BLUETOOTH SWIPE READER
JAVA APPLET
PROGRAMMING REFERENCE MANUAL

PART NUMBER 99875530-1

DECEMBER 2010

MAGTEK®

REGISTERED TO ISO 9001:2008

1710 Apollo Court
Seal Beach, CA 90740
Phone: (562) 546-6400
FAX: (562) 546-6301
Technical Support: (651) 415-6800
www.magtek.com

REVISIONS

Rev Number	Date	Notes
1.01	23 Dec 2010	Initial Release

SOFTWARE LICENSE AGREEMENT

IMPORTANT: YOU SHOULD CAREFULLY READ ALL THE TERMS, CONDITIONS AND RESTRICTIONS OF THIS LICENSE AGREEMENT BEFORE INSTALLING THE SOFTWARE PACKAGE. YOUR INSTALLATION OF THE SOFTWARE PACKAGE PRESUMES YOUR ACCEPTANCE OF THE TERMS, CONDITIONS, AND RESTRICTIONS CONTAINED IN THIS AGREEMENT. IF YOU DO NOT AGREE WITH THESE TERMS, CONDITIONS, AND RESTRICTIONS, PROMPTLY RETURN THE SOFTWARE PACKAGE AND ASSOCIATED DOCUMENTATION TO THE ABOVE ADDRESS, ATTENTION: CUSTOMER SUPPORT.

TERMS, CONDITIONS, AND RESTRICTIONS

MagTek, Incorporated (the "Licensor") owns and has the right to distribute the described software and documentation, collectively referred to as the "Software".

LICENSE: Licensor grants you (the "Licensee") the right to use the Software in conjunction with MagTek products. LICENSEE MAY NOT COPY, MODIFY, OR TRANSFER THE SOFTWARE IN WHOLE OR IN PART EXCEPT AS EXPRESSLY PROVIDED IN THIS AGREEMENT. Licensee may not decompile, disassemble, or in any other manner attempt to reverse engineer the Software. Licensee shall not tamper with, bypass, or alter any security features of the software or attempt to do so.

TRANSFER: Licensee may not transfer the Software or license to the Software to another party without the prior written authorization of the Licensor. If Licensee transfers the Software without authorization, all rights granted under this Agreement are automatically terminated.

COPYRIGHT: The Software is copyrighted. Licensee may not copy the Software except for archival purposes or to load for execution purposes. All other copies of the Software are in violation of this Agreement.

TERM: This Agreement is in effect as long as Licensee continues the use of the Software. The Licensor also reserves the right to terminate this Agreement if Licensee fails to comply with any of the terms, conditions, or restrictions contained herein. Should Licensor terminate this Agreement due to Licensee's failure to comply, Licensee agrees to return the Software to Licensor. Receipt of returned Software by the Licensor shall mark the termination.

LIMITED WARRANTY: Licensor warrants to the Licensee that the disk(s) or other media on which the Software is recorded are free from defects in material or workmanship under normal use.

THE SOFTWARE IS PROVIDED AS IS. LICENSOR MAKES NO OTHER WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Because of the diversity of conditions and PC hardware under which the Software may be used, Licensor does not warrant that the Software will meet Licensee specifications or that the operation of the Software will be uninterrupted or free of errors.

IN NO EVENT WILL LICENSOR BE LIABLE FOR ANY DAMAGES, INCLUDING ANY LOST PROFITS, LOST SAVINGS, OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE, OR INABILITY TO USE, THE SOFTWARE. Licensee's sole remedy in the event of a defect in material or workmanship is expressly limited to replacement of the Software disk(s) if applicable.

GOVERNING LAW: If any provision of this Agreement is found to be unlawful, void, or unenforceable, that provision shall be removed from consideration under this Agreement and will not affect the enforceability of any of the remaining provisions. This Agreement shall be governed by the laws of the State of California and shall inure to the benefit of MagTek, Incorporated, its successors or assigns.

ACKNOWLEDGMENT: LICENSEE ACKNOWLEDGES THAT HE HAS READ THIS AGREEMENT, UNDERSTANDS ALL OF ITS TERMS, CONDITIONS, AND RESTRICTIONS, AND AGREES TO BE BOUND BY THEM. LICENSEE ALSO AGREES THAT THIS AGREEMENT SUPERSEDES ANY AND ALL VERBAL AND WRITTEN COMMUNICATIONS BETWEEN LICENSOR AND LICENSEE OR THEIR ASSIGNS RELATING TO THE SUBJECT MATTER OF THIS AGREEMENT.

QUESTIONS REGARDING THIS AGREEMENT SHOULD BE ADDRESSED IN WRITING TO MAGTEK, INCORPORATED, ATTENTION: CUSTOMER SUPPORT, AT THE ABOVE ADDRESS, OR E-MAILED TO support@magtek.com.

TABLE OF CONTENTS

SECTION 1. FEATURES AND SPECIFICATIONS

INTRODUCTION

This document describes the properties, methods and events provided by the Java Applet component to communicate with MagTek Bluetooth Readers. For additional information, refer to the reader the Bluetooth Reader specific manual.

SECTION 2. METHODS

In general, most methods return an integer of 0 to indicate a successful call. Otherwise, a non-zero integer value is returned to inform that there is an error, and the error message can be retrieved with the "GetLastErrMessage()" method.

AUTOCONNECT METHOD

Returns an integer value.

Syntax
int AutoConnect()

Remarks
The Java Applet uses the default setting and tries to connect to the available ports in the system until it finds the first MagTek Bluetooth MSR reader that it can communicate with. AutoConnect returns a zero (0) if the connection is established, otherwise a non-zero for an unsuccessful auto-connection. Moreover, it is recommended that one should use function named SetReaderType(readerType) prior to AutoConnect () call, where readerType of 1 is MagTek Bluetooth card reader, and readerType of 0 is for a standard RS232 reader type.

Default Serial Port Parameters
Baud Rate: 9600
Parity: N
Data Bits: 8
Stop Bits: 1

Example
```
int SetReaderType(1);          // 1: Bluetooth;  0:  standard RS232 card reader.
int AutoConnect();
```

The AutoConnect returns zero (0) if the connection is established with a MagTek Bluetooth card reader.

CARDDATA METHOD

Returns a string value indicating the whole card-data-swiped.

Syntax
String CardData()

CARDDATAMASKED METHOD

Returns a string value indicating the masked card data.

Syntax
String CardDataMasked()

CARDENCODETYPE METHOD

Returns a string value indicating the card encode type from the reader.

Syntax
String CardEncodeType()

Remarks
Always returning an empty string at this time. (RFU)

CLOSEPORT METHOD

Returns an integer value indicating the status of the ClosePort method.

Syntax
int ClosePort(*String port*)

Remarks
Returns zero (0) if the closes successfully, otherwise a non-zero for error.

The **ClosePort** method syntax has this part:

Part	Data Type	Description
port	String	The port name is a combination of the word "COM" and the port number, e.g. The Bluetooth has port 7, therefore, the portName is COM7. If port name is not provided, the ClosePort() function will close the port that the applet has previously opened.

Example
int stat = **ClosePort**("COM7");
This returns a zero (0) if the port can be close.

DEVICEPORT METHOD

Sets the serial port number.

Syntax
void DevicePort(*int devPortNumber*)

The **DevicePort** method syntax has this part:

Part	Data Type	Description
devPortNumber	int	Serial device port number

DEVICESERIALNUMBER METHOD

Returns a string value indicating the device serial number from the reader.

Syntax
String DeviceSerialNumber()

DEVICESETTINGS METHOD

Returns an integer value indicating the status of the setting.

Syntax
int DeviceSettings(String *settings*)

Remarks
Sets the basic serial port parameter. The DeviceSettings returns a zero (0) if the setting is OK, otherwise a non-zero for error.

The **DeviceSettings** method syntax has these parts:

Part	Data Type	Description
settings	String	The settings consist of baud rate, parity, data bits and stop bits.

Example
int stat = DeviceSetting(“9600, N, 8, 1”);
This returns a zero (0) if the setting is OK.

DUKPTKSN METHOD

Returns a string value indicating the key serial number from the reader.

Syntax
String DUKPTKSN()

FINDELEMENT METHOD

Returns particular information after parsing out specific card data.

Syntax
String FindElement(*int whatTrack, String strStart, int offset, String strEnd*)

Remarks
Returns particular information after parsing out specific card data based on parameters selected by the user.

The **FindElement** method has these parts:

Part	Data Type	Description
whatTrack	int	User input track
strStart	String	User input start
offset	int	User input offset
strEnd	String	User input End

Example
String var = FindElement(2, ";", 0, "=");

FINDELEMENTMASKED METHOD

Returns particular information after parsing out specific card data.

Syntax
String FindElementMasked(*int whatTrack, String strStart, int offset, String strEnd*)

Remarks
Returns particular information after parsing out specific card data based on parameters selected by the user.

The **FindElement** method has these parts:

Part	Data Type	Description
whatTrack	int	User input track
strStart	String	User input start
offset	int	User input offset
strEnd	String	User input End

Example
String var = FindElementMasked(2, ";", 0, "=");

GETCURRENTPORTNAME METHOD

Return the name of the port that's opened.

Syntax
String GetCurrentPortName()

Remarks
Returns the port name, e.g. COM7

GETEXPIRATIONMONTH METHOD

Returns the Expiration Month as a string.

Syntax
String GetExpirationMonth()

GETEXPIRATIONYEAR METHOD

Returns the Expiration Year as a string.

Syntax
String GetExpirationYear ()

GETLASTERRMESSAGE METHOD

Returns the last error message that the applet has detected.

Syntax
String GetLastErrMessage()

Example
byte [] output;
int stat = SendDataSync("00011000", output); // send in the wrong length

if(stat != 0)
  alert("Error Message: " + GetLastErrMessage()); // display the last error message.

output: Error Message: send request error

GETFNAME METHOD

Returns the First Name of the track data.

Syntax
String GetFName()

GETFNAMEMASKED METHOD

Returns the First Name from the masked track data.

Syntax
String GetFNameMasked()

GETLNAME METHOD

Returns the Last Name of the track data.

Syntax
String GetLName()

GETLNAMEMASKED METHOD

Returns the Last Name from the masked track data.

Syntax
String GetLNameMasked()

GETTRACK METHOD

Returns card data from a specific track (1,2,3)

Syntax
String GetTrack(*int iTk)*

The **GetTrack** method syntax has these parts:

Part	Data Type	Description
iTk	int	Track 1,2,3

GETTRACKMASKED METHOD

Returns the masked card data if supported by the reader from a specific track (1,2,3)

Syntax
String GetTrackMasked(*int iTk)*

The **GetTrackMasked** method syntax has these parts:

Part	Data Type	Description
iTk	int	Track 1,2,3

GETPAN METHOD

Returns PAN.

Syntax
String GetPAN()

GETPORTNAMES METHOD

Returns an array of port names that are available for connections.

Syntax
String[] GetPortNames()

GETPORTNAMESSTRING METHOD

Returns a string of port names that are available for connections

Syntax
String GetPortNamesString()

GETPORTOPENSTATUS METHOD

Returns a boolean indicating if the port is open or close.

Syntax
boolean GetPortOpenStatus()

GETREADERTYPE METHOD

Gets the current setting of the reader interface type.

Syntax
int GetReaderType()

Remarks
1 = Bluetooth; 0 = other reader

GETUPDATESPDRIVERSTATUS METHOD

Returns an integer value indicating if the status of the serial port driver installation.

Syntax
int GetUpdateSPDriverStatus()

Remarks
Retrieves the status of the serial port driver installed. If the returning value is zero (0) then the status is OK, otherwise non-zero for error.

MAGNEPRINTDATAHEX METHOD

Returns the MagnePrint data if supported by the reader in 2 Byte Hex format.

Syntax
String MagnePrintDataHexString()

MPRINTDATA METHOD

Returns a byte-array of MagnePrint data.

Syntax
byte[] MPrintDData()

MPRINTLEN METHOD

Returns an integer value indicating the length of MagnePrint data.

Syntax
int MPrintLen()

MPRINTSTATUS METHOD

Returns an integer value indicating the status of MagnePrint data.

Syntax
int MPrintStatust()

OPENPORT METHOD

Returns an integer value indicating the status of the OpenPort method. This OpenPort(port) function provides a short-cut to connect to a Magtek Bluetooth card reader using default settings: baud rate of 9600, parity of 'N', data bits of 8, and stop bits of 1.

Syntax
int OpenPort(*String port*)

Remarks
Returns zero (0) if the port can be open, otherwise a non-zero for error.

The **OpenPort** method syntax has this part:

Part	Data Type	Description
portName	String	Name of a port to open such as COM1, COM7

Example
String portName = "COM7";
int stat = OpenPort(portName);
This returns a zero (0) if the port can be open.

OPENPORT METHOD 2

Returns an integer value indicating the status of the OpenPort method.

Syntax
int OpenPort**(***String sPortname, String sBaudRate, String sDataBits, String sStopBits, String sParityType*)

Remarks
Returns a zero (0) if the port can be open, otherwise a non-zero for error.

The **OpenPort** method has these parts:

Part	Data Type	Description
sPortname	String	The port name is a combination of the word "COM" and the port number, e.g. If the Bluetooth is connected to port 7, then the portName is COM7.
sBaudRate	String	Baud Rate
sDataBits	String	Data Bits
sStopBits	String	Stop Bits
sParityType	String	Parity

Example
int stat = OpenPort("COM7", "9600", "8", "1", "N");
This returns a zero (0) if the port opened with the parameters.

PORTOPEN METHOD

Open or Close the serial port.

Syntax
boolean PortOpen(*boolean open*)

Remarks
This function is for backward compatibility with OCX. The new OpenPort method is also available.

The **PortOpen** method has this part:

Part	Data Type	Description
open	Boolean	open can be true or false

Example
// PortOpen requires three steps, sets the device port, sets the device settings and call PortOpen(true).
int stat = 1;
DevicePort(7); // assume my port is 7
stat = DeviceSettings("9600,N,8,1");
PortOpen(true); // open the port
PortOpen(false); // close the port

READERID METHOD

Returns an empty string . (RFU)

Syntax
String ReaderID()

RXTIMEOUT METHOD

Sets RX timeout. (RFU)

Syntax
void RXTimeout(*int rxtimeout*)

The **RXTimeout** method has this part:

Part	Data Type	Description
Rxtimeout	int	RX Timeout

SENDDATA METHOD

Returns a string value from the reader.

Syntax
String SendData(*String sRequest*)

Remarks
Send specific command to the Bluetooth reader. The length of the sRequest is automatically calculated by the component. If the command fails or encounters an error, the error will be returned and the user needs to parse it.

The **SendData** method has this part:

Part	Data Type	Description
sRequest	String	Command

Example
String response = SendData (“0010”);
The response will be a hexadecimal string returned from the reader. An empty string returned indicates that the communication between the applet and the reader might most likely be disconnected.

SENDDATAWLEN METHOD

Returns a string value from the reader.

Syntax
String SendDataWlen(*String sRequest*)

Remarks
Send specific command to the Bluetooth reader. The length of the sRequest is not automatically calculated by the component. If the command fails or encounters an error, the error will be return and the user needs to parse it.

The **SendDataWlen** method has this part:

Part	Data Type	Description
sRequest	String	Command

Example
String response = SendDataWlen("000110");

The response will be a hexadecimal string returned from the reader. An empty string returned indicates that the communication between the applet and the reader might most likely be disconnected.

SENDDATASYNC METHOD

Returns an integer indicating the execution of the command.

Syntax
int SendDataSync(*String sRequest, byte[] response*)

Remarks
Send specific command to the Bluetooth reader. The length of the sRequest is automatically calculated by the component. The result is sent through the response parameter and the result can also be retrieved with SwipeCommandOutput() method. If SendDataSync returns a zero (0) the command executes successfully, otherwise a non-zero for error.

The **SendDataSync** method has these parts:

Part	Data Type	Description
sRequest	String	Command
response	Byte[]	Response by reference

Example
byte [] response;
int stat = SendDataSync("0010", response);
This returns a zero (0) if the command executes successfully. To retrieve the output, use SwipeCommandOutput() method.

The response will be the data part of the returning hex string from the reader.

SENDDATASYNCWLEN METHOD

Returns an integer indicating the execution of the command.

Syntax
int SendDataSyncWlen(*String sRequest, byte[] response*)

Remarks
Send specific command to the Bluetooth reader. The result is sent through the response parameter and the result can also be retrieved with SwipeCommandOutput() method. If SendDataSyncWlen returns a zero (0) the command executes successfully, otherwise a non-zero for error.

The **SendDataSyncWlen** method has these parts:

Part	Data Type	Description
sRequest	String	Command
response	Byte[]	Response by reference

Example
byte [] response;
int stat = SendDataSyncWlen("000110", response);
This returns a zero (0) if the command executes successfully. To retrieve the output, use SwipeCommandOutput() method. The response will be the data part of the returning hex string from the reader.

SEQUENCENUMBER METHOD

Returns an integer value indicating the sequence number from the reader. Currently returns 0 only. (RFU)

Syntax
int SequenceNumber()

SESSIONID METHOD

Returns a bye-array indicating the session ID from the reader.

Syntax
byte[] SessionID()

SESSIONIDHEXSTRING METHOD

Returns the MagnePrint data if supported by the reader in 2 Byte Hex format.

Syntax
String SessionIDHexString()

SETREADERTYPE METHOD

Sets the type of the reader.

Syntax
void SetReaderType(*int iRdrInterface*)

The **SetReaderType** method has this part:

Part	Data Type	Description
iRdrInterface	int	1 = Bluetooth; 0 = Generic RS232 reader

SWIPECOMMANDOUTPUT METHOD

Returns the result as string of a previously sent command.

Syntax
String SwipeCommandOutput()

Remarks
Returns the result as a string based on the input command.

TRACK1LENGTH METHOD

Returns an integer value indicating the length of track1 data.

Syntax
int Track1Length()

TRACK2LENGTH MEDTHOD

Returns an integer value indicating the length of track2 data.

Syntax
int Track2Length()

TRACK3LENGTH MEDTHOD

Returns an integer value indicating the length of track3 data.

Syntax
int Track3Length()

TRACK1LENGTHMASKED MEDTHOD

Returns an integer value indicating the length of masked track1 data.

Syntax
int Track1LengthMasked()

TRACK2LENGTHMASKED MEDTHOD

Returns an integer value indicating the length of masked track2 data.

Syntax
int Track2LengthMasked()

TRACK3LENGTHMASKED MEDTHOD

Returns an integer value indicating the length of masked track3 data.

Syntax
int Track3LengthMasked()

SECTION 3. EVENTS

1) On a card data change event:

In the event of card data change, the applet will return the card-swipe data by calling the Java script provided in the para name of RunJSOnCardDataChange. The Java script to be called must have one parameter to receive the card-swipe data. For example, the applet will return the card-swipe data by calling the "SetCardData" Java script in the following example. The "SetCardData" function prototype is as follows:

function SetCardData(cardSwipedData)
{ //Do something on a card data change event. }

```
<script language="JavaScript">
if(window.navigator.appName.toLowerCase().indexOf("netscape")!=-1)
{ // set object for Netscape:
      document.getElementById('dvObjectHolder').innerHTML ="<applet
      type=\"application/x-java-applet;version=1.5\"" +
      "codebase =\".\"" +
      "archive = \"JMTBluetoothReader.jar\"" +
      "code=\"JMTBTCardReader.class\"" +
      "name=\"MTRS232MSR\"" +
      "scriptable=\"true\"" +
      "style=\"visibility:hidden;\"" +
      "mayscript=\"mayscript\"" +
      "pluginspage=\"http://java.com/en/download/index.jsp\"" + ">" +
      "<param name=\"cache_option\" value=\"No\">" +
      "<param name=\"classloader_cache\" value=\"true\">" +
      "<param name= \"ReportJavaPluginVersion\" value=\"no\">" +
      "<param name= \"RunJSOnCardDataChange\" value=\"SetCardData\">" +
"</applet>";
}
else if(window.navigator.appName.toLowerCase().indexOf('internet
Explorer')!=-1){ //set object for IE
      document.getElementById('dvObjectHolder').innerHTML = "<object
      type=\"application/x-java-applet;version=1.5\"" +
      "codebase =\".\"" +
      "archive = \"JMTBluetoothReader.jar\"" +
      "code=\"JMTBTCardReader.class\"" +
      "name=\"MTRS232MSR\"" +
      "height=\"0\" width=\"0\" >" +
      "<param name=\"mayscript\" value=\"true\">" +
      "<param name=\"classloader_cache\" value=\"true\">" +
      "<param name=\"cache_option\" value=\"No\">" +
      "<param name= \"ReportJavaPluginVersion\" value=\"no\">" +
      "<param name= \"RunJSOnCardDataChange\" value=\"SetCardData\">" +
" </object>"
}
</script>
```

2) If the param name "ReportJavaPluginVersion" is set to "yes", the Java script "ReportJavaPluginVersion" should be available to be called and will have the following function prototype:

```
function ReportJavaPluginVersion(ver) {//Do something }
```