MAGTEK®
SECURITY FROM THE INSIDE

oDynamo

MagTek Common Message Structure (MTCMS)
Programmer's Reference Manual (Microsoft .NET/Java/Applet)

September 2019

Manual Part Number:
D998200160-20

REGISTERED TO ISO 9001:2015

Table 0.1 Revisions

Rev Number	Date	Notes
10	05/24/2018	Initial Release
20	09/17/2019	Update Java System Requirements.

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 ADDRESS ON THE FRONT PAGE OF THIS DOCUMENT, 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 ADDRESS LISTED IN THIS DOCUMENT, OR E-MAILED TO SUPPORT@MAGTEK.COM.

Table of Contents

1 Introduction

This document provides instructions for software developers who want to create .NET / Java software solutions that include a MagTek Common Message Structure (MTCMS) device connected to a Windows PC.

1.1 System Requirements

Language	Requirements
.NET	• Windows 7 • Windows 8, 8.1 • Windows 10 • Microsoft .NET Framework 4.5 or above.
Java Library	• Windows 7 • Windows 8, 8.1 • Windows 10 • Java Build Platform: JDK 1.8, 32-bit and above. • Minimum Java Runtime requirement: 8
Java Applet	• Windows 7 • Windows 8, 8.1 • Windows 10 Tested web browsers: • Internet Explorer 11 • Java Runtime requirements: Java 8. • Tested Java Runtime Environments: 8u221

2 How to Set Up the MagTek CMS SDK for Java Demo

2.1 How to Set Up the Java Library With the 32-bit JRE/JVM

MagTek highly recommends using the 32-bit version of Java when using the PCI PED Java applet, regardless of whether you are using a 32-bit or 64-bit version of Windows.

1) Uninstall any existing instances of the 64-bit Java Runtime Environment (JRE) or Java Development Kit (JDK). Leaving them installed can cause runtime failures, as the library may fail to load.
2) Download and install the latest version of the 32-bit Java Development Kit (JDK).
3) If you opted to manually copy the SDK dependencies from a master development workstation to the target workstation where it will be used, follow these steps:
   a) On the master workstation, navigate to the root of the SDK **\Library\Java**
   b) Open the **\x86** subfolder and copy all the files to the target workstation's **C:\Windows\System32** folder for x86 systems, or to the target workstation's **C:\Windows\SysWOW64** folder for x64 systems.
4) Connect the device to the workstation using a USB cable. Windows will install the device drivers automatically. Wait for Windows to report the driver installation is complete.
5) Launch a Windows command prompt as an Administrator.
6) **cd** to the root of the folders where the SDK is installed.
7) Type **runsample.bat** and press **Enter** to launch the Java Demo software.

2.2 How to Set Up the Java Library With the 64-bit JRE/JVM

MagTek highly recommends using the 32-bit version of Java if you intend to use the Java applet, regardless of whether you are using a 32-bit or 64-bit version of Windows.

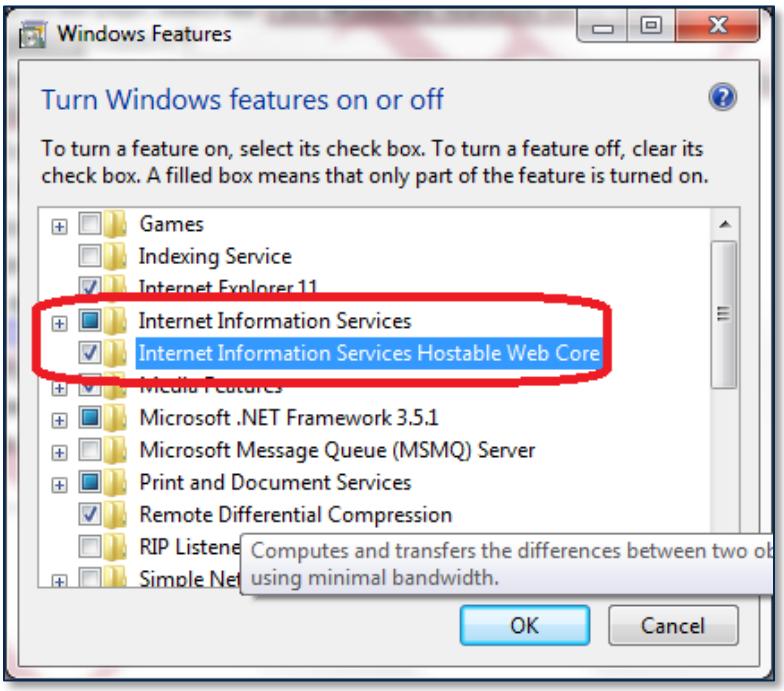
1) Uninstall any existing instances of the 32-bit Java Runtime Environment (JRE) or Java Development Kit (JDK). Leaving them installed can cause runtime failures, as the library may fail to load.
2) Download and install the latest version of the 64-bit Java Development Kit (JDK).
3) If you opted to manually copy the SDK dependencies from a master development workstation to the target workstation where it will be used, follow these steps:
   a) On the master workstation, navigate to the root of the SDK **\Library\Java**
   b) Open the **\x64** subfolder and copy all the files to the target workstation's **C:\Windows\System32** folder.
4) Connect the device to the workstation using a USB cable. Windows will install the device drivers automatically. Wait for Windows to report the driver installation is complete.
5) Launch a Windows command prompt as an Administrator.
6) **cd** to the root of the folders where the SDK is installed.
7) Type **runsample.bat** and press **Enter** to launch the Java Demo software.

2.3 How to Set Up the Applet With the 32-bit JRE/JVM

MagTek highly recommends using the 32-bit version of Java when using the Java applet, regardless of whether you are using a 32-bit or 64-bit version of Windows.

To set up the Java applet using the 32-bit version of Java on either a 32-bit or 64-bit version of Windows, follow these steps:

1) Follow the steps in section **2.1 How to Set Up the Java Library With the 32-bit JRE/JVM**. Having a working JVM, working Java library, working drivers, and working DLLs are prerequisites for using the applet.
2) Verify Java is installed and that the Internet Explorer Java plugin is working correctly by using Oracle's Java applet test page, usually provided as a link or auto-launch at the end of installation.
3) On the Windows 7 workstation you will use for development, enable Internet Information Services 7 (IIS) as follows:
   a) Log in to a Windows 7 workstation using an administrator account.
   b) Launch the Windows **Control Panel**.
   c) Select the **Programs and Features** item to open the **Programs and Features** page.
   d) On the left side of the page, select the **Turn Windows features on or off** link to launch the **Windows Features** window.
   e) Turn on the checkboxes for **Internet Information Services** and **Internet Information Services Hostable Web Core**.



   f) Press the **OK** button to launch a progress window. Wait for Windows to install IIS.
4) Launch a web browser and navigate to **//localhost**. Verify the IIS default page appears as shown in **Figure 2-1**.

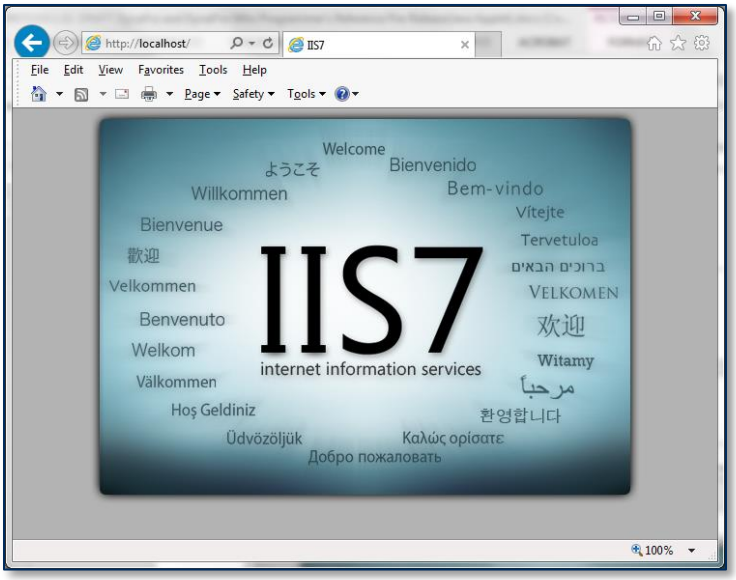


Figure 2-1 - IIS Default Page

5) If it does not already exist, create a **MTPPSCRA** folder in **C:\inetpub\wwwroot**. If it does exist, delete its contents.
6) On the workstation where the SDK is installed, navigate to the folder where it is installed.
7) Open the **Sample Code\Java Applet\Object\Signed** subfolder.
8) Copy the contents of the subfolder to **C:\inetpub\wwwroot\MTCMS**.
9) Open the **Sample Code\Java Applet\Object\x86** subfolder.
10) Copy the contents of the subfolder to **C:\inetpub\wwwroot\MTCMS**.

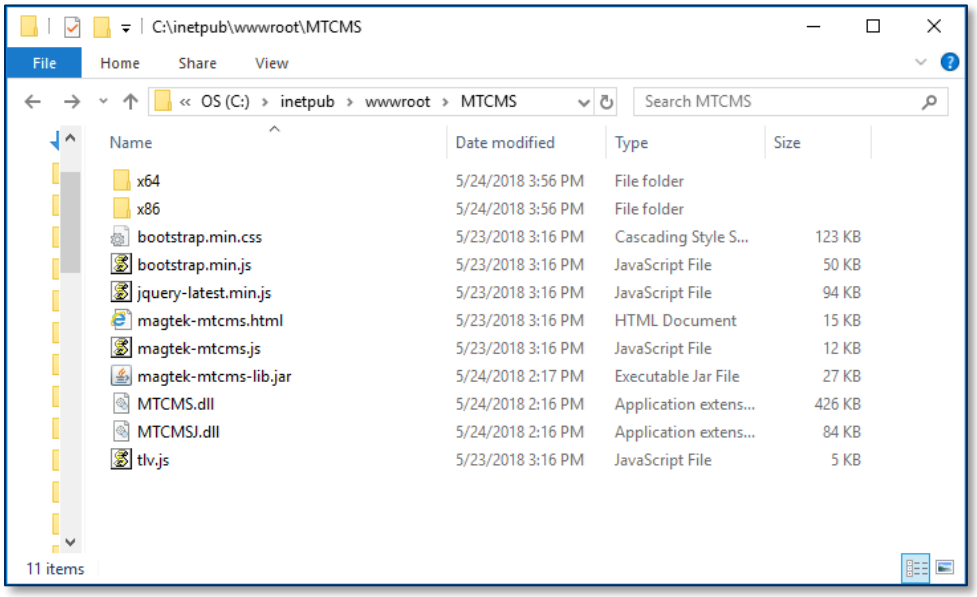
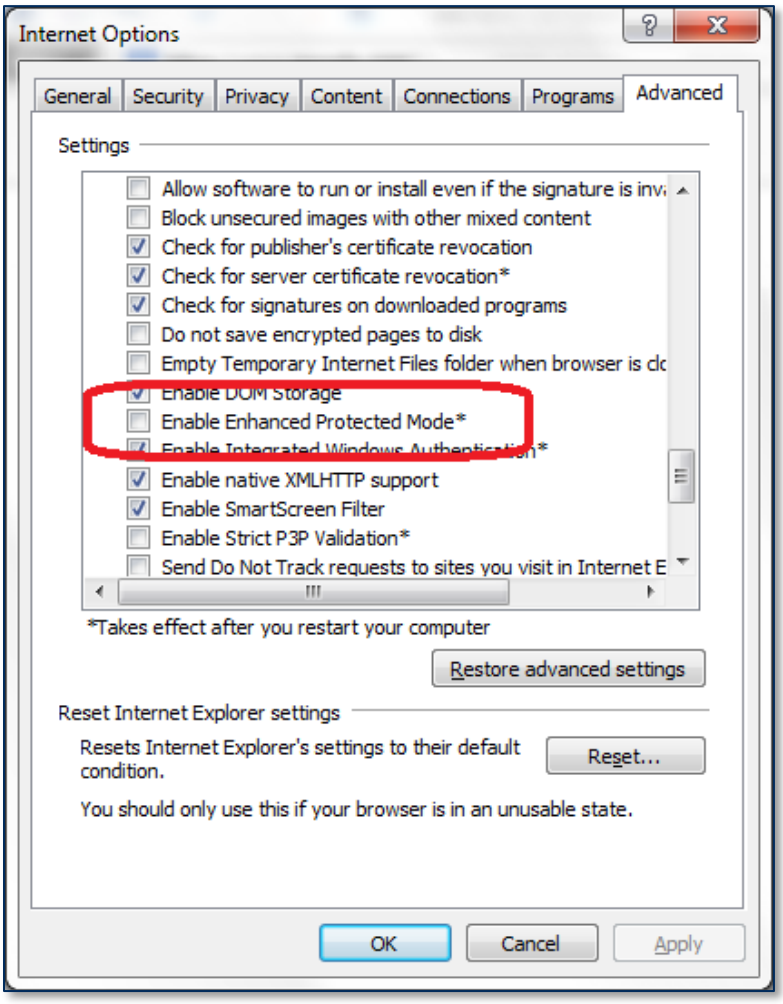


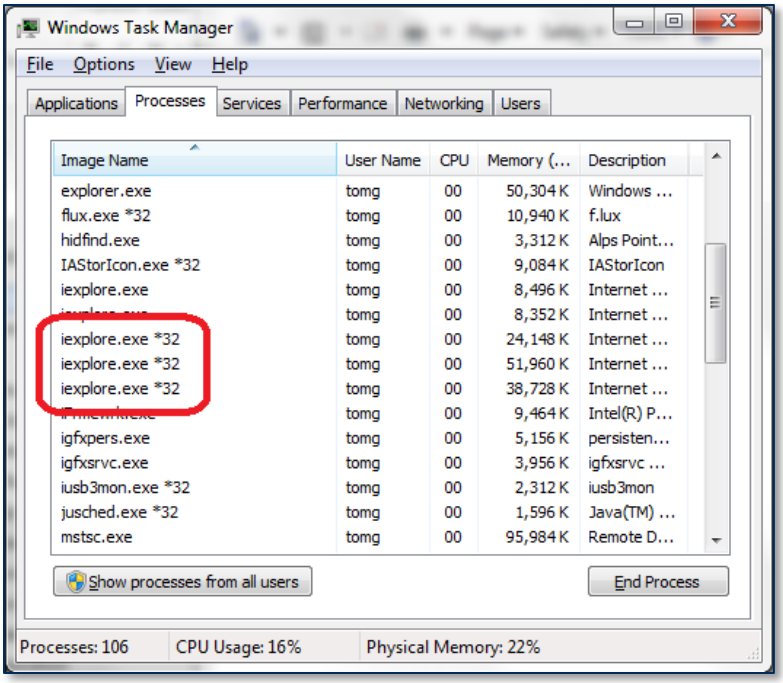
Figure 2-2 - inetpub Structure

11) Connect the device to the workstation using a USB cable. Windows will install the device drivers automatically. Wait for Windows to report the driver installation is complete.

12) Open Internet Explorer as an administrator.
13) If you are using a 64-bit version of Windows with IE11, make sure to launch directly in 32-bit mode using the iexplore.exe found in **C:\Program Files (x86)**. Verify you are running in 32-bit mode using the **Help** > **About** menu.
14) If you are running a 64-bit version of Windows with IE10 or higher, choose the **Internet options** that enable 32-bit mode / disable **Protected Mode** for the zone you are accessing. Also turn **OFF** the checkbox for **Enhanced Protected Mode** in the **Internet Options** > **Advanced** tab.

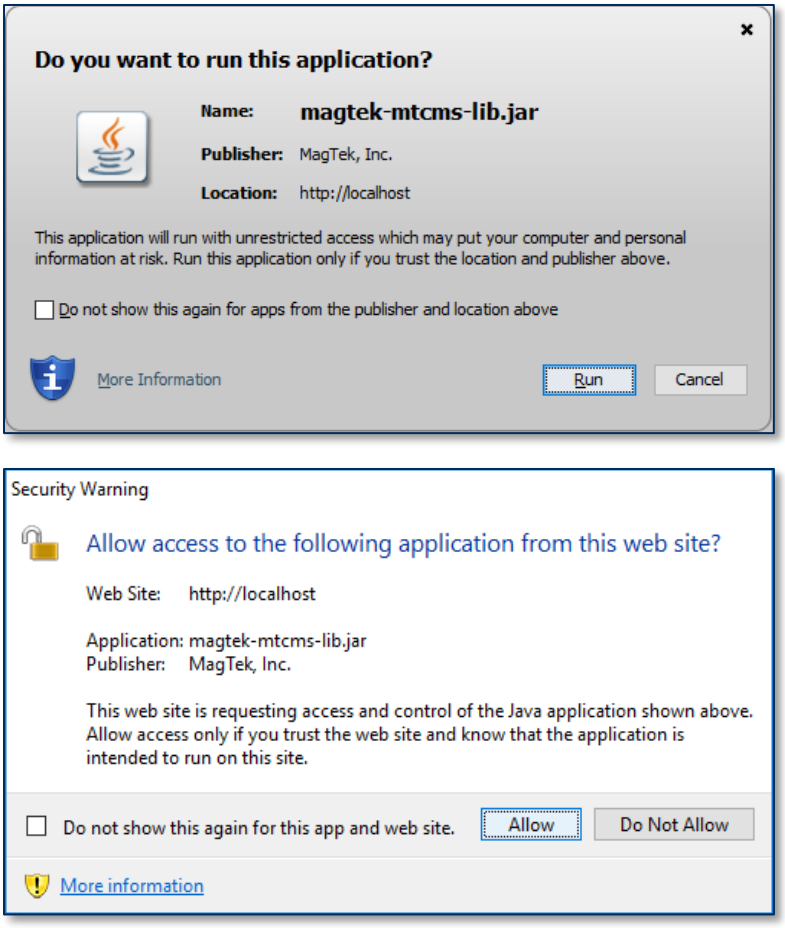


15) If you changed the value of the **Enable Enhanced Protected Mode** checkbox, restart Windows.
16) Open Windows Task Manager (**Ctrl-Alt-Del** > **Start Task Manager**).
17) Open the **Processes** tab and sort by **Image Name**.
18) Note the number and location of all **iexplore.exe *32** processes.
19) In Internet Explorer, navigate to http://localhost/MTCMS/magtek-mtcms.html.
20) In the Windows Task Manager **Processes** tab, find the new process for the Internet Explorer tab you just opened and make sure it is running in 32-bit mode (**iexplore.exe *32** instead of **iexplore.exe**).



21) Close the **Windows Task Manager** window.

22) Internet Explorer will display a welcome page and will pop up a **Do you want to run this application?** window. Press the **Run** button and the **Allow** button to run the Java applet.



23) On the welcome page, press the **Connect** button.

24) Press the **Get Info** button. The large text box in the browser will display device information.

25) Use the buttons and fields on the welcome page to test the connection to the device.

26) After installation on the workstation is complete, future browser sessions do not require the user to launch Internet Explorer as an administrator to use the applet.

2.4 How to Modify Manifest

The **Caller-Allowable-Codebase** attribute is used to identify the domains from which JavaScript code can make calls to your RIA without security prompts. Set this attribute to the domain that hosts the JavaScript code. If a call is made from JavaScript code that is not located in a domain specified by the **Caller-Allowable-Codebase** attribute, the call is blocked. To specify more than one domain, separate the domains by a space, for example:

```
Caller-Allowable-Codebase: *.yahoo.com *.google.com *.magtek.com *
```

The **Application-Library-Allowable-Codebase** attribute identifies the locations where your signed RIA is expected to be found. This attribute is used to determine what is listed in the Location field for the security prompt that is shown to users when the JAR file for your RIA is in a different location than the JNLP file or HTML page that starts your RIA. If the files are not in the locations identified, the RIA is

blocked. Set this attribute to the domains where the JAR file, JNLP file, and HTML page are located. To specify more than one domain, separate the domains by a space, for example:

```
Application-Library-Allowable-Codebase: *.yahoo.com *.google.com
*.magtek.com *
```

For more information regarding the JAR File Manifest Attributes for Security, please visit this website http://docs.oracle.com/javase/7/docs/technotes/guides/jweb/security/manifest.html

In order to modify the Manifest file, please follow these steps. You need to do this for: **magtek-mtcms-lib.jar**

1) Find installation folder by default, the installation folder is: **\Sample Code\Java Applet\Object\Unsigned**
2) Launch the command prompt and extract the META-INF/MANIFEST.MF from the jar file.
```
jar xf magtek-mtcms-lib.jar  META-INF/MANIFEST.MF
```
3) Open **MANIFEST.MF** and look for the **Caller-Allowable-Codebase** and **Application-Library-Allowable-Codebase** and add your website URL to the list like the example above.
4) Update the manifest to the jar file.
```
jar umf META-INF/MANIFEST.MF magtek-mtcms-lib.jar
```

2.5 How to Sign JAR

These instructions provide an overview of obtaining and using Sun Java signing and a digital certificate. Please follow this instruction to sign and verify both **MTCMSJavaSample.jar** and **magtek-mtcms-lib.jar**

1) Make sure your machine has the latest Java JDK installed.
2) Generate a public/private key pair by entering the following command, specifying an alias for your keystore:
```
keytool -genkey -keyalg rsa -alias MyCert
```
3) Generate a certificate signing request (CSR) by entering the following command:
```
keytool -certreq -alias MyCert
```
After prompting you to enter the password for your keystore, keytool will generate a CSR.
4) Save the certificate received from the Certificate provider as Certname.p7b.
5) Import your Digital Certificate by entering the following command:
```
keytool -import -alias MyCert -file Certname.p7b
```
In this string, keytool is requested to import the Digital ID “Certname.cer” into the keystore MyCert.
6) Bundle your applet into a Java Application Resource (JAR) file by entering the following command:
```
jar cvf C:\MTCMSJavaSample.jar
```
7) Sign the files by using jarsigner to sign the JAR file, using the private key you saved in your keystore:
```
jarsigner C:\Magtek-ppscra-applet.jar MyCert
jarsigner C:\magtek-mtcms-lib.jar MyCert
```

8) Verify the output of your signed JAR file by entering the following command:

```
jarsigner -verify -verbose -certs C:\Magtek-ppscra-applet.jar
jarsigner -verify -verbose -certs C:\magtek-mtcms-lib.jar
```

Please visit this website https://docs.oracle.com/javase/tutorial/deployment/jar/signing.html for more information regarding signing JAR files.

3 How to Set Up the MagTek CMS SDK for .NET Projects

Custom Windows software installed on a host PC can communicate with MagTek Common Message Structure (MTCMS) devices via USB, network interface, or serial interface using the MTCMS library.

The supported platforms for .NET projects include Windows 7, Windows 8/8.1, and Windows 10. The .NET project should contain references to the main library file: **`MTCMSNET.dll`**.

To add the MagTek CMS library to a .NET project in Microsoft Visual Studio, follow these steps:

1) Create or open your .NET project in Visual Studio.
2) Copy the following DLL file from the **Library** folders to the library folder of your software project:
   - `MTCMSNET.dll`
3) In the Visual Studio Solution Explorer, right-click the project and select **Add Reference** to show the **Add Reference** window.
4) Select the **Browse** tab and press the **Browse…** button.
5) Navigate to your library folder, select **`MTCMSNET.dll`**, then press the **Add** button.
6) In your custom software, create an instance of **`MTDevice`**. For examples, see the source code included with the **MTCMSNETDemo** project and/or **Appendix A** Code Examples.
7) Begin using the features provided by the MTCMS library.

4 MTDevice Class Methods

After creating an instance of the MTDevice class in your software project, use the methods described in this section to communicate with MagTek CMS device.

4.1 requestDeviceList

This method initiates request to discover devices that are visible to the host using the specified connection interface. The OnDeviceList event will provide information regarding the available devices once the discovery process is completed.

```
public void requestDeviceList(MTConnectionType connectionType)
```

Parameters:

Parameter	Description

connectionType	MTConnectionType value:
	`MTConnectionType.USB,`
	`MTConnectionType.IP,`
	`MTConnectionType.Serial`

Return Value: None

4.2 setConnectionType

This method sets the connection type of the device..

```
public void setConnectionType(MTConnectionType connectionType)
```

Parameters:

Parameter	Description

connectionType	MTConnectionType value:
	`MTConnectionType.USB,`
	`MTConnectionType.IP,`
	`MTConnectionType.Serial`

Return Value: None

4.3 setAddress

This method sets the address of the device.

```
public void setAddress(string deviceAddress)
```

Parameters:

Parameter	Description
deviceAddress	String value of the address.

The following table shows the address formats supported by the different connection types:

<table>
<tr><th>Connection Type</th><th>Address Format</th></tr>
<tr><td>USB</td><td>[PATH]

<table><tr><th>Parameter</th><th>Description</th></tr><tr><td>[PATH]</td><td>The OS specific device path to the USB device. The path is normally retrieved from the Address property of MTDeviceInformation.</td></tr></table></td></tr>
<tr><td>IP</td><td>[IPA]
or
[IPA]:[PORT]

<table><tr><th>Parameter</th><th>Description</th></tr><tr><td>[IPA]</td><td>The IP address of the device in dotted-quad notation (i.e. 192.178.1.123).</td></tr><tr><td>[PORT]</td><td>The TCP port of the device.
(Default: 5000)</td></tr></table></td></tr>
</table>

Serial

PORT=**[PORT]**,
BAUDRATE=**[BAUDRATE]**,
DATABITS=**[DATABITS]**,
PARITY=**[PARITY]**,
STOPBITS=**[STOPBITS]**,
HANDSHAKE=**[HANDSHAKE]**,
STARTINGBYTE=**[STARTINGBYTE]**,
ENDINGBYTE=**[ENDINGBYTE]**,
CRCMODE=**[CRCMODE]**

Parameter	Description
[PORT]	The OS specific device path to the serial port (i.e. COM4).
[BAUDRATE]	The data baud rate . (Default: 9600)
[DATABITS]	The data bits per byte. (Default: 8)
[PARITY]	The parity checking protocol. (Default: NONE). Supported Values: NONE,EVEN,ODD,SPACE,MARK
[STOPBITS]	The number of stop bits per byte. (Default: 1) Supported Values: 1,1.5,2
[HANDSHAKE]	The handshaking protocol for serial port transmission of data. (Default: NONE) Supported Values: NONE,RTS,XONXOFF,RTSXONSOFF
[STARTINGBYTE]	The special character used as the starting byte for each message. (Default is empty string) An empty string indicates no special character is used as the starting byte for each message.
[ENDINGBYTE]	The special character used as the ending byte for each message. (Default is 0x0A)

Connection Type	Address Format	
		An empty string indicates no special character is used as the ending byte for each message.
	[CRCMODE]	A value of 0 indicates CRC is disabled, otherwise CRC is enabled. (Default: 0)

Return Value: None

4.4 setDeviceID

This method sets the device ID.

```
public void setDeviceID(string deviceID)
```

Parameters:

Parameter	Description
deviceID	String value of the device ID.

Return Value: None

4.5 openDevice

This method opens the connection to the device.

```
public void openDevice()
```

Parameters: None

Return Value: None

4.6 closeDevice

This method closes the connection to the device.

```
public void closeDevice()
```

Parameters: None

Return Value: None

4.7 isDeviceConnected

This method returns whether the device is connected or not.

```
public bool isDeviceConnected()
```

Parameters: None

Return Value:
Return true if the device is connected. Otherwise, return false.

4.8 sendDataString

This method sends a data string to the device.

```
public int sendDataString(string dataString)
```

Parameters:

Parameter	Description
dataString	Data to be sent in hexadecimal string format.

Return Value:

- 0 = Success (MTDevice.SEND_SUCCESS)
- 9 = Error (MTDevice.SEND_ERROR)
- 15 = Busy (MTDevice.SEND_BUSY)

4.9 sendDataBytes

This method sends data bytes to the device.

```
public int sendDataBytes(byte[] dataBytes)
```

Parameters:

Parameter	Description
dataBytes	Data to be sent in byte array format.

Return Value:

- 0 = Success (MTDevice.SEND_SUCCESS)
- 9 = Error (MTDevice.SEND_ERROR)
- 15 = Busy (MTDevice.SEND_BUSY)

4.10 sendMTCMSMessage

This method sends a MagTek CMS message to the device.

```
public int sendMTCMSMessage(MTCMSMessage message)
```

Parameters:

Parameter	Description
message	MTCMSMessage to be sent to the device.

Return Value:

- 0 = Success (MTDevice.SEND_SUCCESS)
- 9 = Error (MTDevice.SEND_ERROR)
- 15 = Busy (MTDevice.SEND_BUSY)

5 MTCMSMessage Class Methods

This class allows building MagTek CMS messages to be used in communications with MagTek CMS devices.

5.1 MTCMSMessage

This constructor method builds an MTCMSMessage instance with the provided values.

```
public void MTCMSMessage(int messageType, int applicationID, int
commandID, int dataTag, byte[] data)
```

Parameters:

Parameter	Description
messageType	MessageType value
applicationID	ApplicationID value
commandID	CommandID value
dataTag	Data Tag value
data	Data value

Return Value: None

5.2 MTCMSMessage

This constructor method builds an MTCMSMessage instance with the provided byte array value.

```
public void MTCMSMessage(byte[] messageBytes)
```

Parameters:

Parameter	Description
messageBytes	Message byte array value

Return Value: None

5.3 setMessageType

This method sets the value of MessageType.

```
public void setMessageType(int messageType)
```

Parameters:

Parameter	Description
messageType	MessageType value

Return Value: None

5.4 setApplicationID

This method sets the value of ApplciationID.

```
public void setMessageType(int messageType)
```

Parameters:

Parameter	Description
applicationID	ApplicationID value

Return Value: None

5.5 setCommandID

This method sets the value of CommandID.

```
public void setCommandID(int commandID)
```

Parameters:

Parameter	Description
commandID	CommandID value

Return Value: None

5.6 setResultCode

This method sets the value of ResultCode.

```
public void setResultCode (int resultCode)
```

Parameters:

Parameter	Description
resultCode	ResultCode value

Return Value: None

5.7 setData

This method sets the value of Data Tag and the value of Data.

```
public void setData (int dataTag, byte[] data)
```

Parameters:

Parameter	Description
dataTag	Data Tag value
data	Data value

Return Value: None

5.8 getMessageType

This method returns the value of MessageType.

```
public int getMessageType()
```

Return Value: MessageType value.

5.9 getApplicationID

This method returns the value of ApplicationID.

```
public int getApplciationID()
```

Return Value: ApplicationID value.

5.10 getCommandID

This method returns the value of CommandID.

```
public int getCommandID()
```

Return Value: CommandID value.

5.11 getResultCode

This method returns the value of ResultCode.

```
public int getResultCode()
```

Return Value: ResultCode value.

5.12 getDataTag

This method returns the value of Data Tag.

```
public int getDataTag()
```

Return Value: Data Tag value.

5.13 getData

This method returns the value of Data.

```
public byte[] getData()
```

Return Value: Data value.

5.14 getMessageBytes

This method returns the message bytes.

```
public byte[] getMessageBytes()
```

Return Value: Message bytes.

6 MTCMSRequestMessage Class Methods

This class is a subclass of MTCMSMessage. This class instantiates an MTCMSMessages instance with MessageID value set to the Request message type.

6.1 MTCMSRequestMessage

This constructor method builds and initializes an MTCMSRequestMessage instance with the provided values.

```
public void MTCMSRequestMessage(int applicationID, int commandID, int
dataTag, byte[] data)
```

Parameters:

Parameter	Description
applicationID	ApplicationID value
commandID	CommandID value
dataTag	Data Tag value
data	Data value

Return Value: None

7 MTCMSResponseMessage Class Methods

This class is a subclass of MTCMSMessage. This class instantiates an MTCMSMessages instance with MessageID value set to the Response message type.

7.1 MTCMSResponseMessage

This constructor method builds and initializes an MTCMSResponseMessage instance with the provided values.

```
public void MTCMSResponseMessage(int applicationID, int commandID, int
dataTag, byte[] data)
```

Parameters:

Parameter	Description
applicationID	ApplicationID value
commandID	CommandID value
dataTag	Data Tag value
data	Data value

Return Value: None

8 MTCMSNotificationMessage Class Methods

This class is a subclass of MTCMSMessage. This class instantiates an MTCMSMessages instance with MessageID value set to the Notification message type.

8.1 MTCMSNotificationMessage

This constructor method builds and initializes an MTCMSNotificationtMessage instance with the provided values.

```
public void MTCMSNotificationMessage(int applicationID, int commandID,
int dataTag, byte[] data)
```

Parameters:

Parameter	Description
applicationID	ApplicationID value
commandID	CommandID value
dataTag	Data Tag value
data	Data value

Return Value: None

9 MTDevice Events

9.1 OnDeviceList

This event occurs when device information is available.

```
public event DeviceListHandler OnDeviceList

public delegate void DeviceListHandler(object sender,
MTConnectionType connectionType, List<MTDeviceInformation> deviceList)
```

Parameter	Description
Sender	Object representing the publisher of the event.

connectionType	MTConnectionType value:
	`MTConnectionType.USB,`
	`MTConnectionType.IP,`
	`MTConnectionType.Serial`

deviceList	A list of MTDeviceInformation objects

9.2 OnDeviceConnectionStateChanged

This event occurs when the connection state of the device is changed.

```
public event DeviceConnectionStateHandler
OnDeviceConnectionStateChanged

public delegate void DeviceConnectionStateHandler(object sender,
MTConnectionState state)
```

Parameter	Description
sender	Object representing the publisher of the event.

state	MTDeviceState value indicating the state of the device:
	`Disconnected`
	`Connecting`
	`Error`
	`Connected`
	`Disconneting`

9.3 OnDeviceDataString

This event occurs when a response is received from the device.

```
public event DeviceDataStringHandler OnDeviceDataString

public delegate void DeviceDataStringHandler(object sender, string
dataString)
```

Parameter	Description
sender	Object representing the publisher of the event.
dataString	Hexadecimal string representing data received.

9.4 OnDeviceDataBytes

This event occurs when a response is received from the device.

```
public event DeviceDataBytesHandler OnDeviceDataBytes

public delegate void DeviceDataBytesHandler(object sender, byte[]
dataBytes)
```

Parameter	Description
sender	Object representing the publisher of the event.
dataBytes	Byte array representing data received.

9.5 OnDeviceResponseMessage

This event occurs when a response is received from the device.

```
public event DeviceResponseMessageHandler OnDeviceResponseMessage

public delegate void DeviceResponseMessageHandler(object sender,
MTCMSResponseMessage response)
```

Parameter	Description
sender	Object representing the publisher of the event.
response	MTCMSResponseMessage representing data received.

9.6 OnDeviceNotificationMessage

This event occurs when a response is received from the device.

```
public event DeviceNotificationMessageHandler
OnDeviceNotificationMessage

public delegate void DeviceNotificationMessageHandler(object sender,
MTCMSNotificationMessage notification)
```

Parameter	Description
sender	Object representing the publisher of the event.
notification	MTCMSNotificationMessage representing data received.

Appendix A Code Examples (.NET)

A.1 Initialize Device

```
MTDevice device = new MTDevice();
if (device != null)
{
      device.setConnectionType(MTConnectionType.USB);

      device += OnDeviceConnectionStateChanged;
      device += onDeviceResponseMessage;
      device += onDeviceNotificationMessage;
}
```

A.2 Connect to Device

```
if (device != null)
{
      device.openDevice();
}
```

A.3 Send Data String to Device

```
String dataString = "C00101C10100C20114";
device.sendDataString(dataString);
```

A.4 Send Data Bytes to Device

```
Byte[] dataBytes = new Byte[9] {0xC0,0x01,0x01,
                                0xC1,0x01,0x00,
                                0xC2,0x01,0x14};
device.sendDataBytes(dataBytes);
```

A.5 Send MTCMSMessage to Device

```
MTCMSMessage message = new MTCMSMessage(0x01,0x00,0x14, 0xC4, null);
device.sendMTCMSMessage(message);
```

A.6 Send MTCMSRequestMessage to Device

```
MTCMSRequestMessage request = new MTCMSRequestMessage
                                (0x00,0x14, 0xC4, false);
device.sendMTCMSMessage(request);
```

A.7 Receiving Connection State Updates from Device

```
protected void OnDeviceConnectionStateChanged(object sender,
MTConnectionState state)
{
      if (state == MTConnectionState.Connected)
      {
            sendToDisplay("Device is Connected");
      }
      else if (state == MTConnectionState.Disconnected)
```

```
      {
            sendToDisplay("Device is Disconnected");
      }
}
```

A.8 Receiving Response Message from Device

```
protected void OnDeviceResponseMessage(object sender,
MTCMSResponseMessage response)
{
      sendToDisplay("[Response Message]"
      sendToDisplay("AppID: " + response.getApplicationID());
      sendToDisplay("CommandID: " + response.getCommandID());
      sendToDisplay("ResultCode: " + response.getResultCode());
}
```

A.9 Receiving Notification Message from Device

```
protected void OnDeviceNotificationMessage(object sender,
MTCMSNotificationMessage notification)
{
      sendToDisplay("[Notification Message]"
      sendToDisplay("AppID: " notification. getApplicationID() );
      sendToDisplay("CommandID: " + notification.getCommandID());
      sendToDisplay("ResultCode: " + notification.getResultCode());
}
```

A.10 Close Device

```
if (device != null)
{
      device.closeDevice();
}
```