zia Reference Manual

0

Generated by Doxygen 1.4.5

Tue Dec 6 02:44:41 2005

Contents

Chapter 1

zia Directory Hierarchy

1.1 zia Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

Chapter 2

zia Namespace Index

2.1 zia Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 3

zia Hierarchical Index

3.1 zia Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

zia Data Structure Index

4.1 zia Data Structures

Here are the data structures with brief descriptions:

Chapter 5

zia Data Structure Documentation

5.1 dataman::buffer Class Reference

buffer class

`#include <buffer.hh>`

Public Member Functions

- **buffer** (const unsigned char *, **size_t**)

  Fetch a buffer from the c-like buffer passed in argument.

- **buffer** (const **buffer** &)

  Fetch a buffer from the buffer passed in argument.

- **buffer** (**sysapi::file::handle_t** &)

  Fetch a buffer from a file.

- **size_t size** () const

  return the size of the buffer

- char * **c_str** () const

  return an allocated c-like string

- void **display** () const

  turn the buffer into a human readable form and dump it on std::cout

- unsigned char * **dup** () const

  return a duplicated buffer. Memory is allocated for the new buffer.

- void **reset** ()

  Reset the buffer, deallocating memory if already allocated.

- **buffer operator+** (const **buffer** &)

  Return a new buffer form by adding this one and the one passed as argument.

- **buffer** & **operator+=** (const **buffer** &)

  Add the buffer passed in argument to this one, return this one.

- **buffer** & **operator=** (const **buffer** &)

  Delete this buffer if already allocated and affect to the one passed as argument.

- unsigned char & **operator[]** (int)

  Return the byte at index i in the buffer.

- **operator unsigned char** * ()

  Return a pointer to the internal buffer. Memory is not allocated.

5.1.1 Detailed Description

buffer class

Buffers are used to manage memory allocated resources, in order to be able to internally specialize a storage method for a given resouce. Furthermore, centralizing memory managed resources helps in program debugging.

Definition at line 24 of file buffer.hh.

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

- C:/home/texane/wip/ept3/zia/branches/ziahttpd-mod/src/include/dataman/**buffer.hh**

5.2 server::core Class Reference

Server core.

`#include <core.hh>`

Public Member Functions

- **dataman::conf** & **conf** ()

Static Public Attributes

- static **service** * **services_**

Friends

- class **service**
- class **http::session_manager**

5.2.1 Detailed Description

Server core.

The zia http server is a modular one. As Apache, the server is divided into modules, so that the core is very minimalistic. It contains io, thread and module managers, where as the two first ones could be exported...

Definition at line 28 of file core.hh.

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

- C:/home/texane/wip/ept3/zia/branches/ziahttpd-mod/src/include/server/core.hh

5.3 thrman::ioselect Class Reference

Inheritance diagram for thrman::ioselect::

5.3.1 Detailed Description

Definition at line 33 of file ioselect.hh.

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

- C:/home/texane/wip/ept3/zia/branches/ziahttpd-mod/src/include/server/ioselect.hh

5.4 http::message Class Reference

Public Member Functions

- std::map< std::string, std::string > & **getquery** ()
- std::map< std::string, std::string > & **postquery** ()

5.4.1 Detailed Description

Definition at line 34 of file message.hh.

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

- C:/home/texane/wip/ept3/zia/branches/ziahttpd-mod/src/include/http/message.hh

5.5 server::modman Class Reference

Module manager.

`#include <modman.hh>`

Public Types

- enum **stageid_t**

Public Member Functions

- bool **load_at_beginning** (const std::string &, bool priviledged=false, bool activ=true)
  load a module at the beginning of the list.
- bool **load_at_end** (const std::string &, bool priviledged=false, bool activ=true)
  load a module at the end of the list.
- bool **load** (const std::string &, const std::string &, const std::string &, bool=false, bool=true, bool=true)
  load a module between two others.
- bool **reload** (const std::string &, bool priviledged=false, bool activ=true)
  reload a module if it exists, preserving the position in the list.
- bool **unload** (const std::string &)
  Unload an existing module.
- bool **start** (const std::string &)
  Start the module.
- bool **stop** (const std::string &)
  Stop the module.
- bool **state** (const std::string &, int &)
  Tell why the module is in state...
- **module** * **operator[]** (const std::string &)
  Get the module identified by id.

5.5.1 Detailed Description

Module manager.

Handle module management. Provide methods to handle module dependencies, cold module reloading/unloading, running related function. Modules can have one of the two priviledge level. A priviledged module can access to the server core datas. Modules have to register hooks being called at different stages of the request processing flow. See the API documentation for more information on how a request is processed by the server core. **TODOLIST:**

1. Improve the module system, in order to include list of pending sessions for a given module(modules have to pass information).

2. The above problem might be solved by adding a current_operation attribute in the session, in order for the module to check completion status of the services it called.

Definition at line 25 of file modman.hh.

5.5.2 Member Function Documentation

5.5.2.1 bool server::modman::load (const std::string & *after_id*, const std::string & *my_id*, const std::string & *before_id*, bool *priviledged* = `false`, bool *activ* = `true`, bool *load_missing* = `true`)

load a module between two others.

Parameters:

after_id path identifying the module to load after

my_id path identifying the module to load

before_id path identifying the module to load before

priviledged wether or not the module is a priviledged one

activ is the module activated at loading

load_missing load the missing module

Returns:

false on error (either the module is not found, permission denied...).

Load the module identified by my_id AFTER after_id, and BEFORE before_id. If the one or all module doesn't exist, the boolean load_missing decides wether or not to load them.

5.5.2.2 bool server::modman::load_at_beginning (const std::string & *id*, bool *priviledged* = `false`, bool *activ* = `true`)

load a module at the beginning of the list.

Parameters:

id Path identifying the module

priviledged wether or not the module is a priviledged one

activ is the module activated at loading

Returns:

false on error (either the module is not found, permission denied...).

Load a module at the beginning of the modlist_.

5.5.2.3 bool server::modman::load_at_end (const std::string & *id*, bool *priviledged* = `false`, bool *activ* = `true`)

load a module at the end of the list.

Parameters:

id Path identifying the module

priviledged wether or not the module is a priviledged one

activ is the module activated at loading

Returns:

false on error (either the module is not found, permission denied...).

Load a module at the end of the modlist_.

5.5.2.4 server::module & server::modman::operator[] (const std::string & *id*)

Get the module identified by id.

Parameters:

id Path identifying the module

Returns:

A reference to the module pointer contained in modlist_

Get the module identified by id.

5.5.2.5 bool server::modman::reload (const std::string & *id*, bool *priviledged* = `false`, bool *activ* = `true`)

reload a module if it exists, preserving the position in the list.

Parameters:

id Path identifying the module

priviledged wether or not the module is a priviledged one

activ is the module activated at loading

Returns:

false if the module isnot present, or cannot be accessed.

Reload a module at the same place in modlist_.

5.5.2.6 bool server::modman::start (const std::string & *id*)

Start the module.

Parameters:

id Path identifying the module

Returns:

false if the module isnot present or already running.

Start the module.

5.5.2.7 bool server::modman::state (const std::string & *id*, int & *st*)

Tell why the module is in state...

Parameters:
id Path identifying the module
st Code of the module state

Returns:
false if the module isnot present.

Tell why the module is in state...

5.5.2.8 bool server::modman::stop (const std::string & *id*)

Stop the module.

Parameters:
id Path identifying the module

Returns:
false if the module isnot present or not running.

Stop the module.

5.5.2.9 bool server::modman::unload (const std::string & *id*)

Unload an existing module.

Parameters:
id Path identifying the module

Returns:
false if the module isnot present.

Unload the module identified by id in modlist_.

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

- C:/home/texane/wip/ept3/zia/branches/ziahttpd-mod/src/include/server/modman.hh
- C:/home/texane/wip/ept3/zia/branches/ziahttpd-mod/src/include/server/module.hh

5.6 server::module Class Reference

Modules implement server functionalities extension.

`#include <module.hh>`

Public Types

- typedef bool(∗ **hook_t**)(**http::session** &, **server::core** ∗, int &reason)
- enum **reason_t**
- enum **role_t**
- enum **statecode_t**

Data Fields

- **sysapi::shared_object::handle_t hobj_**
- **hook_t hk_create_con_**
- **hook_t hk_get_rqstmetadata_**
- **hook_t hk_get_rqstdata_**
- **hook_t hk_parse_rqstmetadata_**
- **hook_t hk_alter_rqstdata_**
- **hook_t hk_build_respmetadata_**
- **hook_t hk_build_respdata_**
- **hook_t hk_alter_respdata_**
- **hook_t hk_alter_respmetadata_**
- **hook_t hk_send_response_**
- **hook_t hk_release_con_**
- std::string **name_**
- bool **priviledged_**
- **role_t role_**
- **statecode_t stcode_**
- bool **running_**

5.6.1 Detailed Description

Modules implement server functionalities extension.

Modules register hooks to be called at different stages of the request processing flow. The flow is broken into X stages. Here is a detailed explaination of the steps involved:

1. Create a new internet socket for the incoming connection
   (a) **CON_CREATION HOOK**: For ssl module, a special socket is to be created
2. Read data from the socket
   (a) **GET_RQSTMETADATA_HOOK**: For ssl module, a special read function
   (b) **GET_RQSTADATA_HOOK**
3. Actually process the request
   (a) **PARSE_REQUEST_METADATA_HOOK**: (in the case we are not dealing with proto)

   (b) **ALTER_REQUEST_DATA_HOOK**: (for mod_alias, mod_mime...)

4. Build response, including metadata building and content generation
   (a) **BUILD_RESPMETADATA_HOOK**: Construct response status and header lines...
   (b) **BUILD_RESPDATA_HOOK**: cgi execution, go reading a file on disk, generate error pages...
5. Last chance to alter repsonse before it is sent to client
   (a) **ALTER_RESPDATA_HOOK**: A previously loaded module might want to see how the response was modified by other after it has processed it
   (b) **ALTER_RESPMETADATA_HOOK**
6. Send the response to the client
   (a) **SEND_RESPONSE_HOOK**
7. Release the session connection
   (a) **CON_RELEASING_HOOK** Other notes on modules:

There can be multiple hooks registered for a given stage, allowing hook chaining.

1. Non priviledged modules can only access the current session, containing informations about the current request (buffer, internal representation, accessed resource...).
2. Priviledged modules can access the server core internals.
3. Modules have dependencies, handled at load time.
4. Modules may have roles (?)
5. Modules can be in a running state or not

Definition at line 26 of file module.hh.

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

- C:/home/texane/wip/ept3/zia/branches/ziahttpd-mod/src/include/server/module.hh

5.7 http::msgdata Class Reference

Http messages data manipulation.

`#include <msgdata.hh>`

Public Member Functions

- std::string & **operator[]** (const std::string &)
  get string data from a key value
- std::string & **method_string** ()
- std::string & **version_string** ()
- std::string & **uri_string** ()
- bool & **body** ()

5.7.1 Detailed Description

Http messages data manipulation.

This class have 2 Functionality first one is parse and store request from the client second is build the response line (status line + header lines) Module can access to the header lines for get or out information.

Definition at line 24 of file msgdata.hh.

5.7.2 Member Function Documentation

5.7.2.1 std::string & http::msgdata::operator[] (const std::string &)

get string data from a key value

Parameters:
key value

Returns:
string data

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

- C:/home/texane/wip/ept3/zia/branches/ziahttpd-mod/src/include/http/msgdata.hh

5.8 dataman::resource Class Reference

Inheritance diagram for dataman::resource::

5.8.1 Detailed Description

Definition at line 24 of file resource.hh.

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

- C:/home/texane/wip/ept3/zia/branches/ziahttpd-mod/src/include/dataman/resource.hh

5.9 dataman::resource Class Reference

Inheritance diagram for dataman::resource::

5.9.1 Detailed Description

Definition at line 24 of file resource.hh.

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

- C:/home/texane/wip/ept3/zia/branches/ziahttpd-mod/src/include/dataman/resource.hh

5.10 server::service Class Reference

Services exported by the server to modules.

`#include <service.hh>`

Public Types

- typedef **server::core** * **security_token_t**
- enum **eventid_t**

Public Member Functions

- virtual void **echo** (const std::string &)

  Echo the message.

- virtual bool **load_module** (const **security_token_t** &, const std::string &, const std::string &, const std::string &)

  load the module between and after another ones.

- virtual bool **unload_module** (const **security_token_t** &, const std::string &)

  Unload the module.

- virtual bool **stat_module** (const **security_token_t** &, const std::string &)

  Stat the module.

5.10.1 Detailed Description

Services exported by the server to modules.

Modules have to perform actions when processing data; Some of those actions need interactions with server internals, for instance io related operation. In order to do so, the server export services to modules by the way of the service class. **TODOLIST: (zihappy members involved here)**

1. Add a way to communicate error code between services and modules
2. See for service method names
3. Define configuration access
4. Don't use **dataman::buffer**(p. 10) instead of unsigned char*
5. Don't use **sysapi::socket_in::handle_t**(p. ??)
6. Think about a callback system for the server to communicate with the module
7. Defines an information vector for module stating (modstat_t)

Definition at line 26 of file service.hh.

5.10.2 Member Function Documentation

5.10.2.1 void server::service::echo (const std::string & *msg*) [virtual]

Echo the message.

Parameters:

msg Message the module wants the server to ouput

Returns:

no returned value

Let the server output a message, testing purpose

5.10.2.2 bool server::service::load_module (const security_token_t & *tok*, const std::string & *after*, const std::string & *target*, const std::string & *before*) [virtual]

load the module between and after another ones.

Parameters:

tok Security token passed by the server to module at hook call time

after Name of the module after which I wanna be loaded

target The name of the module to be loaded

before Name of the module before which I wanna be loaded

Returns:

True if the module is loaded, false otherwise.

Load the module after and before other ones. This operation can be denied by the server if the security token is invalid.

5.10.2.3 bool server::service::stat_module (const security_token_t & *tok*, const std::string & *target*) [virtual]

Stat the module.

Parameters:

target Name of the module to be stated.

Returns:

False if the module cannot be found(the module has not been loaded); Else return true.

Stat the module; **Stat structure has not yet been defined**

5.10.2.4 bool server::service::unload_module (const security_token_t & *tok*, const std::string & *target*) [virtual]

Unload the module.

Parameters:

tok Security token passed by the server to module at hook call time

target Name of the module to be unloaded

Returns:

True if the module has been unloaded, false otherwise.

Unload the module. Security token is not yet implemented, but see the above describtion.

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

- C:/home/texane/wip/ept3/zia/branches/ziahttpd-mod/src/include/server/service.hh

5.11 http::session Class Reference

Http related request data storage class.

`#include <session.hh>`

Public Member Functions

- **sysapi::socket_in::handle_t** & **hsock_con** ()
- **sysapi::socket_in::handle_t** & **hsock_srv** ()
- **dataman::resource** * **resource** ()
- **dataman::resource** * **resource_in** ()
- std::list< **dataman::buffer** > & **hdrlines_in** ()
- **dataman::buffer** & **content_in** ()
- **dataman::buffer** & **hdrlines_out** ()
- **dataman::buffer** & **content_out** ()
- **http::uri** & **uri** ()
- **http::msgdata** & **info_in** ()
- **http::msgdata** & **info_out** ()
- bool & **persistent** ()
- bool & **chunked** ()
- bool & **first_chunk** ()
- bool & **last_chunk** ()
- bool & **handleio** ()
- **dataman::conf** & **conf** ()

Data Fields

- **server::service** * **services_**

Friends

- class **session_manager**
- class **server::service**
- class **server::core**
- class **server::modman**

5.11.1 Detailed Description

Http related request data storage class.

As the request is processing, the core has to maintain the request changing state and data associated. In order to do so, the `http::session` is used. This is this chunk of data that is passed to modules for them to interact with the request processing flow.

Definition at line 39 of file session.hh.

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

- C:/home/texane/wip/ept3/zia/branches/ziahttpd-mod/src/include/http/session.hh

5.12 http::session_manager Class Reference

5.12.1 Detailed Description

Definition at line 148 of file session.hh.

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

- C:/home/texane/wip/ept3/zia/branches/ziahttpd-mod/src/include/http/session.hh

5.13 server::sockioman Class Reference

Inheritance diagram for server::sockioman::

Public Types

- typedef bool(* **sockiohandler_t**)(**sysapi::socket_in::handle_t** &, **dataman::buffer** *, **sysapi::socket_in::error_t** &)

Data Structures

- struct **sockio**

5.13.1 Detailed Description

Definition at line 31 of file sockioman.hh.

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

- C:/home/texane/wip/ept3/zia/branches/ziahttpd-mod/src/include/server/sockioman.hh

5.14 server::sockioman::sockio Struct Reference

Data Fields

- bool **done_**
- bool **used_**
- **sysapi::socket_in::handle_t hsock_**
- **dataman::buffer * rdbuf_**
- **dataman::buffer * wrbuf_**
- **sockiohandler_t onread_**
- **sockiohandler_t onwrite_**
- **sockiohandler_t onclose_**
- **sockiohandler_t ontimeout_**
- **sockioman * ioman_**

5.14.1 Detailed Description

Definition at line 46 of file sockioman.hh.

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

- C:/home/texane/wip/ept3/zia/branches/ziahttpd-mod/src/include/server/sockioman.hh

5.15 C:/home/texane/wip/ept3/zia/branches/ziahttpd-mod/src/include/dataman/buffer.hh File Reference

Buffer class.

```
#include <string>
#include <cstdlib>
#include <http/uri.hh>
#include <sysapi/sysapi.hh>
```

Namespaces

- namespace **dataman**

Data Structures

- class **dataman::buffer**
  buffer class

5.15.1 Detailed Description

Buffer class.

Buffers are used to manage memory allocated resources, in order to be able to internally specialize a storage method for a given resouce. Furthermore, centralizing memory managed resources helps in program debugging.

Definition in file **buffer.hh**.