

HTTP Request Processing:

(process_request_internal)

Callback legend:

A AAA Process All callbacks until the result is neither "DECLINED" nor "OK"

S SSS Process callbacks until the result differs from "DECLINED" (which results in one Single successful callback)

See also procedure `run_method()`

modify Request-URI:
`ap_unescape_URL();`
`ap_getparents()`

Read Configuration:
`location_walk()`

S `ap_translate_name`

This routine gives our module an opportunity to translate the URI into an actual filename. If we don't do anything special, the server's default rules (Alias directives and the like) will continue to be followed. The return value is OK, DECLINED, or HTTP_mumble. If we return OK, no further modules are called for this phase.

(`r->proxyreq == NOT_PROXY`)
`&& (r->method_number == M_TRACE)`

Read Configuration:
`directory_walk()`
`file_walk()`
`location_walk()`

A `ap_header_parse`

This routine is called to give the module a chance to look at the request headers and take any appropriate specific actions early in the processing sequence. The return value is OK, DECLINED, or HTTP_mumble. If we return OK, any remaining modules with handlers for this phase will still be called.

A `ap_check_access`

This routine is called to check for any module-specific restrictions placed upon the requested resource. (See the `mod_access` module for an example.) The return value is OK, DECLINED, or HTTP_mumble. All modules with an handler for this phase are called regardless of whether their predecessors return OK or DECLINED. The first one to return any other status, however, will abort the sequence (and the request) as usual.

Authorization check

check_access(r) returns	some auth required(r)	auth_type(r)	satisfies(r) =	
			ALL	ANY
OK	T	NULL	CHK	CHK
	F	NULL	NOP	<code>d_d(ISE, "p.a. AT n sl", r)</code> NOP
E# x	T	NULL	<code>d_d(E#, "chk_acs", r)</code>	CHK
	F	NULL	<code>d_d(E#, "chk_acs", r)</code>	<code>d_d(ISE, "p.a. AT n sl", r)</code> <code>d_d(E#, "chk_acs", r)</code>

S `ap_check_userid`

This routine is called to check the authentication information sent with the request (such as looking up the user in a database and verifying that the [encrypted] password sent matches the one in the database). The return value is OK, DECLINED, or some HTTP_mumble error (typically HTTP_UNAUTHORIZED). If we return OK, no other modules are given a chance at the request during this phase.

S `ap_check_auth`

This routine is called to check to see if the resource being requested requires authorisation. The return value is OK, DECLINED, or HTTP_mumble. If we return OK, no other modules are called during this phase. If "all" modules return DECLINED, the request is aborted with a server error.

(`check result: OK`)
`&& (auth_type(r) != NULL)`

(`check result: OK`)
`&& (auth_type(r) != NULL)`

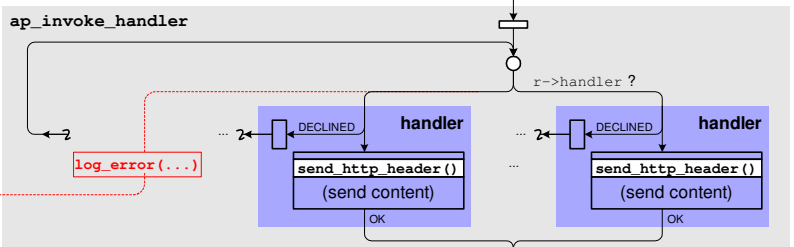
`r->proxyreq != NOT_PROXY`
`&& r->parsed_uri.scheme != NULL`
`&& strcmp(r->parsed_uri.scheme, "http") == 0`

S `ap_find_types (type_checker)`

This routine is called to determine and/or set the various document type information bits, like Content-type (via `r->content_type`), language, et cetera. The return value is OK, DECLINED, or HTTP_mumble. If we return OK, no further modules are given a chance at the request for this phase.

A `ap_run_fixups`

This routine is called to perform any module-specific fixing of header fields, et cetera. It is invoked just before any content-handler.



Now we declare our content handlers, which are invoked when the server encounters a document which our module is supposed to have a chance to see. (See `mod_mime`'s `SetHandler` and `AddHandler` directives, and the `mod_info` and `mod_status` examples, for more details.) Since content handlers are dumping data directly into the connexion (using the `r()` routines, such as `rputs()` and `rprintf()`) without intervention by other parts of the server, they need to make sure any accumulated HTTP headers are sent first. This is done by calling `send_http_header()`. Otherwise, no header will be sent at all, and the output sent to the client will actually be HTTP-uncompliant.

ap_send_http_trace

The return value instructs the caller concerning what happened and what to do next:
 OK ("we did our thing")
 DECLINED ("this isn't something with which we want to get involved")
 HTTP_mumble ("an error status should be reported")

ap_finalize_request_protocol

`finalize_request_protocol` is called at completion of sending the response. Its sole purpose is to send the terminating protocol information for any wrappers around the response message body (i.e., transfer encodings). It should have been named `finalize_response`.

ap_die

`status == DONE`

