MAGTEK®
SECURITY FROM THE INSIDE

DynaFlex, DynaProx

For MMS Devices

OID Converter Console Manual

January 2023

Manual Part Number:
D998200557-10

REGISTERED TO ISO 9001:2015

Table 0.1 – Revisions

Rev Number	Date	Notes
10	January 9, 2023	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 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.

DEMO SOFTWARE / SAMPLE CODE: Unless otherwise stated, all demo software and sample code are to be used by Licensee for demonstration purposes only and MAY NOT BE incorporated into any production or live environment. The PIN Pad sample implementation is for software PIN Pad test purposes only and is not PCI compliant. To meet PCI compliance in production or live environments, a third-party PCI compliant component (hardware or software-based) must be used.

Table of Contents

1 Introduction

This document provides instructions to use the OID Converter Console. It is part of a larger library of documents designed to assist Secure Card Readers implementers, which includes the following documents available from MagTek:

- ***D998200380 MAGTEK UNIVERSAL SDK PROGRAMMER'S MANUAL (MICROSOFT .NET)***
- ***D998200383 DynaFlex Family Programmer's Manual (COMMANDS)***

2 How to use the OID Converter Console Demo

The following instructions are for using the DynaFlex, DynaProx OID Converter Console on a Windows operating system.

The console app accepts 2 input parameters: OID & Command Data. The result is the data to pass to MTUSDK commands `getConfigInfo()` and `setConfigInfo()`.

2.1 Get Configuration

When getting a configuration, only input the OID.

1) Launch a command terminal where the `oidconverter.exe` is located.
2) Type `oidconverter` and an OID in decimal format as shown below, then ENTER.
   The OID may also be typed in hexadecimal as commented out below.

```
c:\temp\oidconverter 2.1.2.2.2.1


//When using hexadecimal OID
//c:\temp\oidconverter 020102020201
```

3) The console app displays the Input OID and Converted OID as shown below.

```
C:\temp>oidconverter 2.1.2.2.2.1
Input OID: 2.1.2.2.2.1
Converted OID string: E108E206E204E202C100
```

4) Use the Converted OID when getting a property value with the MTUSDK API `getConfigInfo()`.

```
Input OID: 2.1.2.2.2.1
Converted OID string: E108E206E204E202C100
```

5) When calling the function `getConfigInfo`, parameter `configType` is the first number of the OID. `2.1.2.2.2.1`

```
byte[] getConfigInfo(
           byte configType,
           byte[] data);
```

```
byte configType = 0x02;
byte[] data = {0xE1,0x08,0xE2,0x06,0xE2,0x04,0xE2,0x02,0xC1,0x00};
byte[] response = devConfig.getConfigInfo(configType, data);
```

2.2 Set Configuration

When setting a configuration, input the OID and the Command Data.

1) Launch a command terminal where the `oidconverter.exe` is located.
2) Type `oidconverter`, an OID, in decimal format, and a Value in hexadecimal as shown below, then ENTER.

   The OID may also be typed in hexadecimal as commented out below.

```
C:\temp>oidconverter 1.2.3.1.1.1 03000000

//When typed in hexadecimal.
//C:\temp>oidconverter 010203010101 03000000
```

3) The console app displays the Input OID and Converted OID as shown below.

```
C:\temp>oidconverter 1.2.3.1.1.1 03000000
Input OID: 1.2.3.1.1.1
Input Command Data: 03000000
Converted OID string: E20CE30AE108E106C10403000000
```

4) Use the Converted OID when setting a property value with the MTUSDK API `setConfigInfo()`.

```
Input OID: 1.2.3.1.1.1
Input Command Data: 03000000
Converted OID string: E20CE30AE108E106C10403000000
```

5) When calling the function `getConfigInfo`, parameter `configType` is the first number of the OID. `1.2.3.1.1.1`

```
byte[] setConfigInfo(
           byte configType,
           byte[] data,
           IConfigurationCallback callback);
```

```
byte configType = 0x01;
byte[] data = {0xE2,0x0C,0xE3,0x0A,0Xe1,0x08,0xE1,
               0x06,0xC1,0x04,0x03,0x00,0x00,0x00};
int response = devConfig.setConfigInfo(configType, data, this);
```

3 OID Conversion Method

This is an example of how to manually convert a property OID in decimal format into a constructed TLV command data.

1) ConfigType is the first number (node 6) to the left of the OID.

```
    [2] . 1 .  2 .  2 .  2 .  1
node(6)                  node(1)
```

2) To construct the remaining nodes, logical OR as follows.

```
OR node 1 with C0
OR nodes 2 - 5 with E0
```

Result:

```
    1 .  2 .  2 .  2 .  1
OR E0   E0   E0   E0   C0
 = E1   E2   E2   E2   C1
```

3) Add Lengths to each node to include all lower nodes and data.

```
->08 ->06 ->04 ->02 ->00
```

4) The fully Constructed TLV command data:

```
E108 E206 E204 E202 C100 (E108E206E204E202C100)
```

5) Use the configType and command data for the SDK APIs.

```
byte configType = 02;
byte[] data = {0xE1,0x08,0xE2,0x06,0xE2,0x04,0xE2,0x02,0xC1,0x00};
byte[] response = devConfig.getConfigInfo(configType, data);
```

Example Get: Device Model Name

```
[2] . 1  .  2  .  2  .  2  .  1
CT    En    En    En    En    Cn
CT    EnLL->EnLL->EnLL->EnLL->Cn00
02    E108->E206->E204->E202->C100

ConfigType = 02
Command data = E108E206E204E202C100
```

Example Set: User Event Notifications Enable

```
[1] . 2  .  3  .  1  .  1  .  1  = 03000000
CT    En    En    En    En    Cn = 03000000
CT    EnLL->EnLL->EnLL->EnLL->CnLLValue
01    E20C->E30A->E108->E106->C10403000000

ConfigType = 01
Command data = E20CE30AE108E106C10403000000
```

Notation	Description
CT	Configuration type
En	Converted tag for a node
LL	TLV Length for that node
->	All TLVs to the right
Cn	Converted tag for the first node