



Seminole Embedded Webserver

Features

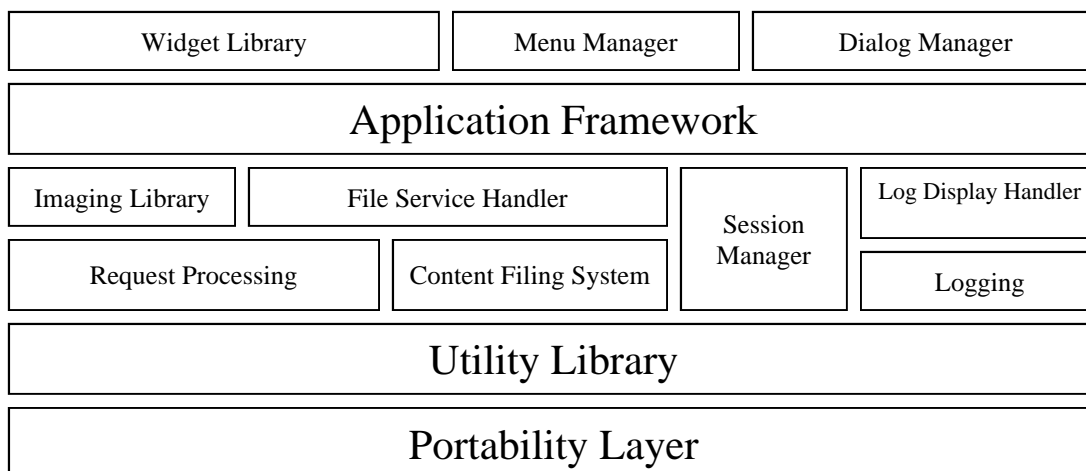
- Compact code size, small heap footprint, high performance.
- Portable code
- Content file system
- Form handling (including support for multipart MIME—file uploads)
- SSL support and digest authentication
- Supports HTTP versions 0.9, 1.0, and 1.1 on IPv4 and IPv6
- Session manager
- Object-oriented, modular design
- Optional event-driven application framework
- Integrated build system for both Windows and UNIX host environments
- Highly modular code base for customized feature sets
- Discovery Service
- Complete source code included

Description

Seminole is an embeddable webserver toolkit designed to be non-invasive and easily retrofitted to existing applications, lightweight with low resource consumption, and highly reliable with proper standards compliance and security safeguards. Written using a subset of C++, Seminole provides a modular, high-level API which simultaneously insulates the client programmer from complicated protocol details while allowing total control over low-level operation when necessary. Common uses for Seminole include:

- Providing web interfaces for embedded devices for monitoring or configuration.
- Allowing distributed method invocation (SOAP, for example).
- Creating context-sensitive help engines which leverage open standards.
- Enabling legacy systems to export data using a modern, commodity protocol.

Software Architecture



GladeSoft, Inc.

1-561-213-6177
<http://gladesoft.com>

Copyright © 2011 Gladesoft Inc.
All rights reserved.

Seminole Webserver



Modular Architecture

Two primary concerns amongst embedded systems developers are memory usage and code size. Seminole is architected as a set of loosely-coupled modules that can be used without requiring the bloat incurred by excluded features. An additional measure to keeping code size small is a set of compile-time options that allow developers to select between various tradeoffs (i.e. space vs. time).

An object-oriented design allows multiple handlers to lay claim to portions of the URL namespace. Seminole includes several pre-written handlers for basic functions (redirects, file serving, etc). Handlers can also be custom written for applications.

These handlers are far more powerful than the "CGI scripts" in other web servers. In true embedded style, it is possible to let all server-side processing execute in the same address space – after all, few embedded operating systems support multiple processes. Of course, Seminole's flexible design does not preclude external scripts should the target environment provide for files, scripting languages, and multiple processes.

Target Platforms

Seminole is designed to be highly portable to various hardware architectures and operating systems. This is accomplished by isolating all of the platform specific code into a single module, called the "portability layer." This layer presents an interface that the remainder of Seminole (the portable code) can use to access features of the base platform.

Several stock implementations of the portability layer are provided for common operating systems:

- POSIX (Solaris, Linux, μ CLinux, BSD, Irix, etc.)
- Win32 / WinCE
- VxWorks
- μ C/OS-II
- eCos
- QNX / Neutrino

New target operating systems can be added easily using the existing portability layers as a base. Full source code in C++ is included. Furthermore, great care was taken to use only the features of C++ that are available in most common implementations and don't conflict with the typical embedded systems development model (so called "Embedded C++").

The interface to the network is also abstracted and supports multiple transports simultaneously (IPv4, IPv6, SSL, serial, custom, etc).



Content File System

Seminole does not require that the target operating system provide a file system. Web content is processed by a set of tools running on the host computer and bound into a special format designed specifically to conserve space yet provide all the necessary metadata that the HTTP protocol requires. Content can be compressed on a file-by-file basis as well as conditionally preprocessed at compile-time to avoid runtime processing overhead for such basic mechanisms as a common look and feel.

Standard HTML generation tools may be used by the host-based tools. The host tools also provide a foundation for internationalization support. Multiple content packages can be distributed and/or included in a single image alongside the Seminole webserver.

Intelligent Template System

Serving dynamic content is key to a good web interface. Seminole supports the easy generation of dynamic content by offering a template system. The template system allows content to be separated from code. Templates are not just done via simple substitution but rather can use loops and conditionals (as well as substitution) to expose the data model of your application. Additionally, unlike many other template systems, the syntax of the template directives is checked by the host tools of the content filing system – not at runtime. This early detection of errors can help make content development more cost effective by preventing errors that are normally only seen in later phases of development. As an added bonus because no syntax checking of templates must be performed at runtime, the Seminole template code is more efficient.

Form Handling

Seminole handles three different types of form parameter decoding: query string, data via the HTTP `POST` method, and multipart-MIME. The latter method allows Seminole to receive files from the HTTP client. This is useful for remotely updating the firmware on embedded devices. The query string method and the `POST` method are not mutually exclusive; they can be used simultaneously to generate two independent parameter lists. In addition, utility routines are provided for (optionally) quickly searching (via hashing) parameter lists.

SSL Support

To enhance security, Seminole provides a framework for using alternative transports to straight TCP. The most common use of this facility is to interface with SSL (Secure Sockets Layer) libraries. Interfaces to `MatrixSSL` from PeerSec Networks (<http://www.peersec.com>) and `OpenSSL` are included.



Session Manager

HTTP is a stateless protocol which is less than ideal for complex web interfaces. Seminole provides a session manager that retains state across multiple HTTP requests securely. Sessions can optionally be expired after an idle period for enhanced security. Session tokens (identifiers on the client representing the unique session) can be stored either in CGI parameters or client-side cookies.

Imaging Library

Seminole provides a graphics engine that is capable of doing charting and graphing. This feature allows devices to report data graphically to enhance usability. The imaging library provides a canvas object and drawing tools. Once drawing is complete the image is sent out as a GIF file automatically by Seminole.

XML Parser

Seminole also includes a “streamy” XML parser for AJAX or XML-RPC style applications. The XML parser provides an optional zero-copy interface for pushing in data from HTTP request bodies. XML documents can be parsed either SAX style or by building a DOM. A simplified XPath mechanism for querying and modifying the DOM is also provided.

Application Framework

Building robust web-based interfaces requires complex logic to handle the problems introduced by the stateless nature of HTTP. The Seminole Application Framework provides a MVC (Model-View-Controller) based engine to insulate applications from the complexity of HTTP and HTML. The application framework supports internationalization, security, and parameter validation combined with an easy-to-use interface. Despite its power and flexibility the application framework is compact and efficient allowing it to be used on low-end microprocessors like those typically found in cost-sensitive networking equipment.

Security

Seminole provides a framework for HTTP authentication. For additional security, HTTP digest authentication can optionally be used. Digest authentication avoids passwords being transmitted in plain text and prevents replay attacks. Digest authentication is a useful alternative when there are insufficient resources for an SSL solution. Digest sessions (logically, managed by the session manager) can also optionally be expired due to inactivity for added security. Seminole also was designed to be resistant to denial-of-service attacks that can consume scarce resources on an embedded system. Resource usage policies are in one spot and easily configured.



Performance

Small code size is only one aspect of Seminole's impact on your system. Despite its small footprint Seminole has a number of optimizations to be a high performance server. The data paths avoid unnecessary memory copies and the architecture is designed for high-performance for dynamic content — the most common use of an embedded webserver. On systems with CPU caches the compact footprint of Seminole avoids cache thrashing.

Seminole includes many options to tune size, performance, and feature set. The code is structured so that unused features are not linked in if they are not used.

Discovery Service

To help locate Seminole instances on a network a UDP (multicast) discovery server and client are included. These packages not only allow automatic discovery of endpoints but also can display endpoint status. Get an overview of an entire network of embedded devices without complex configuration. Instances of Seminole can discover each other on the network and form self-organizing networks of devices. Seminole includes two discovery clients — both with full source code: A portable Java applet and a Win32-based client. Furthermore the Win32-based client uses a generic library that is part of Seminole. This allows embedded devices to locate each other on a network.

Integrated Build System

Seminole provides an easy-to-use, template-driven build system. Although completely optional, the build system allows for easy integration into existing systems with a minimum of fuss. The flexible build system compiles Seminole and handles content building on both UNIX and Windows host platforms. The built components can then be simply integrated into the existing application. For situations where no existing application exists, the build system can be easily extended to compile application-specific code as well as building Seminole. The default build target for Seminole is a set of header files and library files configured for the target that can then be easily linked into your final application.



Documentation

An extensive programmers reference manual details each and every Seminole API. A “Getting Started Guide” is also provided to get programmers up and running with the basic features of Seminole quickly. The internal design and operation of Seminole is documented within the extensively commented source code for advanced understanding. Example code demonstrating the features of Seminole is also included.

HTTP Client

Seminole also includes an HTTP client. The client can be used, for example, to implement a software update mechanism by fetching firmware files from a public website. With the XML parser Seminole can help embedded systems participate in communicating with XML over HTTP to application servers.

WebDAV Server

Seminole can also function as a WebDAV server. This allows an instance of Seminole to be mounted as a file system on most popular operating systems. The storage for the WebDAV server does not even have to be a real file system; any storage mechanism that can be wrapped with a simple API can be exported.

For pricing or more detailed technical information, please contact our sales department at sales@gladesoft.com or call us at 1-561-213-6177.