Smart Solutions for Embedded Success

SMB

Server Message Block

Features

- 100% portable ANSI-C
- Ports easily to any kernel/ network stack
- Ports available for Linux/BSD sockets, Win32/Winsock, RTKernel-32/RTIP
- Small self-contained memory footprint
- Runs in singled-thread, multi-thread, or polled mode
- Uses EBS virtual file system (with modules for ertfs, rtfiles, dos/windows, linux, nfs, mfs)
- Supports share- and user-level security with encrypted passwords
- Supports printing
- 6 Months free email and phone support

SMB (Server Messaging Block) is the network protocol used by all variants of Microsoft Windows to share files and printers over a LAN or WAN. It runs on top of TCP/IP as an application-level protocol. Embedding RT-SMB in your system makes it available as a disk or printer resource to any Windows PC as well as any Linux or Unix variant running Samba. RT-SMB is designed from the bottom up for small real-time systems. It is high-performance, has a small footprint, is robust, and portable.

FOOTPRINT

(Measurements were taken with no debugging information and no optimization.)

With a small build (i.e., able to handle 5 simultaneous connections, 4 helper threads, 1 user per connection, 10 open files per session, and default transaction buffer size), the code segment for SMB takes up 85 Kb. The data segment for SMB takes up 65 Kb.

***Total footprint*: 150 Kb.**

With a larger build (i.e., able to handle 15 simultaneous connections, 10 helper threads, 3 users per connection, 20 open files per session, and default transaction buffer size), the code segment for SMB takes up 85 Kb. The data segment for SMB takes up 262 Kb.

***Total footprint*: 347 Kb.**

PORTING

To port RT-SMB to a new kernel/network stack, you need to define a set of functions for that platform. For an example of this, see smbrtip.h and smbrtip.c. Following is a description of each function. First, some explanation of platform-defined data types:

RTSMB_SEM - *is a platform-specific mutex variable.*
RTSMB_TIME - *is a platform-specific variable used to keep track of elapsed time.*

THREADING

```
BBOOL THREAD_StartThread (PNET_THREAD pThread);
```

This starts a thread that will run until completion then die. If threads are unsupported, just always return FALSE.

SEMAPHORES

```
BBOOL SEM_CreateSem (RTSMB_SEM *sem);
void SEM_ClaimSem (RTSMB_SEM *sem);
void SEM_ReleaseSem (RTSMB_SEM *sem);
```

These all handle semaphores (which are actually used as mutexes) so that RT-SMB can control multiple threads. Again, if threads are unsupported, you can just return FALSE when creating a thread.

TIME

```
RTSMB_TIME TIME_GetTime (void);
BBOOL TIME_IsPast (RTSMB_TIME time);
int TIME_Compare (RTSMB_TIME time1, RTSMB_TIME time2);
RTSMB_TIME TIME_AddMS (RTSMB_TIME time, dword ms);
dword TIME_SubtractInMS (RTSMB_TIME time1, RTSMB_TIME time2);
```

These are standard operations that RT-SMB needs for calculating elapsed time. Most should be trivial to implement, depending on platform support for time. All operations deal with millisecond times, so the system time must be at least that precise.

PRINTING

```
int PRINT_Init (int n);
void PRINT_PrintByte (int n, byte b);
```

If printing support is desired, you must implement a way to initialize the nth printer (nth is defined loosely. Usually, it relates to parallel port n). In addition, the only interaction RT-SMB needs with the printer is the ability to print a byte at a time.

SUPPORTED SMB MESSAGES

Negotiate SessionSetupAndx LogoffAndx TreeConnectAndx TreeConnect TreeDisconnect Echo Seek Flush Close Delete	Rename Move Copy Open OpenAndx Read ReadAndx ReadRaw Write WriteAndx WriteRaw	WriteAndClose WritePrintFile DeleteDirectory CheckDirectory Transaction2FindFirst Transaction2FindNext FindClose2 Transaction2QueryPathInformation Transaction2QueryFileInformation OpenPrintFile ClosePrintFile	Create CreateNew CreateDirectory CreateTemporary ProcessExit QueryInformation QueryInformation2 Search QueryInformationDisk	NTCancel – *It is handled correctly, but has no effect.* SetInformation – *It is handled correctly, but has no effect.* SetInformation2 – *It is handled correctly, but has no effect.*

PARTIALLY SUPPORTED SMB MESSAGES

Transaction2QueryFSInformation – *Our virtual abstraction layer does not yet provide all the needed information*

UNSUPPORTED SMB MESSAGES

NTCreateAndx – *Uses NT-specific arguments* NTTransactCreate LockingAndx – *No form of byte locking is supported* LockAndRead – *No form of byte locking is supported* LockByteRange – *No form of byte locking is supported* UnlockByteRange – *No form of byte locking is supported* WriteAndUnlock – *No form of byte locking is supported* NTTransactNotifyChange Transaction2GetDFSReferral – *DFS is not supported* Transaction2ReportDFSInconsistency NTTransactIOCTL – *Only useful on NT*	NTTransactQuerySecurityDesc – *Only useful on NT* NTTransactSetSecurityDesc – *Only useful on NT* ReadMPX – *This is for connection-less transports, which we don't support* WriteMPX – *This is for connection-less transports, which we don't support* Transaction2Open2 – *No support for extended attributes* Transaction2SetPathInformation – *VFS does not yet support what we need for this* Transaction2SetFileInformation – *VFS does not yet support what we need for this* GetPrintQueue – *Support is being worked on* We are compatible with clients for WinXP, Windows 95, Windows 98, Samba We have ported RT-SMB to Linux and Win32.

API

rtsmb_read_config – Reads a configuration file rtsmb_share_add_tree – Adds a disk share rtsmb_share_add_ipc – Adds a control share (always needed) rtsmb_share_add_printer – Adds a print share rtsmb_share_remove – Disables a share rtsmb_set mode – Sets user mode or share mode rtsmb_get_mode – Finds out what mode RT-SMB is running in rtsmb_register_group – Registers a user group rtsmb_register_user – Registers a user	rtsmb_delete_user – Deletes a user rtsmb_add_user_to_group – Adds a user to a group rtsmb_remove_user_from_group – Removes a user from a group rtsmb_set_group_permission – Sets the access rights of a group rtsmb_init – Initializes RT-SMB rtsmb_pollos_cycle – Handle some requests in non-blocking mode rtsmb_cycle – Handle some requests in blocking mode rtsmb_shutdown – Shutdown RT-SMB

CONFIGURATION PARAMETERS

RTSMB_ENCRYPTION *If you want RT-SMB to handle encrypted passwords sent by the client, enable this.* NUM_THREADS *The maximum number of threads that will be created to handle incoming connections. Set to 0 to disable multithreading.* MAX_SESSIONS *The maximum number of sessions that RT-SMB will handle simultaneously.* MAX_UIDS_PER_SESSION *The maximum number of users logged on for each session. This is usually low.* MAX_FIDS_PER_SESSION *The maximum number of open files that a session can maintain.* MAX_FIDS_PER_TREE *The maximum number of open files that one share can maintain.* MAX_FIDS_PER_UID *The maximum number of open files that one user can maintain.*	MAX_SEARCHES_PER_UID *The maximum number of searches a user can maintain simultaneously.* MAX_TREES_PER_SESSION *The maximum number of shares a session can access simultaneously.* MAX_SHARES *The maximum number of shares that can be defined.* MAX_GROUPS MAX_USERS *The maximum number of groups and users.* SMALL_BUFFER_SIZE *The size of the normal SMB buffer. Used for everyday transactions.* BIG_BUFFER_SIZE *The size of the large SMB buffer. Used for raw reads and writes.* NUM_BIG_BUFFERS *The number of big buffers to keep around. This limits the number of simultaneous raw reads and writes. Set to 0 to disable raw reading and writing.*

EBSnet, Inc. • 39 Court Street • Groton, MA 01450
Voice **978.448.9340** • Fax **978.448.6376** • **800.428.9340** (U.S. Only)
sales@ebsnetinc.com

ebsnetinc.com

ebsnet