GSM Protocol Stack

SHM-NT (Shared Mem Gadget for WinNT/Win2000)

API-Description

Author: Condat AG
Alt Moabit 91d
10559 Berlin
Germany

Date: May 24, 2002
ID: xxxx.0x0.00.00x
Status: being processed

0 Document Control

Condat AG
Alt Moabit 91d
10559 Berlin
Germany

Telephone: +49.30.39094-0
Fax: +49.30.39094-300
Internet: www.condat.de
E-mail: gsm@condat.de

0.1 Document History

ID	Author	Date	Status	Remarks
8415.090.00.001	RME	May. 24, 02	Being Processed	Initial

0.2 References

[] No references up to now.

0.3 Abbreviations

SHM	Shared Memory
MMU	Memory Management Unit. Special Processing unit, usually part of a CPU
G23 Target System	Hardware which executes G23
PC	Personal Computer
ACI	Application Control Interface (AT Commands)
G23	The Condat implementation of Layers 2 and 3 of the GSM Protocol Stack
G23 Target System	Hardware which executes G23
MM	Mobility Management
MMI	Man Machine Interface
Target System	Shortened form of 'G23 Target System'

0.4 Terms

Shared Memory	A chunk of physical memory, mapped by OS into several user address spaces. Mapping usually done by address translation, which is performed by a MMU.
Chunk	A portion of Memory.

1 Introduction

G23 is a software package implementing Layers 2 and 3 of the ETSI-defined GSM air interface signaling protocol, and as such represents the part of a GSM mobile station's protocol software which is both, platform and manufacturer independent. Therefore, G23 can be viewed as a building block providing standardized functionality through generic interfaces for easy integration.

The G23 suite of products consists of the following items:

- Layers 2 and 3 for speech & short message services,
- Layers 2 and 3 for fax & data services,
- Application Control Interface/AT Command Interface,
- MMI and MMI Framework (MFW) and
- Test and integration support tools.

This document describes the interface (API) of the SHM-NT Gadget for WinNT/Win2000. SHM-NT is introduced to overcome some performance issues using the Windows shared memory approach using "memory mapped files" and to get rid of the resulting pointer addressing constraints in behalf of shared memory address map mismatch. SHM-NT is based on the undocumented Zw*() Windows system calls of the native Win32 API and therefore will only be useable with WinNT & Win2000 (WinXP expected), but not with Win Millenium/98/95/3.x.

2 General Application Manual

2.1 Environment / Installation

Currently, the SHM-NT DLL is build using a MS-DeveloperStudio Project File: \gpf\shm-nt\msdev\shm_nt.dsw. It can be used to build \gpf\bin\debug\shm_nt.dll and \gpf\bin\debug\shm_list.exe, as well.

2.2 Getting started

An involved SHM-NT client needs to include "shm_nt.h" from \gpf\shm_nt\inc. Instrument your source code with the appropriate SHM_NT call described in this document. After compilation, simply link \gpf\bin\debug\shm_nt.lib to your executable.

2.3 Command line parameters for shm_list.exe

The SHM-NT dynamic link library is accompanied by a small utilty named shm_list.exe, residing in \gpf\shm-t\bin. "shm_list.exe" lists some internal data structures of the shared memory gadget and is intended to support development and debugging of the gadget itself and/or clients using SHM-NT. "shm_list.exe" supports some flags to specify its behaviour:

shm_list -s name [addr]

create/map a section "name" and try to allocate a chunk of memory.If an optional addr is given, an internal heap state dump is performed, followed by a memory dump starting at given addr.

shm_list -t name

performs a check with iterated alloc/frees on this section

shm_list -l name ...

performs an internal heap state dump on the given section names

shm_list -L

gives a complete list of all sections created by all SHM-NT clients including names, addresses, sizes and ranges

3 The API

int shm_section(char * name, unsigned long int rsize, unsigned long int * phdl);
creates a "raw" *shared memory section*, to be managed by user. The section is a bare memory area without any operations on it. Though, there is no classic alloc/free available. If likewise is desired, shm_heap() should be used instead.

int shm_heap(char * name, unsigned long int rsize, unsigned long int * phdl, BOOL forcelnit);
creates a *shared memory heap* providing classic malloc/free. This call is internally based on shm_section(). So, a shared heap is always a (extension of a) shared section, but not vice versa.

int shm_delete_section(unsigned long int hdl);
deletes a shared memory section or a shared heap.

void * shm_alloc(unsigned long int hdl, unsigned long int size);
allocates a chunk in the denoted shared heap.

int shm_free(void * addr);
releases a chunk in the denoted shared heap.

int shm_exit();
unmaps all sections from a calling client. It does not delete any section irrespective the internal sections. These are deleted, if they are the solely remainder (no other SHM-NT sections/heaps existing). This behaviour may change in the near Future.

int shm_map(char * name, unsigned long int rsize, unsigned long int * paddr);
map a single, specific shared memory section/heap into the caller's address space.

void shm_map_all();
map ALL current existing (created remotely in the meantime), locally unmapped shared memory sections/heaps into the caller's address space.

long int shm_map_by_exeption(EXCEPTION_POINTERS* EP);
map ALL current existing (created remotely in the meantime), locally unmapped shared memory sections/heaps into the caller's address space by "*trap on use.* Usage of this call implies exploitation of the Windows try/except scheme. A user who wants to use this function has to instrument the affected code. An Example follows:

```
int q_read(
        __try
        {
         [...main q_read code...]
        }
        __except (shm_map_by_exeption(GetExceptionInformation()))
)
```

void shm_list_pools();
A service/debug function. Prints information (stdout) about all existing SHM-NT shared memory address sections/heaps, each qualified with it's name, address, size and address range.

void shm_list_heap(FILE * outf, char * name);

A service/debug function. Lists internal management data of a shared memory heap to file.

Used in Program shm_list.exe, not really necessary for implementation.

4 Known problems and future tasks

This paragraph is meant to show which bugs are already found (but not removed yet) and to provide an impression of future plans concerning this product.

4.1 Known bugs

4.2 „Soon implemented“

4.3 „Nice to have“