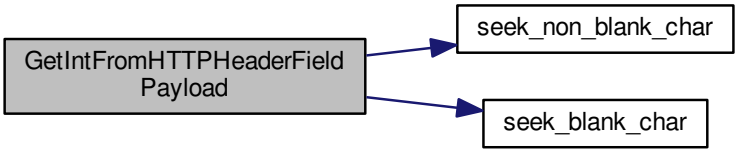
String	with the extension type

Return values

contentTypeEnumerator	

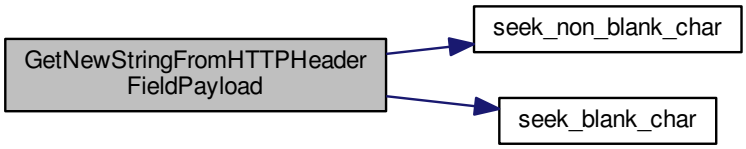
6.71.1.15 unsigned int GetIntFromHTTPHeaderFieldPayload (char * *request,* unsigned int *request_length*)

Here is the call graph for this function:



6.71.1.16 char* GetNewStringFromHTTPHeaderFieldPayload (char * *request,* unsigned int *request_length*)

Here is the call graph for this function:



6.71.1.17 int ReducePathSlashes_Inplace (char * *filename*)

Filenames may contain ///// with an arbitrary number of slashes , we convert them to a single slash ,.

Parameters

Input	string

Return values

1=Success,0=Failure	

6.71.1.18 char* RequestHTTPWebPage (char * *hostname,* unsigned int *port,* char * *filename,* unsigned int *max_content*)

A very basic http client for testing connections and maybe in the future make AmmarServers communicate with each other.

Parameters

Hostname	to connect to
Port	to connect to
Filename	to download
Maximum	size of response to carry

Return values

Pointer	to requested page,0=Failure

Here is the call graph for this function:

6.71.1.19 int seek_blank_char (char * *input,* char * *input_end*)

6.71.1.20 int seek_non_blank_char (char * *input,* char * *input_end*)

6.71.1.21 unsigned int ServerThreads_DropRootUID ( )

Drop Root UID , if we have one (and according to server_configuration.h)

Here is the call graph for this function:

6.71.1.22 int setSocketTimeouts (int *clientSock*)

Enforce socket timeouts declared in server_configuration.h and configuration files to socket.

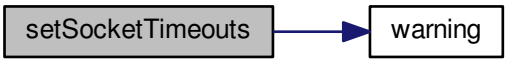
Parameters

Socket	to change

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.71.1.23 int StripGETRequestQueryAndFragment (char * *filename,* char * *query,* unsigned int *max_query_length*)

6.71.1.24 int StripHTMLCharacters_Inplace (char * *filename,* int *enable_security*)

HTML characters should be converted to plain c byte chars after we get them , this poses some security threats since this might allow "weird" bytes to get set that in conjunction with an overflow somewhere else might trick the server into executing ,.

Parameters

Input	string
Enforce	security that filters out possibly unwanted bytes..!! , bytes larger than 255 are always filtered since the sec_byte can be also triggered by ZZ or any ascii value out of 0-F for (see code) ..!

Return values

1=Success,0=Failure	

6.71.1.25 int StripVariableFromGETorPOSTString (char * *input,* char * *var_id,* char * *var_val,* unsigned int *var_val_length*)

TODO : A decent implementation here..! , input is like "idname=idvalue&idname2=idvalue2&idname3=idvalue3" , var_id is the value we are looking for var_val is the payload which has space allocated as declared in var_val_length

Here is the call graph for this function:

6.71.1.26 int stristr (char * *str1CAPS,* unsigned int *str1_length,* char * *str2CAPS,* unsigned int *str2_length,* unsigned int * *pos_found*) [inline]

6.71.1.27 int stristr2Caps (char * *str1,* unsigned int *str1_length,* char * *str2CAPS,* unsigned int *str2_length,* unsigned int * *pos_found*) [inline]

6.71.1.28 int strToUpcase (char * *strTarget,* char * *strSource,* unsigned int *strLength*)

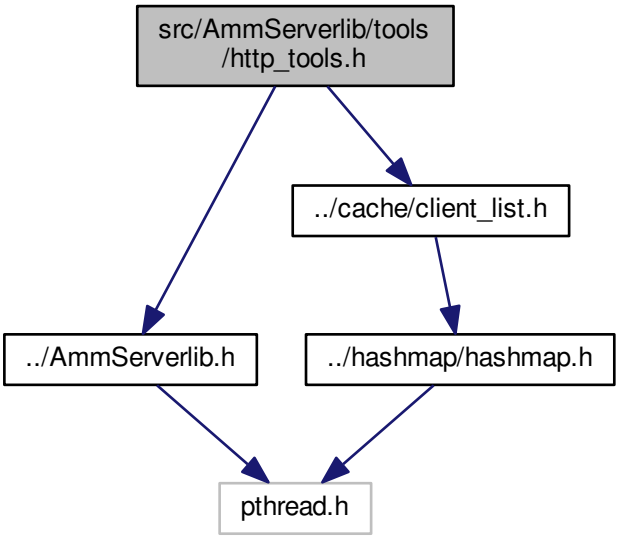
6.71.1.29 int trim_last_empty_chars (char * *input,* unsigned int *input_length*)

6.72 src/AmmServerlib/tools/http_tools.h File Reference

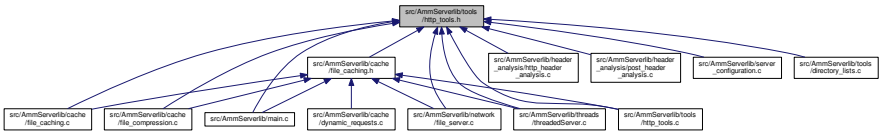
A collection of tools required by the server and gathered here since they do a very specific job.

```
#include "../AmmServerlib.h"
#include "../cache/client_list.h"
```

Include dependency graph for http_tools.h:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef unsigned int contentType

Enumerations

- enum contentTypeEnumerator {
  NO_FILETYPE =0, RESERVED_CTE_VALUE, TEXT, IMAGE,
  AUDIO, VIDEO, EXECUTABLE, FOLDER }

Functions

- unsigned int ServerThreads_DropRootUID ()

  Drop Root UID , if we have one (and according to server_configuration.h)
- char FileExistsAmmServ (char *filename)

  Check if file Exists.
- char DirectoryExistsAmmServ (char *dirpath)

  Check if directory Exists.
- int GetExtentionType (char *theextension)

  Convert an Extension Type to a contentTypeEnumerator.
- int GetContentType (char *filename, char *contentType, unsigned int contentTypeLength)

  Convert a filename to a contentType.
- int GetExtensionImage (char *filename, char *theimagepath, unsigned int theimagepath_length)

  Return template image for specific content type (for directory listings etc)
- int FindIndexFile (struct AmmServer_Instance *instance, char *webserver_root, char *directory, char *indexfile)
- int StripGETRequestQueryAndFragment (char *filename, char *query, unsigned int max_query_length)
- int StripVariableFromGETorPOSTString (char *input, char *var_id, char *var_val, unsigned int var_val_length)
- int GetDateString (char *output, char *label, unsigned int now, unsigned int dayofweek, unsigned int day, unsigned int month, unsigned int year, unsigned int hour, unsigned int minute, unsigned int second)
- int strToUpcase (char *strTarget, char *strSource, unsigned int strLength)
- int CheckHTTPHeaderCategoryAllCaps (char *lineCAPS, unsigned int line_length, char *potential_strCAPS, unsigned int *payload_start)
- int CheckHTTPHeaderCategory (char *line, unsigned int line_length, char *potential_strCAPS, unsigned int *payload_start)
- int trim_last_empty_chars (char *input, unsigned int input_length)
- int seek_non_blank_char (char *input, char *input_end)
- int seek_blank_char (char *input, char *input_end)
- unsigned int GetIntFromHTTPHeaderFieldPayload (char *request, unsigned int request_length)
- char * GetNewStringFromHTTPHeaderFieldPayload (char *request, unsigned int request_length)
- int encodeToBase64 (char *src, unsigned s_len, char *dst, unsigned d_len)

  Convert a string to base64 , required for the authorization tokens.
- int StripHTMLCharacters_Inplace (char *filename, int enable_security)

  HTML characters should be converted to plain c byte chars after we get them , this poses some security threats since this might allow "weird" bytes to get set that in conjunction with an overflow somewhere else might trick the server into executing ,.
- int ReducePathSlashes_Inplace (char *filename)

  Filenames may contain ///// with an arbitrary number of slashes , we convert them to a single slash ,.
- int FilenameStripperOk (char *filename)

  Strip filename and security check it.
- char * RequestHTTPWebPage (char *hostname, unsigned int port, char *filename, unsigned int max_content)

  A very basic http client for testing connections and maybe in the future make AmmarServers communicate with each other.
- int freeString (char **str)

  Free C string and set it to 0.

- int setSocketTimeouts (int clientSock)

  Enforce socket timeouts declared in server_configuration.h and configuration files to socket.
- clientID findOutClientIDOfPeer (struct AmmServer_Instance *instance, int clientSock)

  Tool that resolve a client socket to its IP , then uses it to try to clientList_GetClientId and returns the id number.

6.72.1 Detailed Description

A collection of tools required by the server and gathered here since they do a very specific job.

Author

Ammar Qammaz (AmmarkoV)

6.72.2 Typedef Documentation

6.72.2.1 typedef unsigned int contentType

6.72.3 Enumeration Type Documentation

6.72.3.1 enum contentTypeEnumerator

Enumerator

NO_FILETYPE

RESERVED_CTE_VALUE

TEXT

IMAGE

AUDIO

VIDEO

EXECUTABLE

FOLDER

6.72.4 Function Documentation

6.72.4.1 int CheckHTTPHeaderCategory (char * *line*, unsigned int *line_length*, char * *potential_strCAPS*, unsigned int * *payload_start*)

Here is the call graph for this function:

6.72.4.2 int CheckHTTPHeaderCategoryAllCaps (char * *lineCAPS,* unsigned int *line_length,* char * *potential_strCAPS,* unsigned int * *payload_start*)

Here is the call graph for this function:

6.72.4.3 char DirectoryExistsAmmServ (char * *dirpath*)

Check if directory Exists.

Parameters

Path	to directory

Return values

1=Exists,0=Does	not Exist

6.72.4.4 int encodeToBase64 (char * *src,* unsigned *s_len,* char * *dst,* unsigned *d_len*)

Convert a string to base64 , required for the authorization tokens.

Parameters

Input	string
Input	string length
Output	string
Input	Maximum output length

Return values

1=Success,0=Failure	

6.72.4.5 char FileExistsAmmServ (char * *filename*)

Check if file Exists.

Parameters

Path	to file

Return values

1=Exists,0=Does	not Exist

6.72.4.6 int FilenameStripperOk (char * *filename*)

Strip filename and security check it.

Parameters

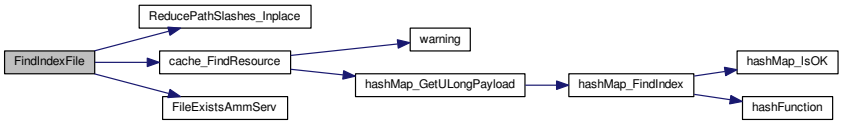
Pointer	to string pointer to be analyzed

Return values

1=Ok,0=Failed	

6.72.4.7 int FindIndexFile (struct AmmServer_Instance * *instance*, char * *webserver_root*, char * *directory*, char * *indexfile*)

Here is the call graph for this function:



6.72.4.8 clientID findOutClientIDOfPeer (struct AmmServer_Instance * *instance*, int *clientSock*)

Tool that resolve a client socket to its IP , then uses it to try to clientList_GetClientId and returns the id number.

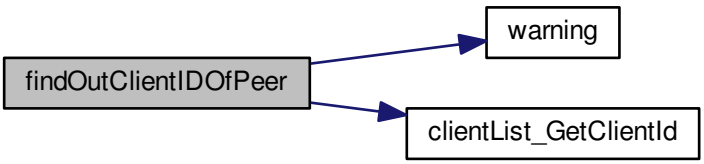
Parameters

An	AmmarServer instance
client	socket

Return values

ClientID	or ,0=Failure

Here is the call graph for this function:



6.72.4.9 int freeString (char ** *str*)

Free C string and set it to 0.

Parameters

Pointer	to string pointer to be freed

Return values

1=Success,0=Failure	

6.72.4.10 int GetContentType (char * *filename,* char * *contentType,* unsigned int *contentTypeLength*)

Convert a filename to a contentType.

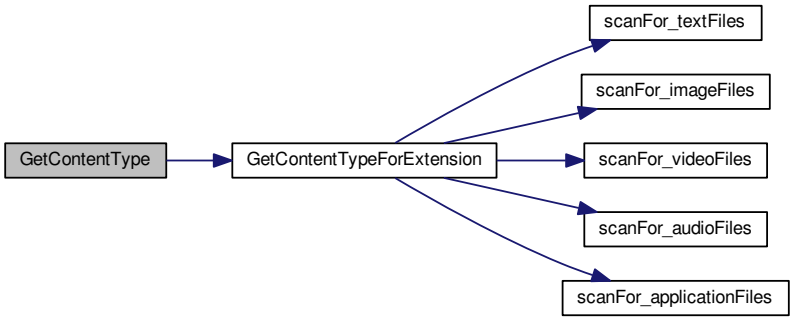
Parameters

String	with the filename we want to examine
Output	String with the contentType
Output	contentType length

Return values

contentTypeEnumerator	

Here is the call graph for this function:



6.72.4.11 int GetDateString (char * *output,* char * *label,* unsigned int *now,* unsigned int *dayofweek,* unsigned int *day,* unsigned int *month,* unsigned int *year,* unsigned int *hour,* unsigned int *minute,* unsigned int *second*)

6.72.4.12 int GetExtensionImage (char * *filename,* char * *theimagepath,* unsigned int *theimagepath_length*)

Return template image for specific content type (for directory listings etc)

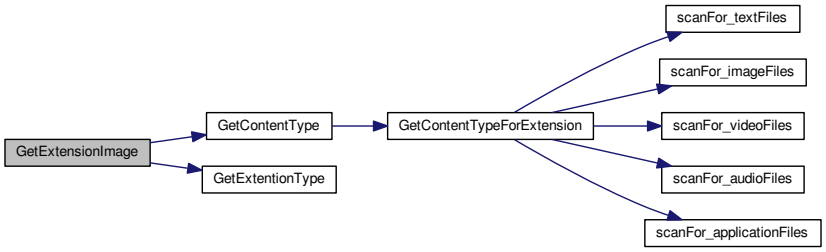
Parameters

Filename	of file
Path	to Image
Length	of path to Image

Return values

1=Exists,0=Does	not Exist

Here is the call graph for this function:



6.72.4.13 int GetExtentionType (char * *theextension*)

Convert an Extension Type to a contentTypeEnumerator.

Parameters

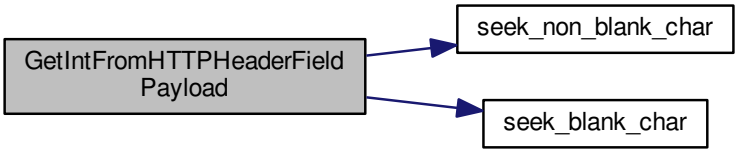
String	with the extension type

Return values

contentTypeEnumerator	

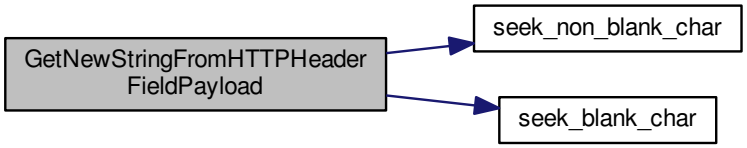
6.72.4.14 unsigned int GetIntFromHTTPHeaderFieldPayload (char * *request,* unsigned int *request_length*)

Here is the call graph for this function:



6.72.4.15 char* GetNewStringFromHTTPHeaderFieldPayload (char * *request,* unsigned int *request_length*)

Here is the call graph for this function:



6.72.4.16 int ReducePathSlashes_Inplace (char * *filename*)

Filenames may contain ///// with an arbitrary number of slashes , we convert them to a single slash ,.

Parameters

Input	string

Return values

1=Success,0=Failure	

6.72.4.17 char* RequestHTTPWebPage (char * *hostname,* unsigned int *port,* char * *filename,* unsigned int *max_content*)

A very basic http client for testing connections and maybe in the future make AmmarServers communicate with each other.

Parameters

Hostname	to connect to
Port	to connect to
Filename	to download
Maximum	size of response to carry

Return values

Pointer	to requested page,0=Failure

Here is the call graph for this function:

6.72.4.18 int seek_blank_char (char * *input*, char * *input_end*)

6.72.4.19 int seek_non_blank_char (char * *input*, char * *input_end*)

6.72.4.20 unsigned int ServerThreads_DropRootUID ( )

Drop Root UID , if we have one (and according to server_configuration.h)

Here is the call graph for this function:

6.72.4.21 int setSocketTimeouts (int *clientSock*)

Enforce socket timeouts declared in server_configuration.h and configuration files to socket.

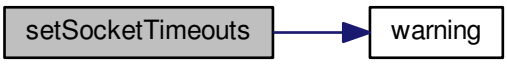
Parameters

Socket	to change

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.72.4.22 int StripGETRequestQueryAndFragment (char * *filename*, char * *query*, unsigned int *max_query_length*)

6.72.4.23 int StripHTMLCharacters_Inplace (char * *filename*, int *enable_security*)

HTML characters should be converted to plain c byte chars after we get them , this poses some security threats since this might allow "weird" bytes to get set that in conjunction with an overflow somewhere else might trick the server into executing ,.

Parameters

Input	string
Enforce	security that filters out possibly unwanted bytes..!! , bytes larger than 255 are always filtered since the sec_byte can be also triggered by ZZ or any ascii value out of 0-F for (see code) ..!

Return values

1=Success,0=Failure	

6.72.4.24 int StripVariableFromGETorPOSTString (char * *input,* char * *var_id,* char * *var_val,* unsigned int *var_val_length*)

TODO : A decent implementation here..! , input is like "idname=idvalue&idname2=idvalue2&idname3=idvalue3" , var_id is the value we are looking for var_val is the payload which has space allocated as declared in var_val_length

Here is the call graph for this function:

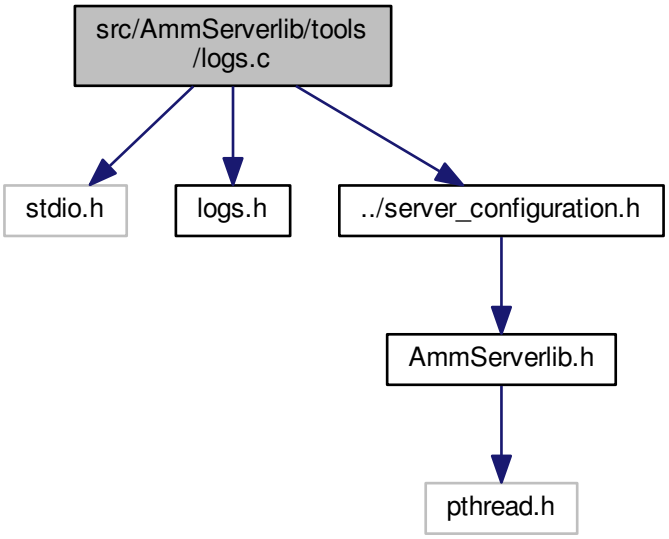
6.72.4.25 int strToUpcase (char * *strTarget,* char * *strSource,* unsigned int *strLength*)

6.72.4.26 int trim_last_empty_chars (char * *input,* unsigned int *input_length*)

6.73 src/AmmServerlib/tools/logs.c File Reference

```
#include <stdio.h>
#include "logs.h"
#include "../server_configuration.h"
```

Include dependency graph for logs.c:



Functions

- void error (char ∗msg)

  Log Function to output Errors.
- void warning (char ∗msg)

  Log Function to output warnings.
- int AccessLogAppend (char ∗IP, char ∗DateStr, char ∗Request, unsigned int ResponseCode, unsigned long ResponseLength, char ∗Location, char ∗Useragent)
- int ErrorLogAppend (char ∗IP, char ∗DateStr, char ∗Request, unsigned int ResponseCode, unsigned long ResponseLength, char ∗Location, char ∗Useragent)

6.73.1 Function Documentation

6.73.1.1 int AccessLogAppend (char ∗ *IP,* char ∗ *DateStr,* char ∗ *Request,* unsigned int *ResponseCode,* unsigned long *ResponseLength,* char ∗ *Location,* char ∗ *Useragent*)

6.73.1.2 void error (char ∗ *msg*)

Log Function to output Errors.

Parameters

String	with message To log

6.73.1.3 int ErrorLogAppend (char ∗ *IP,* char ∗ *DateStr,* char ∗ *Request,* unsigned int *ResponseCode,* unsigned long *ResponseLength,* char ∗ *Location,* char ∗ *Useragent*)

6.73.1.4 void warning (char * *msg*)

Log Function to output warnings.

Parameters

String	with message To log

6.74 src/AmmServerlib/tools/logs.h File Reference

Logging functions.

This graph shows which files directly or indirectly include this file:

Macros

- #define NORMAL "\033[0m"
- #define BLACK "\033[30m" /* Black */
- #define RED "\033[31m" /* Red */
- #define GREEN "\033[32m" /* Green */
- #define YELLOW "\033[33m" /* Yellow */
- #define BLUE "\033[34m" /* Blue */
- #define MAGENTA "\033[35m" /* Magenta */
- #define CYAN "\033[36m" /* Cyan */
- #define WHITE "\033[37m" /* White */
- #define BOLDBLACK "\033[1m\033[30m" /* Bold Black */
- #define BOLDRED "\033[1m\033[31m" /* Bold Red */
- #define BOLDGREEN "\033[1m\033[32m" /* Bold Green */
- #define BOLDYELLOW "\033[1m\033[33m" /* Bold Yellow */
- #define BOLDBLUE "\033[1m\033[34m" /* Bold Blue */
- #define BOLDMAGENTA "\033[1m\033[35m" /* Bold Magenta */
- #define BOLDCYAN "\033[1m\033[36m" /* Bold Cyan */
- #define BOLDWHITE "\033[1m\033[37m" /* Bold White */

Functions

- void error (char *msg)

  Log Function to output Errors.
- void warning (char *msg)

  Log Function to output warnings.
- int AccessLogAppend (char *IP, char *DateStr, char *Request, unsigned int ResponseCode, unsigned long ResponseLength, char *Location, char *Useragent)
- int ErrorLogAppend (char *IP, char *DateStr, char *Request, unsigned int ResponseCode, unsigned long ResponseLength, char *Location, char *Useragent)

6.74.1 Detailed Description

Logging functions.

Author

Ammar Qammaz (AmmarkoV)

6.74.2 Macro Definition Documentation

6.74.2.1 #define BLACK "\033[30m" /* Black */

6.74.2.2 #define BLUE "\033[34m" /* Blue */

6.74.2.3 #define BOLDBLACK "\033[1m\033[30m" /* Bold Black */

6.74.2.4 #define BOLDBLUE "\033[1m\033[34m" /* Bold Blue */

6.74.2.5 #define BOLDCYAN "\033[1m\033[36m" /* Bold Cyan */

6.74.2.6 #define BOLDGREEN "\033[1m\033[32m" /* Bold Green */

6.74.2.7 #define BOLDMAGENTA "\033[1m\033[35m" /* Bold Magenta */

6.74.2.8 #define BOLDRED "\033[1m\033[31m" /* Bold Red */

6.74.2.9 #define BOLDWHITE "\033[1m\033[37m" /* Bold White */

6.74.2.10 #define BOLDYELLOW "\033[1m\033[33m" /* Bold Yellow */

6.74.2.11 #define CYAN "\033[36m" /* Cyan */

6.74.2.12 #define GREEN "\033[32m" /* Green */

6.74.2.13 #define MAGENTA "\033[35m" /* Magenta */

6.74.2.14 #define NORMAL "\033[0m"

6.74.2.15 #define RED "\033[31m" /* Red */

6.74.2.16 #define WHITE "\033[37m" /* White */

6.74.2.17 #define YELLOW "\033[33m" /* Yellow */

6.74.3 Function Documentation

6.74.3.1 int AccessLogAppend (char * *IP,* char * *DateStr,* char * *Request,* unsigned int *ResponseCode,* unsigned long *ResponseLength,* char * *Location,* char * *Useragent*)

6.74.3.2 void error (char * *msg*)

Log Function to output Errors.

Parameters

String	with message To log

6.74.3.3 int ErrorLogAppend (char * *IP,* char * *DateStr,* char * *Request,* unsigned int *ResponseCode,* unsigned long *ResponseLength,* char * *Location,* char * *Useragent*)

6.74.3.4 void warning (char * *msg*)

Log Function to output warnings.

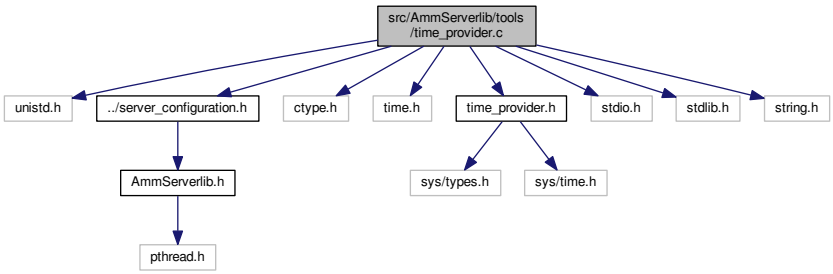
Parameters

String	with message To log

6.75 src/AmmServerlib/tools/time_provider.c File Reference

```
#include <unistd.h>
#include "../server_configuration.h"
#include <ctype.h>
#include <time.h>
#include "time_provider.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

Include dependency graph for time_provider.c:



Functions

- unsigned long GetTickCountAmmServ ()

  GetTickCount like call for functions wanting to get monotonic values in milliseconds.
- int GetDateString (char *output, char *label, unsigned int now, unsigned int dayofweek, unsigned int day, unsigned int month, unsigned int year, unsigned int hour, unsigned int minute, unsigned int second)

  Get a string back with date and time.
- int start_timer (struct time_snap *val)

  Start a timer using a time_snap structure.
- unsigned long end_timer (struct time_snap *val)

  End a started timer and get back the results.

Variables

- const char * days [] = {"Sun","Mon","Tue","Wed","Thu","Fri","Sat"}
- const char * months [] = {"Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"}

6.75.1 Function Documentation

6.75.1.1 unsigned long end_timer (struct time_snap * *val*)

End a started timer and get back the results.

Parameters

time_snap	structure that holds the timer data

Return values

Elapsed	time since start_timer , needs to be divided by 1000 to get msecs , and by 1000000 to get seconds..

6.75.1.2 int GetDateString (char * *output*, char * *label*, unsigned int *now*, unsigned int *dayofweek*, unsigned int *day*, unsigned int *month*, unsigned int *year*, unsigned int *hour*, unsigned int *minute*, unsigned int *second*)

Get a string back with date and time.

Parameters

Pointer	to where Output String should be stored
Pointer	to Label String
Flag	to control if we want to override values with the current time
Unsigned	Integer Day of Week Value
Unsigned	Integer Day
Unsigned	Integer Month
Unsigned	Integer Year
Unsigned	Integer Hour
Unsigned	Integer Minute
Unsigned	Integer Second

Return values

1=Success,0=Failure	

6.75.1.3 unsigned long GetTickCountAmmServ ( )

GetTickCount like call for functions wanting to get monotonic values in milliseconds.

Return values

Milliseconds	

6.75.1.4 int start_timer (struct time_snap * *val*)

Start a timer using a time_snap structure.

Parameters

time_snap	structure that holds the timer data

Return values

1=Success,0=Failure	

6.75.2 Variable Documentation

6.75.2.1 const char* days[ ] = {"Sun","Mon","Tue","Wed","Thu","Fri","Sat"}

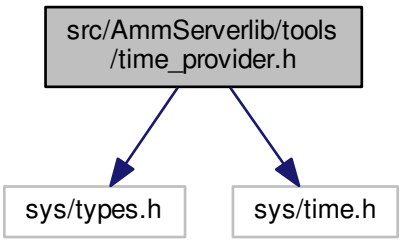
6.75.2.2 const char* months[ ] = {"Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"}

6.76 src/AmmServerlib/tools/time_provider.h File Reference

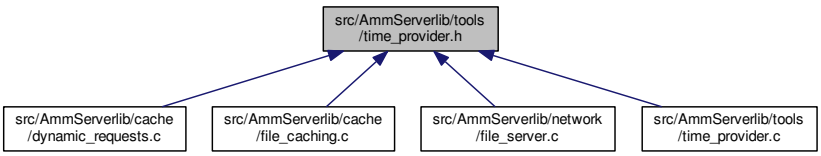
Timer functions.

```
#include <sys/types.h>
#include <sys/time.h>
```

Include dependency graph for time_provider.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct time_snap

Functions

- unsigned long GetTickCountAmmServ ()

  GetTickCount like call for functions wanting to get monotonic values in milliseconds.

- int GetDateString (char *output, char *label, unsigned int now, unsigned int dayofweek, unsigned int day, unsigned int month, unsigned int year, unsigned int hour, unsigned int minute, unsigned int second)

  Get a string back with date and time.

- int start_timer (struct time_snap *val)

  Start a timer using a time_snap structure.

- unsigned long end_timer (struct time_snap *val)

  End a started timer and get back the results.

6.76.1 Detailed Description

Timer functions.

Author

Ammar Qammaz (AmmarkoV)

6.76.2 Function Documentation

6.76.2.1 unsigned long end_timer (struct time_snap * *val*)

End a started timer and get back the results.

Parameters

time_snap	structure that holds the timer data

Return values

Elapsed	time since start_timer , needs to be divided by 1000 to get msecs , and by 1000000 to get seconds..

6.76.2.2 int GetDateString (char * *output*, char * *label*, unsigned int *now*, unsigned int *dayofweek*, unsigned int *day*, unsigned int *month*, unsigned int *year*, unsigned int *hour*, unsigned int *minute*, unsigned int *second*)

Get a string back with date and time.

Parameters

Pointer	to where Output String should be stored
Pointer	to Label String
Flag	to control if we want to override values with the current time
Unsigned	Integer Day of Week Value
Unsigned	Integer Day
Unsigned	Integer Month
Unsigned	Integer Year
Unsigned	Integer Hour
Unsigned	Integer Minute
Unsigned	Integer Second

Return values

1=Success,0=Failure	

6.76.2.3 unsigned long GetTickCountAmmServ ( )

GetTickCount like call for functions wanting to get monotonic values in milliseconds.

Return values

Milliseconds	

6.76.2.4 int start_timer (struct time_snap * *val*)

Start a timer using a time_snap structure.

Parameters

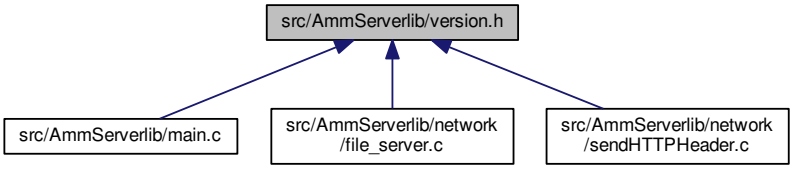
time_snap	structure that holds the timer data

Return values

1=Success,0=Failure	

6.77 src/AmmServerlib/version.h File Reference

This graph shows which files directly or indirectly include this file:



Macros

- #define RC_FILEVERSION 0,27,124,537
- #define RC_FILEVERSION_STRING "0, 27, 124, 537\0"

6.77.1 Macro Definition Documentation

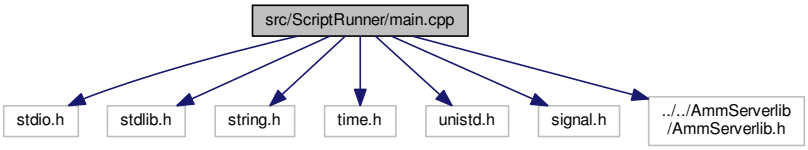
6.77.1.1 #define RC_FILEVERSION 0,27,124,537

6.77.1.2 #define RC_FILEVERSION_STRING "0, 27, 124, 537\0"

6.78 src/ScriptRunner/main.cpp File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <unistd.h>
#include <signal.h>
#include "../../AmmServerlib/AmmServerlib.h"
```

Include dependency graph for main.cpp:



Macros

- #define MAX_BINDING_PORT 65534
- #define ENABLE_PASSWORD_PROTECTION 0
- #define ENABLE_CHAT_BOX 0
- #define MAX_COMMAND_SIZE 2048
- #define DEFAULT_BINDING_PORT 8080
- #define ADMIN_BINDING_PORT 8082
- #define ENABLE_ADMIN_PAGE 0

Functions

- void replaceChar (char ∗input, char findChar, char replaceWith)
- void ∗ prepare_index_content_callback (struct AmmServer_DynamicRequest ∗rqst)
- int getBackCommandLine (char ∗command, char ∗what2GetBack, unsigned int what2GetBackMaxSize)
- void ∗ prepare_stats_content_callback (struct AmmServer_DynamicRequest ∗rqst)
- void ∗ prepare_base_image (struct AmmServer_DynamicRequest ∗rqst)
- void ∗ prepare_top_image (struct AmmServer_DynamicRequest ∗rqst)
- void joystickExecute (float x, float y)
- void execute (char ∗command, char ∗param)
- void ∗ store_new_configuration_callback (struct AmmServer_DynamicRequest ∗rqst)
- void ∗ prepare_form_content_callback (struct AmmServer_DynamicRequest ∗rqst)
- int init_dynamic_content ()
- void close_dynamic_content ()
- void termination_handler (int signum)
- int main (int argc, char ∗argv[])

Variables

- char admin_root [MAX_FILE_PATH] ="admin_html/"
- char webserver_root [MAX_FILE_PATH] ="public_html/"
- char templates_root [MAX_FILE_PATH] ="public_html/templates/"
- char ∗ page =0
- unsigned int pageLength =0
- struct AmmServer_Instance ∗ default_server =0
- struct AmmServer_Instance ∗ admin_server =0
- struct AmmServer_RequestOverride_Context GET_override ={{0}}
- struct AmmServer_RH_Context indexPage ={0}
- struct AmmServer_RH_Context settings ={0}
- struct AmmServer_RH_Context stats ={0}

- struct AmmServer_RH_Context form ={0}
- struct AmmServer_RH_Context chatbox ={0}
- struct AmmServer_RH_Context base_image ={0}
- struct AmmServer_RH_Context top_image ={0}
- struct AmmServer_RH_Context random_chars ={0}

6.78.1 Macro Definition Documentation

6.78.1.1 #define ADMIN_BINDING_PORT 8082

6.78.1.2 #define DEFAULT_BINDING_PORT 8080

6.78.1.3 #define ENABLE_ADMIN_PAGE 0

6.78.1.4 #define ENABLE_CHAT_BOX 0

6.78.1.5 #define ENABLE_PASSWORD_PROTECTION 0

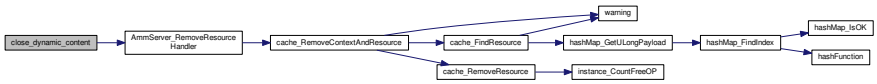
6.78.1.6 #define MAX_BINDING_PORT 65534

6.78.1.7 #define MAX_COMMAND_SIZE 2048

6.78.2 Function Documentation

6.78.2.1 void close_dynamic_content ( )

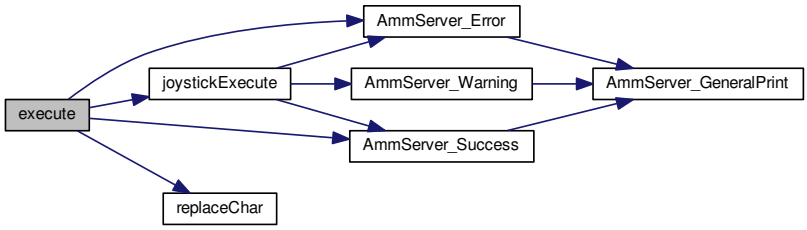
Here is the call graph for this function:



6.78.2.2 void execute (char * *command*, char * *param*)

bin/bash -c "

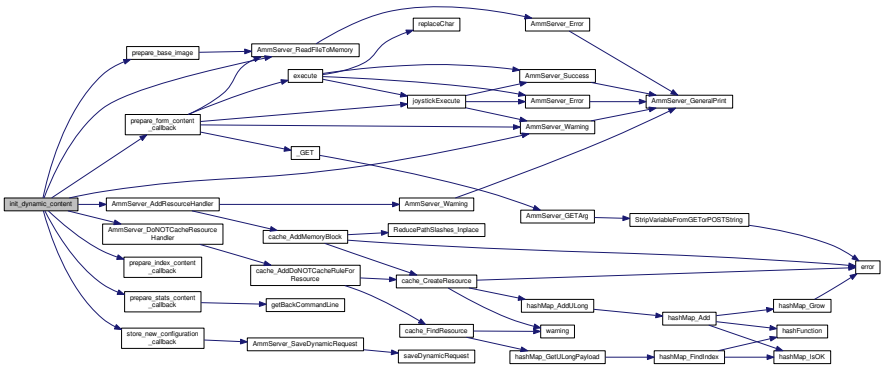
Here is the call graph for this function:



6.78.2.3 int getBackCommandLine (char * *command,* char * *what2GetBack,* unsigned int *what2GetBackMaxSize*)

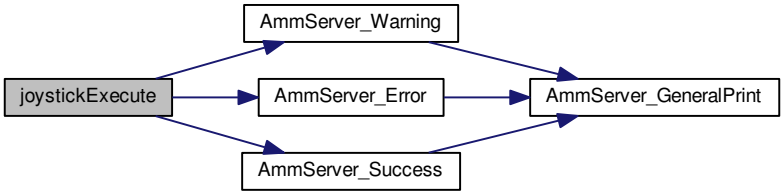
6.78.2.4 int init_dynamic_content ( )

Here is the call graph for this function:



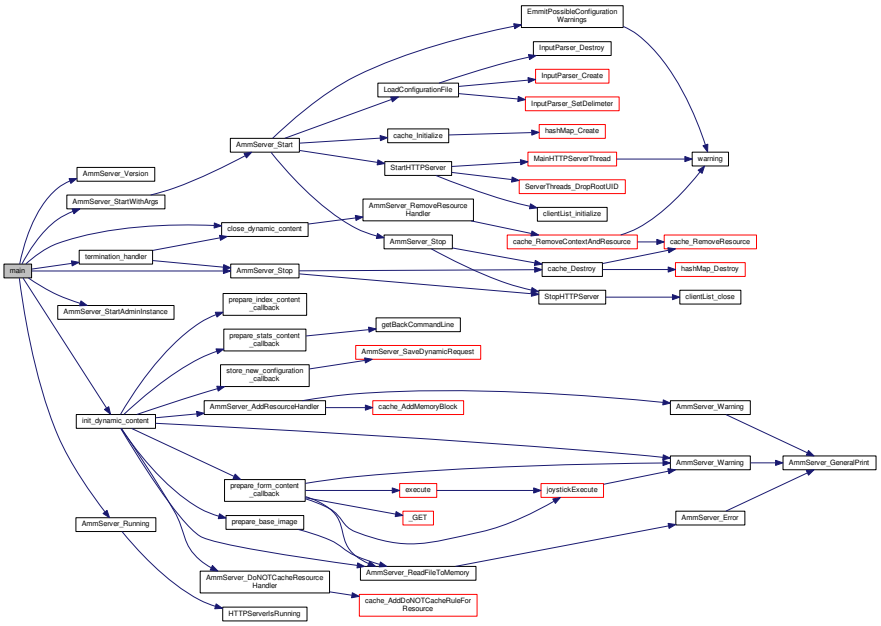
6.78.2.5 void joystickExecute (float *x,* float *y*)

Here is the call graph for this function:



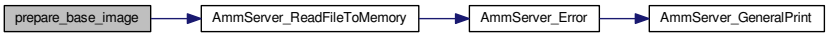
6.78.2.6 int main (int *argc*, char * *argv[]*)

Here is the call graph for this function:



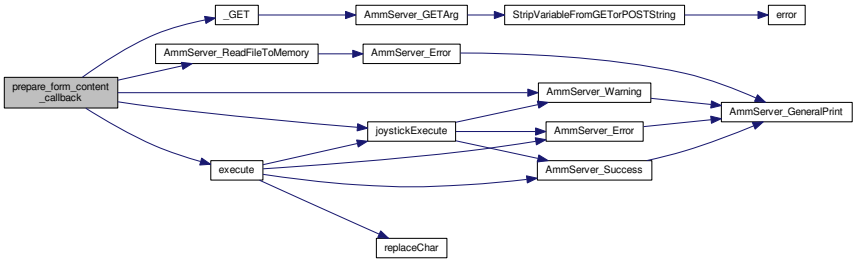
6.78.2.7 void* prepare_base_image (struct AmmServer_DynamicRequest * *rqst*)

Here is the call graph for this function:



6.78.2.8 void* prepare_form_content_callback (struct AmmServer_DynamicRequest * *rqst*)

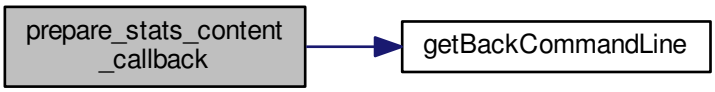
Here is the call graph for this function:



6.78.2.9 void* prepare_index_content_callback (struct AmmServer_DynamicRequest * *rqst*)

6.78.2.10 void* prepare_stats_content_callback (struct AmmServer_DynamicRequest * *rqst*)

Here is the call graph for this function:



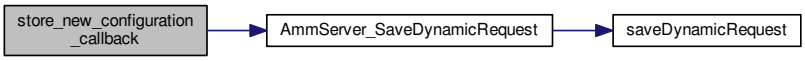
6.78.2.11 void* prepare_top_image (struct AmmServer_DynamicRequest * *rqst*)

Here is the call graph for this function:

6.78.2.12 void replaceChar (char * *input,* char *findChar,* char *replaceWith*)

6.78.2.13 void* store_new_configuration_callback (struct AmmServer_DynamicRequest * *rqst*)

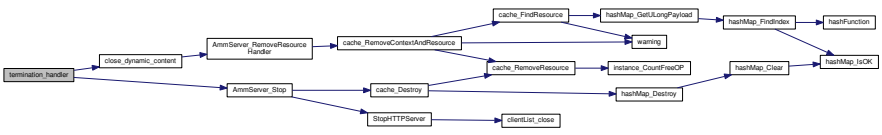
Here is the call graph for this function:



6.78.2.14 void termination_handler (int *signum*)

Dynamic content code ..! END ----------—

Here is the call graph for this function:



6.78.3 Variable Documentation

6.78.3.1 char admin_root[MAX_FILE_PATH] ="admin_html/"

6.78.3.2 struct AmmServer_Instance* admin_server =0

6.78.3.3 struct AmmServer_RH_Context base_image ={0}

6.78.3.4 struct AmmServer_RH_Context chatbox ={0}

6.78.3.5 struct AmmServer_Instance* default_server =0

Dynamic content code ..! START!

6.78.3.6 struct AmmServer_RH_Context form ={0}

6.78.3.7 struct AmmServer_RequestOverride_Context GET_override ={{0}}

6.78.3.8 struct AmmServer_RH_Context indexPage ={0}

6.78.3.9 char* page =0

6.78.3.10 unsigned int pageLength =0

6.78.3.11 struct AmmServer_RH_Context random_chars ={0}

6.78.3.12 struct AmmServer_RH_Context settings ={0}

6.78.3.13 struct AmmServer_RH_Context stats ={0}

6.78.3.14 char templates_root[MAX_FILE_PATH] ="public_html/templates/"

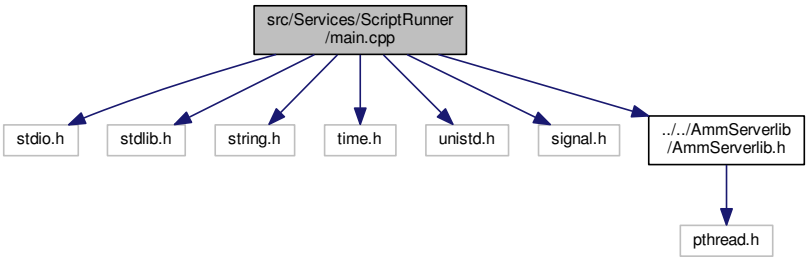
6.78.3.15 struct AmmServer_RH_Context top_image ={0}

6.78.3.16 char webserver_root[MAX_FILE_PATH] ="public_html/"

6.79 src/Services/ScriptRunner/main.cpp File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <unistd.h>
#include <signal.h>
#include "../../AmmServerlib/AmmServerlib.h"
```

Include dependency graph for main.cpp:



Macros

- #define MAX_BINDING_PORT 65534
- #define ENABLE_PASSWORD_PROTECTION 0
- #define ENABLE_CHAT_BOX 0
- #define MAX_COMMAND_SIZE 2048
- #define DEFAULT_BINDING_PORT 8080
- #define ADMIN_BINDING_PORT 8082
- #define ENABLE_ADMIN_PAGE 0

Functions

- char FileExistsTest (char ∗filename)
- char EraseFile (char ∗filename)
- unsigned int StringIsHTMLSafe (char ∗str)
- void replaceChar (char ∗input, char findChar, char replaceWith)
- void ∗ prepare_index_content_callback (struct AmmServer_DynamicRequest ∗rqst)
- int getBackCommandLine (char ∗command, char ∗what2GetBack, unsigned int what2GetBackMaxSize)
- void ∗ prepare_stats_content_callback (struct AmmServer_DynamicRequest ∗rqst)
- void ∗ prepare_base_image (struct AmmServer_DynamicRequest ∗rqst)
- void ∗ prepare_top_image (struct AmmServer_DynamicRequest ∗rqst)
- void joystickExecute (float x, float y)
- void execute (char ∗command, char ∗param)
- void ∗ store_new_configuration_callback (struct AmmServer_DynamicRequest ∗rqst)
- void ∗ prepare_form_content_callback (struct AmmServer_DynamicRequest ∗rqst)
- int init_dynamic_content ()
- void close_dynamic_content ()
- void termination_handler (int signum)
- int main (int argc, char ∗argv[])

Variables

- char admin_root [MAX_FILE_PATH] ="admin_html/"
- char webserver_root [MAX_FILE_PATH] ="public_html/"
- char templates_root [MAX_FILE_PATH] ="public_html/templates/"
- char ∗ page =0
- unsigned int pageLength =0
- struct AmmServer_Instance ∗ default_server =0

- struct AmmServer_Instance * admin_server =0
- struct AmmServer_RequestOverride_Context GET_override ={{0}}
- struct AmmServer_RH_Context indexPage ={0}
- struct AmmServer_RH_Context settings ={0}
- struct AmmServer_RH_Context stats ={0}
- struct AmmServer_RH_Context form ={0}
- struct AmmServer_RH_Context chatbox ={0}
- struct AmmServer_RH_Context base_image ={0}
- struct AmmServer_RH_Context top_image ={0}
- struct AmmServer_RH_Context random_chars ={0}

6.79.1 Macro Definition Documentation

6.79.1.1 #define ADMIN_BINDING_PORT 8082

6.79.1.2 #define DEFAULT_BINDING_PORT 8080

6.79.1.3 #define ENABLE_ADMIN_PAGE 0

6.79.1.4 #define ENABLE_CHAT_BOX 0

6.79.1.5 #define ENABLE_PASSWORD_PROTECTION 0

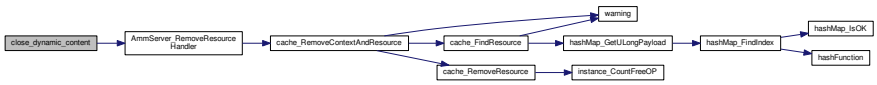
6.79.1.6 #define MAX_BINDING_PORT 65534

6.79.1.7 #define MAX_COMMAND_SIZE 2048

6.79.2 Function Documentation

6.79.2.1 void close_dynamic_content ( )

Here is the call graph for this function:

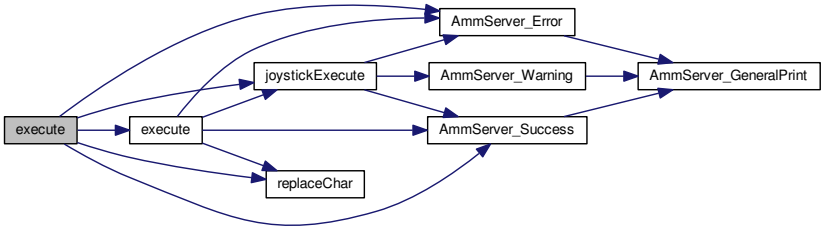


6.79.2.2 char EraseFile (char * *filename*)

6.79.2.3 void execute (char * *command,* char * *param*)

bin/bash -c "

Here is the call graph for this function:

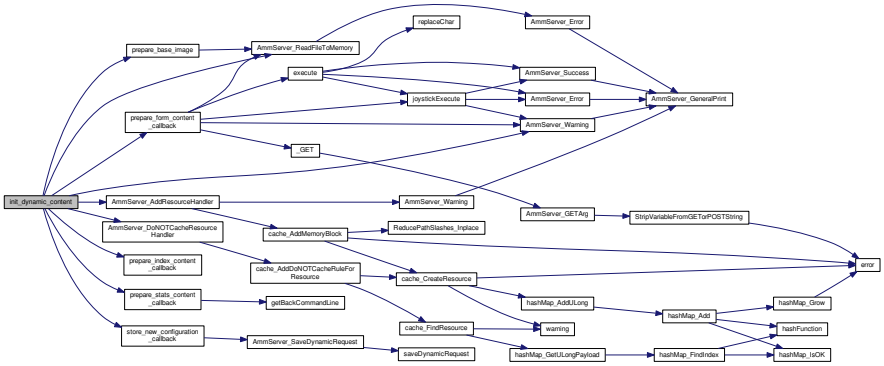


6.79.2.4 char FileExistsTest (char * *filename*)

6.79.2.5 int getBackCommandLine (char * *command*, char * *what2GetBack*, unsigned int *what2GetBackMaxSize*)

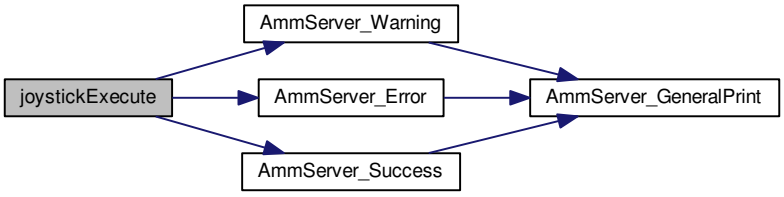
6.79.2.6 int init_dynamic_content ( )

Here is the call graph for this function:



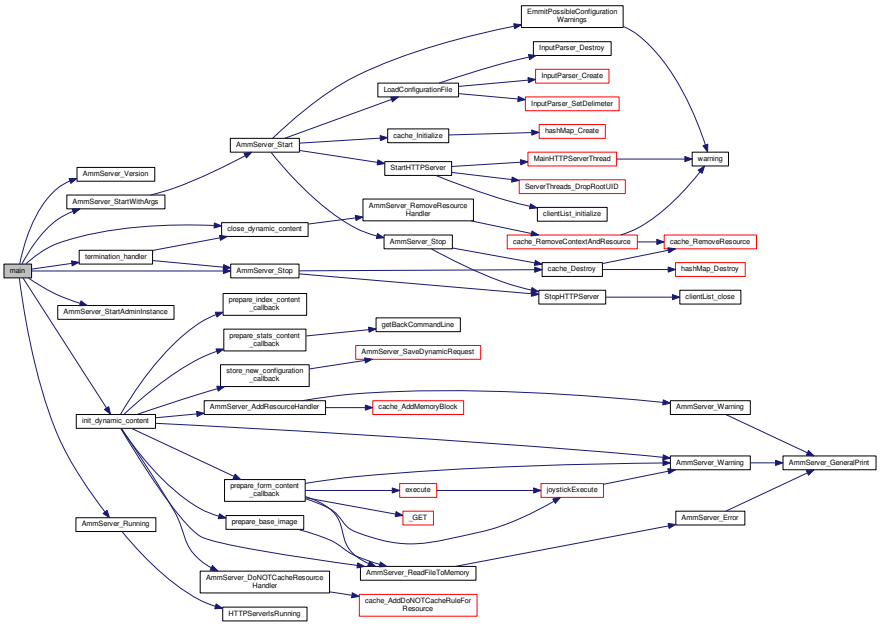
6.79.2.7 void joystickExecute (float *x*, float *y*)

Here is the call graph for this function:



6.79.2.8 int main (int *argc*, char * *argv[]*)

Here is the call graph for this function:

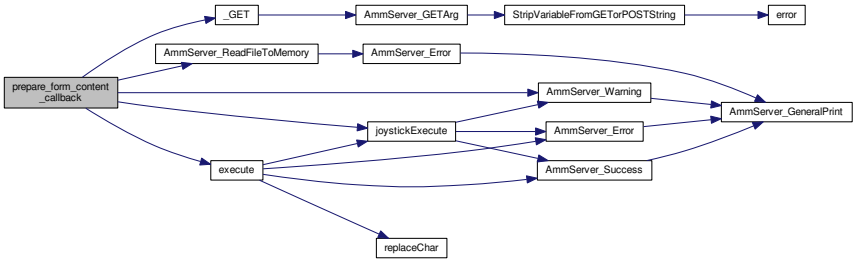


6.79.2.9 void* prepare_base_image (struct AmmServer_DynamicRequest * *rqst*)

Here is the call graph for this function:

6.79.2.10 void* prepare_form_content_callback (struct AmmServer_DynamicRequest * *rqst*)

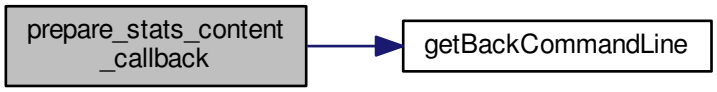
Here is the call graph for this function:



6.79.2.11 void* prepare_index_content_callback (struct AmmServer_DynamicRequest * *rqst*)

6.79.2.12 void* prepare_stats_content_callback (struct AmmServer_DynamicRequest * *rqst*)

Here is the call graph for this function:



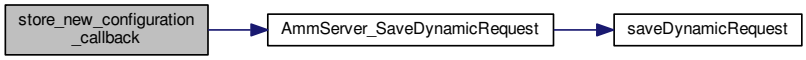
6.79.2.13 void* prepare_top_image (struct AmmServer_DynamicRequest * *rqst*)

Here is the call graph for this function:

6.79.2.14 void replaceChar (char * *input,* char *findChar,* char *replaceWith*)

6.79.2.15 void* store_new_configuration_callback (struct AmmServer_DynamicRequest * *rqst*)

Here is the call graph for this function:

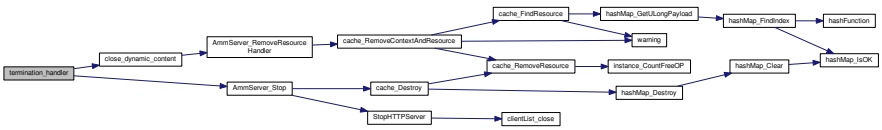


6.79.2.16 unsigned int StringIsHTMLSafe (char * *str*)

6.79.2.17 void termination_handler (int *signum*)

Dynamic content code ..! END ----------—

Here is the call graph for this function:



6.79.3 Variable Documentation

6.79.3.1 char admin_root[MAX_FILE_PATH] ="admin_html/"

6.79.3.2 struct AmmServer_Instance* admin_server =0

6.79.3.3 struct AmmServer_RH_Context base_image ={0}

6.79.3.4 struct AmmServer_RH_Context chatbox ={0}

6.79.3.5 struct AmmServer_Instance* default_server =0

Dynamic content code ..! START!

6.79.3.6 struct AmmServer_RH_Context form ={0}

6.79.3.7 struct AmmServer_RequestOverride_Context GET_override ={{0}}

6.79.3.8 struct AmmServer_RH_Context indexPage ={0}

6.79.3.9 char* page =0

6.79.3.10 unsigned int pageLength =0

6.79.3.11 struct AmmServer_RH_Context random_chars ={0}

6.79.3.12 struct AmmServer_RH_Context settings ={0}

6.79.3.13 struct AmmServer_RH_Context stats ={0}

6.79.3.14 char templates_root[MAX_FILE_PATH] ="public_html/templates/"

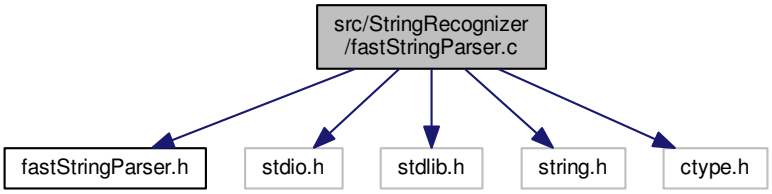
6.79.3.15 struct AmmServer_RH_Context top_image ={0}

6.79.3.16 char webserver_root[MAX_FILE_PATH] ="public_html/"

6.80 src/StringRecognizer/fastStringParser.c File Reference

```
#include "fastStringParser.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

Include dependency graph for fastStringParser.c:



Macros

- #define MAXIMUM_LINE_LENGTH 1024
- #define MAXIMUM_LEVELS 123
- #define ACTIVATED_LEVELS 3

Functions

- void convertTo_ENUM_ID (char ∗sPtr)
- int fastStringParser_addString (struct fastStringParser ∗fsp, char ∗str)
- struct fastStringParser ∗ fastStringParser_initialize (unsigned int totalStrings)
- int fastStringParser_hasStringsWithNConsecutiveChars (struct fastStringParser ∗fsp, unsigned int ∗resStringResultIndex, char ∗Sequence, unsigned int seqLength)
- unsigned int fastStringParser_countStringsForNextChar (struct fastStringParser ∗fsp, unsigned int ∗resStringResultIndex, char ∗Sequence, unsigned int seqLength)
- void addLevelSpaces (FILE ∗fp, unsigned int level)
- int printIfAllPossibleStrings (FILE ∗fp, struct fastStringParser ∗fsp, char ∗Sequence, unsigned int seqLength)
- int printAllEnumeratorItems (FILE ∗fp, struct fastStringParser ∗fsp, char ∗functionName)
- int recursiveTraverser (FILE ∗fp, struct fastStringParser ∗fsp, char ∗functionName, char ∗cArray, unsigned int level)
- int export_C_Scanner (struct fastStringParser ∗fsp, char ∗functionName)

  Export a C Scanner source code.
- struct fastStringParser ∗ fastSTringParser_createRulesFromFile (char ∗filename, unsigned int totalStrings)

  Read a file and create C files that parse the input.
- int fastStringParser_close ()

Variables

- struct fastStringParser ∗ fspHTTPHeader = 0
- char acceptedChars [] ="ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789-_"

6.80.1 Macro Definition Documentation

6.80.1.1 #define ACTIVATED_LEVELS 3

6.80.1.2 #define MAXIMUM_LEVELS 123

6.80.1.3 #define MAXIMUM_LINE_LENGTH 1024

6.80.2 Function Documentation

6.80.2.1 void addLevelSpaces (FILE * *fp*, unsigned int *level*)

6.80.2.2 void convertTo_ENUM_ID (char * *sPtr*) [inline]

6.80.2.3 int export_C_Scanner (struct **fastStringParser** * *fsp*, char * *filename*)

Export a C Scanner source code.

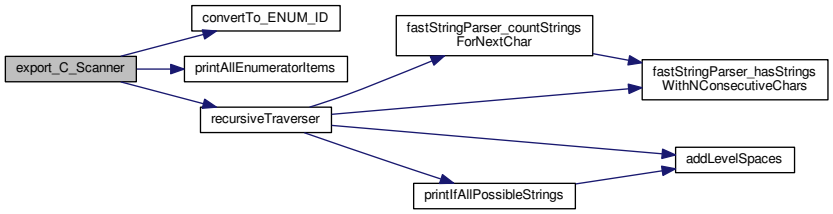
Parameters

Structure	to hold all the intermediate state
Name	of the current function

Return values

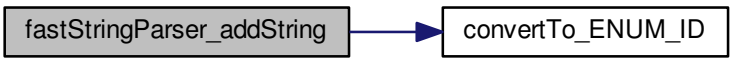
1=Success,0=Failure	

Here is the call graph for this function:



6.80.2.4 int fastStringParser_addString (struct **fastStringParser** * *fsp*, char * *str*)

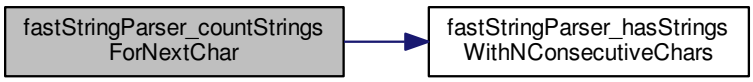
Here is the call graph for this function:



6.80.2.5 int fastStringParser_close ( )

6.80.2.6 unsigned int fastStringParser_countStringsForNextChar (struct **fastStringParser** * *fsp,* unsigned int * *resStringResultIndex,* char * *Sequence,* unsigned int *seqLength*)

Here is the call graph for this function:



6.80.2.7 struct **fastStringParser*** fastSTringParser_createRulesFromFile (char * *filename,* unsigned int *totalStrings*)

Read a file and create C files that parse the input.

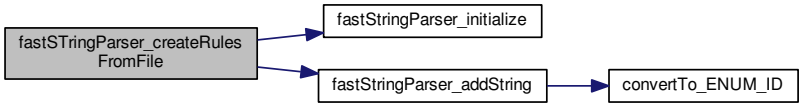
Parameters

Filename	of the current function
Total	Number of Strings

Return values

fastStringParser	context,0=Failure

Here is the call graph for this function:



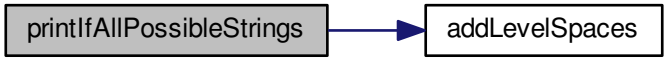
6.80.2.8 int fastStringParser_hasStringsWithNConsecutiveChars (struct **fastStringParser** * *fsp,* unsigned int * *resStringResultIndex,* char * *Sequence,* unsigned int *seqLength*)

6.80.2.9 struct **fastStringParser*** fastStringParser_initialize (unsigned int *totalStrings*)

6.80.2.10 int printAllEnumeratorItems (FILE * *fp,* struct **fastStringParser** * *fsp,* char * *functionName*)

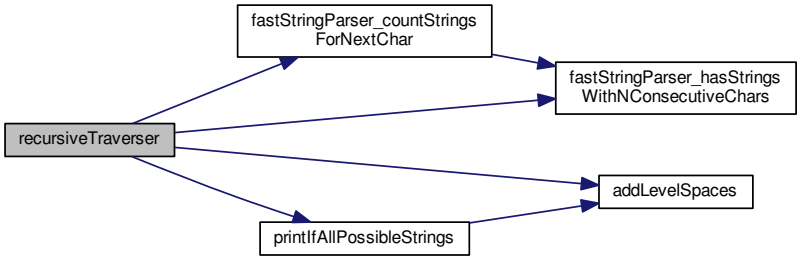
6.80.2.11 int printIfAllPossibleStrings (FILE * *fp,* struct **fastStringParser** * *fsp,* char * *Sequence,* unsigned int *seqLength*)

Here is the call graph for this function:



6.80.2.12 int recursiveTraverser (FILE * *fp,* struct **fastStringParser** * *fsp,* char * *functionName,* char * *cArray,* unsigned int *level*)

Here is the call graph for this function:



6.80.3 Variable Documentation

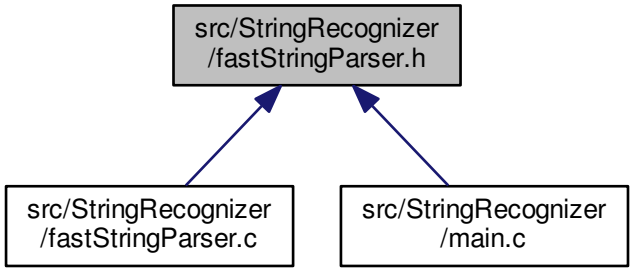
6.80.3.1 char acceptedChars[ ] ="ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789-_"

6.80.3.2 struct **fastStringParser*** fspHTTPHeader = 0

6.81 src/StringRecognizer/fastStringParser.h File Reference

A tool that converts a file with words (each word on a new line) to C code (see automata) for fast string checking.

This graph shows which files directly or indirectly include this file:



Data Structures

- struct fspString

  Internal Structure to hold a string and its id for further processing.
- struct fastStringParser

  Internal Structure that holds all the string parser context.

Functions

- int export_C_Scanner (struct fastStringParser ∗fsp, char ∗filename)

  Export a C Scanner source code.
- struct fastStringParser ∗ fastSTringParser_createRulesFromFile (char ∗filename, unsigned int totalStrings)

  Read a file and create C files that parse the input.

6.81.1 Detailed Description

A tool that converts a file with words (each word on a new line) to C code (see automata) for fast string checking.

Author

Ammar Qammaz (AmmarkoV)

6.81.2 Function Documentation

6.81.2.1 int export_C_Scanner (struct fastStringParser ∗ *fsp*, char ∗ *filename*)

Export a C Scanner source code.

Parameters

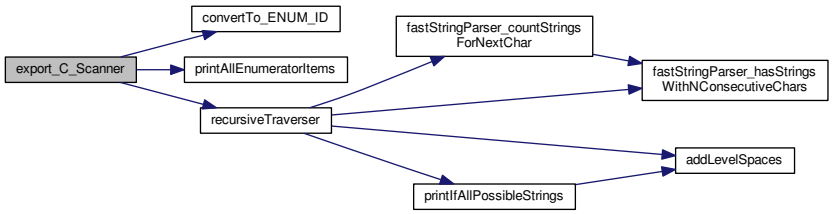
Structure	to hold all the intermediate state

Name	of the current function

Return values

1=Success,0=Failure	

Here is the call graph for this function:



6.81.2.2 struct fastStringParser* fastSTringParser_createRulesFromFile (char * *filename*, unsigned int *totalStrings*)

Read a file and create C files that parse the input.

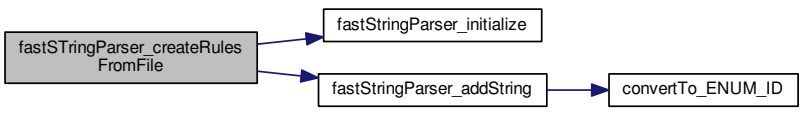
Parameters

Filename	of the current function
Total	Number of Strings

Return values

fastStringParser	context,0=Failure

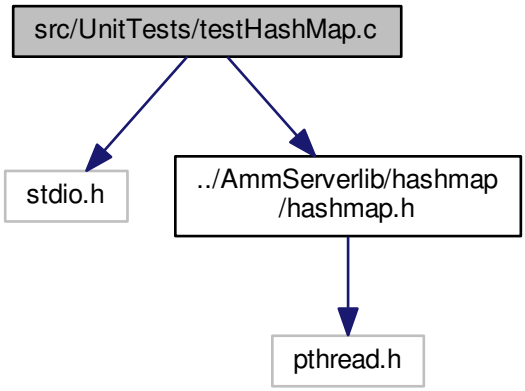
Here is the call graph for this function:



6.82 src/UnitTests/testHashMap.c File Reference

```
#include <stdio.h>
#include "../AmmServerlib/hashmap/hashmap.h"
```

Include dependency graph for testHashMap.c:



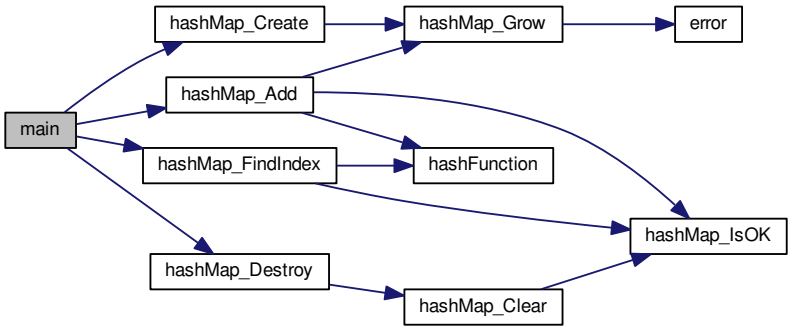
Functions

- int main (int argc, char ∗argv[])

6.82.1 Function Documentation

6.82.1.1 int main (int *argc,* char ∗ *argv[]*)

Here is the call graph for this function:



Index