Subject: VoiceSaver ® Mailbox Provisioning Application Programmer's Interface (API)

Parwan Electronics Corporation
January 14, 2003
By: Ovidiu Dascalu
(732)-290-1900, ovidiu@voicesaver.com
No. 2003-0045

TECHNICAL MEMORANDUM

Overview

This API gives mailbox handling control to a third party application.
Such an application gets connected through a TCP/IP socket to the VoiceSaver and requests a read/write transaction. The web administration feature of VoiceSaver is this way extended to include the API.

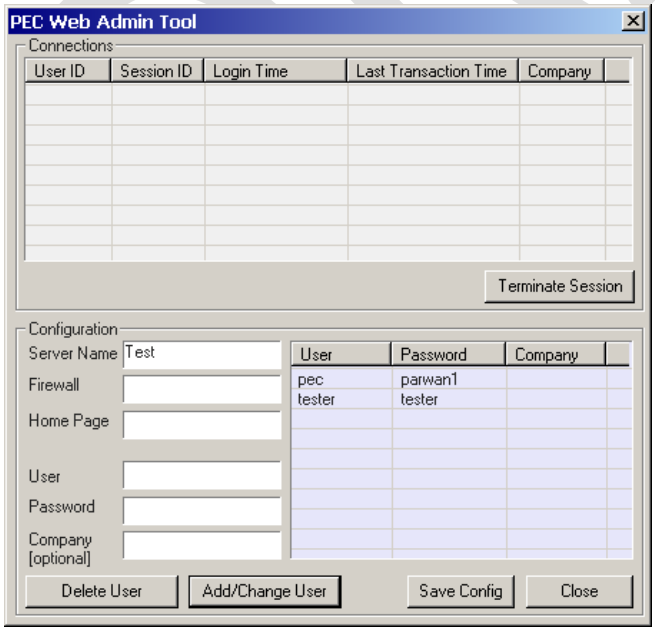
The client application opens a socket connection to VoiceSaver, sends a request (read/write), gets a response (a mailbox structure or a confirmation for read) and disconnects. A username and password should be provided for each transaction.

The TCP/IP Connection

The VoiceSaver is listening on port 5000 (eee58) for incoming connections. Supposing that VoiceSaver machine has an IP: 192.168.0.30, the client should open a TCP/IP connection on 192.168.0.30 on port 5000, or the port designated by parameter eee58.

The authentication

The authorized users are managed through the Web Admin Tool.
(VoiceSaver View Menu)



The READ Transaction

Description:

Requests the mailbox structure for a specific mailbox.

The client needs to send the following data:

Transaction Type:		READ	(16 bytes) // 0 terminated string
Packet Length	:	integer	(4 bytes) // 16 + 4 + 16 + 16 + 16 + 4
User Name	:	<valid user name>	(16 bytes) // 0 terminated string
User Password	:	<valid user password>	(16 bytes) // 0 terminated string
Mailbox	:	<valid mailbox no>	(16 bytes) // 0 terminated string
Record	:	<valid record no>	(4 bytes) // read the box from this record

The client should wait for the following response if no error occurs:

Transaction type:		DATA	(16 bytes) // 0 terminated string
Packet Length	:	integer	(4 bytes) // 16 + 4 + 16 + 4 + // + sizeof (Mailbox Structure)
Mailbox	:	<mailbox no requested>	(16 bytes) // 0 terminated string
Record	:	integer	(4 bytes) // the requested box record
Data	:	<Mailbox Structure>	(sizeof (Mailbox Structure) bytes)

In case of an error or failed authorization the response will be:

Transaction type:		ERROR	(16 bytes) // 0 terminated string
Packet Length	:	integer	(4 bytes) // 16 + 4 + 256
Error Message	:	<error description>	(256 bytes) // 0 terminated string

Possible errors:

- Malformed packet
- Invalid username or password
- Invalid mailbox
- Invalid record number
- System error

Notes:

- The bad(malformed) packets will be discarded on the server side (VoiceSaver)
- When the mailbox field is NULL (empty) the record determines which box to be read
- To ignore the record set this value to -1
- When both mailbox and record are specified the record is ignored
- At least one of mailbox and record must be valid

The WRITE Transaction

Description:

Requests the update of a mailbox structure or the addition of a mailbox structure If the mailbox does not exist it a new one is added, otherwise an update transaction is performed.

The client needs to send the following data:

Transaction Type:		WRITE	(16 bytes) // 0 terminated string
Packet Length	:	integer	(4 bytes) // 16 + 4 + 16 + 16 + 16 + 4 // + sizeof (Mailbox Structure)
User Name	:	<valid user name>	(16 bytes) // 0 terminated string
User Password	:	<valid user password>	(16 bytes) // 0 terminated string
Mailbox	:	<valid mailbox>	(16 bytes) // 0 terminated string
Record	:	<valid record no>	(4 bytes) // write this record
Data	:	<Mailbox Structure>	(sizeof (Mailbox Structure) bytes)

To Add a new record set the record number to -1. For an updated set the record to the corresponding box record number. To get the valid record number for an existing box the client should performed a READ transaction first.

The client should wait for the following response if no error occurs:

Transaction type:		UPDATED	(16 bytes) // 0 terminated string
Packet Length	:	integer	(4 bytes) // 16 + 4 + 16 + 4
Mailbox	:	<the updated mailbox>	(16 bytes) // 0 terminated string
Record	:	integer	(4 bytes) // the requested box record

Or

Transaction type:		ADDED	(16 bytes) // 0 terminated string
Packet Length	:	integer	(4 bytes) // 16 + 4 + 16 + 4
Mailbox	:	<the new mailbox>	(16 bytes) // 0 terminated string
Record	:	integer	(4 bytes) // the requested box record

In case of an error or failed authorization the response will be:

Transaction type:		ERROR	(16 bytes) // 0 terminated string
Packet Length	:	integer	(4 bytes) // 16 + 4 + 256
Error Message	:	<error description>	(256 bytes) // 0 terminated string

Possible errors:

- Malformed packet
- Invalid username or password
- Invalid mailbox
- Invalid record number

- Invalid data (mailbox specific data check)
- System error

Notes:

- The bad(malformed) packets will be discarded on the server side (VoiceSaver)
- The mailbox fields must be valid according to the VoiceSaver Box Specifications
- The synchronization id should match and the key field should be the same as the box number.

The Mailbox Structure

```
/*=============================================*/
/* STRUCTURE FOR DATA RECORDS                  */
/*=============================================*/

#pragma pack(1)


#define BOX_SIZ  13
#define PWD_SIZ  15
#define EXT_SIZ 37

typedef struct {
      char key[14];           // Record Key Field, Generaly as boxid */
      char boxtype[4];        // Message Box Type */
      char boxno[BOX_SIZ+1];  // Message box number */
      char supbox[BOX_SIZ+1]; // Supervisor Box Number */
      char boxpwd[PWD_SIZ+1]; // Password */
      char extens[EXT_SIZ];   // PBX Extension */
      char extens01[EXT_SIZ]; // PBX Extension */
      char extens02[EXT_SIZ]; // PBX Extension */
      char extens03[EXT_SIZ]; // PBX Extension */
      char extens04[EXT_SIZ]; // PBX Extension */
      char extens05[EXT_SIZ]; // PBX Extension */
      char extens06[EXT_SIZ]; // PBX Extension */
      char extens07[EXT_SIZ]; // PBX Extension */
      char extens08[EXT_SIZ]; // PBX Extension */
      char extens09[EXT_SIZ]; // PBX Extension */

      char boxmsgs[5];        // Maximum number of messages */
      char boxsecs[5];        // Maximum message size in seconds */
      char boxenable[4];      // Enable=1 Disable=0 */
      char elevel[4];         // Experience Level 1 to 9 */
      char boxclass[4];       // Box Class of Service
      char drive[4];          // Drive Code 1=A 2=B 3=C 4=D 5=E ... */
      char ndays[4];          // Keep Messages for So Many Days */
      char ncallers[4];       // Number of Caller can Leave Messages */
      char sync[5];           // Synchronize String for Checking */
      char pask[4];           // 1=Page 2=AskAndPage 3=NoPage */
      char boxbeep[56];       // Beeper 1 Number to Dial */
      char beepcnt[3];        // How Many Times Beeped So far */
      char beepmax[4];        // Beeper Maximum Try Count */
      char beepmsgs[4];       // Beep after So Many Messages */
```

```
      char beeptype[4];       // Beeper Type */
      char boxbeep2[40];      // Beeper 2 Number to Dial */
      char loff[40];          // Message Waiting Light Off String */
      char beepcnt2[4];       // How Many Times Beeped So far */
      char beepmax2[4];       // Beeper Maximum Try Count */
      char beep2type[4];      // Beeper Type */
      char beepmsgs2[5];      // Beep after So Many Messages */
      char zBeepInterval[5];  // BEEP INTERVAL IN MINUTES - 0=FOR EVERY MESSAGE
      char wakebeep[BOX_SIZ+1];// Escalation Mail box
      char company[10];        // Company or Group
      char zBeeperStartHour[4];// START BEEPING AFTER THIS HOUR IN MILITARY
      char zBeeperStopHour[4]; // STOP BEEPING AFTER THIS HOUR in MILTARY
unsigned long incalls;         // Number of Inbound Calls */
unsigned long outcalls;        // Outbound Calls Made */
unsigned long cbeeps;          // Connected Beeps */
unsigned long intimeuse;       // In  time Use in 6 Second units */
unsigned long outtimeuse;      // Out time Use in 6 Second Units  */
      long totmsgs;           // Number of Total Messages */
      char nbr[4];            // Number of Rings for Call Transfer  */
      char dnd[4];            // Do Not Disturb */
      char wake[8];           // Wake Up Time */
      char wdays[10];         // Wake Up Days to Consider */
      char wrings[4];         // Wake Up Rings before Quiting */
      char ntime[10];         // Notification Time */
unsigned long nNewMsgs;       // Total new messages
unsigned char nflag[4];       // New Message Count */
      char promptset[4];      // Prompt Set To Use */
      char boxname[50];       // Box Identifcation Name */
      char screen[4];         // Call Screening On or Off */
unsigned long tBmsg;          // Subscriber Broadcast play once time remember,
      char Beeperdnd[2];      // Disable Beeper Flag, Mostly used by
      char zEmailAddress[64]; // SUBSCRIBER'S EMAIL ADDRESS
      char cEmailCode[2];     // EMAIL ACTIVATION CODE, 1=ACTIVATE, 0=DEACTIVATE
      char zDomainName[14];   // MAIL BOX DOMAIN LOCATION FOR NETWORKING
      char zForwardMbox[BOX_SIZ+1];// MSG FORWARD MAIL BOX
      char zSamplingRate[4];  // MAIL BOX VOICE SAMPLING RATE IN KILO
BITS PER SECOND, E.G. 024,  032, 064
      char zClassOptions[20]; // EACH CHARACTER WILL MEAN ONE OPTION, C=CALLER ID.
F=FOLLOW ME
unsigned long nBalanceDue;  // BALANCE DUE AMOUNT IN CENTS FOR THE MAIL BOX,
      char zIntLength[6];     // Introduction Length
      char RFU[80];           // Reserved For Future */
}rec_t;
```