EXCELLA MDX

WINDOWS API SPECIFICATIONS
PROGRAMMING REFERENCE MANUAL

PART NUMBER 99875391-4.01

AUGUST 2009

Confidential

This document contains the proprietary information of MagTek. Its receipt or possession does not convey any rights to reproduce or disclose its contents or to manufacture, use or sell anything it may describe. Reproduction, disclosure or use without specific written authorization of MagTek is strictly forbidden.

Unpublished – All Rights Reserved

MAGTEK®

REGISTERED TO ISO 9001:2000

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

Copyright© 2004 – 2009
MagTek®, Inc.
Printed in the United States of America

Information in this document is subject to change without notice. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of MagTek, Inc.

MagTek is a registered trademark of MagTek, Inc.
Excella™ is a trademark of MagTek, Inc.
Microsoft® is a trademark of Microsoft, Inc.

REVISIONS

Rev Number	Date	Notes
1	30 May 08	Initial Release
2	18 February 09	
3	20 May 09	
4	27 Aug 09	Added new XML tags, updated software license agreement, added sections & headers

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

TABLES AND FIGURES

SECTION 1. OVERVIEW

This manual contains the following sections:

Section 1. Overview
Section 2. Excella MDX Software Architecture – includes flow diagrams, screen captures, and several "How To" descriptions.
Section 3. Excella MDX API – describes Excella MDX device API functions and return codes.
Section 4. Keys Sent to Device – lists and explains keys sent to the Excella MDX device by the application.
Section 5. Keys Received From Device – lists and explains keys received from the Excella MDX device.
Section 6. Other Keys Available from Device – lists additional keys available from Excella MDX device

REQUIREMENTS

The following item is required for software installation:

P/N 22359103, API/Demo for Excella MDX (CD)

This CD installs MagTek USB Drivers, MTXMLMCR.dll and Demo program.

SECTION 2. EXCELLA MDX SOFTWARE ARCHITECTURE

The architecture of the system is shown in Figure 2-1. Descriptions of the terms and operations used follow the illustration.

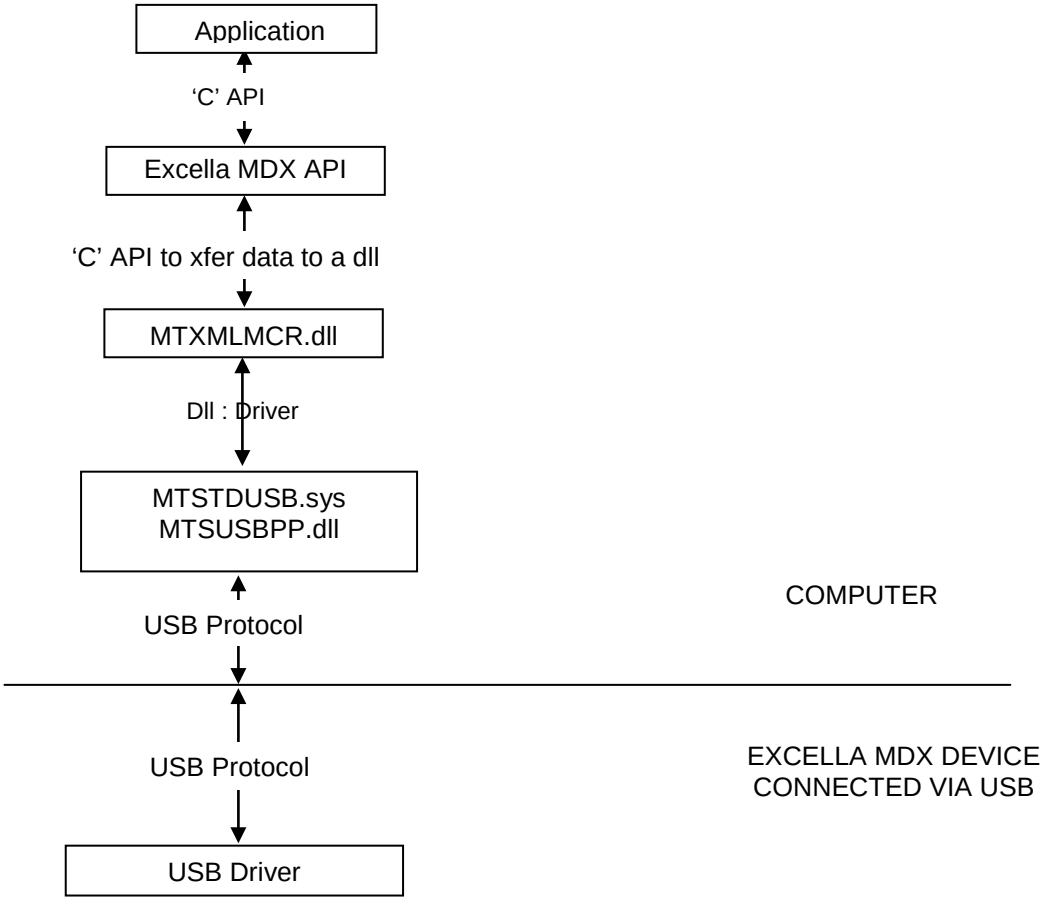

Figure 2-1. Excella Software Architecture

TERM DESCRIPTION

MTXMLMCR.dll. Application provides API to upper level application to talk to the device.

MTSTDUSB.sys : MagTek USB driver file, located in Windows\System32\drivers folder**.**

MTSUSBPP.dll : MagTek USB driver file, located in Windows\System32 folder**.**

API FUNCTIONS

Table 2-1 lists functions provided by **MTXMLMCR.dll**. Please refer to Section 3 for a complete description of these functions.

Table 2-1. Functions

NAME	DESCRIPTION
MTMICRGetDevice	Get device name of all devices present.
MTMICROpenDevice	Opens device with the given name.
MTMICRCloseDevice	Closes device with the given name.
MTMICRQueryInfo	Sends a request to a given device name to query for info on a given query parameter.
MTMICRSendCommand	Sends a single command to a given device name.
MTMICRProcessCheck	Sends a scan request to a given device to scan with a given process options.
MTMICRGetImage	Sends request to the given device name to get an image of a previously scanned check.
MTMICRSetValue	Adds a key/value pair to a given section.
MTMICRSetIndexValue	Adds a key/value pair with index to a given section.
MTMICRGetValue	Returns value of a key/value pair from a given section.
MTMICRGetIndexValue	Returns value of a key/value pair with index from a given section.

SECTION 3. EXCELLA MDX API

MTMICRGetDevice

MTMICRGetDevice function returns the device name of the device present in the system.

```
ULONG MTMICRGetDevice (
        DWORD       dwDeviceContext,
        char        *pcDevName
);
```

Parmeters

dwDeviceContext

This is the device number of the device. This must be set to 1 to get the first device in the system. Increment it by 1 to get the next device name.

pcDevName

Pointer to the buffer where the device name will be stored.

Return Values

MICR_ST_OK
MICR_ST_DEVICE_NOT_FOUND
MICR_ST_BAD_PARAMETER

Remarks

If the function succeeds, the return value is MICR_ST_OK. The device name of the device is filled in the buffer pointed by the parameter pcDevName.

If there is no device present or there is no device that is associated with the given dwDeviceContext, then the function fails and MICR_ST_DEVICE_NOT_FOUND is returned.

If there is no memory allocated for the pcDevName parameter, then MICR_ST_BAD_PARAMETER is returned.

Example

```
#define DEVICE_NAME_LEN 128
int i=1;
DWORD dwResult;
char pcDevName[DEVICE_NAME_LEN]="";
while ((dwResult = MTMICRGetDevice(i,(char*) pcDevName)) != MICR_ST_DEVICE_NOT_FOUND)
{
        // Device found, increment the device number
        i++;
}
```

MTMICROpenDevice

MTMICROpenDevice function opens the device with the given device name.

```
ULONG MTMICROpenDevice (
        char            *pcDevName
);
```

Parmeters

pcDevName

Pointer to null terminated string that specifies the name of the device to open. Use function **MTMICRGetDevice** to obtain the device name.

Return Values

If the function succeeds, the return value is MICR_ST_OK.

MICR_ST_OK
MICR_ST_BAD_DEVICE_NAME
MICR_ST_DEVICE_NOT_FOUND
MICR_ST_DEVICE_NOT_RESPONDING
MICR_ST_MSXML_NOT_FOUND
MICR_ST_MSXML_FAILED

Remarks

If the pcDevName is NULL, the return value is MICR_ST_BAD_DEVICE_NAME.
If no device is found, the return value is MICR_ST_DEVICE_NOT_FOUND.
If MSXML is not installed, return value is MICR_ST_MSXML_NOT_FOUND
If MSXML cannot be instantiated, return value is MICR_ST_MSXML_FAILED
If device is found but cannot connect, return value is MICR_ST_DEVICE_NOT_RESPONDING

Example

```
#define DEVICE_NAME_LEN 128
int i=1;
DWORD dwResult;
char pcDevName[DEVICE_NAME_LEN]="";
while ((dwResult = MTMICRGetDevice(i,(char*) pcDevName)) != MICR_ST_DEVICE_NOT_FOUND)
{
        if (MTMICROpenDevice (pcDevName) == MICR_ST_OK)
        {
                ///close the device
                MTMICRCloseDevice (pcDevName);
        }
        i++;
}
```

MTMICRCloseDevice

MTMICRCloseDevice function closes the device with the given device name.

```
ULONG MTMICRCloseDevice (
        char            *pcDevName
);
```

Parameters

pcDevName
: Pointer to null terminated string containing the name of the device to close. This is the device that is previously opened using function **MTMICROpenDevice**.

Return Values

MICR_ST_OK
MICR_ST_BAD_DEVICE_NAME
MICR_ST_DEVICE_NOT_FOUND

Remarks

If the pcDevName is NULL, the return value is MICR_ST_BAD_DEVICE_NAME.

Example

```
#define DEVICE_NAME_LEN 128
int i=1;
DWORD dwResult;
char pcDevName[DEVICE_NAME_LEN]="";

// Get the device name at device number "i"
while ((dwResult = MTMICRGetDevice(i,(char*) pcDevName)) != MICR_ST_DEVICE_NOT_FOUND)
{
     // Success in getting the device name

    // Open this device
    if (MTMICROpenDevice (pcDevName) == MICR_ST_OK)
    {
        // Now close this device
        MTMICRCloseDevice (pcDevName);
    }
    else
    {
        // Error while opening this device.
    }

    i++;
}
```

MTMICRQueryInfo

MTMICRQueryInfo function inquires data of a given section name from the given device name.

ULONG MTMICRQueryInfo (
char *pcDevName,
char *pcSection,
char *pcSectionData,
DWORD *pdwLength
);

Parameters

pcDevName
Pointer to null terminated string containing device name.

pcSection
Pointer to null terminated string containing the section name. e.g. DeviceCapabilities, DeviceUsage, DeviceStatus, etc.

pcSectionData
Pointer to the buffer that is used to store the data of the inquiry section.

pdwLength
Specifies the size of the pcSectionData buffer.

Return Values

MICR_ST_OK
MICR_ST_DEVICE_NOT_OPEN
MICR_ST_DEVICE_NOT_RESPONDING
MICR_ST_DEVICE_CONNECTION_ERROR
MICR_ST_REQUEST_TIMEDOUT
MICR_ST_CONNECT_REQUEST_TIMEDOUT
MICR_ST_ERR_INTERNET_CONNECT
MICR_ST_ERR_HTTP_OPEN_REQUEST
MICR_ST_ERR_HTTP_SEND_REQUEST
MICR_ST_NOT_ENOUGH_MEMORY
MICR_ST_BAD_DEVICE_NAME
MICR_ST_BAD_QUERY_PARM
MICR_ST_BAD_BUFFER
MICR_ST_BAD_BUFFER_LENGTH
MICR_ST_USB_GET_DATA_FAILED
MICR_ST_INET_GET_DATA_FAILED

Remarks

If the function succeeds MICR_ST_OK is returned.
If the device fails to respond, MICR_ST_DEVICE_NOT_RESPONDING is returned.
If the memory allocated for pcSectionData buffer that is not big enough to store the data of the inquiry section,
MICR_ST_NOT_ENOUGH_MEMORY is returned.
Error results from bad connection with device can be one of the following:

MICR_ST_CONNECT_REQUEST_TIMEDOUT
MICR_ST_REQUEST_TIMEDOUT
MICR_ST_DEVICE_NOT_RESPONDING
MICR_ST_ERR_INTERNET_CONNECT
MICR_ST_ERR_HTTP_OPEN_REQUEST
MICR_ST_ERR_HTTP_SEND_REQUEST
MICR_ST_USB_GET_DATA_FAILED
MICR_ST_INET_GET_DATA_FAILED

Example

```
#define BUFFER_LEN 4096
char pResult [BUFFER_LEN];
DWORD resultLength;
resultLength = BUFFER_LEN;
MTMICRQueryInfo(pcDeviceName, "DeviceCapabilities",pResult,&resultLength);
```

MTMICRSendCommand

MTMICRSendCommand function is used to send a single command to the device.

```
ULONG MTMICRSendCommand (
        char        *pcDevName,
        char        *pcCommand,
        char        *pcResult,
        DWORD       *pdwLength
);
```

Parameters

pcDevName
Pointer to null terminated string containing device name.

pcCommand
Pointer to null terminated string containing the command string.

pcResult
Pointer to the buffer that is used to store results returning from device.

pdwLength
Specifies the size of the pcResult buffer.

Return Values

MICR_ST_OK
MICR_ST_DEVICE_NOT_OPEN
MICR_ST_DEVICE_NOT_RESPONDING
MICR_ST_DEVICE_CONNECTION_ERROR
MICR_ST_REQUEST_TIMEDOUT
MICR_ST_CONNECT_REQUEST_TIMEDOUT
MICR_ST_ERR_INTERNET_CONNECT
MICR_ST_ERR_HTTP_OPEN_REQUEST
MICR_ST_ERR_HTTP_SEND_REQUEST
MICR_ST_NOT_ENOUGH_MEMORY
MICR_ST_ERR_CREATE_EVENT
MICR_ST_BAD_DEVICE_NAME
MICR_ST_BAD_PARAMETER
MICR_ST_INSUFFICIENT_DISKSPACE
MICR_ST_USB_GET_DATA_FAILED
MICR_ST_INET_GET_DATA_FAILED
MICR_ST_HTTP_HEADER_NOT_FOUND

Remarks

When sending command “ClearPathExit” or command “ClearPathEntry” to the device, the device is trying to clear the document path. If there is document in the path, the document is removed through the exit point if the command is “ClearPathExit” or the document is removed through the entry point if the command is “ClearPathEntry.”

Example

```
char szResult [4096];
char szDeviceName[4096];
char Buffer[256];
DWORD dwBufferLength ;
DWORD dwStatus;

szDeviceName = “Excella”; // assuming this device has been opened

dwBufferLength = 4096;

if (MTMICRSendCommand(szDeviceName, "ClearPathEntry", szResult , & dwBufferLength)== MICR_ST_OK)
{
        dwBufferLength = 256;
        dwStatus = MTMICRGetValue(szResult, SECTION_COMMAND_STATUS,"ReturnCode", Buffer,
                                                                                & dwBufferLength);
       if (dwStatus == MICR_ST_OK)
       {
              dwStatus = atol (Buffer);

              if (dwStatus == 0)
                     prinft ("Clear path done!");
              else
                     .printf ("Clear path failed!");

       }
}
```

MTMICRProcessCheck

MTMICRProcessCheck function sends a scan check request with the given process options to the given device name. When the device completes the scan request, the result of the scan is returned in the buffer pcDocInfo.

```
ULONG MTMICRProcessCheck (
        char        *pcDevName,
        char        *pcProcessOptions,
        char        *pcDocInfo,
        DWORD       *pdwDocInfoSize
);
```

Parameters

pcDevName

Pointer to null terminated string containing device name.

pcProcessOptions

Pointer to a buffer containing the options to be used in processing the check. The options are stored in the buffer by using function **MTMICRSetValue** or function **MTMICRSetIndexValue**.

pcDocInfo

Pointer to the buffer containing the information returned from the device. The returned information contains command status, MICR data, and image information.

pdwDocInfoSize

Size of the buffer pcDocInfo.

Return Values

MICR_ST_OK
MICR_ST_NOT_ENOUGH_MEMORY
MICR_ST_PROCESS_CHECK_FAILED
MICR_ST_BAD_DATA
MICR_ST_BAD_BUFFER
MICR_ST_BAD_BUFFER_LENGTH
MICR_ST_BAD_DEVICE_NAME
MICR_ST_DEVICE_NOT_OPEN
MICR_ST_DEVICE_NOT_RESPONDING
MICR_ST_REQUEST_TIMEDOUT
MICR_ST_CONNECT_REQUEST_TIMEDOUT
MICR_ST_ERR_INTERNET_CONNECT
MICR_ST_ERR_HTTP_OPEN_REQUEST
MICR_ST_ERR_HTTP_SEND_REQUEST
MICR_ST_INET_GET_DATA_FAILED
MICR_ST_USB_GET_DATA_FAILED
MICR_ST_HTTP_HEADER_NOT_FOUND

Remarks

If the function succeeds MICR_ST_OK is returned.
If there is no data returns from device, error MICR_ST_PROCESS_CHECK_FAILED is returned
If pcProcessOptions is NULL, error MICR_ST_BAD_DATA is returned
If pcDocInfo is NULL, error MICR_ST_BAD_BUFFER is returned
If the size of returned data is larger than the value specified in pdwDocInfoSize, error
MICR_ST_NOT_ENOUGH_MEMORY is returned
If the function fails to get HTTP header information, MICR_ST_HTTP_HEADER_NOT_FOUND is returned.
Error results from bad connection with device can be one of the following:
MICR_ST_CONNECT_REQUEST_TIMEDOUT
MICR_ST_REQUEST_TIMEDOUT
MICR_ST_DEVICE_NOT_RESPONDING
MICR_ST_ERR_INTERNET_CONNECT
MICR_ST_ERR_HTTP_OPEN_REQUEST
MICR_ST_ERR_HTTP_SEND_REQUEST
MICR_ST_USB_GET_DATA_FAILED
MICR_ST_INET_GET_DATA_FAILED

Example

```
#define BUFFER_LEN 4096

char docInfo [BUFFER_LEN];
char options [BUFFER_LEN];
char Device[4096] ="";
DWORD docInfoSize;;

// Set up options using function MTMICRSetValue or function MTMICRSetIndexValue
// Use function MTMICRGetDevice to get device name for variable “Device”

docInfoSize = BUFFER_LEN;
dwStatus=MTMICRProcessCheck(Device, options, docInfo, &docInfoSize);

// Use MTMICRGetValue and MTMICRGetIndexValue to parse the docInfo.
```

MTMICRGetImage

MTMICRGetImage function sends to the given device name a request for image data results from a previously scan.

```
ULONG MTMICRGetImage (
char        *pcDevName,
char        *pcImageID,
char        *pcBuffer,
DWORD       *pdwLength
);
```

Parameters

pcDevName
: Pointer to null terminated string containing device name.

pcImageID
: the identification of the requested image

pcBuffer
: Pointer to buffer for storing the image data.

pdwLength
: Specifies the size of the pcBuffer.

Return Values

MICR_ST_OK
MICR_ST_DEVICE_NOT_RESPONDING
MICR_ST_IMAGE_NOT_FOUND
MICR_ST_NOT_ENOUGH_MEMORY
MICR_ST_UNKOWN_ERROR
MICR_ST_BAD_BUFFER
MICR_ST_BAD_IMAGE_NAME
MICR_ST_BAD_DEVICE_NAME
MICR_ST_BAD_BUFFER_LENGTH
MICR_ST_DEVICE_NOT_OPEN
MICR_ST_CONNECT_REQUEST_TIMEDOUT
MICR_ST_REQUEST_TIMEDOUT
MICR_ST_DEVICE_NOT_RESPONDING
MICR_ST_ERR_INTERNET_CONNECT
MICR_ST_ERR_HTTP_OPEN_REQUEST
MICR_ST_ERR_HTTP_SEND_REQUEST
MICR_ST_USB_GET_DATA_FAILED
MICR_ST_INET_GET_DATA_FAILED

Remarks

If the function succeeds, MICR_ST_OK is returned.
If the requested image is not found, MICR_ST_IMAGE_NOT_FOUND is returned.
If the device fails to respond to the command, the return value is MICR_ST_DEVICE_NOT_RESPONDING.
If the size of the buffer uses to store the image is not large enough, MICR_ST_NOT_ENOUGH_MEMORY is returned.

Error results from bad connection with device can be one of the following:

MICR_ST_CONNECT_REQUEST_TIMEDOUT
MICR_ST_REQUEST_TIMEDOUT
MICR_ST_DEVICE_NOT_RESPONDING
MICR_ST_ERR_INTERNET_CONNECT
MICR_ST_ERR_HTTP_OPEN_REQUEST
MICR_ST_ERR_HTTP_SEND_REQUEST
MICR_ST_USB_GET_DATA_FAILED
MICR_ST_INET_GET_DATA_FAILED

Example

```
#define BUFFER_LEN 512
// DocInfo is returned by the MTMICRProcessCheck()

void GetNthImages (char *Device, unsigned int nIndex, char *DocInfo)
{
        char key [512];
        DWORD bufferSize;
        char cValue[BUFFER_LEN]="";
        DWORD nValueSize;

        // Get the image size from DocInfo
        nValueSize = BUFFER_LEN;
        MTMICRGetIndexValue (DocInfo, “ImageInfo”,”ImageSize”, nIndex,  cValue, & nValueSize);

        // Convert size to integer
        bufferSize = atol (cValue);

        // Get the Image Id from DocInfo
        nValueSize = BUFFER_LEN;
        MTMICRGetIndexValue (DocInfo, “ImageInfo”,”ImageSize, nIndex, cValue, & nValueSize);

        if (bufferSize)
        {
                // Allocate memory for image
                ImageBuffer = (char *) VirtualAlloc (NULL, bufferSize, MEM_COMMIT,
                                                                PAGE_READWRITE);


                // Use function MTMICRGetDevice to get device name for variable “Device”
                // Get the image from the device
                MTMICRGetImage (Device, cValue, ImageBuffer, &bufferSize);
                .
                .
                // Free the memory
                VirtualFree (ImageBuffer, bufferSize, MEM_DECOMMIT);
        }
}
```

MTMICRSetValue

MTMICRSetValue function adds a key/value pair to the given device settings specified in pcOptions buffer and in a given section specified in pcSection buffer.

ULONG MTMICRSetValue (

char	***pcOptions,**
char	***pcSection,**
char	***pcKey,**
char	***pcValue,**
DWORD	***pdwLength**

);

Parameters

pcOptions
Pointer to null terminated string containing all key/value pairs.
pcSection
Pointer to null terminated string containing the section name.
pcKey
Pointer to null terminated string containing the key name.
pcValue
Pointer to null terminated string containing the key value. If this is NULL, then the key pcKey is deleted from the section pcSection.
pdwLength:
Pointer to a double word that contains the size of the pcOptions buffer.

Return Values

MICR_ST_OK
MICR_ST_NOT_ENOUGH_MEMORY
MICR_ST_BAD_DATA
MICR_ST_BAD_SECTION_NAME
MICR_ST_BAD_BUFFER_LENGTH

Remarks

The functions return MICR_ST_NOT_ENOUGH_MEMORY, if the pcOptions buffer is not enough to add the new key/value pair. The required size of the buffer is returned in pdwLength.

The minimum size of the buffer should be equal to MTMICR_OPTIONS_BUFFER_SIZE.

The **MTMICRSetValue** function saves the new key/value pair in the pcOptions buffer only. This function does not send this key/value pair to the device. Use function **MTMICRProcessCheck** to send this key/value pair to the device.

If pcKey is NULL, the whole section pcSection will be removed from pcOptions string.
If pcValue is NULL and a key is specified, the specified key will be removed from the specified section
If pdwLength is less than the size of the results pcOptions after new key/value pair is added then
MICR_ST_NOT_ENOUGH_MEMORY is returned

Example

```
char Settings [4096];
DWORD SettingsBufferSize;

// Initialize Settings

// Initialize the Settings variable first
SettingsBufferSize =4096;

// Set a value of “4” to the key “Number” in section “ImageOptions”and store this new key/value pair in the buffer” Settings”
dwStatus=MTMICRSetValue(Settings, "ImageOptions","Number","4", & SettingsBufferSize);
```

MTMICRSetIndexValue

MTMICRSetIndexValue is similar to the **MTMICRSetVaue**, except that, **MTMICRSetIndexValue** adds the Index number to the pcKey before adding the key/value pair to the given Options buffer.

```
ULONG MTMICRSetIndexValue (
        char            *pcOptions,
        char            *pcSection,
        char            *pcKey,
        unsigned int    nIndex,
        char            *pcValue,
        DWORD           *pdwLength
);
```

Parameters

pcOptions
    Pointer to null terminated string containing all key/value pairs.
pcSection
    Pointer to null terminated string containing the section name.
pcKey
    Pointer to null terminated string containing the key name.
nIndex
    Index of the key.
pcValue
    Pointer to null terminated string containing the key value.
pdwLength:
    Pointer to a double word that contains the size of the pcOptions buffer.

Return Values

MICR_ST_OK
MICR_ST_NOT_ENOUGH_MEMORY
MICR_ST_BAD_PARAMETER
MICR_ST_BAD_DATA
MICR_ST_BAD_SECTION_NAME
MICR_ST_BAD_KEY_NAME
MICR_ST_BAD_VALUE_BUFFER
MICR_ST_BAD_BUFFER_LENGTH

Remarks

The function returns MICR_ST_NOT_ENOUGH_MEMORY, if the memory allocated for pcOptions buffer is not big enough to store the additional key/value pair. The required size for the buffer is returned in pdwLength;
The minimum size of the buffer should be equal to MTMICR_OPTIONS_BUFFER_SIZE.
The **MTMICRSetIndexValue** function saves the new key/value pair in the pcOptions buffer only. This function does not send the new key/value pair to the device. Use function **MTMICRProcessCheck** to send this key/value pair to the device.
If parameter **nIndex is less than zero or bigger than 4 then** MICR_ST_BAD_PARAMETER is returned.
If pdwLength is less than the length of the results pcOptions string after new key/value pair is added then
    MICR_ST_NOT_ENOUGH_MEMORY is returned.

Example

```
char Settings [4096];
DWORD SettingsBufferSize;

// Initialize Settings

// Intialize the Settings variable first
SettingsBufferSize =4096;

// The following command will set ImageColor1 = BW under the ImageOptions section.
dwStatus=MTMICRSetIndexValue(Settings, "ImageOptions","ImageColor",1, "BW", &SettingsBufferSize);
```

MTMICRGetValue

MTMICRGetValue function retrieves a key/value pair that was previously stored in the pcDocInfo parameter using **MTMICRSetValue** function .

```
ULONG MTMICRGetValue (
char        *pcDocInfo,
char        *pcSection,
char        *pcKey,
char        *pcValue,
DWORD       *pdwLength
);
```

Parameters

pcDocInfo
: Buffer pointer containing all the key/value pairs.

pcSection
: Pointer to null terminated string containing the section name.

pcKey
: Pointer to null terminated string containing the key name.

pcValue
: Pointer to the buffer that receives the retrieved string.

pdwLength
: Specifies the size of the pcValue buffer.

Return Values

MICR_ST_OK
MICR_ST_NOT_ENOUGH_MEMORY
MICR_ST_ERR_LOAD_XML
MICR_ST_ERR_GET_DOM_POINTER
MICR_ST_BAD_DATA
MICR_ST_BAD_SECTION_NAME
MICR_ST_BAD_KEY_NAME
MICR_ST_BAD_VALUE_BUFFER
MICR_ST_BAD_BUFFER_LENGTH
MICR_ST_KEY_NOT_FOUND

Remarks

MTMICRGetValue finds the key in the pcDocInfo buffer then returns its value in the pcValue.
If **MTMICRGetValue** succeeds it returns MICR_ST_OK.
If pdwLength is less than the size of the returned value then MICR_ST_NOT_ENOUGH_MEMORY is returned and the required size for the pcValue buffer is returned in the pdwLength.
If the key/value pair can not be found, then a NULL is returned for pcValue and error parameter MICR_ST_KEY_NOT_FOUND is returned.

Example

```
char Settings [4096];
char DocInfo [4096];
char device[4096] ="";
DWORD SettingsBufferSize;
DWORD DocInfoSize;
char cValue [1024];
DWORD valueSize;
DWORD dwStatus;

// Initialize Settings

DocInfoSize = 4096;

// Use function MTMICRGetDevice to get device name for variable “device”
// Call MTMICRProcessCheck function to process a document.
dwStatus = MTMICRProcessCheck (device, Settings, DocInfo, &DocInfoSize);

if (dwStatus == MTMICR_ST_OK)
{

        //Let us check the return status from the device
        valueSize = 1024;
        dwStatus=MTMICRGetValue(DocInfo, "CommandStatus", "ReturnCode", cValue, &valueSize);
        if (dwStatus != MICR_ST_OK)
                // error retrieving key value
        else
        {
                // do further process
        }
}
```

MTMICRGetIndexValue

MTMICRGetIndexValue function retrieves a key/value pair that was previously stored in the pcDocInfo parameter. **MTMICRGetIndexValue** function is similar to the function **MTMICRGetValue**. **MTMICRGetIndexValue** function adds index to the key name before searching for the value of the key name pcKey in the pcDocInfo.

```
ULONG MTMICRGetIndexValue (
char            *pcDocInfo,
char            * pcSection,
char            *pcKey,
unsigned int    nIndex,
char            *pcValue,
DWORD           *pdwLength
);
```

Parameters

pcDocInfo
: Buffer pointer containing all the key/value pairs.

pcSection
: Pointer to null terminated string containing the section name.

pcKey
: Pointer to null terminated string containing the key name.

nIndex
: Key Index Number

pcValue
: Pointer to the buffer that receives the retrieved value.

pdwLength
: Specifies the size of the buffer pointed to by the pcValue parameter.

Return Values

MICR_ST_OK
MICR_ST_NOT_ENOUGH_MEMORY
MICR_ST_ERR_LOAD_XML
MICR_ST_ERR_GET_DOM_POINTER
MICR_ST_BAD_DATA
MICR_ST_BAD_SECTION_NAME
MICR_ST_BAD_KEY_NAME
MICR_ST_BAD_VALUE_BUFFER
MICR_ST_BAD_BUFFER_LENGTH
MICR_ST_KEY_NOT_FOUND

Example

```
char Settings [4096];
char DocInfo [4096];
char device[4096] ="";
DWORD SettingsBufferSize;
DWORD DocInfoSize;
char cValue [1024];
DWORD valueSize;
DWORD dwStatus;

// Intialize Settings

DocInfoSize = 4096;

// Use function MTMICRGetDevice to get device name for variable “device”
// Call MTMICRProcessCheck function to process a document.
dwStatus = MTMICRProcessCheck (device, Settings, DocInfo, &DocInfoSize);

if (dwStatus == MTMICR_ST_OK)
{

        //Let us check the return status from the device
        valueSize = 1024;
        dwStatus=MTMICRGetValue(DocInfo, "CommandStatus", "ReturnCode", cValue, &valueSize);
        if (dwStatus != MICR_ST_OK)
                // error retrieving key value
        else
        {
                // Get the key ImageSize1 under ImageInfo section
                dwStatus=MTMICRGetIndexValue(DocInfo, "ImageInfo”, "ImageSize",1, cValue,
                                                                                &valueSize);
                if (dwStatus == MICR_ST_OK)
                {
                }
        }
}
```

SECTION 4. KEYS SENT TO DEVICE

Section 4 describes all the options to be sent by the application to the device for document processing. The application can send information the device using the following *ImageOptions* and *Endorser* sections.

The following keys are included in the *ImageOptions* section:

- Number
- ImageColor#
- Resolution#
- Compression#
- FileType#
- ImageSide#
- CalculateSHA1

The following keys are included in the *ProcessOptions* section:

- DocFeed
- ReadMICR
- MICRQuality

The following keys are included in the *Endorser* section:

- PrintData
- PrintFrontData
- PrintFont
- PrintFrontfont
- PrintFontSize
- PrintFrontFontSize
- BackXPosition
- FrontXPositon
- BackYPosition
- FrontYPosition

SECTION = ImageOptions

The *ImageOptions* section includes keys that control various options to process the check images. Some of the keys below include the symbol '#' to indicate a variable for the image number. The image number can be 1 or 2 (as specified in the *Number* key). Each image requires its own set of options controlled by the keys *ImageColor#, Resolution#, Compression#, FileType#, and ImageSide#*.

Number

This key determines how many images will be captured for each check that is processed. This key is supported by Excella, Excella STX and Excella MDX.

Values	Value Description
numeric value	Specifies number of images to be captured: 1, 2, 3, or 4.
0	If this number equals zero, no image is captured (supported by Excella STX only.)

ImageColor#

This key determines the image rendition for the specified image number. This key is supported by Excella, Excella STX and Excella MDX.

Values	Value Description
BW	Black and white (i.e., bitonal)
GRAY8	8-bit grayscale
COL24	24-bit color (supported by Excella STX only)

Resolution#

This key determines the image resolution for the specified image number. This key is supported by Excella, Excella STX and Excella MDX.

Values	Value Description
100X100	100x100 DPI (dots per inch)
200X200	200x200 DPI (dots per inch)

Compression#

This key determines the algorithm used to compress the captured images. This key is supported by Excella, Excella STX and Excella MDX.

Values	Value Description
GROUP4	Image will be compressed using Group4 compression type
JPEG	Image will be compressed using JPEG compression type
NONE	No compression

FileType#

This key determines the file type for the specified image number. This key is supported by Excella, Excella STX and Excella MDX.

Values	Value Description
TIF	Image format type is tiff
JPG	Image format type is jpg
BMP	Image format type is bmp

ImageSide#

This key determines which side of the check will be associated with specified image number. This key is supported by Excella, Excella STX and Excella MDX.

Values	Value Description
FRONT	Scan the front of image of the check.
BACK	Scan the back image of the check.

CalculateSHA1

This key determines if SHA1 calculation is required for captured images. This key is supported by Excella STX and Excella MDX.

Values	Value Description
YES	SHA1 calculation is required.
NO	SHA1 calculation is NOT required.

SECTION = ProcessOptions

The *ProcessOptions* section includes keys that control various options to process the MSR and MICR data.

DocFeed

This key determines the input tray to be used for document processing. This key is supported by Excella, Excella STX and Excella MDX.

Values	Value Description
MSR	Process Magstripe card from MSR. This value is valid for Excella MDX.

ReadMICR

This key selects the MICR font to be read on the check. This key is supported by Excella and Excella STX.

Values	Value Description
E13B	Decode only E13B character set.
CMC7	Decode only CMC7 character set.
ALL	Auto-detect and decode both E13B and CMC7.
NO	Suppress MICR reading

MICRQuality

This key adjust the quality threshold of MICR data. This key is supported by Excella MDX.

Values	Value Description
numeric string	Value range is from 20 to 95, default setting is 90

SECTION = Endorser

The *Endorser* section includes keys that control various options for printing on the check.

PrintData

This key specifies the text to be printed on the back of the check (i.e., endorsement message). This key is supported by Excella, Excella STX and Excella MDX.

Values	Value Description
string	Specifies text for the endorsement message

PrintFrontData

This key specifies the text to be printed on the front of the check (i.e., franking message). This key is supported by Excella STX and Excella MDX..

Values	Value Description
string	Specifies text for the franking message

PrintFont

This key determines the fonts to be used for the endorsement message. This key is supported by Excella, Excella STX and Excella MDX.

Values	Value Description
INTFONT1	Selects internal Font 1 (5x7 bitmap). Excella and STX only
INTFONT2	Selects internal Font 2 (7x10 bitmap) Excella and STX only
string	Microsoft Fonttype Name, default value “Times New Roman”, MDX only

PrintFrontFont

This key determines the fonts to be used for the franking message. This key is supported by Excella STX and Excella MDX.

Values	Value Description
INTFONT1	Selects internal Font 1 (5x7 bitmap).
INTFONT2	Selects internal Font 2 (7x10 bitmap)
string	Microsoft Fonttype Name, default value “Times New Roman”, MDX only

PrintFontSize

This key determines the font size of the selected font for the endorsement message. This key is supported by Excella MDX only.

Values	Value Description
numeric string	Font size, default setting is “30”

PrintFrontFontSize

This key determines the font size of the selected font for the franking message. This key is supported by Excella MDX only.

Values	Value Description
numeric string	Font size, default setting is “30”

BackXPosition

This key determines the X Position to be used for the endorsement message. This key is supported by Excella MDX only.

Values	Value Description
numeric string	Amount to set X position of endorsement message

FrontXPostion

This key determines the X Position to be used for the franking message. This key is supported by Excella MDX only.

Values	Value Description
numeric string	Amount to set X position of franking message

BackYPosition

This key determines the Y Position to be used for the endorsement message. This key is supported by Excella MDX only.

Values	Value Description
numeric string	Amount to set Y position of endorsement message

FrontYPosition

This key determines the Y Position to be used for the franking message. This key is supported by Excella MDX only.

Values	Value Description
numeric string	Amount to set Y position of franking message

SECTION 5. KEYS RECEIVED FROM DEVICE

Section 5 describes all the Sections (and their corresponding Key-Value pairs) automatically reported by the device to the application after the requested document has been processed. The device reports information using the following sections: *DocInfo*, *ImageInfo* and *MSRInfo*.

The following keys are included in the *DocInfo* section:

- DocUnits
- DocWidth
- DocHeight
- MICRFont
- MICRRaw
- MICRAcct
- MICRAmt
- MICRAux
- MICRBankNum
- MICRChkType
- MICRCountry
- MICRDecode
- MICREPC
- MICROnUs
- MICROut
- MICRSerNum
- MICRTPC
- MICRTransit
- MICRParseSts0
- MICRParseSts1

The following keys are included in the *ImageInfo* section:

- ImageSize#
- ImageURL#
- ImageSHA1Key#
- Number

The following keys are included in the *MSRInfo* section:

- MPData
- MPStatus
- TrackData1
- TrackData2
- TrackData3
- TrackData1Masked
- TrackData2Masked
- TrackData3Masked
- ReaderEncryptStatus
- DeviceSerialNumber
- EncryptedSessionID

SECTION = DocInfo

The *DocInfo* section includes keys that report on various attributes of the check document and information on the MICR data read from the check.

DocUnits

This key specifies the units of measurement used to report on check document dimensions (see the *DocWidth* and *DocHeight* keys below). This key is supported by Excella, Excella STX and Excella MDX.

Values	Value Description
ENGLISH	Dimensions in thousands of an inch
METRIC	Dimensions in millimeters

DocWidth

This key specifies the width of the check document based on the scanned image. This key is supported by Excella, Excella STX and Excella MDX.

Values	Value Description
numeric value	Width dimension of scanned document

DocHeight

This key specifies the height of the check document based on the scanned image. This key is supported by Excella, Excella STX and Excella MDX.

Values	Value Description
numeric value	Height dimension of scanned document.

MICRFont

This key specifies the MICR font detected and read from the check. This key is supported by Excella, Excella STX and Excella MDX.

Values	Value Description
E13B	E13B font was detected on the check read
CMC7	CMC7 font was detected on the check read

Note

The parsed MICR fields below only apply to U.S. checks with E13B font. CMC7 Checks are not parsed, and only the raw MICR line is reported.

MICRRaw

This key specifies the unformatted (as-is) MICR data read from the check. This key is supported by Excella, Excella STX and Excella MDX.

Values	Value Description
string	Unformatted MICR data

MICRAcct

This key specifies the account number MICR field read from the check. This key is supported by Excella, Excella STX and Excella MDX.

Values	Value Description
string	Account number MICR field

MICRAmt

This key specifies the amount MICR field read from the check. This key is supported by Excella, Excella STX and Excella MDX.

Values	Value Description
string	Amount MICR field

MICRAux

This key specifies the Auxiliary On-Us MICR field read from the check. This key is supported by Excella, Excella STX and Excella MDX.

Values	Value Description
string	Auxiliary On-Us MICR field

MICRBankNum

This key specifies the 4-digit ABA Identifier MICR field read from the check. This key is supported by Excella, Excella STX and Excella MDX.

Values	Value Description
string	ABA Institution Identifier (bank number) MICR field

MICRChkType

This key specifies the check type based on the presence on the Auxiliary On-Us MICR field. This key is supported by Excella, Excella STX and Excella MDX.

Values	Value Description
PERSONAL	Auxiliary On-Us MICR field NOT present
BUSINESS	Auxiliary On-Us MICR field present

MICRCountry

This key specifies the country of origin based on known MICR line formats. This key is supported by Excella, Excella STX and Excella MDX.

Values	Value Description
USA	Check drawn in USA
CANADIAN	Check drawn in Canada
MEXICAN	Check drawn in Mexico

MICRDecode

This key indicates whether a MICR decode/read error was detected. This key is supported by Excella, Excella STX and Excella MDX.

Values	Value Description
NONE	No MICR data detected
OK	MICR decode/read was successful
ERROR	Error detected in MICR decode/read operation

MICREPC

This key specifies the EPC MICR field read from the check. This key is supported by Excella, Excella STX and Excella MDX.

Values	Value Description
string	EPC MICR field

MICROnUs

This key specifies the On-Us MICR field read from the check. This key is supported by Excella, Excella STX and Excella MDX.

Values	Value Description
string	On-Us MICR field

MICROut

This key specifies the formatted MICR output data read from the check as defined by the *MICRFmtCode* key in Section=*ProcessOptions*. This key is supported by Excella, Excella STX and Excella MDX.

Values	Value Description
string	Formatted MICR output data

MICRSerNum

This key specifies the sequence number MICR field read from the check. This key is supported by Excella, Excella STX and Excella MDX.

Values	Value Description
string	Sequence number (or check number) MICR field

MICRTPC

This key specifies the TPC MICR field read from the check. This key is supported by Excella, Excella STX and Excella MDX.

Values	Value Description
string	TPC MICR field

MICRTransit

This key specifies the 9-digit Transit MICR field read from the check. This key is supported by Excella, Excella STX and Excella MDX.

Values	Value Description
string	Transit MICR field

MICRParseSts0

This key specifies a 4-digit status/error code reported by the device after parsing the MICR fields read from the check. This key is supported by Excella, Excella STX and Excella MDX.

Values	Value Description
numeric value	For specific status/error code information refer to table 6-9

Table 5-1. MICRPARSESTS0

Digit	
1st Digit	0 – OK MICR
	1 – Low MICR
	2 – No MICR
2nd Digit	0 – Standard Check
	1 – Business Check
	2 – Mexican Check
	3 – Canadian Check
3rd Digit	0 – No Status
	1 – Amount Present
	2 – Short Account
	3 – Short Account + Amount Present
	4 – No Check#
	5 – No Check# + Amount Present
	6 – No Check# + Short Account
	7 – No Check# + Short Account + Amount Present
4th Digit	0 – No Errors
	1 – Chk#
	2 – Account
	3 – Account + Chk#
	4 – Transit
	5 – Transit + Chk#
	6 – Transit + Account
	7 – Transit + Account + Chk#

MICRParseSts1

This key specifies a 2-digit status/error code reported by the device after parsing the MICR fields read from the check. This key is supported by Excella, Excella STX and Excella MDX.

Values	Value Description
numeric value	For specific status/error code information refer to table 6-10

Table 5-2. MICRPARSESTS1

PRIORITY	MICRParseSts1	TYPE	DESCRIPTION
10	01	Error	No MICR data: no transit and no account found
9	09	Status	Mexican check
8	08	Status	Canadian check
7	05	Error	No transit, bad character, bad length, bad check digit
6	07	Error	No account, bad character
5	04	Error	Bad character in check number
5	04	Status	No check number
4	12	Status	Short Account (maybe caused by mis-parsed check#)
3	03	Status	Low MICR signal, good read
2	10	Status	Business check
1	11	Status	Amount field present
0	00	Status	No error, check OK

Notes:

- The LED indicator will turn red on all error conditions.
- The absence of a check number is not considered an error.
- If a multiple error occurs, the error or status code with the highest priority is reported.
- All unreadable MICR characters are transmitted as an “?” ASCII character (hex 3F), except for Format 00xx.

SECTION = ImageInfo

The *ImageInfo* section includes keys that report on various attributes of the scanned images. Some of the keys below include the symbol '#' to indicate a variable for the image number. The image number can be 1 or 2 (up to 2 images as specified in the *Number* key below). The device reports image information using the keys *ImageSize#*, *ImageURL#*, and *ImageSHA1#*.

ImageSize#

This key specifies the size (in bytes) for the specified Image number. The number of scanned images and their availability is determined by the *Number* key in Section=*ImageOptions*. This key is supported by Excella, Excella STX and Excella MDX.

Values	Value Description
numeric value	Number of bytes

ImageURL#

This key specifies the URL string needed to request the specified Image number using the **MTMICRGetImage** API function. This key is supported by Excella, Excella STX and Excella MDX.

Values	Value Description
string	URL string to indicate internal file path for the Image.

ImageSHA1Key#

This key contains the SHA1 key calculation for the specified Image number (if requested and available). This key is supported by Excella STX and Excella MDX.

Values	Value Description
string	SHA1 Key string for the image.

Number

This key specifies the number of scanned images ready and available from the device. This key is supported by Excella, Excella STX and Excella MDX.

SECTION = MSRInfo

The *MSRInfo* section includes keys that report on the information captured from the magnetic stripe card. This section is supported by Excella STX and Excella MDX.

MPData

This key specifies the MagnePrint data read from the magnetic stripe card. This key is supported by Excella STX and Excella MDX.

Values	Value Description
String	Please contact MagTek for the use of MagnePrint data

MPStatus

This key specifies the MagnePrint status reported by the device. This key is supported by Excella STX and Excella MDX.

Values	Value Description
String	Please contact MagTek for the use of MagnePrint data

TrackData1

This key specifies the Track 1 data read from the magnetic stripe card. This key is supported by Excella STX and Excella MDX.

Values	Value Description
String	Track 1 data on magnetic stripe card

TrackData2

This key specifies the Track 2 data read from the magnetic stripe card. This key is supported by Excella STX and Excella MDX.

Values	Value Description
String	Track 2 data on magnetic stripe card

TrackData3

This key specifies the Track 3 data read from the magnetic stripe card. This key is supported by Excella STX and Excella MDX.

Values	Value Description
String	Track 3 data on magnetic stripe card

TrackData1Masked

This key specifies the Masked Track 1 data read from the magnetic stripe card. The card holder's name and the expiration date are unmasked; all other characters are masked. This key is supported by Excella STX and Excella MDX.

Values	Value Description
String	Masked Track 1 data on magnetic stripe card

TrackData2Masked

This key specifies the Masked Track 2 data read from the magnetic stripe card. This key is supported by Excella STX and Excella MDX.

Values	Value Description
String	Masked Track 2 data on magnetic stripe card

TrackData3Masked

This key specifies the Masked Track 3 data read from the magnetic stripe card. This key is supported by Excella STX and Excella MDX.

Values	Value Description
String	Masked Track 3 data on magnetic stripe card

ReaderEncryptStatus

This key specifies the reader encryption status reported by the device. The two byte binary field contains information on DUKPT key exhaust status, initial DUKPT key, Encryption status, Authentication, and timed out error during swipe. This key is supported by Excella STX and Excella MDX.

Values	Value Description
String	Please see below for the formatting of this string

Bit 0 = DUKPT Keys exhausted
Bit 1 = Initial DUKPT key Injected
Bit 2 = Encryption Enabled
Bit 3 = Authentication Required
Bit 4 = Timed Out waiting for user to swipe card
Bits 5–15 = Unassigned (always set to Zero)

DeviceSerialNumber

This key contains the device serial number. This key is supported by Excella STX and Excella MDX.

Values	Value Description
string	Device Serial Number

EncryptedSessionID

This key contains the encrypted version of the current Session ID. This key is supported by Excella STX and Excella MDX.

Values	Value Description
string	Encrypted current Session ID

SECTION 6. OTHER KEYS AVAILABLE FROM DEVICE

Section 6 describes other Sections (and their corresponding key-value pairs) available from the device upon query by the application.

The following keys are included in the *DeviceStatus* section:

- State

The following keys are included in the *DeviceCapabilities* section:

- Firmware
- MachineType

SECTION = DeviceStatus

The *DeviceStatus* section includes keys that report general operational status of the device.

State

This key specifies the general status of the device. This key is supported by Excella and Excella STX.

Values	Value Description
ONLINE	Device is connected and online; device is ready to process check or msr.
CHECKREADY	Device send check image data to the host.
MSRREADY	Device send MSR data to the host.

SECTION = DeviceCapabilities

The *DeviceCapabilities* section includes keys that report on the general capabilities of the device.

Firmware

This key specifies the version of the firmware installed in the device. This key is supported by Excella, Excella STX and Excella MDX.

Values	Value Description
string	Firmware version

MachineType

This key indicates the type of Excella device connected. This key is supported by Excella, Excella STX and Excella MDX.

Values	Value Description
EXCELLA	The device connected is Excella
EXCELLASTX	The device connected is Excella STX
MDX	The device connected is Excella MDX