

EBS HTTP Server Library

Introduction

The EBS HTTP Server Library provides server-side HTTP functionality for embedded platforms. This allows programmers to implement an HTTP server from within embedded applications without the need to understand the details of the HTTP protocol. The library handles all of the details for transacting with HTTP clients. Applications include Web-based devices, cloud computing, implementing REST-ful and other Service Oriented Architectures.

The EBS HTTP Server Library supports the following features:

- Implementation of robust HTTP server
- HTTP 1.1
- Authentication support
- Per-client access lists and blocking
- Keep-alive
- Single or multi-threaded operation

Usage Guidelines

Basic usage is straightforward. First, `HTTP_ServerInit` is called to initialize the server. Then each path on the server needs to be set up. This is done by calling `HTTP_ServerAddPath` providing a user-implemented, `HTTPRequestHandler` callback. This callback is called whenever a client connects to the server and requests the specified path. When the callback is called, the following structures are passed:

`HTTPServerRequestContext` – contains the server context. Mainly for passing to other API calls

`HTTPSession` - contains the state of the connection to the client

`HTTPRequest` – (the most useful structure) contains the information about the HTTP request that was issued by the client

Once the server is set up and the paths are added, you must then call `HTTP_ServerProcessOneRequest` repeatedly so that the server gains execution to handle server requests. In the single-threaded mode, requests are handled on the main thread when `HTTP_ServerProcessOneRequest` is called. For the multi-threaded mode of operation, making this call allows the server to service the worker threads.

To simplify request handing, `HTTP_ServerAddVirtualFile` can be called. With this function you can assign a memory array to be returned when a specific path is requested by a client.

The API call, `HTTP_ServerAddPostHandler` provides simplified HTTP POST processing.

Request Handler API

A set of API's is provided to simplify request processing inside the request callbacks. See the documentation for `HTTP_ServerInitResponse`, `HTTP_ServerSetDefaultHeaders`, `HTTP_ServerSendError`, and `HTTP_ServerReadHeaders`. These functions simplify the processing of HTTP requests made by the client.

Example Code

The best way to understand the interface to the HTTP Server Library is through example. The source distribution contains several example programs. Below is a list of examples provided.

- Simple Server – outlines the steps presented in the Usage Guidelines section above.
- Advanced Server – adds forms processing, virtual file registration etc.

API Reference

Below is a list of API's for the HTTP Server Library.

HTTP_ServerInit

Initialize an HTTP Server instance.

HTTP_ServerDestroy

Destroy an HTTP Managed Server instance.

HTTP_ServerAddAuthRealm

Set up an authentication "realm" – mapping between local paths and users .

HTTP_ServerRemoveAuthRealm

Remove an authentication "realm".

HTTP_ServerSetAllowedClients

Set the ip addresses of clients that are allowed to connect to a given server context.

HTTP_ServerSetBlockedClients

Set the ip addresses of clients that are blocked from accessing a given server context.

HTTP_ServerAddPath

Add a URL and service callback functions to handle when the URL is accessed from a client.

HTTP_ServerRemovePath

Release a URL service callback function that was assigned by HTTP_ServerAddPath.

HTTP_ServerAddVirtualFile

Assign a memory array to be returned when the URL is requested.

HTTP_ServerAddPostHandler

Add a URL and service callback functions to when a client POSTs to the URL.

HTTP_ServerProcessOneRequest

Wait for a client to connect and process one request.

HTTP_ServerStopHelperThreads

Kills all server worker threads. Used in the multithreaded configuration only.

HTTP_ServerGetPort

Returns the port number that the server is listening on.

HTTP_ServerInitResponse

Initializes an HTTP response object.

HTTP_ServerSetDefaultHeaders

Sets the default HTTP header for a given response object.

HTTP_ServerSendError

For a given request context, return an error code to the client.

HTTP_ServerValidateRequest

Not currently implemented.

HTTP_ServerSendAuthChallenge

Not currently implemented.

HTTP_ServerConnectSetKeepAlive

Turns keep-alive on or off for a given server context.

HTTP_ServerConnectionClose

Closes the connection for a given request context..

HTTP_ServerReadHeaders

Extract the HTTP header for the current request.

HTTP_ServerRequestGetLocalAddr

Returns the locally-bound name for the socket pertaining to the given request context.

TBD – detailed API descriptions

