

## RMB Windows Extender Reference Manual

### Contents:

<b>1</b>	<b>INTRODUCTION</b> .....	<b>2</b>
<b>2</b>	<b>ARCHITECTURE</b> .....	<b>2</b>
2.1	MESSAGING SERVICE .....	3
2.1.1	<i>Message Format</i> .....	3
<b>3</b>	<b>CSUB SERVICES</b> .....	<b>4</b>
3.1	MESSAGING SERVICE .....	4
3.1.1	<i>Introduction</i> .....	4
3.1.2	<i>functions list</i> .....	4
3.1.3	<i>sample code</i> .....	5
3.2	CONFIGURATION SERVICE .....	6
3.2.1	<i>introduction</i> .....	6
3.2.2	<i>functions list</i> .....	6
3.2.3	<i>sample code</i> .....	6
3.2.4	<i>potential problems</i> .....	6
3.3	LOGGING SERVICE .....	6
3.4	ERROR VALUES.....	7
<b>4</b>	<b>ACTIVEX COMPONENT</b> .....	<b>8</b>
4.1	PROPERTIES .....	8
4.2	METHODS .....	8
4.3	EVENTS .....	8
4.4	METHODS RESERVED FOR TEST USAGE .....	8
4.5	SPY DIALOGUE.....	8

## 1 Introduction

This document describes RMB Windows Extender architecture and interfaces.

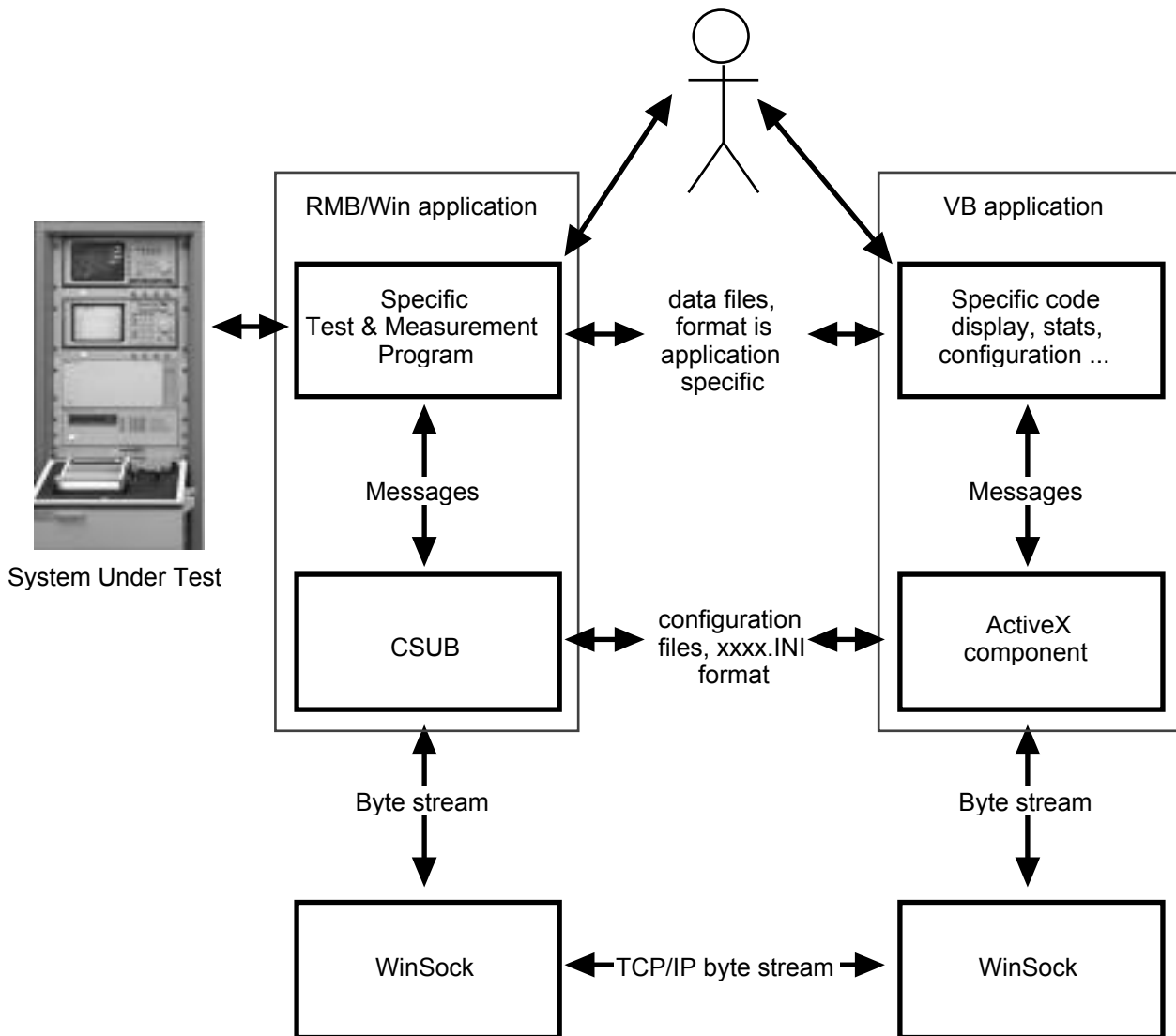
RMB Windows Extender offers the following services:

- message transfer between a RMB application and an ActiveX container application
- configuration file access for RMB applications

RMB Windows's extender components are:

- a CSUB is named « Rmbwext.csb » or « Rmbwext63.csb » and the associated file « Rmbwext.dll »
- an ActiveX component « RmbWinExtender.ocx ». It can be used from any valid ActiveX container application.

## 2 architecture



## 2.1 Messaging service

The communication media is TCP/IP. The components use the WINSOCK standard API, in stream mode. TCP is byte oriented, it's RMBWinExtender role to manage the messages. The components (CSUB, ActiveX) implement that function by adding/removing additional control bytes.

The version 1 of RMBWinExtender uses only characters oriented Data (ANSI), in order to offer simple meaningful spying facilities. The RMB application is a server. When it is ready it listens on a TCP port. Only the connection establishment is asymmetric. After that the communication is full-duplex, and the two applications can release the connection.

The applications send and receive messages made of:

- a type : integer value from 1 to 19999
- additional data : from 0 to 19999 ANSI characters

### 2.1.1 Message Format

Each message has a header, containing the following information:

- a magic string
- the message type
- the total message length, header included

The type and length are short integers that are transferred as 4 hexadecimal ANSI characters [0-9A-F], MSB first

decimal	hexadecimal	characters transmitted
1	0x0001	0001
15	0x000f	000F
61473	0xf021	F021

The complete message is:

	size in bytes	encoding	comments
magic string	4	« ALBK »	4 ASCII characters « A » is first, « K » is last
total message length, header included	4	4 hexadecimal digits [0-9A-F] encoding an unsigned 16 bits integer	maximum 16384 (0x4000) bytes
message type	4	4 hexadecimal digits [0-9A-F] encoding an unsigned 16 bits integer	application messages can use values between 1 to 19999 included, all other values are reserved
message data	variable, = total length - 12	ANSI characters	

## 3 CSUB services

### 3.1 Messaging service

#### 3.1.1 Introduction

RMB Server application must call:

- « Tcplisten », after that call the CSUB listens for TCP connection requests. If client or network releases that connection, a new connection can be established even if RMB application does not call Tcplisten again. The only reason to call Tcplisten again is after a call to Tcpstop, or to change socket port number.
- as many times as it likes :
  - « Tcpgetstate », to get connection status and number of pending messages in reception FIFO queue
  - « Tcpsend », to send a message
  - « Tcpreceive », to read and remove the first pending message in the reception queue
- « Tcpstop », to release current connection, and to stop server role. (Connection requests will no longer be accepted).
- if « Tcpgetstate » function is not called regularly, (two seconds period by example), CSUB will not react to Windows messages that signals events on the socket (connection, deconnexion, incoming message). **The application must call regularly Tcpgetstate.**

#### 3.1.2 functions list

name	role
Tcplisten	CSUB listens on specified port number and accepts connection requests
Tcpsend	send a message
Tcpstop	CSUB stops listening on socket, release current connection
Tcpreceive	reads and removes first message in reception FIFO queue
Tcpgetstate	get connection status and number of pending messages in reception FIFO queue
Tcpsenderror	send a badly formatted message (used only by Bourbaky Validation programs)

Tcplisten(INTEGER port, INTEGER senderrmode)

- parameters are not modified
- after that call, CSUB listens on specified port
- when a connection request is received :
  - it is always accepted
  - if there was already a communication link established, the new connection is immediately released. The first connection remains open.
- after a release by client or network, RMB server application does not need to call Tcplisten again
- senderrmode specifies what to do if RMB application tries to send a message when it can't be sent (connection is closed) see below for detailed information.
- if port number is in use, returns error CrPortAddrInUse (5006)

Tcpsend(INTEGER Typmessage, Message\$)

- parameters are not modified
- sends the specified message
- if message can't be sent, (system buffers full, closed connection)
  - next call to Tcpgetstate will give error number
  - returned value depends of « Tcplistentre senderrmode » parameter value :
    - if senderrmode = 0 , returned value is always 0
    - if senderrmode = 1 , returned value is error number. Application must provide « ON ERROR » code.

Tcpstop

- release current connection if any
- CSUB stops listening on socket, connection requests are ignored

Tcpreceive(INTEGER Typmessage, Message\$)

- reads first pending message and removes it from FIFO queue
- after return, parameters have been modified :

- Typemessage = 0, if there was no pending message
  - Message\$ = Empty String
- Typemessage <> 0, type of received message
  - Message\$ = additional message data

Tcpgetstate(INTEGER Active, INTEGER Msgrcvpendingbb, INTEGER Senderrcr)

- get connection status and number of pending messages in reception FIFO queue
- after return, parameters have been modified:
  - Active = 0 (no connection) or 1 (connection established)
  - Msgrcvpendingbb = number of pending messages in reception FIFO queue, at disposition of application
  - Senderrcr = 0 or error number if there was a problem during last send

Tcpsenderror(INTEGER Typerror, Errorstring\$)

- sends a badly formatted message
- used to test ActiveX reactions by Bourbaky validation programs
- the only valid reason to call that from a user application is to test that Client application contains correct error code

### 3.1.3 sample code

```

80  Tcplisten(1800,0)    ! begins Server role on port 1800
90  !===== loop for a while
100 INTEGER N,Msgnbread,Msgpendingnb,Socketopened, Typmsg
101 DIM Pumpmsg$(256), Msg$(256)
110 Msgnbread=0
120 FOR N=1 TO 60
130  WAIT 1            ! wait 1 second to simulate real code
150  Tcpgetstate(Socketopened,Msgpendingnb)
160  ! prepare a message giving information about application status
170  Pumpmsg$="pumping time left="&VAL$(60-N)&" sockOpen="&VAL$(Socketopened)&
      ", Msgnbread="&VAL$(Msgnbread)

180  DISP Pumpmsg$
190  !== while there are pending messages
200  WHILE (Msgpendingnb>0)
220    Tcpreceive(Typmsg,Msg$)    ! read first pending message
230    Msgnbread=Msgnbread+1
240    PRINT "received msg, typ=";Typmsg;" data=<<<"Msg$;">>>"
250    Tcpsend(33,"received msg number "&VAL$(Msgnbread))    ! send ack message
251    Msgpendingnb=Msgpendingnb-1
252  END WHILE
253  !=====
261  Tcpsend(289,Pumpmsg$) ! message type = 289, as senderrmode = 0, message is just destroyed if connection is closed
280  NEXT N

.....
530  Tcpstop    !===== close connection, stops Server role

```

## 3.2 Configuration Service

### 3.2.1 introduction

RMB Win Extender offers some simple functions that simplify access to a configuration file.

That file must be formatted as a Windows « .INI » file.

The default configuration file is « RMBWinExtender.ini » in the Windows directory

### 3.2.2 functions list

name	parameters	role	comment
Cnffileis	Filename\$	declare the configuration file name « Filename\$ »	can be relative or absolute path
Cnfreastring	Sectionname\$, Paramname\$, Paramvalue\$, Defaultvalue\$	read parameter value	
Cnfwriting	Sectionname\$, Paramname\$, Paramvalue\$	write a parameter value	

### 3.2.3 sample code

```

90 !==== select a specific .INI file
100 Cnffileis("E:\dvw\bky\hpbasic\bkycnf\bkycnf.ini")
110 !==== read last run Date and Time
120 DIM Ts$(64)
130 DIM Tskey$(64)
140 Tskey$="Lastrun"
150 Cnfreastring(Tskey$,"date",Ts$,"unknown" )
160 PRINT "Date lastrun = <<"&Ts$&">>"
170 Cnfreastring(Tskey$,"time",Ts$," unknown ",)
180 PRINT "Time lastrun = <<"&Ts$&">>"
190 !==== write current Date and Time
191 Cnfwriting(Tskey$,"date",DATES$(TIMEDATE))
200 Cnfwriting(Tskey$,"time",TIMES$(TIMEDATE))

```

### 3.2.4 potential problems

The implementation uses Windows standard configuration API.

This API uses a data cache, so if you modify the .INI file with a standard text editor and that your application is already active, there can be a delay of some seconds before your application can read the new values.

## 3.3 Logging Service

RMBWinExtender CSUB uses log files to trace what is happening on RMB's application side.

RMBWinExtender logs in this file :

- important function calls (Tcplisten, Tcpstop)
- connection state changes (Connection, Release)
- messages of type less than or equal to user-defined values
- detailed error descriptions

The logging is done in a list of files that implements a kind of circular buffer. The log files are :

- created in the application current directory

- named « RMBWinExtenderLogN.txt », where N can be 0 to 9
- when the current file size is FileMaxSize (64 Kilobytes default value)
  - the logging service increments current file index (modulo N)
  - removes next file
  - begins writing in it
- when RMBWinExtender starts, it begins by using the most recently modified in the list

The logging service is configured with default values, which can be modified by specifying other values in the configuration file.

Those values are read when the user application calls « Cnffileis »

The logging service is configured by the following values of the « LOGSERVICE » section :

name	default value	valid range	
FileMaxIndex	1	1 to 9	maximum index value in the circular file list
FileMaxSize	0x10000	0x1000 to 0x1000000	
LogInfos	1	0 and 1	if value is 0, connection state changes are not logged
LogSendTypMax	0	0 to 19999	log sent message if type <= value
LogRcvTypMax	0	0 to 19999	log received message if type <= value
LogDirectory	« . » application starting directory	any valid directory path	example : C:\Temp\dir

#### sample configuration file :

logging is done in 4 files (indexes 0 to 3), whose size are 4 kilobytes,

```
[LOGSERVICE]
FileMaxIndex=4
FileMaxSize=0x1000
LogRcvTypMax=800
LogSendTypMax=500
```

### 3.4 Error Values

All error is logged in the files created by the logging service.

The value returned by RMBWinExtender CSUB functions can be :

value	Symbolic name	can be returned by	meaning
0	CrOk	any function	everything is fine
5000	CrException	any function	an exception was thrown and caught, error detailed description was written in RMBWinExtender log file
5001	CrNbParam	any function	incorrect parameters number for this function
5003	CrNotListening	Tepsend, Tcpreceive, Tepgetstate	the function () must be called only after a call to TcpListen, and before a call to Tcpstop
5004	CrNotConnected	Tepsend, Tcpreceive	cannot send or receive as socket is not connected
5005	CrSendWouldBlock	Tepsend	too many files in Winsock system buffers, the send would block. The message was discarded
5006	CrPortAddrInUse	Tcplisten	TCP port is already reserved
5011-5019	CrInvalidParameter	any function	invalid function parameter N (1 to 9)

## 4 ActiveX component

### 4.1 properties

name	
HostName As String	hostname of the computer where RMB application is executing
Port As Integer	TCP/IP port where RMB application is listening
LogWindow as Boolean	opens a modeless Form that enables connection spying

### 4.2 methods

name	
Sub Connect()	request TCP/IP connection, using HostName and Port current values. If the OCX accept the call (does not throw a parameter syntax error), an asynchronous event will give the result of the connection request : ConnexionStateChanged if OK, Error if KO.
Sub Disconnect()	closes TCP/IP connection. The OCX does NOT signal a "ConnexionStateChanged event" after that request
Sub SendData(typeMsg As Integer, data As String)	sends a message on TCP/IP connection
Function IsOpened() As Boolean	true if Connect method was called and : Disconnect method was not called since Request was not rejected by TCP/IP stack
Function IsConnected () As Boolean	When true, messages can be sent to and received from the HPBW application

### 4.3 events

ConnectionStateChanged(ByVal connected As Boolean)	signaled when the Connection state change is a consequence of an event external to the client application : connection accept by TCP/IP HPBW server, connection close, ...
MsgReceived(ByVal typeMsg As Integer, data As String)	signaled when a message is received from the HPBW application
Error(ByVal number As Long, description As String)	signaled if there is a communication error

### 4.4 methods reserved for test usage

Sub SendDataWithError(Error As Integer)

This method is used to send a message with a format error, in order to test the CSUB reactions.

### 4.5 spy dialogue

A modeless Form gives a lot of information about what is happening. That Form can be loaded/unloaded by using the « LogWindow » property of the component.

It displays :

- counters : connection numbers, release numbers, messages sent, received, ..
- a trace of the last messages sent/received with a type less or equal to user-specifiable values