



beyond  
payment

**UPOS**

## **Developer's Guide**

**Telium Devices**

UPOS Developer's Guide  
Part Number DIV350825 Rev. C  
Released October, 2011.  
Copyright © 2011, Ingenico Corp. All rights reserved.

Ingenico Inc.	Ingenico Canada Ltd.
6195 Shiloh Road, Suite D	79 Torbarrie Road, Toronto, Ontario
Alpharetta, GA 30005	Canada M3L 1G5
Tel: 678.456.1200	Tel: 416.245.6700
Fax: 678.456.1201	Fax: 416.245.6701
<a href="http://www.ingenico-us.com">www.ingenico-us.com</a>	<a href="http://www.ingenico-us.com">www.ingenico-us.com</a>

#### **North American Customer Support**

Tel: 888.900.8221  
Fax: 905.795.9343  
Email: [customersfirst.us@ingenico.com](mailto:customersfirst.us@ingenico.com)

#### **Customer Service Centers:**

In the U.S.A.	Canada
6195 Shiloh Road, Suite D	6520 Gottardo Court
Alpharetta, GA 30005	Mississauga, Ontario, L5T 2A2

No part of this publication may be copied, distributed, stored in a retrieval system, translated into any human or computer language, transmitted, in any form or by any means, without the prior written consent of Ingenico. Ingenico and the Ingenico logo are registered trademarks of Ingenico Corp. All other brand names and trademarks appearing in this guide are the property of their respective holders.

Information in this document is subject to change without notice.

The information contained herein is considered an intellectual property of Ingenico and as such should be treated as confidential information to be reviewed only by authorized employees covered under the executed Mutual Non-Disclosure signed between our companies. Ingenico Corp © 2011. All rights reserved.

# Contents

---

<b>Contents .....</b>	<b>i</b>
<b>Revision History .....</b>	<b>7</b>
<b>1. Introduction to UPOS .....</b>	<b>9</b>
1.1. About This Manual.....	9
1.2. Definitions.....	10
1.3. UPOS Device Drivers .....	10
1.4. Compatible Devices.....	10
1.5. Connecting and Powering the PIN Pad .....	11
<b>2. Getting Started with the JPOS Integration Kit .....</b>	<b>13</b>
2.1. JPOS Overview .....	13
2.2. JPOS Integration Kit Contents (SD40-01.00) .....	13
2.3. Installation .....	13
2.3.1. Installing JRE.....	13
2.3.2. Installing JavaCOMM API on Windows .....	13
2.3.3. Installing JavaCOMM within JRE .....	14
2.3.4. Installing JavaCOMM Outside of JRE .....	14
2.3.5. Installing the Java XML Parser.....	14
2.3.6. Installing Apache Log4J Logging API .....	14
2.3.7. Troubleshooting Installation Issues .....	15
2.4. JavaPOS Configuration .....	15
2.5. JPOS Entry Editor .....	16
2.5.1. Setting the Java Path .....	16
2.5.2. Editing Standard Properties with the JPOS Entry Editor .....	17
2.5.3. Editing Bus Properties with the JPOS Entry Editor.....	17
2.5.4. Adding Vendor Properties with the JPOS Entry Editor .....	17
2.5.5. Editing Vendor Properties with the JPOS Entry Editor.....	18
2.5.6. Removing Vendor Properties with the JPOS Entry Editor .....	18
<b>3. Getting Started with the OPOS Integration Kit.....</b>	<b>19</b>
3.1. OPOS Overview .....	19
3.2. OPOS Integration Kit Contents (SD36-2.00).....	20
3.3. Installation .....	22
3.3.1. Typical Install .....	22
3.3.2. Silent Installation.....	24
3.4. OPOS Configuration .....	26
3.5. Uninstalling the Software .....	37

3.5.1. Typical Uninstall.....	37
3.5.2. Silent Uninstall.....	39
3.6. Forms .....	39
3.7. Using the OPOS Controls .....	39
3.8. Setting Up the System Registry.....	40
<b>4. Direct I/O.....</b>	<b>43</b>
4.1. The DirectIO Method.....	43
4.1.1. Get Health Statistics .....	44
4.1.2. Set UIA Variable .....	47
4.1.3. Get UIA Variable .....	49
4.1.4. Configure PIN Pad Lights .....	50
4.1.5. Reboot the PIN Pad.....	51
4.1.6. Transmit Check Box Data.....	52
4.1.7. Transmit Survey Data .....	52
4.1.8. Clear Line Display Element.....	52
4.1.9. Clear Screen .....	53
4.1.10. Display Text At.....	53
4.1.11. Position Text Cursor .....	53
4.1.12. Get Last Error.....	54
4.1.13. Save File .....	54
4.1.14. Run File.....	54
4.1.15. Retrieve File .....	55
4.1.16. File Status.....	55
4.2. Best Practices .....	55
4.3. Usage Samples .....	55
4.3.1. Store a Form .....	55
4.3.2. Display a Form .....	56
<b>5. Line Display.....</b>	<b>57</b>
5.1. General Information .....	57
5.2. Usage Samples .....	57
5.2.1. Clear a Window.....	57
5.2.2. Write Text .....	57
<b>6. MSR.....</b>	<b>59</b>
6.1. General Information.....	59
6.2. Usage Samples .....	59
6.2.1. Read All Tracks of Card Data.....	59
6.2.2. SetDeviceEnabled Error Reporting .....	59
6.2.3. Request Sentinels.....	59
6.2.4. Parse MSR Data .....	60

<b>7. Signature Capture .....</b>	<b>61</b>
7.1. General Information .....	61
7.2. Usage Samples .....	61
7.2.1. Define the Signature Format .....	61
7.2.2. Define the OPOS Conversion Formats .....	61
7.2.3. Handle Signature Capture .....	61
<b>8. SigDisplay Control.....</b>	<b>63</b>
8.1. General Information .....	63
8.2. Summary .....	63
8.2.1. Properties .....	63
8.2.2. Methods.....	63
8.3. Properties .....	64
8.3.1. GetDrawBorder .....	64
8.3.2. SetDrawBorder .....	64
8.3.3. GetDrawBackground.....	64
8.3.4. SetDrawBackground .....	64
8.3.5. GetPenWidth .....	65
8.3.6. SetPenWidth.....	65
8.3.7. GetDisplayNumPoints .....	65
8.3.8. SetDisplayNumPoints.....	65
8.4. Methods.....	66
8.4.1. SetOPOSBCNIBBLESignatureData.....	66
8.4.2. SetOPOSBCNIBBLESignatureDataX .....	66
8.4.3. SetSignatureData .....	67
8.4.4. SetSignatureDataX .....	67
8.4.5. GetSignatureType.....	68
8.4.6. GetSignatureTypeString.....	68
8.4.7. WriteSignatureToFile .....	68
8.4.8. ConvertSignatureToImageBuffer .....	68
<b>9. PIN Pad.....</b>	<b>71</b>
9.1. General Information .....	71
9.2. Encryption Key Formats.....	71
9.2.1. Master/Session Key Serial Number Format .....	71
9.2.2. DUKPT Key Serial Number Format .....	71
9.3. Usage Samples .....	71
9.3.1. Handle PIN Entry .....	71
9.3.2. Request Key ID or Key Slot Information .....	72
9.3.3. Retrieve PIN Block.....	73
<b>10. OPOS Usage Samples.....</b>	<b>75</b>

10.1.	Implementing a Terms and Conditions Screen .....	75
10.2.	Implementing a Survey Screen .....	75
10.3.	Simulating a Transaction .....	75
10.4.	Using OPOS for the .NET Application .....	75
10.4.1.	PosExplorer .....	75
10.4.2.	.NET Interop .....	76
<b>11.</b>	<b>UIA Prompts .....</b>	<b>77</b>
11.1.	Prompts (PROMPTS.xml) .....	77
11.2.	Prompts (CUSTPROMPTS.xml) .....	81
<b>Appendix A.</b>	<b>Differences Between U32 OPOS and Telium OPOS .....</b>	<b>83</b>
A.1.	At A Glance .....	83
A.2.	OPOS Control Support .....	83
A.2.1.	Form Control .....	83
A.2.2.	Direct I/O .....	84
A.3.	OPOS Control Panel .....	84
A.3.1.	Installation .....	84
A.3.2.	General Tab .....	84
A.3.3.	PIN Pad Tab .....	84
A.3.4.	MSR Tab .....	85
A.3.5.	Signature Tab .....	85
A.3.6.	Line Display Tab .....	85
A.4.	OPOS Test Application .....	85
<b>Appendix B.</b>	<b>U.S. Retail Telium Download Application (TDA) .....</b>	<b>87</b>
B.1.	Accessing the Telium Download Application .....	87
B.2.	Configuring PIN Pad Communication Settings .....	88
B.2.1.	Setting Communication Type .....	88
B.2.2.	Configuring Serial Connection Settings .....	88
B.2.3.	Configuring Ethernet Connection Settings .....	90
B.2.4.	Configuring Tailgate Connection Settings .....	93
B.3.	Exiting the Telium Download Application .....	93
<b>Appendix C.</b>	<b>5992 Mode .....</b>	<b>95</b>
C.1.	Special Functions .....	95
C.2.	Line Displays .....	95
C.3.	Fonts .....	95
C.4.	Required Scripting .....	96
C.5.	5992 Mode Font Tables .....	96
C.5.1.	Font Types .....	96
C.5.2.	Point Size Values .....	96
C.5.3.	Examples .....	97

<b>Appendix D. JPOS Test Application.....</b>	<b>99</b>
D.1. Getting Started .....	99
D.2. Updating JPOS Configuration.....	99
D.3. The JavaPOS Test Application Interface .....	100
D.4. DirectIO .....	101
D.5. Running the JPOS Test App .....	102
<b>Appendix E. OPOS Test Application.....</b>	<b>103</b>
E.1. Installing the OPOS Test Application .....	104
E.2. Application Interface.....	108
E.2.1. Device Name Tab.....	108
E.2.2. Line Display Tab .....	109
E.2.3. MSR Tab .....	110
E.2.4. PIN Pad Tab.....	111
E.2.5. Sig Cap Tab.....	112
E.2.6. Device Info Tab .....	113
E.2.7. Direct I/O .....	113
E.3. Working With the Test Application .....	115
E.3.1. Opening, Claiming and Enabling a Control.....	115
E.3.2. Changing Variable Text With The Test Application .....	115
E.3.3. Configuring MSR LEDs .....	116
<b>Appendix F. Telium PIN Pad USB Settings .....</b>	<b>119</b>
F.1. For HID Communication.....	119
F.2. For CDC Communication .....	119

## Notes



# Revision History

Manual Revision	Application Revision	Changes
C	OPOS v1.4.1.7 JPOS v1.13.1 TDA 1.4.3	<p>4.1 – The DirectIO Method:</p> <ul style="list-style-type: none"> <li>Added new Direct I/O commands.</li> <li>Added Direct I/O result codes with labels and definitions.</li> </ul> <p>4.1.4 – Configure :</p> <ul style="list-style-type: none"> <li>Added new section.</li> </ul> <p>4.1.12 – Get Last Error:</p> <ul style="list-style-type: none"> <li>Added new section.</li> </ul> <p>B.2 – Configuring PIN Pad Communication Settings:</p> <ul style="list-style-type: none"> <li>Updated with new TDA menu structure.</li> </ul> <p>E.2.7 – Direct I/O:</p> <ul style="list-style-type: none"> <li>Added new section.</li> </ul> <p>E.3.3 – Configuring MSR LEDs:</p> <ul style="list-style-type: none"> <li>Added new section.</li> </ul>
B	Various	<p>3.4 – OPOS Configuration:</p> <ul style="list-style-type: none"> <li>Updated images.</li> <li>Added and updated configuration steps.</li> </ul> <p>4.1 – The DirectIO Method:</p> <ul style="list-style-type: none"> <li>Added new Direct I/O commands.</li> <li>Added Direct I/O result codes with labels and definitions.</li> </ul> <p>4.1.2 – Set UIA Variable:</p> <ul style="list-style-type: none"> <li>Added new variable.</li> </ul> <p>4.1.10 – Display Text At :</p> <ul style="list-style-type: none"> <li>Added new section.</li> </ul> <p>4.1.15 – Retrieve File:</p> <ul style="list-style-type: none"> <li>Added new section.</li> </ul> <p>4.1.16 – File Status:</p> <ul style="list-style-type: none"> <li>Added new section.</li> </ul> <p>Appendix C - 5992 Mode:</p> <ul style="list-style-type: none"> <li>Added new appendix.</li> </ul> <p>D.2. – Application Interface:</p> <ul style="list-style-type: none"> <li>Updated images.</li> <li>Updated tables.</li> </ul> <p>D.3.2 – Changing Variable Text With The Test Application:</p> <ul style="list-style-type: none"> <li>Added new section.</li> </ul>
A	Various	Initial release.

## Notes

# 1. Introduction to UPOS

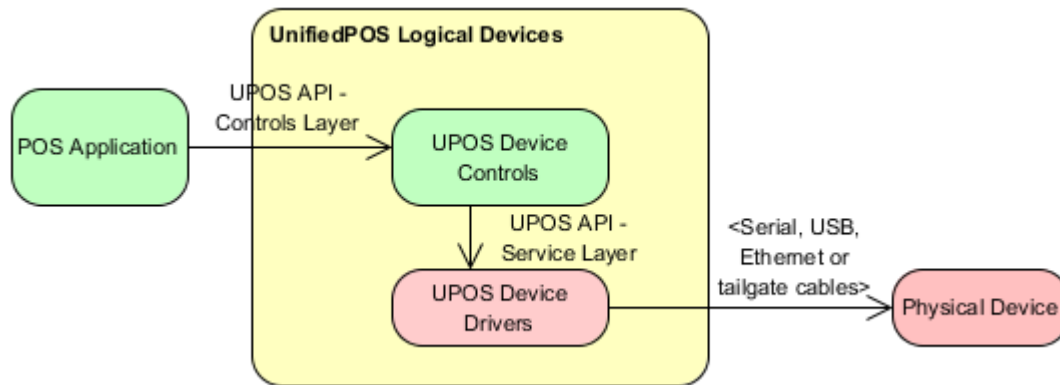
---

## 1.1. About This Manual

---

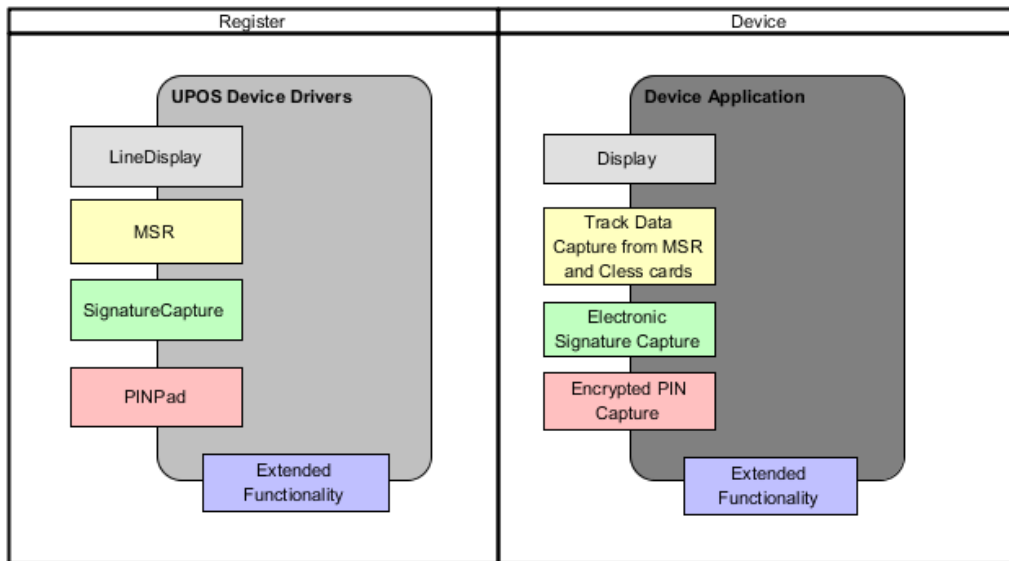
This document covers Ingenico's UPOS-based payment solutions for Telium devices.

There are two main components for each Ingenico UPOS solution: a PIN-pad-based UPOS application (Ingenico's UIA application in most cases) and POS-based UPOS drivers. The POS application calls on the standard UPOS Device Control Layer to utilize functionality implemented in Ingenico UPOS Device Drivers.



**Figure 1 - UPOS Architecture**

All of Ingenico's UPOS-based solutions follow distributed application architecture. POS-based UPOS device drivers and UIA can be used to implement all the capabilities made available by logical device controls as defined by UPOS specification.



**Figure 2 - Ingenico UPOS distributed application.**

## 1.2. Definitions

Term	Definition
JCL	Java Configuration Loader
JPOS	Java for Point of Service
OPOS	OLE for Retail Point of Service
UIA	UPOS Interface Application
UPOS	Unified Point of Service

## 1.3. UPOS Device Drivers

Ingenico UPOS device drivers for Telium 2 devices implement the following UPOS Device Controls:

- LineDisplay
- MSR – Magnetic Stripe Reader
- SignatureCapture
- PINPad

## 1.4. Compatible Devices

Ingenico's UPOS-based solutions are compatible with the following Ingenico PIN pad models:

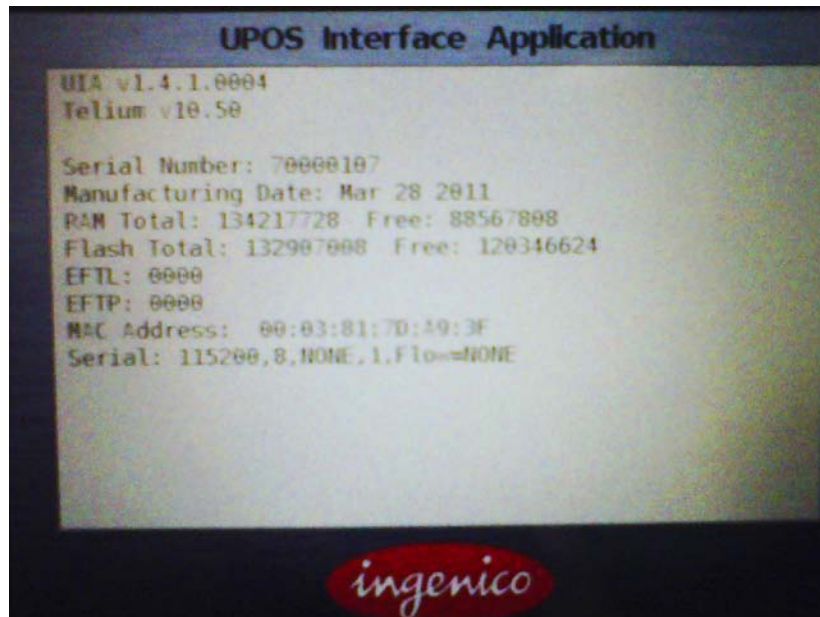
- iSC250
- iSC350

- iPP320
- iPP350

## 1.5. Connecting and Powering the PIN Pad

---

Figure 3 below shows a sample UPOS Interface Application splash screen. For any UPOS-based POS applications to interface with the PIN pad correctly, the UPOS Interface Application (UIA) must be present on your Telium PIN pad. Your version and memory information may differ from what is shown.



*Figure 3 - UPOS interface application splash screen.*

## Notes

## 2. Getting Started with the JPOS Integration Kit

---

### 2.1. JPOS Overview

---

The Ingenico JavaPOS Integration Kit lets developers create Java-based POS applications that interface with UPOS controls in order to drive Ingenico PIN pad functionality.

### 2.2. JPOS Integration Kit Contents (SD40-01.00)

---

Ingenico's JPOS Integration Kit contains the following:

- Ingenico documentation
- JavaPOS drivers
- Telium utilities
- Telium UIA terminal application
- Telium UIA data samples and configuration files

### 2.3. Installation

---

Ingenico JavaPOS PIN pad services depend on the following:

- JRE 1.4 or 1.5
- JavaCOMM API
- Java XML Parser
- Log4J Logging API

The JavaPOS Test App and the JavaPOS device services implementation can run on any platform with Sun compliant Java environment.

#### 2.3.1. Installing JRE

JPOS developers may download a SUN Java Runtime package from the Oracle software download web site at <http://www.oracle.com/technetwork/indexes/downloads/index.html>.

After downloading, install the package by running the downloaded installation file.

#### 2.3.2. Installing JavaCOMM API on Windows

The Ingenico JavaPOS implementation uses the JavaCOMM API package to communicate with peripherals connected via serial port.

It is important to note that platform-specific implementations of the JavaCOMM API are available from several vendors (e.g. Sun, IBM, and serialio.com). Ingenico recommends the use of SUN's Windows implementation of the JavaCOMM API, which can be obtained from <http://java.sun.com/products/javacomm/downloads/index.html>. The JavaCOMM API should be installed within the java library path.



SUN's JavaCOMM API for Windows is included with Ingenico's JPOS Integration Kit.

### 2.3.3. Installing JavaCOMM within JRE

To install the SUN JavaCOMM API package for Windows within JRE, follow these steps:

1. Unzip the JavaCOMM-Win32.zip file to a temporary folder.
2. Copy the following three files to the specified Java runtime folders. In this example, the Java runtime is in folder: "C:\Program Files\Java\jre1.5.0\_22".

comm.jar	C:\Program Files\Java\jre1.5.0_22\lib\ext
javax_comm.properties	C:\Program Files\Java\jre1.5.0_22\lib
win32.dll	C:\Program Files\Java\jre1.5.0_22\bin

### 2.3.4. Installing JavaCOMM Outside of JRE

To install the SUN JavaCOMM API package for Windows outside of JRE follow these steps:

1. Unzip the JavaCOMM-Win32.zip file to a temporary folder.
2. Ingenico provided JavaCOMM-Win32.zip file contains the JavaCOMM for Windows.
3. Copy the following three files to the appropriate location on your local drive as defined by the classpath and Java library path:

comm.jar	This file should be defined in classpath.
javax_comm.properties	This file should be defined in the Java library path.
win32.dll	This file should be defined in the Java library path.

For help with installation problems, refer to the readme.html file provided with JavaCOMM API files and the Java Communications API FAQ available at <http://java.sun.com/products/javacomm/reference/faqs/index.html>.

### 2.3.5. Installing the Java XML Parser

Ingenico recommends the use of Xerces Java, a validating XML parser from Apache Software Foundation. The Java Configuration Loader must load JPOS registry entries by parsing the XML-based JPOS configuration file (jpos.xml).

An XML Document Type Definition file (jcl.dtd, included with the kit) must also be available to the validating XML parser.

The Ingenico JPOS package includes xerces.jar, which can be placed in your classpath, or within the JRE extension folder.

### 2.3.6. Installing Apache Log4J Logging API

Logging in Ingenico JavaPOS device services is supported via Log4J. When reporting an issue, it is helpful to submit a log file with the logging level set to debug.



Apache Log4J can be downloaded from <http://logging.apache.org/log4j/docs/download.html>.

The Ingenico JPOS Logger can be integrated into application specific Log4J configuration file (configlog.xml).

To install the Apache Log4J Logging API, follow these steps:

1. Run the installer, making sure that the API is installed within the JRE extension folder or in a location defined in the classpath.
2. Specify the location of Log4J configuration file (jposconfig.xml) in the JPOS Configuration file (ijpos.xml). See JavaPOS Configuration on page 15 for more information on editing the JPOS configuration file.

```
< prop name="Configlog" type="String"
value=" ./res/configlog.xml " />
```

3. Edit Log4J configuration file (configlog.xml)
  - a. Set the level of logging in Log4J configuration file (configlog.xml).
  - b. Set the location of output log file.
  - c. Set the Appender.
  - d. Set the Loggers.

### 2.3.7. Troubleshooting Installation Issues

Most Eclipse based RADs have an issue with the java library path. If entering java library path in the VM arguments of Run Dialog results in `NoSuchPortException` when open is called on Ingenico device, try placing `javax.comm.properties` file in the JRE/lib folder.

`NoSuchPortException` is thrown Validate that the following three files are in the library/classpath path:

- `win32comm.dll`
- `comm.jar`
- `javax.comm.properties`

## 2.4. JavaPOS Configuration

---

The JPOS configuration file (ijpos.xml) holds JPOS data specific to Ingenico applications.

The location of the configuration file is specified in `jpos.properties` file contained within `jpos113.jar` (or `ijpos113.jar` when using an Ingenico build of the JPOS API). The `jpos.properties` file resides in the `jpos/res` folder, and has the following entry for location:

```
jpos.config.populatorFile=./res/ijpos.xml
```

If the JPOS Configuration Loader is unable to find the configuration, it returns a Service Not Found error.

If the JPOS configuration file is meant to be packaged within a JAR file, the `jpos.properties` file entry for location would be the following:

```
#jpos.config.populatorFile=./res/ijpos.xml
```

jpos.config.populatorFileJAR=res/jpos/ijpos.xml

**Info** The Java Configuration Loader is hardcoded to search for the jpos.properties file within the jpos113.jar file [jpos/res/jpos.properties].

## 2.5. JPOS Entry Editor

The Java Configuration Loader comes with a Java-based GUI that can be used to edit the JavaPOS configuration XML file.

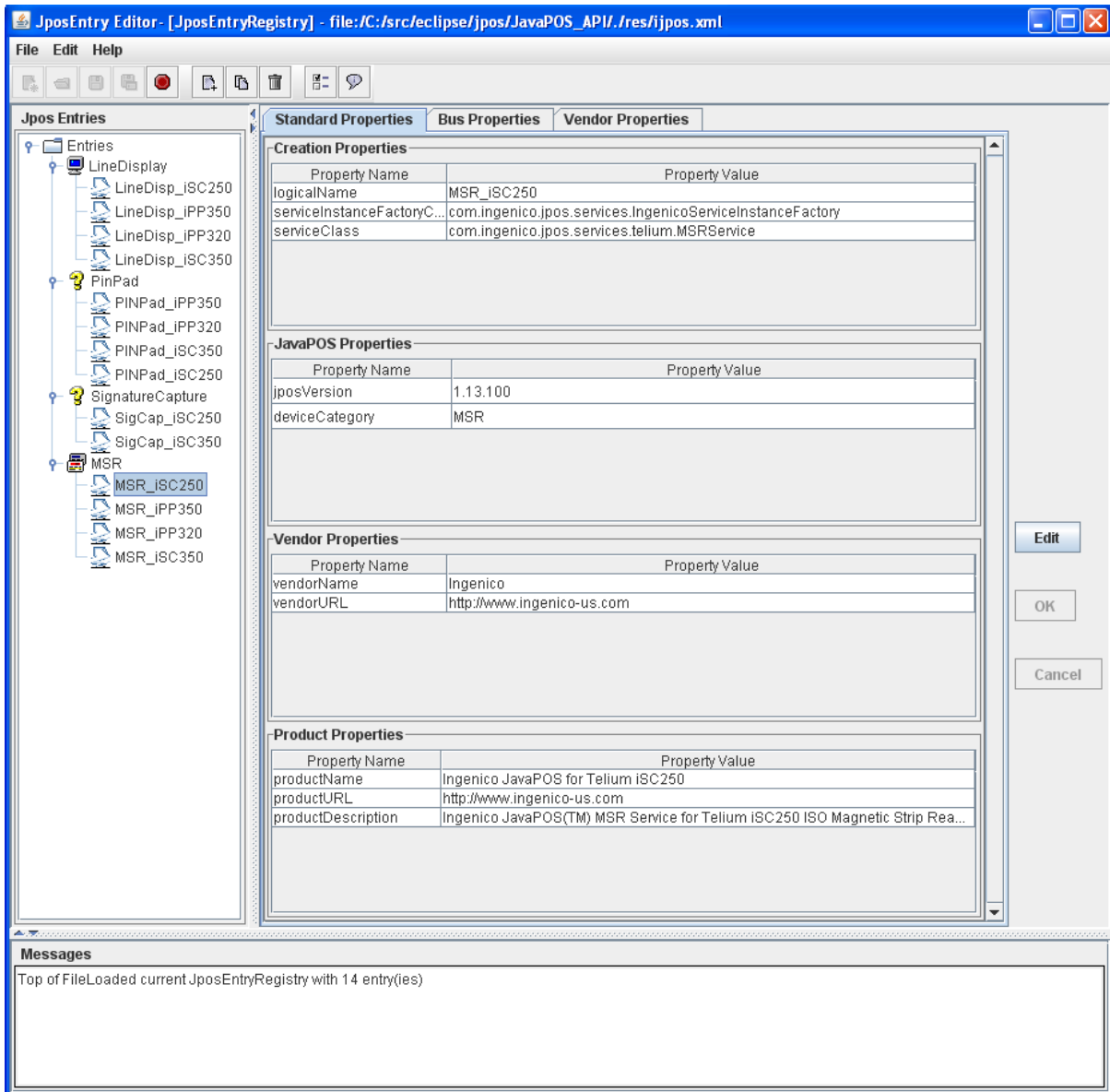


Figure 4 - The JPOS Entry Editor's main screen.

### 2.5.1. Setting the Java Path

In order for the JPOS Configuration Tool and Test Application to work properly, users must first modify the corresponding batch files to specify their system's Java path.

To modify JPOS batch files to specify the correct Java path, follow these steps:

1. Using a plain-text editor, open the \*.BAT file that you wish to modify.
2. Modify the third line in the \*.BAT file to specify the installation path for your system's JRE:

```
set JAVA_HOME=C:\"Program Files"\Java\jre6
```



*Batch files for both the JPOS Entry Editor and the Test Application MUST specify the correct path for the system's Java installation in order to work properly.*

### 2.5.2. Editing Standard Properties with the JPOS Entry Editor

To edit JPOS standard properties with the JPOS Entry Editor, follow these steps:

1. Click the entry that corresponds to the UPOS control and PIN pad that you wish to edit properties for in the Jpos Entries panel.
2. Click the Standard Properties tab.
3. Click Edit.
4. Update the desired property by clicking on the corresponding field and changing it by typing in a new value or selecting a new value from the drop-down.
5. Once you have finished editing the standard properties, click OK to save your changes.

### 2.5.3. Editing Bus Properties with the JPOS Entry Editor

To edit JPOS standard properties with the JPOS Entry Editor, follow these steps:

1. Click the entry that corresponds to the UPOS control and PIN pad that you wish to edit properties for in the Jpos Entries panel.
2. Click the Bus Properties tab.
3. Click Edit.
4. Update the desired property by clicking on the corresponding field and changing it by typing in a new value or selecting a new value from the drop-down.
5. Once you have finished editing bus properties, click OK to save your changes.

### 2.5.4. Adding Vendor Properties with the JPOS Entry Editor

To add JPOS vendor properties with the JPOS Entry Editor, follow these steps:

1. Click the entry that corresponds to the UPOS control and PIN pad that you wish to edit properties for in the Jpos Entries panel.
2. Click the Vendor Properties tab.
3. Click Edit.
4. Click Add.
5. Click the Property Name field and type a name for the new property.

6. Click the Property Value field and type a value to assign to the new property.
7. Click the Property Type field and select the data type that you wish to assign to the new property from the drop-down.
8. Click OK to save your changes.

### 2.5.5. Editing Vendor Properties with the JPOS Entry Editor

To edit JPOS standard properties with the JPOS Entry Editor, follow these steps:

1. Click the entry that corresponds to the UPOS control and PIN pad that you wish to edit properties for in the Jpos Entries panel.
2. Click the Vendor Properties tab.
3. Click Edit.
4. Choose one of the following:
5. Edit Property Names and Property Values by clicking the corresponding field and typing in the new value, or edit Property Types by clicking the corresponding field and selecting a new type from the drop-down.
6. Once you have finished editing vendor properties, click OK to save your changes.

### 2.5.6. Removing Vendor Properties with the JPOS Entry Editor

To remove JPOS vendor properties with the JPOS Entry Editor, follow these steps:

1. Click the entry that corresponds to the UPOS control and PIN pad that you wish to edit properties for in the Jpos Entries panel.
2. Click the Vendor Properties tab.
3. Click Edit.
4. Select the property that you wish to delete, then click Remove. A confirmation window displays.
5. Click Yes to confirm that you would like to delete the property from the JPOS configuration file.
6. Click OK to save your changes.

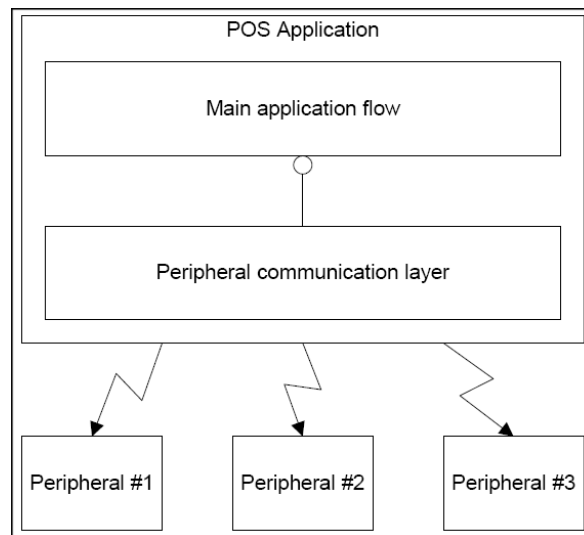
## 3. Getting Started with the OPOS Integration Kit

### 3.1. OPOS Overview

Object linking and embedding for retail point of sale (OPOS) is an object-based programming environment for the development of point of sale (POS) applications. OPOS allows developers to run their applications across a wide range of PIN pads and peripherals. This provides increased flexibility and reduces the effort required to ensure cross-platform operations for POS applications developed using the OPOS standard. OPOS reduces the cost of development usually associated with developing applications for proprietary hardware peripherals.

Epson, NCR, and ICL Retail Systems, in conjunction with Microsoft, developed the OPOS specification. The OPOS specification defines a set of POS device interfaces based on Microsoft's object linking and embedding (OLE) architecture. The OLE control architecture allows development in environments such as Visual Basic and Visual C++, to create POS application software in a device independent manner.

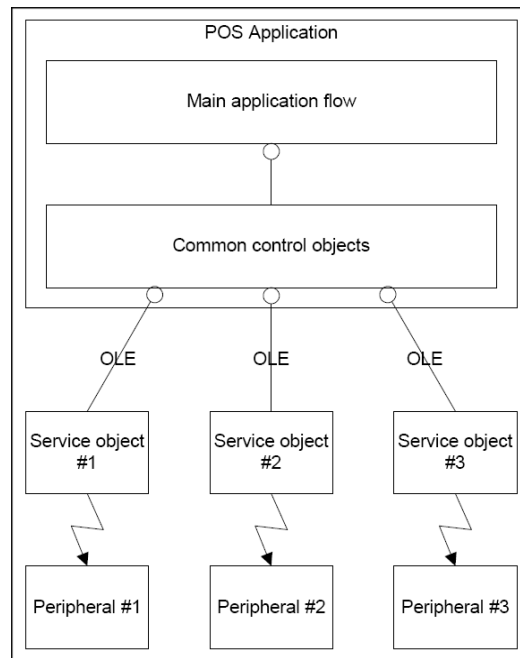
In the past, developers had to create device-specific communication layers in the POS application for each peripheral (see below).



**Figure 5 - Architectural overview.**

To add a new peripheral or install another model of the same type of peripheral, the communication layer had to be modified to support the new device. This usually meant that a new version of the entire application had to be created.

Using OLE control architecture, POS applications communicate with a defined interface (see below). This interface remains the same even though a peripheral change occurred.



**Figure 6 - OLE architectural overview.**

Because the application interface to the common control object does not change, when a new peripheral is added or a peripheral is changed, the application does not have to change (assuming the application supports the device type to begin with). To add a new device, simply replace the service object and the device. The application will not know the difference.

For detailed information on the standard OPOS controls, see the UnifiedPOS v1.13 specification, available in the ARTS Standards section of the Association for Retail Technology Standards website (<http://www.nrf-arts.org>).

### 3.2. OPOS Integration Kit Contents (SD36-2.00)

Your OPOS Integration Kit contains all of the materials necessary for writing a Windows-based application for a Telium PIN pad:

- OPOS for the Ingenico Telium2.exe, which will install the following:
  - Ingenico OPOS Service Object (*IngSC350SO.dll*)
  - Standard OPOS Control Objects:
    - *OPOSLineDisplay.ocx*
    - *OPOSMSR.ocx*
    - *OPOSPINPad.ocx*
    - *OPOSSigCap.ocx*

- OPOS for Telium Control Panel Application
- OPOS for Telium Configuration file (*IngenicoConfig.xml*)
- *Telium Utilities* folder, which contains the following:
  - OPOS Test Application
  - Telium Tools folder:
    - Telium Form Builder, Data Packaging Tool, and SAT installer
    - Telium Script Builder
    - Telium LLT
- Telium UPOS Interface Application:
  - Production loads:
    - iPP350:
      - LLT load
      - \*.OGZ load
    - iSC350:
      - LLT load
      - \*.OGZ load
      - USB memory stick load
  - Mock-up loads:
    - iPP350 (LLT load only)
    - iSC350:
      - LLT load
      - USB memory stick load
  - Data and Parameters:
    - Prompt Files
    - Forms
    - Images
- Ingenico Documents:
  - Getting Started document
  - This manual (DIV350781 - OPOS Developer's Guide for Telium)
  - User's guides for supported Telium PIN pads
- Release Notes

## 3.3. Installation

### 3.3.1. Typical Install

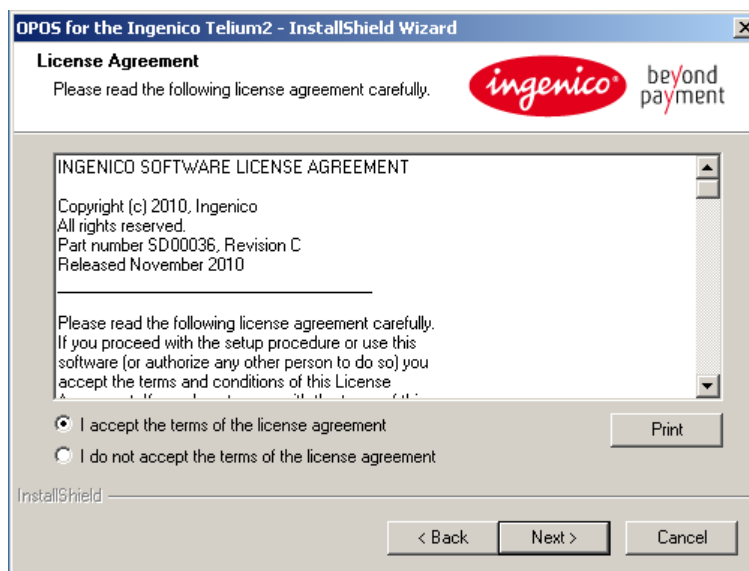
Before installing the OPOS Integration Kit, you must uninstall any previous versions of OPOS on your PC. See Uninstalling the Software on page 37 for more information.

To install your OPOS Integration Kit, do the following:

1. Exit all open programs.
2. Open the *OPOS for the Ingenico Telium2.exe* file.
3. The Welcome window displays. Click **Next**.

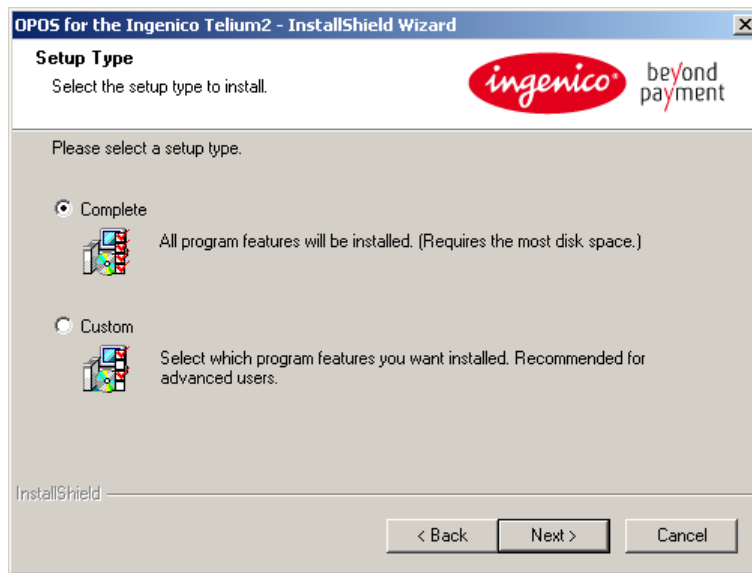


4. The License Agreement window displays. Read through the agreement and if you agree, click **I accept the terms of the license agreement** and click **Next** to accept it.

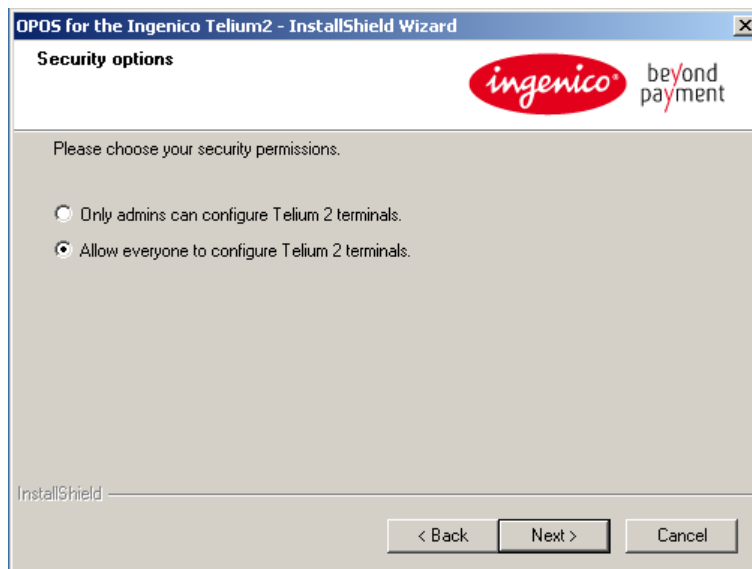




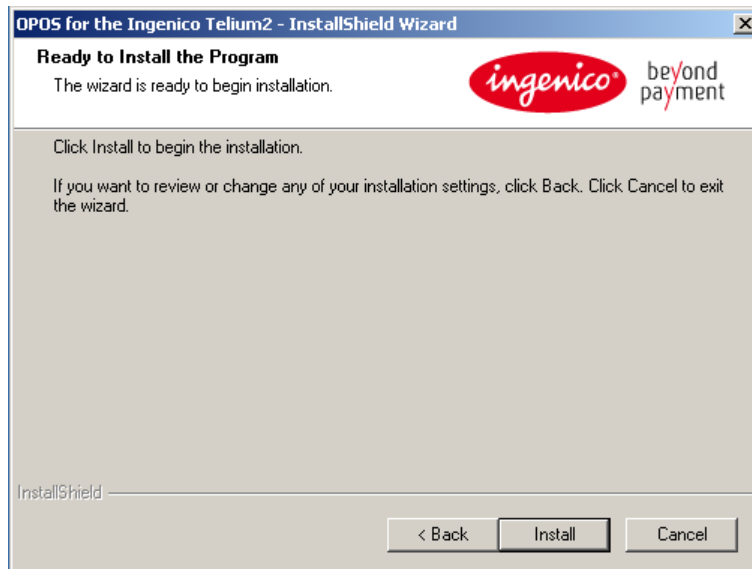
5. The Setup Type window displays. Select **Complete** to install all program features, then click **Next** to continue.



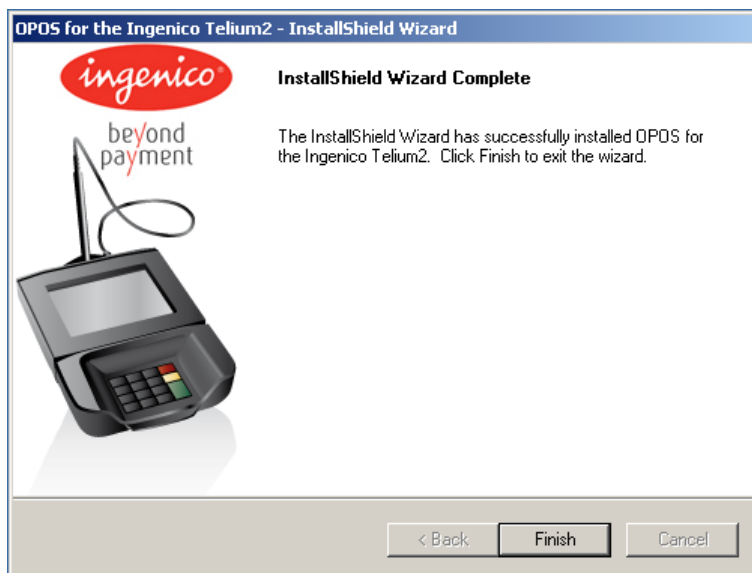
6. The Security options window displays. Select whether you only want administrators to configure Telium 2 PIN pads or if you want everyone to be able to configure Telium 2 PIN pads. Click **Next**.



7. The Ready to Install the Program window displays. Click Install to install the program. The program will install in the directory: C:\Program Files\Ingenico\OPOS for the Ingenico Telium2.



8. The InstallShield Wizard Complete window displays. Click Finish.

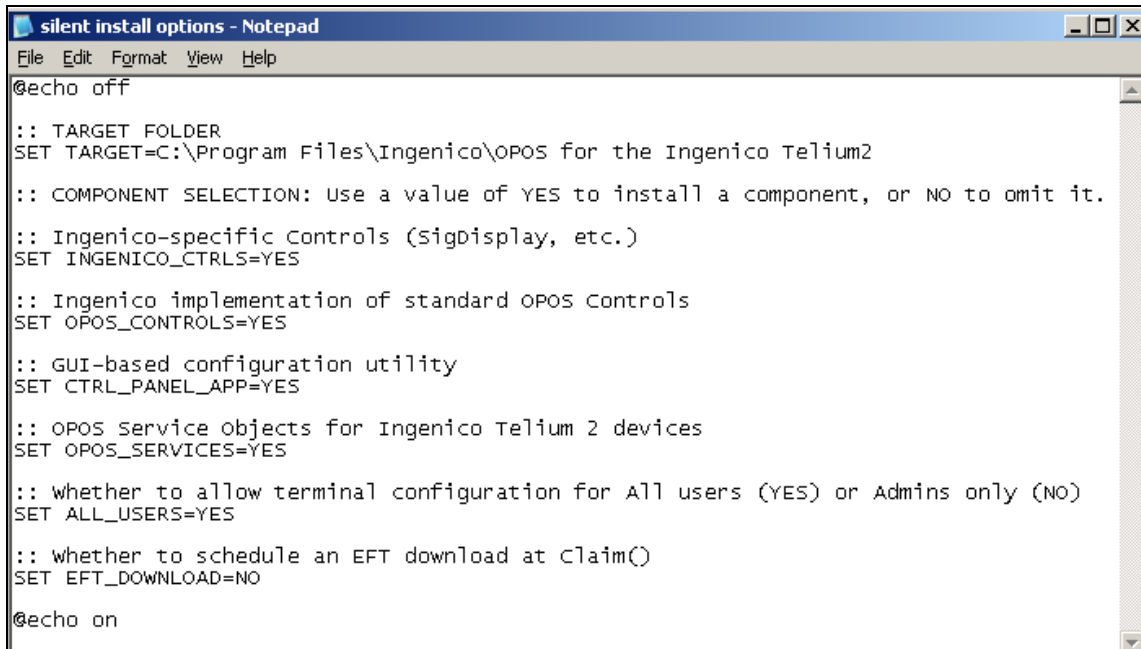


### 3.3.2. Silent Installation

The OPOS Integration Kit for Telium 2 also includes a silent installation feature that allows the kit to be installed with minimum user input.

### 3.3.2.1. Configuring a Silent Install

Silent install users can configure the silent install flow by editing the parameters specified in the *silent install options.bat* file provided with the OPOS Integration Kit using a plain text editor:



```
silent install options - Notepad
File Edit Format View Help
@echo off
:: TARGET FOLDER
SET TARGET=C:\Program Files\Ingenico\OPOS for the Ingenico Telium2
:: COMPONENT SELECTION: Use a value of YES to install a component, or NO to omit it.
:: Ingenico-specific Controls (SigDisplay, etc.)
SET INGENICO_CTRL=YES
:: Ingenico implementation of standard OPOS Controls
SET OPOS_CONTROLS=YES
:: GUI-based configuration utility
SET CTRL_PANEL_APP=YES
:: OPOS Service Objects for Ingenico Telium 2 devices
SET OPOS_SERVICES=YES
:: Whether to allow terminal configuration for All users (YES) or Admins only (NO)
SET ALL_USERS=YES
:: Whether to schedule an EFT download at Claim()
SET EFT_DOWNLOAD=NO
@echo on
```

**Figure 7 – silent install options.bat file contents.**

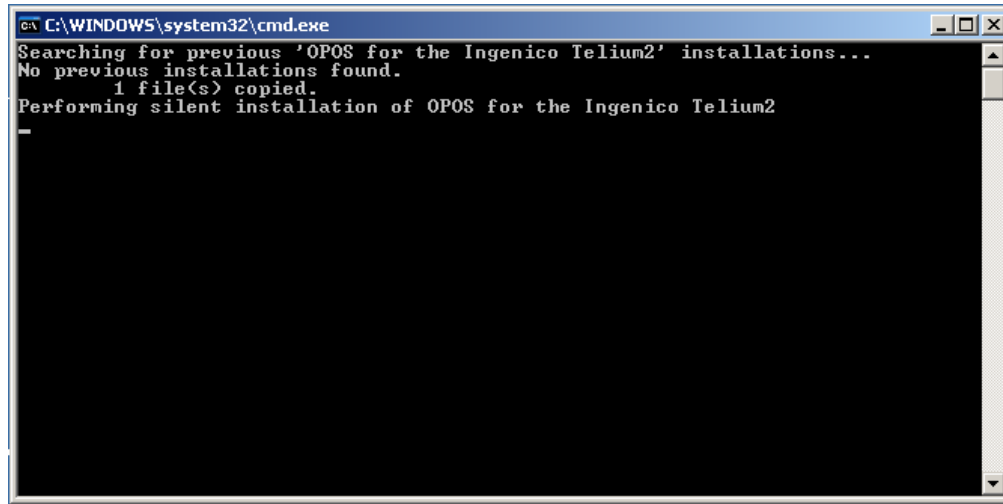
It is important to note that the OPOS Control Panel silent install WILL NOT overwrite an IngenicoConfig.xml file created by a previous installation. Users who wish to replace the IngenicoConfig.xml file during a silent install will need to modify the silent install options.bat file to add the following code:

```
@echo copy XML to Install directory
SET TARGET=C:\Program Files\Ingenico\OPOS for the Ingenico
Telium2
COPY /Y IngenicoConfiguration.xml "%TARGET%"
```

### 3.3.2.2. Executing a Silent Install

The silent install process can be initiated by executing the *silent install.bat* file provided with the OPOS Integration Kit.

The following screen displays:



```
C:\WINDOWS\system32\cmd.exe
Searching for previous 'OPOS for the Ingenico Telium2' installations...
No previous installations found.
1 file(s) copied.
Performing silent installation of OPOS for the Ingenico Telium2
```

The screen disappears once the OPOS Integration Kit has been installed.

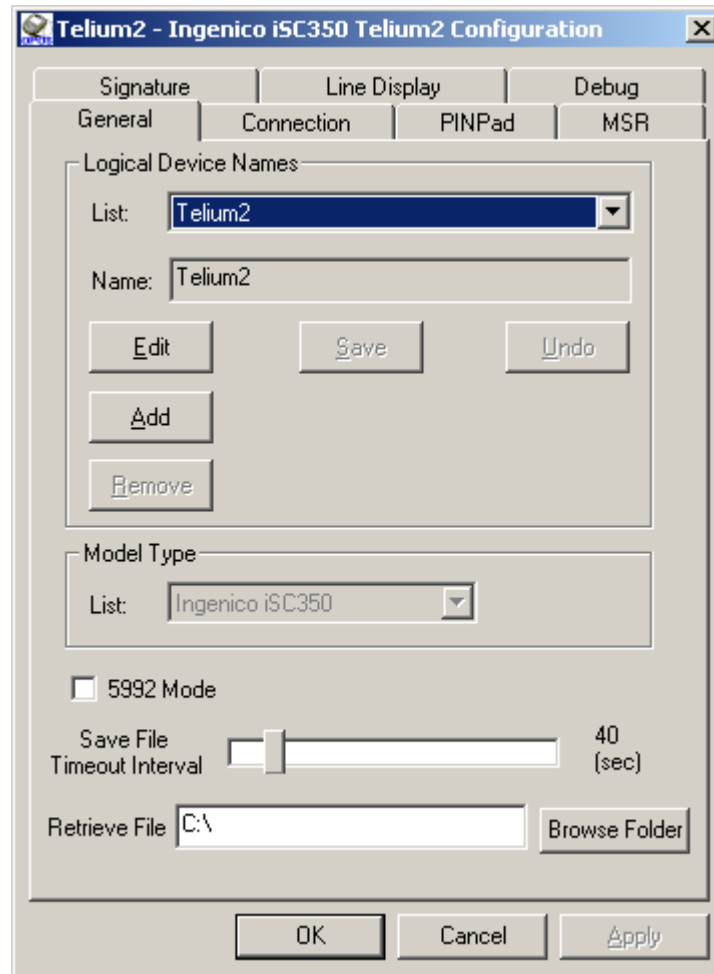
### 3.4. OPOS Configuration

---

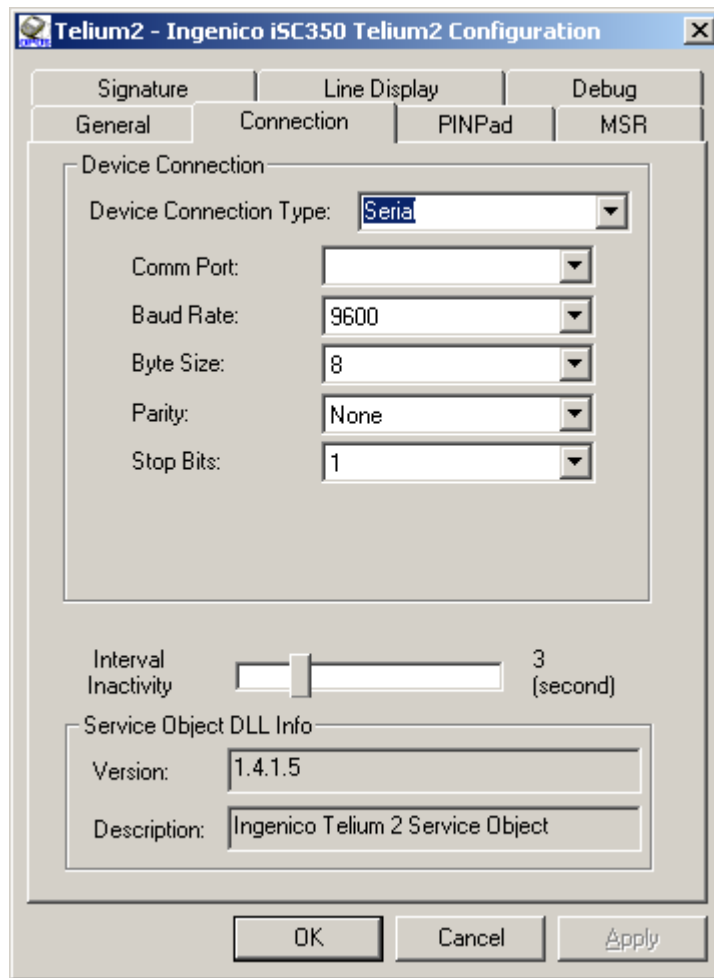
Host communication parameters are configured in the *IngenicoConfiguration.xml* file using the OPOS Control Panel:

1. Open Start > All Programs > INGENICO > OPOS for the Ingenico Telium 2 > Ingenico Telium 2 Setup.

The Ingenico iSeriesTelium2 window displays.



2. On the **General** tab, you can choose to **Edit**, **Add**, or **Remove** a Logical Device if desired.
3. Pick the correct Logical Device **Model Type** from the corresponding drop-down list.
4. Check the 5992 Mode checkbox to enable 5992 mode.  
***Info*** See 5992 Mode on page 95 for more information.
5. Pick the desired **Save File Timeout Interval** using the corresponding slider.
6. Specify the location to save any files retrieved from the PIN pad in the **Retrieve File** field.



7. Click the Connection tab and select the desired connection type in the **Device Connection Type** drop-down list:
  - a. If using **Ethernet**, specify an IP address and a port number for that IP address. The Ethernet setting is typically used in conjunction with a “terminal server” device. The terminal server usually occupies one IP address and provides multiple RS-232 serial ports.
  - b. If using **Serial**, fill in the fields as follows:
    - i. In the **Comm Port** box, select the appropriate communications port.
    - ii. In the **Baud Rate** box, select the desired baud rate.
    - iii. In the **Byte Size** box, select 8.
    - iv. In the **Parity** box, select **None**.
    - v. In the **Stop Bits** box, select 1.

8. Click the PINPad tab.

The screenshot shows the 'Telium2 - Ingenico iSC350 Telium2 Configuration' dialog box with the 'PINPad' tab selected. The dialog has a title bar with a close button. Below the title bar are four tabs: 'Signature', 'Line Display', 'Debug', and 'PINPad'. The 'PINPad' tab is active, showing the following configuration options:

- Pin Entry Form:** A text field labeled 'Form name:' containing the text 'PINSC350.K3Z'.
- PIN Entry Timeouts (in seconds):** Two sliders. The 'First Key' slider is set to 15, and the 'Inter Key' slider is also set to 15.
- PIN Entry min/max input digits:** Two sliders. The 'Min input Key' slider is set to 4, and the 'Max input Key' slider is set to 12.
- Configure Options:** A checkbox labeled 'Clear Screen after Pin Entry' which is currently unchecked.
- Configure Forms:** A button labeled 'Form Config'.

At the bottom of the dialog are three buttons: 'OK', 'Cancel', and 'Apply'.

- a. **Form name:** Enter the PIN Entry form name that exists in *IngenicoConfig.xml* and in the device.
- b. **PIN Entry Timeouts:** Use the sliders to specify the timeout interval, in seconds. The First Key slider is the timeout before the first key is pressed during PIN entry. The Inter Key slider is the timeout for any two consecutive key presses during PIN entry.
- c. **PIN Entry min/max input digits:** Use the sliders to specify the minimum input key and the maximum input key in digits. The minimum input key value must be less than or equal to the maximum input key value.
- d. Check the **Clear Screen after Pin Entry** check box to clear the screen after PIN entry.
- e. Click the **Form Config** button to enable/disable the display of specific PIN entry forms for various OPOS control events.

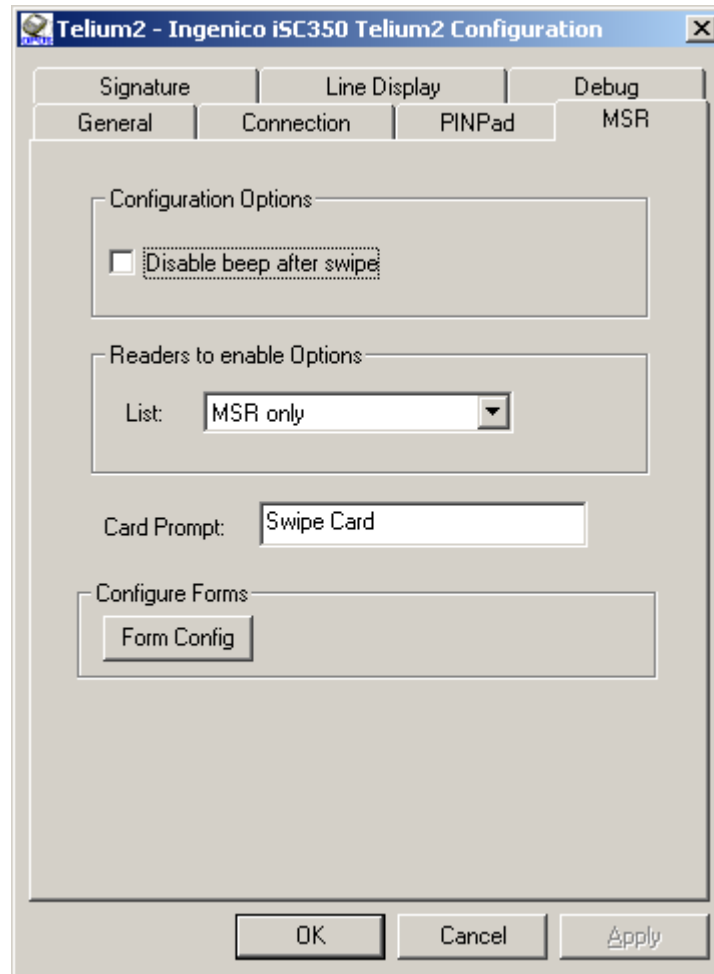
The image shows a 'Form Configuration' window with a blue title bar and a close button. Inside, there's a 'Forms' section containing six sub-sections: 'Claim', 'Enable Device', 'Disable Device', 'Data Event', 'Error Event', and 'Release'. Each sub-section has an 'Enable' checkbox and a 'Name' text field. All checkboxes are unchecked, and all 'Name' fields contain the text 'PINSC350.K3Z'. At the bottom of the window are two buttons: 'Save and Exit' and 'Exit Only'.

In the Form Configuration window, check the **Enable** checkbox and specify a form **Name** in the corresponding field to assign forms to specific events for the corresponding OPOS control.

Click **Save and Exit** or **Exit Only** to close the Form Configuration window.



9. Click the MSR tab.




- a. Check the **Disable beep after swipe** check box if you want to turn off the beep that sounds when you swipe a card. This option disables both error and regular swipe beeps.
- b. Select the type of card reader to enable by picking the desired option from the Readers to Enable Options – List drop-down menu.
- c. Make sure the **Card Prompt** field specifies the prompt text you wish to use.
- d. Click the **Form Config** button to enable/disable the display of specific PIN entry forms for various OPOS control events.

The image shows a 'Form Configuration' dialog box with a title bar containing a close button. Inside, there is a 'Forms' section with several sub-sections, each containing an 'Enable' checkbox and a 'Name' text field. The 'Enable Device' checkbox is checked, while the others are not. All 'Name' fields contain the text 'SWIPE640.K3Z'. At the bottom, there are two buttons: 'Save and Exit' and 'Exit Only'.

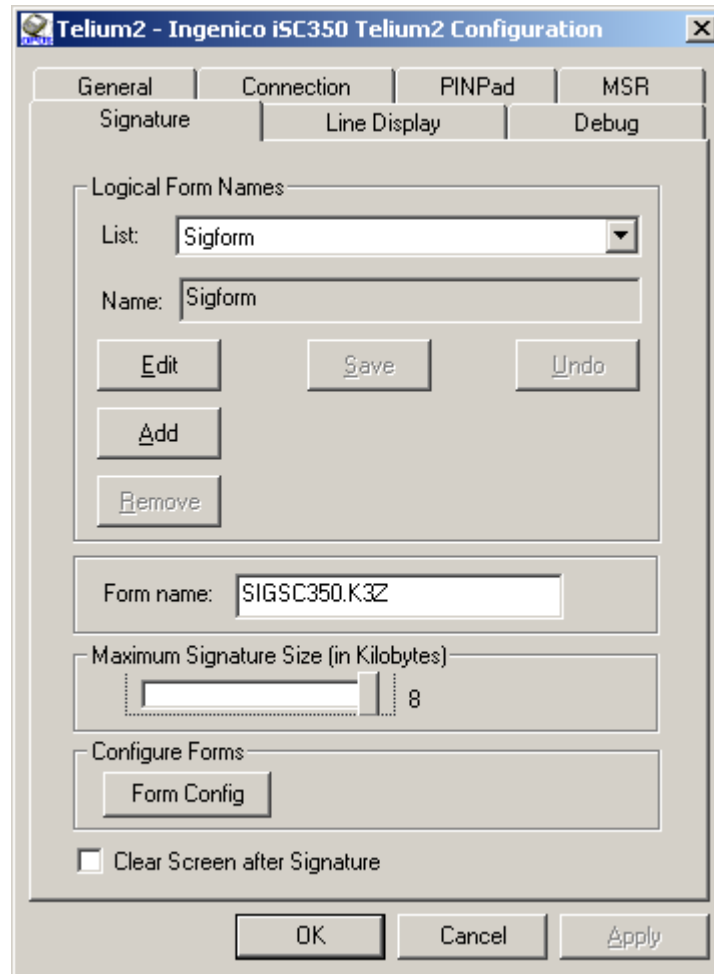
Event	Enable	Name
Claim	<input type="checkbox"/>	SWIPE640.K3Z
Enable Device	<input checked="" type="checkbox"/>	SWIPE640.K3Z
Disable Device	<input type="checkbox"/>	SWIPE640.K3Z
Data Event	<input type="checkbox"/>	SWIPE640.K3Z
Error Event	<input type="checkbox"/>	SWIPE640.K3Z
Release	<input type="checkbox"/>	SWIPE640.K3Z

In the Form Configuration window, check the **Enable** checkbox and specify a form **Name** in the corresponding field to assign forms to specific events for the corresponding OPOS control.

 *The Enable Device form is required for the OPOS MSR control to work correctly.*

Click **Save and Exit** or **Exit Only** to close the Form Configuration window.

10. Click the Signature tab.



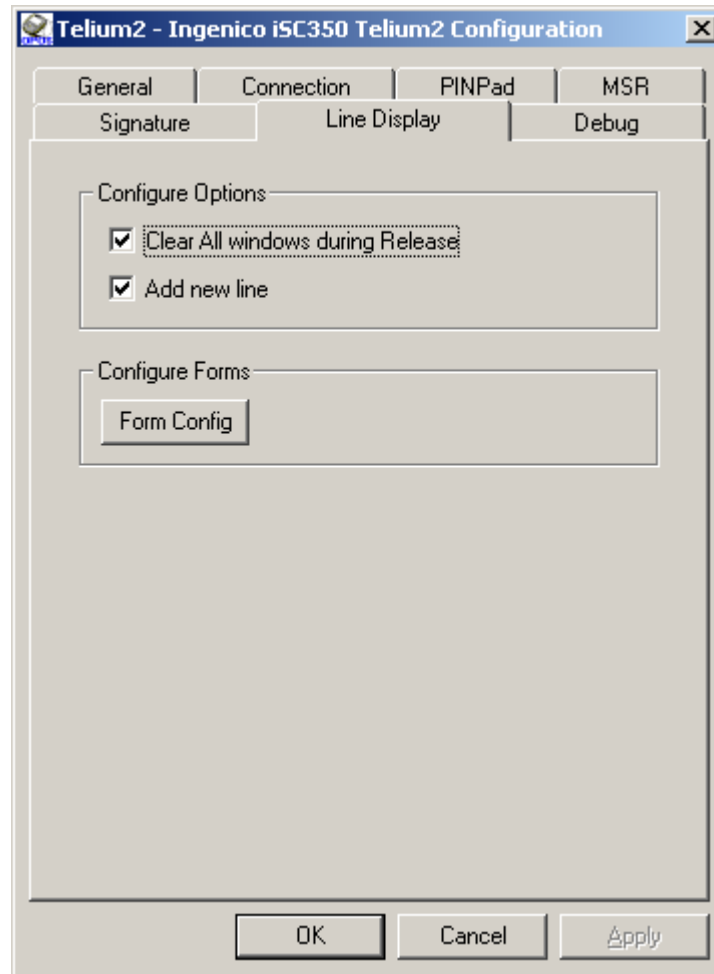
- a. On the Signature tab, you can choose to **Edit**, **Add**, or **Remove** a Logical Form Name if desired.
- b. **Form name:** Enter the Signature form name that exists in *IngenicoConfig.xml* and in the device.
- c. Under **Maximum Signature Size**, use the slider to specify the maximum signature size.
- d. Check the **Clear Screen after Signature** check box to clear the screen after signature.
- e. Click the **Form Config** button to enable/disable the display of specific PIN entry forms for various OPOS control events.

The image shows a 'Form Configuration' window with a blue title bar and a close button. Inside, there's a 'Forms' section containing six sub-sections: 'Claim', 'Enable Device', 'Disable Device', 'Data Event', 'Error Event', and 'Release'. Each sub-section has an 'Enable' checkbox and a 'Name' text field. All checkboxes are unchecked, and all 'Name' fields contain the text 'SIGSC350.K3Z'. At the bottom of the window are two buttons: 'Save and Exit' and 'Exit Only'.

In the Form Configuration window, check the **Enable** checkbox and specify a form **Name** in the corresponding field to assign forms to specific events for the corresponding OPOS control.

Click **Save and Exit** or **Exit Only** to close the Form Configuration window.

11. Click the Line Display tab.



- a. Check the **Clear All windows during Release** checkbox if you would like the PIN pad to clear all windows when the Line Display control is released.
- b. Check the **Add new line checkbox** to enable.
- c. Click the **Form Config** button to enable/disable the display of specific PIN entry forms for various OPOS control events.

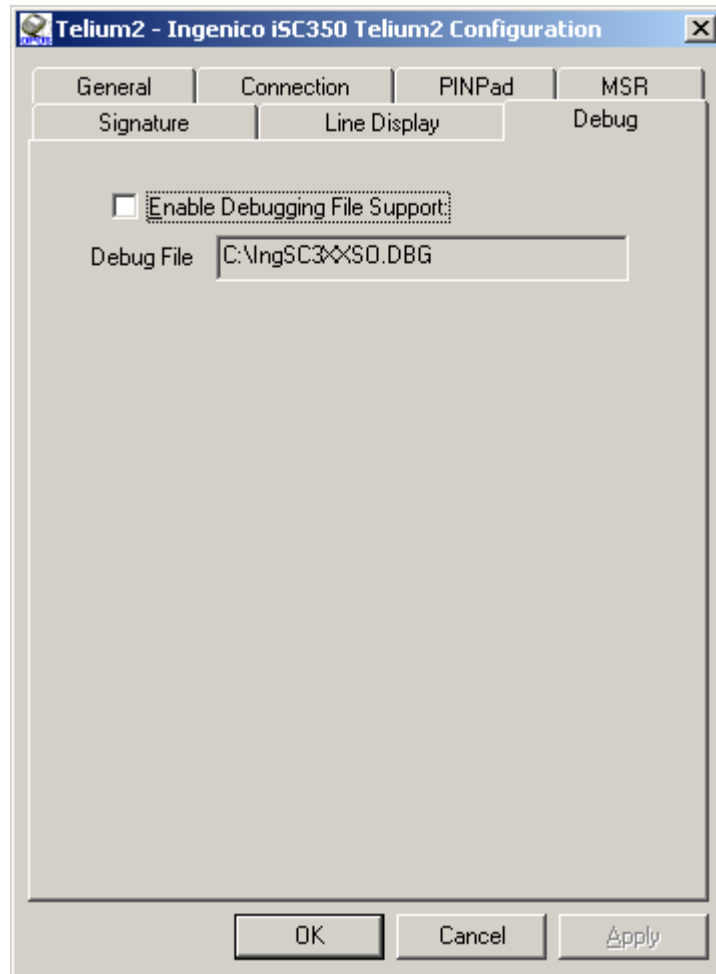
The image shows a 'Form Configuration' window with a blue title bar and a close button. Inside, there's a 'Forms' section containing six sub-sections: 'Claim', 'Enable Device', 'Disable Device', 'Data Event', 'Error Event', and 'Release'. Each sub-section has an 'Enable' checkbox and a 'Name' text field. All checkboxes are unchecked, and all 'Name' fields contain the text 'LDSC350.K3Z'. At the bottom, there are two buttons: 'Save and Exit' and 'Exit Only'.

Form Type	Enable	Name
Claim	<input type="checkbox"/>	LDSC350.K3Z
Enable Device	<input type="checkbox"/>	LDSC350.K3Z
Disable Device	<input type="checkbox"/>	LDSC350.K3Z
Data Event	<input type="checkbox"/>	LDSC350.K3Z
Error Event	<input type="checkbox"/>	LDSC350.K3Z
Release	<input type="checkbox"/>	LDSC350.K3Z

In the Form Configuration window, check the **Enable** checkbox and specify a form **Name** in the corresponding field to assign forms to specific events for the corresponding OPOS control.

Click **Save and Exit** or **Exit Only** to close the Form Configuration window.

12. Click the Debug tab.



- a. Select the **Enable Debugging File Support** check box to turn on debugging support for the Telium 2 PIN pad and generate a debugging file. In the **Debug File** field, specify the path for the debugging file. All OPOS errors will be logged to this file. To turn debugging support off, clear the check box.
13. Once you have completed all the necessary configuration steps, click **OK** to save your changes to the *IngenicoConfiguration.xml* file, or click **Cancel** to discard them.

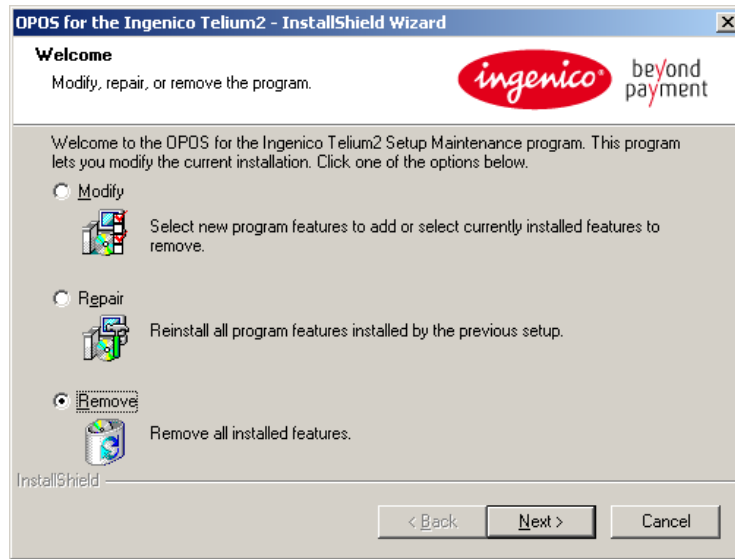
## 3.5. Uninstalling the Software

### 3.5.1. Typical Uninstall

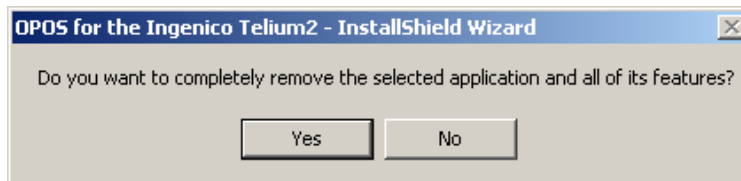
To uninstall the OPOS Integration Kit:

1. Exit all open programs.
2. Open the *OPOS for the Ingenico Telium2.exe* file.

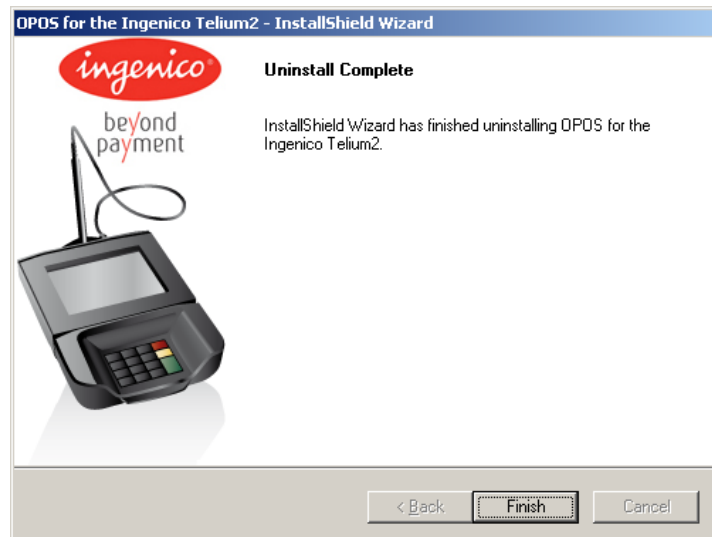
3. The Welcome window displays. Select **Remove** to remove OPOS from your PC. Click **Next**.



4. A warning window displays. Click **Yes** to remove the OPOS Integration Kit.



5. The Uninstall Complete window displays. Click **Finish** to close the screen.

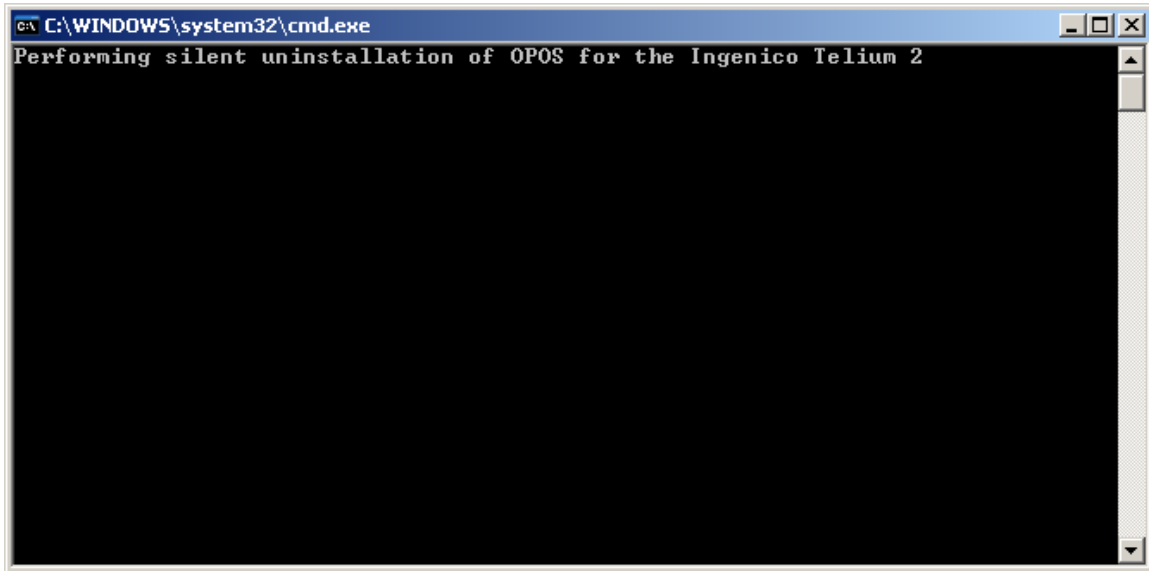




### 3.5.2. Silent Uninstall

The silent uninstall process can be initiated by executing the *silent uninstall.bat* file provided with the OPOS Integration Kit.

The following screen displays:



The screen disappears once the OPOS Integration Kit has been uninstalled.

## 3.6. Forms

---

The Telium platform supports forms in the \*.K3Z file format. They must be packaged as unsigned \*.TGZ files before being loaded onto an Ingenico PIN pad.

If you wish to use custom forms on your Ingenico PIN pad, refer to the DIV350713 Telium Tools User's Guide for additional information.

## 3.7. Using the OPOS Controls

---

In order to use the ActiveX controls listed in the following tables, you must first bring them into your development environment. This can be done in Visual Basic, Visual C++, or .NET environment.

- To insert an OPOS control into a Visual Basic project:
  - Right-click the **VB Controls Toolbox**, select **Components**, and then select the **Controls** tab.
  - Select the controls you would like to insert from the following table.
- To insert an OPOS control into a Visual C++ dialog:
  - Right-click the dialog, and then select **Insert ActiveX Control**.
  - Select the controls you would like to insert from the following table.

- To insert an OPOS Control into a Visual Studio .NET solution:
  - Select **Tools > Add/Remove Toolbox Items > COM Components** tab.
  - Select the controls you would like to insert (see following table). The selected controls become available in the Toolbox toolbar.
  - Click on the toolbar to select the desired type.
  - Click in your project's form to insert the control.

**Table 1: OPOS Controls**

Control	Description
OPOSLineDisplay	This OPOS control allows you to display forms using attribute sets in the Line Display form.
OPOSSigCap	<p>This is an OPOS control that allows an application to retrieve a signature from the PIN pad. This control is provided for users who wish to migrate from other OPOS SigCap applications to the PIN pad.</p> <p>The behavior of this control is highly dependent on the string parameter that is passed to the <b>BeginCapture()</b> method.</p> <p>If the string is not NULL, an attempt is made to match this string with the name of a registry value located under the OPOS SignatureCapture hive. If the match fails, the control returns OPOS_E_NOEXIST.</p>
OPOSMSR	This is an OPOS control that allows an application to get magnetic stripe information from a credit or debit card via the magnetic stripe reader (MSR).
OPOSPINPad	This is an OPOS control that allows an application to manage keys, compute MAC values, and get an encrypted PIN block from the secure PIN entry screen.

For instructions on how to use and format these controls, see the following chapters.

### 3.8. Setting Up the System Registry

OPOS relies on the system registry for proper operation. Each OPOS device has associated registry entries.

All OPOS registry settings are located under:

`HKEY_LOCAL_MACHINE\SOFTWARE\OLEforRetail\ServiceOPOS`

and are further subdivided by OPOS class type. The following is a listing of Ingenico's registry settings for each device.

The OPOS control key contains the following:

<b>LineDisplay\Telium2</b>	<code>OPOS.LineDisplay.SO.IngSC3XX</code>
<b>MSR\Telium2</b>	<code>OPOS.MSR.SO.IngSC3XX</code>
<b>PINPad\Telium2</b>	<code>OPOS.PINPad.SO.IngSC3XX</code>
<b>SignatureCapture\Telium2</b>	<code>OPOS.SigCap.SO.IngSC3XX</code>

The **SignatureCapture** entry in the System Registry holds the location of the *IngenicoConfig.xml* (used for configuring connection details). The connection entries for each device are modified through the control panel applet for the PIN pad. These settings are accessed when you call:

- SigCap.Open("Telium2")
- MSR.Open("Telium2")
- PINPad.Open("Telium2")
- LineDisplay.Open("Telium2")

## Notes

## 4. Direct I/O

Several features of the Telium PIN pads that are not covered by the OPOS specification have been exposed by the **DirectIO** method. It does not matter which Control Object is used to send the following Direct I/O commands, because each underlying service object supports Direct I/O usage equally.

### 4.1. The DirectIO Method

Syntax	<b>LONG DirectIO</b> (LONG Command, LONG* pData, BSTR* pString)																																
Remarks	<p>The required parameters are the following:</p> <ul style="list-style-type: none"><li>• <i>Command</i> - A numeric index corresponding to the desired operation.</li><li>• <i>pData</i> - Supplementary data, dependent on command index. See below.</li><li>• <i>pString</i><ul style="list-style-type: none"><li>– <i>Input</i>: unused</li><li>– <i>Output</i>: When applicable, a string containing the PIN pad's response to the most recently received command.</li></ul></li></ul> <p>The <b>DirectIO</b> method is capable of handling the following commands, as listed by DirectIO.h in your Developer's Documentation installation directory:</p> <table><tr><td>DIO_HEALTH_STATS</td><td>= x0D</td></tr><tr><td>DIO_SET_VARIABLE</td><td>= xA1</td></tr><tr><td>DIO_GET_VARIABLE</td><td>= xA2</td></tr><tr><td>FC_LIGHT_CONTROL</td><td>= xA6</td></tr><tr><td>DIO_DEVICE_RESET</td><td>= x09</td></tr><tr><td>DIO_TRANSMIT_CHECKBOX_DATA</td><td>= x12</td></tr><tr><td>DIO_TRANSMIT_SURVEY_DATA</td><td>= x13</td></tr><tr><td>DIO_CLEAR_LD</td><td>= x30</td></tr><tr><td>DIO_CLEAR</td><td>= x31</td></tr><tr><td>DIO_DISPLAY_TEXT_AT</td><td>= x34</td></tr><tr><td>DIO_TEXT_CURSOR</td><td>= x36</td></tr><tr><td>FC_LAST_ERROR</td><td>= x81</td></tr><tr><td>DIO_SAVE_FILE</td><td>= x91</td></tr><tr><td>DIO_RUN_FILE</td><td>= x92</td></tr><tr><td>DIO_RETRIEVE_FILE</td><td>= x94</td></tr><tr><td>DIO_FILE_STATUS</td><td>= x95</td></tr></table> <p>Each command is explained in the following sections.</p> <p>To have OPOS perform the desired command immediately, set the <i>Command</i> parameter to the appropriate index. Specific instructions regarding each command are provided in the following sections.</p> <p>The only DirectIO command that needs to fire a DirectIOEvent is DIO_RUN_FILE (x92). This DirectIOEvent will only be fired if pData is 1 or 2. The DirectIOEvent will fire after the user has performed some type of input (key presses). Unless otherwise indicated, the <i>pString</i> parameter of this event will contain the PIN pad's response to that command, encoded subject to the current value of the <b>BinaryConversion</b> property. To guarantee the correctness of <i>pString</i>, this property must have a value of OPOS_BC_NIBBLE or OPOS_BC_DECIMAL when the event is fired.</p>	DIO_HEALTH_STATS	= x0D	DIO_SET_VARIABLE	= xA1	DIO_GET_VARIABLE	= xA2	FC_LIGHT_CONTROL	= xA6	DIO_DEVICE_RESET	= x09	DIO_TRANSMIT_CHECKBOX_DATA	= x12	DIO_TRANSMIT_SURVEY_DATA	= x13	DIO_CLEAR_LD	= x30	DIO_CLEAR	= x31	DIO_DISPLAY_TEXT_AT	= x34	DIO_TEXT_CURSOR	= x36	FC_LAST_ERROR	= x81	DIO_SAVE_FILE	= x91	DIO_RUN_FILE	= x92	DIO_RETRIEVE_FILE	= x94	DIO_FILE_STATUS	= x95
DIO_HEALTH_STATS	= x0D																																
DIO_SET_VARIABLE	= xA1																																
DIO_GET_VARIABLE	= xA2																																
FC_LIGHT_CONTROL	= xA6																																
DIO_DEVICE_RESET	= x09																																
DIO_TRANSMIT_CHECKBOX_DATA	= x12																																
DIO_TRANSMIT_SURVEY_DATA	= x13																																
DIO_CLEAR_LD	= x30																																
DIO_CLEAR	= x31																																
DIO_DISPLAY_TEXT_AT	= x34																																
DIO_TEXT_CURSOR	= x36																																
FC_LAST_ERROR	= x81																																
DIO_SAVE_FILE	= x91																																
DIO_RUN_FILE	= x92																																
DIO_RETRIEVE_FILE	= x94																																
DIO_FILE_STATUS	= x95																																

Return	One of the following values is returned by the <b>DirectIO</b> method and placed in the <b>ResultCode</b> property:		
	Code	Label	Definition
	0	OPOS_SUCCESS	The method was successfully executed.
	113	OPOS_E_BUSY	The current service state does not allow this request. For example, if asynchronous output is in progress, certain methods may not be allowed.
	101	OPOS_E_CLOSED	An attempt was made to access a closed device.
	102	OPOS_E_CLAIMED	An attempt was made to access a device that is claimed by another control.
	115	OPOS_E_DEPRECATED	The requested operation can not be performed since it has been deprecated.
	105	OPOS_E_DISABLED	Cannot perform the requested operation while the device is disabled.
	110	OPOS_E_EXISTS	The file name (or other specified value) already exists.
	114	OPOS_E_EXTENDED	A device category-specific error condition occurred. The error condition code is held in an extended error code.
	111	OPOS_E_FAILURE	The device cannot perform the requested procedure, even though the physical device is connected to the system, powered on, and on-line.
	106	OPOS_E_ILLEGAL	An attempt was made to perform an illegal or unsupported operation with the device, or an invalid parameter value was used.
	109	OPOS_E_NOEXIST	The file name (or other specified value) does not exist.
	107	OPOS_E_NOHARDWARE	The physical device is not connected to the system or is not powered on.
	104	OPOS_E_NOSERVICE	The control cannot communicate with the service due to a configuration error.
	103	OPOS_E_NOTCLAIMED	An attempt was made to access an exclusive-use device that must be claimed before the method can be used.
	108	OPOS_E_OFFLINE	The physical device is off-line.
	112	OPOS_E_TIMEOUT	The service timed out waiting for a response from the physical device, or the control timed out waiting for a response from the service.
	F4	BAD_PARAMETER	General failure code for DirectIO commands.
	F7	ERROR	Error, use FC_LAST_ERROR for additional information.
	Default	UNKNOWN RESULT CODE	No return code received.

#### 4.1.1. Get Health Statistics

Parameter	Description
Command	DIO_HEALTH_STATS
pData	Not used.
pString	Not used.

The host sends this command to request general version and configuration information as described in the table below. This command can be used to determine which applications are installed on the device (as well as the corresponding version information).

The table below describes supported variables.

**Table 2: Health Statistics Variable Names**

Variable Name	Variable Description
<b>APP</b>	Internal application name.
<b>APP_BIN_NAME</b>	Application numeric Telium name.
<b>APP_VERS</b>	Application version.
<b>BAD_BLOCK</b>	Accumulated number of nand bad block found (some nand flash blocks may be bads without problem).
<b>BOOSTER_VERS</b>	Version of the Booster (security) processor.
<b>BUILD</b>	Build number.
<b>CLESS</b>	Is terminal capable of reading contactless cards? <ul style="list-style-type: none"> <li>• 0 = No</li> <li>• 1 = Yes</li> </ul>
<b>CLESS_ERRORS</b>	Total number of contactless errors.
<b>CLESS_READS</b>	Incremented when contactless card entry was enabled (“0x41” command) and a contactless card was read.
<b>DEVICE</b>	Name of the PIN pad device.
<b>ENTER_KEY</b>	Incremented when [ENTER] button was pressed while key entry task is enabled on PIN pad.
<b>FIN_KEY</b>	Whether or not a debit key is found in the PIN pad memory: <ul style="list-style-type: none"> <li>• 0 = No</li> <li>• 1 = Yes</li> </ul> <b>Currently out of scope.</b>
<b>FLASH_REFRESH</b>	Accumulated number of nand flash block refresh (1 bit ECC error had been found in NAND flash. Data had been corrected.)
<b>FREE_DFS</b>	Amount of free data file space in bytes.
<b>FREE_RAM</b>	Amount of free program memory in bytes.
<b>GARBAGE_COLLECT</b>	Accumulated number of nand flash garbage collection (physically erasing nand flash).
<b>HOST_CONFIG</b>	Details about the host port configuration.
<b>HOST_PORT</b>	Host communications port type. Possible values are: Ethernet, Serial, USB, Tailgate.
<b>IP_ADDR</b>	IP address of PIN pad. Display of this item can be disabled via configuration file.

Variable Name	Variable Description
MAC_ADDR	Network MAC Address. "xx:xx:xx:xx:xx:xx".
MANUF_NAME	Name of the manufacturer.
MANUF_DATE	Date of manufacture. "mmm dd yyyy".
MOCKUP_MODE	PIN pad is in mockup mode: <ul style="list-style-type: none"> <li>• 0 = No</li> <li>• 1 = Yes</li> </ul>
MSR_CRC	Incremented when checksum error happened while reading tracks. Incremented for each track separately.
MSR_PARITY	Incremented when parity error happened while reading tracks. Incremented for each track separately.
MSR_SWIPES1	Accumulated number of ISO1 magnetic cards swiped.
MSR_ERRORS1	Incremented when any error happened on Track 1 while reading card. Limitation: this parameter will be incremented only if track was read but any errors appeared during verification process (CRC, PARITY, etc). If due to some error the track was not read, but it is present on the card, this parameter will not be incremented – because only three return codes are available in OS, and there is no way to know if the card contains the track and it was not read, or if the card does not contain the track at all.
MSR_SWIPES2	Accumulated number of ISO2 magnetic cards swiped.
MSR_ERRORS2	Incremented when any error happened on Track 2 while reading card. Limitation: this parameter will be incremented only if track was read but any errors appeared during verification process (CRC, PARITY, etc). If due to some error the track was not read, but it is present on the card, this parameter will not be incremented – because only three return codes are available in OS, and there is no way to know if the card contains the track and it was not read, or if the card does not contain the track at all.
MSR_SWIPES3	Accumulated number of ISO3 magnetic cards swiped.
MSR_ERRORS3	Incremented when any error happened on Track 3 while reading card. Limitation: this parameter will be incremented only if track was read but any errors appeared during verification process (CRC, PARITY, etc). If due to some error the track was not read, but it is present on the card, this parameter will not be incremented – because only three return codes are available in OS, and there is no way to know if the card contains the track and it was not read, or if the card does not contain the track at all.
OS_VERSION	Version of the Telium OS installed.
PEN_TOTAL	Incremented after signature capture when terminal was touched with pen in signature box: incremented once for each continuous touch. <b>Currently out of scope.</b>
PIN_CANCEL	Number of PIN entries with CANCEL key pressed.
PIN_ENTER	Number of PIN entries with ENTER key pressed.
RESET	Accumulated number of soft reset. Processor reset required by the software (internal use, value read at startup in Thunder reset controller) - Telium 2 only.
SHUTDOWN	Accumulated number of power shutdowns.



Variable Name	Variable Description
<b>SIG</b>	Is PIN pad capable of capturing signatures? <ul style="list-style-type: none"> <li>• 0 = No</li> <li>• 1 = Yes</li> </ul>
<b>SIG_CANCEL</b>	Incremented when a signature is cancelled on the PIN pad.
<b>SIG_TOTAL</b>	Incremented when a signature is captured on PIN pad. This includes partial signatures captured before a CANCEL keypress.
<b>STARTUPS</b>	Accumulated number of Main processor (Thunder) Operating System starts.
<b>SUBNET_MASK</b>	IP subnet mask of the PIN pad. Display of this item can be disabled via configuration file.
<b>TELUM_VERS</b>	Version number of the Telium system.
<b>TERM_SERIAL#</b>	Ingenico serial number, generated at manufacture.
<b>THUNDER_VERS</b>	Version number of the Thunder (main) processor.
<b>TOUCH</b>	Whether or not the device is capable of capturing screen touches with stylus / finger. <ul style="list-style-type: none"> <li>• 0 = No</li> <li>• 1 = Yes</li> </ul> <b>Currently out of scope.</b>
<b>UP_TIME</b>	Total accumulated running time for the device, in seconds.
<b>USAGE</b>	Incremented when backlight is on. Total use time. <b>Currently out of scope.</b>

#### 4.1.2. Set UIA Variable

Parameter	Description
Command	DIO_SET_VARIABLE
pData	Not used.
pString	Name/value pair to indicate the variable and value to change.

This command allows the user to set UIA variables. Any variable values set with this command will remain even after the current form has been cleared.

The table below describes supported variable names.

**Table 3: Set UIA Variable Names**

Variable Name	Variable Description
<b>ACCOUNT_NAME</b>	Cardholder's name. User for manual credit card entry.
<b>BACKLIGHT</b>	Set display backlight brightness (0 – 100 percent).
<b>CVV2</b>	Credit verification value. Used for manual credit card entry.
<b>CLESSDELAY</b>	Amount of time (in 10 ms increments) to delay for a MSR swipe after a contactless card is detected. Default is 750 ms.

Variable Name	Variable Description
<b>CURRENTLANGUAGE</b>	Current prompt language: <ul style="list-style-type: none"> <li>• 1 = English</li> <li>• 2 = Spanish</li> <li>• 3 = French</li> </ul>
<b>CURRENTWINDOW</b>	Current line display (1,2,3, or 4). In OPOS control, Current Window is indexed 0 to 3. When the OPOS driver sends Current Window information to the UIA, it maps the index to position by incrementing by 1 (window 0 in OPOS control is window 1 in UIA).
<b>EXP_DATE</b>	Expiration date. Used for manual credit card entry. Format: mmyy.
<b>FORMATSPECIFIERINDEX</b>	Specifies the index of the secure format specifier for Clear Text Entry.
<b>KEYBOARDLIGHT</b>	Set keyboard backlight brightness (0 – 100 percent).
<b>KEYBEEP</b>	Beep on keypress: <ul style="list-style-type: none"> <li>• 0 = Off</li> <li>• 1 = On</li> </ul>
<b>KEYINDEX</b>	PIN entry encryption key index (slot).
<b>LDPRESETHEIGHT</b>	Sets the vertical distance between label elements in UIA's NCR 5992 line display form.
<b>MSRBADBEEP</b>	Beeps on bad magnetic stripe card swipe: <ul style="list-style-type: none"> <li>• 0 = Off</li> <li>• 1 = On</li> </ul>
<b>MSRGOODBEEP</b>	Beeps on good magnetic stripe card swipe: <ul style="list-style-type: none"> <li>• 0 = Off</li> <li>• 1 = On</li> </ul>
<b>MSR_NO_LIGHTS_ON_ENABLE</b>	Disables the MSR lights when the MSR reader is enabled: <ul style="list-style-type: none"> <li>• 0 = lights on (default)</li> <li>• 1 = lights off</li> </ul>
<b>MSR_NO_LIGHTS_ON_ERROR</b>	Disables the MSR lights when the MSR detects a bad card read: <ul style="list-style-type: none"> <li>• 0 = lights on (default)</li> <li>• 1 = lights off</li> </ul>
<b>PAN</b>	Primary account number. Used for manual credit card entry.
<b>pIDLEBACKLIGHTLEVEL</b>	Set idle screen backlight brightness (0 – 100 percent).
<b>pIDLEBACKLIGHTTIMEOUT</b>	Amount of idle time (in seconds) before idle screen or script is displayed. Must be greater than 60 seconds.
<b>pIDLESCREEN</b>	Form or script to be displayed on idle.
<b>pIDLESCREENTIMEOUT</b>	Amount of idle time (in seconds) before idle screen or script is displayed. Must be greater than 60 seconds.
<b>PINCARDNUMBER</b>	The account number to be associated with the entered PIN.

Variable Name	Variable Description
<b>PINENCRIPTION</b>	PIN entry encryption type: <ul style="list-style-type: none"> <li>• 0 = Master/Session</li> <li>• 1 = DUKPT</li> </ul>
<b>PINERRORBEEP</b>	Beeeps on PIN entry error: <ul style="list-style-type: none"> <li>• 0 = Off</li> <li>• 1 = On</li> </ul>
<b>PINFIRSTTIMEOUT</b>	Timeout for 1 <sup>st</sup> digit input. Default is 30 seconds. Set to 0 for infinite timeout.
<b>PININTERKEYTIMEOUT</b>	Inter-digit timeout. Default is 30 seconds. Set to 0 for infinite timeout.
<b>PINMAXKEY</b>	Minimum number of digits in PIN. Must be 12 or less. Default is 12.
<b>PINMINKEY</b>	Minimum number of digits in PIN. Must be 4 or greater. Default is 4.
<b>PINSESSIONKEY</b>	Specifies a session key value to be loaded into the terminal. The KEYINDEX variable must already be set to specify which key slot is to be updated.
<b>PROMPTINDEX</b>	Specifies the index of the secure prompt for Clear Text Entry and PIN entry.
<b>SIGMAXLEN</b>	Maximum signature size in bytes returned from UIA. Must be less than 8192.
<b>TABSIZE</b>	Specifies the width of tab stops in line displays. This controls the size of tabs for text to be displayed. It does not alter tab sizes in text that is already displayed.
<b>VARIABLESEPARATOR</b>	The integer value of the separator character used in the Set Variable message. The acceptable values are 1 – 255. The default value is 10 (linefeed). This variable should be set separately from other variables.
<b>5992_MODE</b>	Enables UIA NCR 5992 line display features: <ul style="list-style-type: none"> <li>• 0 = disabled (default)</li> <li>• 1 = enabled</li> </ul>

Syntax:

```
nCmd = DIO_SET_VARIABLE;
*pData = 0x0;
CString csVarName = "BUTTON_ID=01,255\TEXT_ID=5,Ha";
BSTR pString2 = ::SysAllocString(csVarName);
nResult = m_Form.DirectIO(nCmd,pData,&pString2);
```

### 4.1.3. Get UIA Variable

Parameter	Description
Command	DIO_GET_VARIABLE
pData	Ignored.
pString	Variable name. Example: CURRENTWINDOW.

This command allows the user to return the value of the specified variable.

#### 4.1.4. Configure PIN Pad Lights

Parameter	Description
Command	FC_LIGHT_CONTROL
pData	Ignored.
pString	Light control command string specifying desired settings.

This command allows the user to control the behavior of an iSC350's MSR LEDs.

The pString submitted for each light control message can use a maximum of 100 characters and can contain several groups of light control settings separated by colons:

<Light Setting 1>:<Light Setting 2>...

Each light setting can contain values for Light Type, State/Color, Animation Type, Duration and Intensity separated by semi-colons:

<Light Setting> = <Light Type>;<State/Color>;<Animation Type>;<Duration>;<Intensity>

To omit a parameter from a light control setting message, simply leave out the value for that parameter and add a semicolon before the next parameter value.

The light control message can be used to set the following parameters:

Parameter	Description	Allowable Values
State/Color	Activates/deactivates the PIN pad's MSR LED display and selects the active color.	<ul style="list-style-type: none"><li>• RED = only red LEDs are used</li><li>• GREEN = only green LEDs are used</li><li>• ON = only green LEDs are used (although this may change in future implementations)</li><li>• OFF = no LEDs are used</li></ul>
Animation Type	Selects the type of animation to apply to the MSR LED display.	<ul style="list-style-type: none"><li>• L_TO_R = MSR LEDs of the specified color are animated from left to right at a fixed rate</li><li>• R_TO_L = MSR LEDs of the specified color are animated from right to left at a fixed rate</li><li>• BLINK = MSR LEDs of the specified color blink at a fixed rate</li><li>• NONE = no animation</li></ul>
Duration	Selects the amount of time during which lights are on.	<ul style="list-style-type: none"><li>• 1 - 3600 = number of seconds</li></ul> <p><b>Info</b> If a duration value is not specified, the light setting group will apply until the PIN pad reboots, or a new light control command is received.</p>
Intensity	Ignored.	Ignored.

**Info** Light control commands ARE NOT case sensitive.

For best results, light control commands should follow these guidelines:

1. Light control setting groups where State/Color = OFF should omit all other parameters in order to work correctly. Submitting a value for any other parameter after State/Color = OFF will re-activate the MSR LEDs.
2. For best results, always begin light control message strings with a State/Color = OFF light setting group, followed by any other light settings you might wish to apply.
3. For best results, always specify a color when turning on MSR LEDs, i.e., use GREEN or RED rather than ON.
4. After a light control setting expires, the MSR LEDs revert to OFF. To apply a light control setting indefinitely, omit a duration value from the command to activating the MSR LEDs.

The table below shows some examples of valid light control message strings:

Command	LED Behavior
<code>MSR;OFF;;;</code>	Turns off MSR LEDs.
<code>MSR;RED;L_TO_R;20;;</code>	Activates red MSR LEDs with left-to-right animation for a period of 20 seconds. <i>Info This example DOES NOT follow guideline #2 specified above.</i>
<code>MSR;OFF;;;:MSR;GREEN;L_TO_R;;;</code>	Turns off MSR LEDs and turns on setting color to green with left-to-right animation. <i>Info This example follows the guideline #2 specified above.</i>
<code>MSR;OFF;;;:MSR;GREEN;L_TO_R;15;; :MSR;RED;L_TO_R;15;;</code>	Turns off MSR LEDs, turns on setting color to green with left-to-right animation for 15 seconds, then sets color to red with left-to-right animation for 15 seconds. <i>Info This example follows the guideline #2 specified above.</i>

If UIA returns an error in response to a light control message, you can obtain additional information on the error using the Get Last Error Direct I/O command. The possible Return Code responses are:

- 01 – Invalid Light Type <light type>
- 02 – Invalid Light Color/State <color/state>
- 03 – Invalid Animation Type <animation type>
- 04 – Invalid Duration <duration>
- 05 – Invalid Intensity <intensity>
- 06 – Invalid Light Control String

#### 4.1.5. Reboot the PIN Pad

Parameter	Description
-----------	-------------

Parameter	Description
Command	DIO_DEVICE_RESET
pData	Not used.
pString	Not used.

The host sends this command to reset the PIN pad.

The PIN pad will return one of the following:

- In response to a successful reset:
  - m\_dataOutput = 00
  - m\_objOutput = "Successfully Reset the Device.\r\nPlease close DirectIO Window and all OPOS Open Connections."
- In response to a failed reset:
  - m\_objOutput = "Failed to Reset the Device "

#### 4.1.6. Transmit Check Box Data

Parameter	Description
Command	DIO_TRANSMIT_CHECKBOX_DATA
pData	Ignored.
pString	Response message.

The host sends this command to request the state of any check box buttons displayed on the current form. The response message will return the state of each check box type button defined.

#### 4.1.7. Transmit Survey Data

Parameter	Description
Command	DIO_TRANSMIT_SURVEY_DATA
pData	Ignored.
pString	Response message.

The host sends this command to request the state of any survey boxes (sometimes called radio buttons) displayed on the current form. The response message will return the group ID (question number) and selected survey box ID (key ID) for each group defined in the current form.

#### 4.1.8. Clear Line Display Element

Parameter	Description
Command	DIO_CLEAR_LD
pData	Ignored.
pString	Ignored.

The host sends this command to clear the contents of the current line display area of the current window.

#### 4.1.9. Clear Screen

Parameter	Description
Command	DIO_CLEAR
pData	Ignored
pString	<ul style="list-style-type: none"><li>• Input: Ignored</li><li>• Output: The PIN pad's encoded response string.</li></ul>

Use this Direct I/O command to clear the all of the contents of your PIN pad display (for full context, see The DirectIO Method on page 43). Your OPOS Controls will be unaware of this change in display state.

Syntax:

```
Long nData = 0;
WORD wDummy = 0;
BSTR pString = ::SysAllocString(&wDummy);
OPOSDevice.DirectIO(DIO_CLEAR, &nData, &pString);
```

#### 4.1.10. Display Text At

Parameter	Description
Command	DIO_TEXT_AT
pData	<ul style="list-style-type: none"><li>• 0 = for Row/Column display mode without font selection.</li><li>• 1 = for Pixel display mode without font selection.</li><li>• 2 = for Row/Column display mode with font selection.</li><li>• 3 = for Pixel display mode with font selection.</li></ul>
pString	<ul style="list-style-type: none"><li>• <b>For pData = 0</b> – concatenated string with 1 byte row value, 1 byte column value, variable length text to display. Items are separated by a field separator character.</li><li>• <b>For pData = 1</b> – concatenated string with 2 byte x value, 2 byte y value, variable length text to display. Items are separated by a field separator character.</li><li>• <b>For pData = 2</b> – concatenated string with 1 byte row value, 1 byte column value, 1 byte font value, variable length text to display. Items are separated by a field separator character.</li><li>• <b>For pData = 3</b> – concatenated string with 2 byte x value, 2 byte y value, 1 byte font value, variable length text to display. Items are separated by a field separator character.</li></ul>

This DirectIO command is used to display text on the PIN pad screen in a specific font at a specified row and column offset or (x,y) pixel location. The X and Y coordinate values are based on the iSC350's screen dimension of 640 x 480 pixels.

#### 4.1.11. Position Text Cursor

Parameter	Description
Command	DIO_TEXT_CURSOR

Parameter	Description
pData	One byte row and one byte column position converted to a long. For example, to set cursor position to row 8, column 9, use 08, 09 hex and convert to decimal. In this case the value is 2057 so set the pData to 2057.
pString	Ignored.

This command sets the cursor position for the current line display.

#### 4.1.12. Get Last Error

Parameter	Description
Command	FC_LAST_ERROR
pData	Ignored.
pString	Ignored.

This command can be used to get details on the first 1017 bytes of errors logged since this message was last received by UIA.

When message is received, UIA returns a response code and error data.

#### 4.1.13. Save File

Parameter	Description
Command	DIO_SAVE_FILE
pData	<ul style="list-style-type: none"> <li>0 = No reboot needed after file is saved (typical with *.TGZ files).</li> <li>1 = Reboot PIN pad after saving file (typical with *.PGZ or *.OGZ files).</li> </ul>
pString	Fully qualified file name (e.g. c:\temp\forms.agn) passed as an array of bytes.

Stores an archive file on the PIN pad in persistent memory. This file will contain other files such as web pages, images, JavaScript libraries, etc. For production PIN pads the file must be in signed PGZ format. For development (mockup) PIN pads, the file can be in PGZ format. Video files must be downloaded individually. Video files must be in signed MGN format on production PIN pads. On development PIN pads, the format can be MP4 or MGN.

The Save File command is also used to update content on the PIN pad. Please refer to each PIN pad's User Guide for more information on PIN pad updates.

#### 4.1.14. Run File

Parameter	Description
Command	DIO_RUN_FILE
pData	<ul style="list-style-type: none"> <li>0 = If form does not contain any user input (e.g. buttons, edit box)</li> <li>1 = If form has buttons (e.g. control buttons, survey, check boxes)</li> <li>2 = If form is used for clear text entry</li> </ul>
pString	Form file name (e.g. sigcap.k3z) passed as an array of bytes.

Displays a form on the PIN pad.



#### 4.1.15. Retrieve File

Parameter	Description
Command	DIO_RETRIEVE_FILE
pData	N/A
pString	The word “manifest” passed in as an array of bytes.

The Retrieve File command is used to retrieve a PIN pad’s manifest file to a specified location defined using the OPOS Control Panel application. Using the command to retrieve any other type of file from the PIN pad returns the OPOS\_E\_NOEXIST code.

#### 4.1.16. File Status

Parameter	Description
Command	DIO_FILE_STATUS
pData	<ul style="list-style-type: none"><li>1 = Return file status</li></ul>
pString	File name (e.g. sigcap.k3z) passed as an array of bytes.

The File Status command will confirm whether a given file exists on the Telium PIN pad. If the file exists, the command returns a time stamp and the size of the specified file. If the file does not exist, the PIN pad returns the OPOS\_E\_NOEXIST code.

### 4.2. Best Practices

---

Ingenico's OPOS Service Objects are designed to free all resources they consume when the **Close()** method is called. To minimize processor usage, Ingenico recommends calling **Close()** if the Service Object will not be used for an extended period of time, for example: overnight when no transactions are taking place. The **Open()** method can be subsequently called when transactions are resumed.

### 4.3. Usage Samples

---

Ingenico OPOS provides a set of DirectIO commands described under DirectIO.h:

The following shows how to make a direction call:

```
OPOSDevice.DirectIO(command: int32, inout data: int32, inout obj:
object);
```

#### 4.3.1. Store a Form

The following code sample shows how to store a form:

```
long nCmd = DIO_SAVE_FILE;
LONG Data = 0;
long* pData = &Data;
CString csVarName = "INGENICO.PGZ";

// name of file to store
```

```

        BSTR pString2 = csVarName.AllocSysString();
        m_MSR.Open("Telium2");

m_MSR.ClaimDevice (5000);
m_MSR.SetDeviceEnabled(TRUE);
m_MSR.DirectIO(nCmd, Data, pString2);

```

#### 4.3.2. Display a Form

The following code sample shows how to display a form:

```

long nCmd = DIO_RUN_FILE;
LONG Data = 2;
long* pData = &Data;
CString csVarName = "INGENICO.K3Z";

// name of file to display

        BSTR pString2 = csVarName.AllocSysString();
        m_MSR.Open("Telium2");

m_MSR.ClaimDevice (5000);
m_MSR.SetDeviceEnabled(TRUE);
m_MSR.SetDataEventEnabled(TRUE);
m_MSR.DirectIO(nCmd, pData, &pString2);

```

## 5. Line Display

---

### 5.1. General Information

---

Ingenico's implementation of the Line Display control follows the version 1.13 specifications for UnifiedPOS controls exactly.

### 5.2. Usage Samples

---

#### 5.2.1. Clear a Window

The following code sample shows how to clear a window:

```
m_LineDisplay.Open("Telium2");
m_LineDisplay.ClaimDevice (5000);
m_LineDisplay.SetDeviceEnabled(TRUE);
m_LineDisplay.ClearText();
```

#### 5.2.2. Write Text

##### 5.2.2.1. To a Location (Row, Column)

The following code sample shows how to write text to a particular row and column:

```
int row=0, col=3;
m_LineDisplay.Open("Telium2");
m_LineDisplay.ClaimDevice (5000);
m_LineDisplay.SetDeviceEnabled(TRUE);
m_LineDisplay.DisplayTextAt(row, col, "$", DISP_DT_NORMAL));
```

##### 5.2.2.2. To a Variable

The following code sample shows how to write text to a variable:

```
long nCmd = DIO_ SET_UIA_VARIABLE;
LONG Data = 0;
long* pData = &Data;

// TOTAL is name of variable and it is being set to $12.99.
// When a form is displayed that contains the
// variable TOTAL it will show $12.99

CString csVarName = "TOTAL=$12.99";
BSTR pString2 = csVarName.AllocSysString();
m_LineDisplay.Open("Telium2");

m_LineDisplay.ClaimDevice (5000);
m_LineDisplay.SetDeviceEnabled(TRUE);
m_LineDisplay.DirectIO(nCmd, pData, &pString2);
```

## Notes

## 6. MSR

---

### 6.1. General Information

---

Ingenico's implementation of the MSR control follows the version 1.13 specifications for UnifiedPOS controls exactly.

### 6.2. Usage Samples

---

#### 6.2.1. Read All Tracks of Card Data

The following code sample shows how to read all tracks of card data:

```
CString strTrack1, strTrack2, strTrack3;
m_MSR.Open("Telium2");
m_MSR.ClaimDevice (5000);
m_MSR.SetDeviceEnabled(TRUE);
m_MSR.SetDataEventEnabled(TRUE);

// Now check each track for data after Data Event happens

strTrack1 = m_MSR.GetTrack1Data();
strTrack2 = m_MSR.GetTrack2Data();
strTrack3 = m_MSR.GetTrack3Data();
```

#### 6.2.2. SetDeviceEnabled Error Reporting

The following code sample shows how to enable error reporting:

```
m_MSR.Open("Telium2");
m_MSR.ClaimDevice (5000);
m_MSR.SetDeviceEnabled(TRUE);

// use one of the error methods below but not both
// error reporting by card

m_MSR.SetErrorReportingType (MSR_ERT_CARD);

// error reporting by track

m_MSR.SetErrorReportingType (MSR_ERT_TRACK);
```

#### 6.2.3. Request Sentinels

The following code sample shows how to request sentinels:

```
m_MSR.Open("Telium2");
m_MSR.ClaimDevice (5000);
m_MSR.SetDeviceEnabled(TRUE);
m_MSR.SetDataEventEnabled(TRUE);
// if device is capable of transmitting sentinels
// then request the device to send the sentinels
```

```

if (TRUE == m_MSR.GetCapTransmitSentinels())
m_MSR.SetTransmitSentinels(TRUE);

```

#### 6.2.4. Parse MSR Data

The following VB.NET code sample shows how to parse MSR data:

```

Private Sub MSR_DataEvent(ByVal Status As Long)

Dim res As String
UpdateInfo ("MSR DataEvent Received with Status of: " &
Status)
UpdateInfo ("Transmit Sentinels Value: " &
MSR.TransmitSentinels)
txtMSRAccountNum.Text = MSR.AccountNumber
UpdateInfo ("MSR Data Event Received, Count = " & MSR.Count)
UpdateInfo ("Account Number: " & MSR.AccountNumber)
txtMSRAccountNum.Text = MSR.AccountNumber
UpdateInfo ("Track 1: " & MSR.Track1Data)
UpdateInfo ("Track 2: " & MSR.Track2Data)
UpdateInfo ("Track 3: " & MSR.Track3Data)
UpdateInfo ("First Name: " & MSR.FirstName)
UpdateInfo ("Surname: " & MSR.Surname)
If MSR.AutoDisable = False Then cmdEnableMSR_Click

End Sub

```

## 7. Signature Capture

---

### 7.1. General Information

---

Ingenico's implementation of the Signature Capture control follows the version 1.13 specifications for UnifiedPOS controls exactly.

### 7.2. Usage Samples

---

#### 7.2.1. Define the Signature Format

OPOS PointArray holds the signature captured from the device. It consists of an array of (x, y) coordinate points. Each point is represented by four characters: x (low 8 bits), x (high 8 bits), y (low 8 bits), y (high 8 bits).

A special point value is (0xFFFF, 0xFFFF) which indicates the end of a line (that is, a pen lift). Almost all signatures are comprised of more than one line.

#### 7.2.2. Define the OPOS Conversion Formats

The following code sample shows how to define OPOS conversion formats:

```
LONG OPOS_BC_NONE           = 0 ;
LONG OPOS_BC_NIBBLE         = 1 ;
LONG OPOS_BC_DECIMAL        = 2 ;
```

#### 7.2.3. Handle Signature Capture

In the code sample below, SigCap is an instance of the Signature Capture Control Object.

A form is used to provide the user a place to sign. For information on how to generate a form, refer to the Telium UI Developer's Guide that corresponds to the PIN pad model you will be using.

The SignatureCapture.BeginCapture method parameter specifies the logical name of the form to be used for signature capture. The file name must be specified in *IngenicoConfiguration.xml*.

```
<MsrCardPrompt name="CLESS">Tap Card</MsrCardPrompt>
<MsrCardPrompt name="MSRCLESS">Tap or Swipe Card</MsrCardPrompt>
</Items>
</MsrCardPrompts>
- <PinEntryForms>
- <Items>
  <PinEntryModelForm name="PinEntry">PINSC350.K3Z</PinEntryModelForm>
</Items>
</PinEntryForms>
```

After the SignatureCapture control has been opened, claimed, and enabled, the application can call the BeginCapture method to start the signature. The parameter to BeginCapture method must be the logical name of the signature form in *IngenicoConfiguration.xml*.

### 7.2.3.1. Implementing Signature Capture: C Sample Code

The C code sample below shows how to use signature capture:

```
long lResponseCode;
//prepare for signature
m_SigCap.Open("Telium2");
m_SigCap.ClaimDevice(5000);
m_SigCap.DeviceEnabled(TRUE);
m_SigCap.SetDataEventEnabled(TRUE);
m_SigCap.SetBinaryConversion(NIBBLE);
lResponseCode = m_SigCap.BeginCapture("Sigform");
if(lResponseCode!=0x00)
{
    //handle error
}
```

### 7.2.3.2. Implementing Signature Capture: Visual Basic Sample Code

The Visual Basic example below shows how to implement signature capture:

```
With OPOSSig
UpdateInfo ("Using OPOS Signature Capture Control")
'Log file update
.DeviceEnabled = True
.DataEventEnabled = True
.BinaryConversion = 1
res = .BeginCapture(txtBeginCapForm.Text)
'this is registry entry that points to the actual form in the
device.
If res = 0 Then
UpdateInfo ("OPOS Begin Capture Form Displayed Successfully")
Else
UpdateInfo ("Error Displaying Begin Capture Form, Error: " & res)

End If
'Format Guide: "12345678901234567890"
res = Form1.DisplayTextAt(5, 6, "    Please Sign    ", 0)
res = Form1.DisplayTextAt(6, 6, "    With Stylus    ", 0)

If res = 0 Then UpdateInfo ("Display Text Successfully") Else
UpdateInfo ("Error Displaying Text, Error: " & res)
End With
```



## 8. SigDisplay Control

---

### 8.1. General Information

---

Ingenico wrote its own form control, **OPOSSigDisplay.ocx**, as an extension to the OPOS specifications in order to display or save signature capture data from the Telium PIN pads. With respect to common properties and methods, **OPOSSigDisplay** control follows the version 1.13 specifications for UnifiedPOS controls exactly.

### 8.2. Summary

---

This section contains precise usage prerequisites for each property.

#### 8.2.1. Properties

**Table 4: Specific Properties**

Name	Type Access	Initialized Before
GetDrawBorder	Boolean R	Methods
SetDrawBorder	Boolean W	Methods
GetDrawBackground	Boolean R	Methods
SetDrawBackground	Boolean W	Methods
GetPenWidth[Printer]	Long R	Methods, Open
SetPenWidth[Printer]	Long W	Methods, Open
GetDisplayNumPoints	Boolean R	Methods
SetDisplayNumPoints	Boolean W	Methods
GetNumPointsInDisplay	Long R	Methods
SetNumPointsInDisplay	Long W	Methods

#### 8.2.2. Methods

**Table 5: Specific Methods**

Name	May Use After
SetOPOSBCNIBBLESignatureData	Acquiring signature data as LPCSTR
SetOPOSBCNIBBLESignatureDataX	Acquiring signature data as LPCSTR
SetSignatureData	Creating VARIANT with signature data
SetSignatureDataX	Creating VARIANT with signature data

Name	May Use After
GetSignatureType	Methods
GetSignatureTypeString	Methods
WriteSignatureToFile	Methods
ConvertSignatureToImageBuffer	Methods
EnableLiveCapture	Methods
StartLiveCapture	Methods
SetDeviceResolution	Methods

## 8.3. Properties

---

### 8.3.1. GetDrawBorder

Syntax	BOOL GetDrawBorder;
Remarks	<b>GetDrawBorder</b> toggles the drawing of the border around the signature display window. If <b>GetDrawBorder</b> is True (default value), then a border displays. If set to False, no border displays.

### 8.3.2. SetDrawBorder

Syntax	BOOL SetDrawBorder;
Remarks	<b>GetDrawBorder</b> toggles the drawing of the border around the signature display window. If <b>GetDrawBorder</b> is True (default value), then a border displays. If set to False, no border displays.

### 8.3.3. GetDrawBackground

Syntax	<b>BOOL GetDrawBackground;</b>
Remarks	<b>GetDrawBackground</b> toggles the drawing (FillRect(...)) of the background before rendering the signature into the display window. If <b>GetDrawBackground</b> is True (default value), the background drawing displays before the signature is rendered in the display window. If set to False, the signature is rendered first before the background drawing.

### 8.3.4. SetDrawBackground

Syntax	<b>BOOL SetDrawBackground;</b>
Remarks	<b>SetDrawBackground</b> toggles the drawing (FillRect(...)) of the background before rendering the signature into the display window. If <b>SetDrawBackground</b> is True (default value), the background drawing displays before the signature is rendered in the display window. If set to False, the signature is rendered first before the background drawing.

### 8.3.5. GetPenWidth

Syntax	<b>LONG GetPenWidth[Printer];</b>
Remarks	<b>GetPenWidth</b> adjusts the thickness of the pen used to render the signature on the screen or printer device context. <ul style="list-style-type: none"><li>• Default pen width on screen: 1</li><li>• Default pen width on printer: 2</li></ul>

### 8.3.6. SetPenWidth

Syntax	<b>LONG SetPenWidth[Printer];</b>
Remarks	<b>SetPenWidth</b> adjusts the thickness of the pen used to render the signature on the screen or printer device context. <ul style="list-style-type: none"><li>• Default pen width on screen: 1</li><li>• Default pen width on printer: 2</li></ul>

### 8.3.7. GetDisplayNumPoints

Syntax	<b>BOOL GetDisplayNumPoints;</b>
Remarks	<b>GetDisplayNumPoints</b> toggles the textual display of the number of points contained in the signature. The default value is Off. This display is implemented in the display control of the signature window as: <pre>if (m_bDisplayNumPoints) {     CString csTmp;     csTmp.Format("%ld", m_lTotalDisplayedPoints);     pdc-&gt;TextOut(0, 0, csTmp); }</pre>

### 8.3.8. SetDisplayNumPoints

Syntax	<b>BOOL SetDisplayNumPoints;</b>
Remarks	<b>SetDisplayNumPoints</b> toggles the textual display of the number of points contained in the signature. The default value is Off. This display is implemented in the display control of the signature window as: <pre>if (m_bDisplayNumPoints) {     CString csTmp;     csTmp.Format("%ld", m_lTotalDisplayedPoints);     pdc-&gt;TextOut(0, 0, csTmp); }</pre>

## 8.4. Methods

### 8.4.1. SetOPOSBCNIBBLESignatureData

Syntax	<b>void SetOPOSBCNIBBLESignatureData</b> (LPCSTR <i>lpszSigData</i> )
Remarks	<p>This method expects the signature data to be OPOS_BC_NIBBLE binary encoded. The format type of the encoded electronic signature data is autodetected by this function.</p> <p>For example, the data format of the <b>PointArray</b> property from an OPOS SigCap control will always be OPOS_POINT_ARRAY and will conform to OPOS specifications. On the other hand, the data format of the <b>RawData</b> property will be the native data format of the hardware device and depending upon Control Panel configuration can be one of the following:</p> <ul style="list-style-type: none"><li>• CME_4BYTE_RAW</li></ul> <pre>enum enSignatureType {     SIG_NO_DATA,     SIG_NOT_DEFINED,     OPOS_POINT_ARRAY,     CME_4BYTE_RAW, };</pre>

### 8.4.2. SetOPOSBCNIBBLESignatureDataX

Syntax	<b>LONG SetOPOSBCNIBBLESignatureDataX</b> (LPCTSTR <i>lpszSigData</i> , long <i>lSignatureType</i> )
Remarks	<p>This method expects the signature data to be OPOS_BC_NIBBLE binary encoded. This method is similar to the <b>SetOPOSBCNIBBLESignatureData</b> function, except the caller specifies the signature format type as a second parameter. Signature format type recognition is highly reliable, but if the caller knows the signature format type, it is recommended they make use of this function over the previous version of this control.</p> <p>For example, the data format of the <b>PointArray</b> property from an OPOS SigCap control will always be OPOS_POINT_ARRAY and will conform to OPOS specifications. On the other hand, the data format of the <b>RawData</b> property will be the native data format of the hardware device and depending upon Control Panel configuration can be one of the following:</p> <ul style="list-style-type: none"><li>• CME_4BYTE_RAW</li></ul> <pre>enum enSignatureType {     SIG_NO_DATA,     SIG_NOT_DEFINED,     OPOS_POINT_ARRAY,     CME_4BYTE_RAW, };</pre>
Return	<p>One of the following values is returned by the method:</p> <ul style="list-style-type: none"><li>• 0 = Success</li><li>• Any other value = Failure</li></ul>

### 8.4.3. SetSignatureData

Syntax	<code>void <b>SetSignatureData</b> (const VARIANT&amp; varSigData, long lBinaryConversion);</code>
Remarks	<p>This method is designed to take a VARIANT argument containing the electronic signature data. This VARIANT can be any supported Ingenico signature type, and any supported OPOS binary conversion. The <b>SetSignatureData</b> method performs auto detection of the Ingenico or OPOS signature type.</p> <p>When using these functions, it is necessary to provide the OPOS binary conversion to the signature display control so it can convert the data back into a raw data format. OPOS recognizes 3 binary conversion techniques that correspond to the BinaryConversion property on the OPOS SigCap control.</p> <p>The standard OPOS binary conversion constants are:</p> <pre>// OPOS Defined Binary Conversion Constants const LONG OPOS_BC_NONE           = 0; const LONG OPOS_BC_NIBBLE         = 1; const LONG OPOS_BC_DECIMAL        = 2;</pre>

### 8.4.4. SetSignatureDataX

Syntax	<code>LONG <b>SetSignatureDataX</b> (const VARIANT&amp; varSigData, long lBinaryConversion, long lSignatureType)</code>
Remarks	<p>This method is designed to take a VARIANT argument containing the electronic signature data. This VARIANT can be any supported Ingenico signature type, and any supported OPOS binary conversion. The SetSignatureDataX method allows the caller to specify the Ingenico or OPOS signature type, while the SetSignatureData method performs autodetection of the Ingenico or OPOS signature type.</p> <p>When using these functions, it is necessary to provide the OPOS binary conversion to the signature display control can convert the data back into a raw data format. OPOS recognized 3 binary conversion techniques that will correspond to the BinaryConversion property on the OPOS SigCap or OPOS (Extension) Form control.</p> <p>The standard OPOS binary conversion constants are:</p> <pre>// OPOS Defined Binary Conversion Constants const LONG OPOS_BC_NONE           = 0; const LONG OPOS_BC_NIBBLE         = 1; const LONG OPOS_BC_DECIMAL        = 2;</pre>
Return	<p>One of the following values is returned by the method:</p> <ul style="list-style-type: none"><li>• 0 = Success</li><li>• Any other value = Failure</li></ul>

### 8.4.5. GetSignatureType

Syntax	short <b>GetSignatureType</b>
Remarks	<p>Once <b>SetSignatureData[X]</b> or <b>SetOPOSBCNIBBLESignatureData[X]</b> have been called, <b>GetSignatureType</b> and <b>GetSignatureTypeString</b> will return the interrogated signature type to the caller.</p> <p>Return values will be one of the following:</p> <ul style="list-style-type: none"><li>• OPOS_POINT_ARRAY</li><li>• CME_4BYTE_RAW</li><li>• SIG_NOT_DEFINED</li></ul>

### 8.4.6. GetSignatureTypeString

Syntax	BSTR <b>GetSignatureTypeString</b>
Remarks	<p>Once <b>SetSignatureData[X]</b> or <b>SetOPOSBCNIBBLESignatureData[X]</b> have been called, <b>GetSignatureType</b> and <b>GetSignatureTypeString</b> will return the interrogated signature type to the caller.</p> <p>Return values will be one of the following:</p> <ul style="list-style-type: none"><li>• OPOS_POINT_ARRAY</li><li>• CME_4BYTE_RAW</li><li>• UNKNOWN_SIGNATURE_TYPE</li></ul>

### 8.4.7. WriteSignatureToFile

Syntax	LONG <b>WriteSignatureToFile</b> (LPCTSTR <i>lpszOutputFile</i> , long <i>lOutputFormat</i> , long <i>lOutputWidth</i> , long <i>lOutputHeight</i> , BOOL <i>bDrawBorder</i> )
Remarks	<p>This function can be used to render the captured signature data to either a monochrome TIFF or BMP file.</p> <p>The required parameters are the following:</p> <ul style="list-style-type: none"><li>• <i>lpszOutputFile</i> - Specifies a valid file name for the output graphic file. The <i>lOutputFormat</i> can be one of the following:<ul style="list-style-type: none"><li>— #define FF_TIFF = 0</li><li>— #define FF_BMP = 1</li></ul></li><li>• <i>lOutputWidth</i> - Specifies the width of the output image file.</li><li>• <i>lOutputHeight</i> - Specifies the height of the output image file.</li><li>• <i>bDrawBorder</i> - Specifies whether the output graphic has a one-pixel border around the enclosing rectangle.</li></ul>

### 8.4.8. ConvertSignatureToImageBuffer

Syntax	BSTR <b>ConvertSignatureToImageBuffer</b> (long <i>lOutputFormat</i> , long <i>lOutputWidth</i> , long <i>lOutputHeight</i> , BOOL <i>bDrawBorder</i> )
Remarks	<p>This function can be used to render the captured signature data as a monochrome TIFF or BMP and store it in a buffer. This buffer is returned to the caller.</p> <p>The required parameters are the following:</p> <ul style="list-style-type: none"><li>• <i>lOutputFormat</i> - Can be one of the following:</li></ul>

- #define FF\_TIFF = 0
- #define FF\_BMP = 1

- *IOutputWidth* - Specifies the width of the image contained in the buffer.
- *IOutputHeight* - Specifies the height of the image contained in the buffer.
- *bDrawBorder* - Specifies whether the output graphic has a one-pixel border around the enclosing rectangle.

In a .NET environment, the buffer returned by this function is stored in a string object. By default .NET will encode this data into Unicode, which may cause some bytes to be changed. As a result, the signature may appear blurry or contain noise. The following code snippet in C# will convert the buffer data back to a Cp1252 encoding, which eliminates this problem.

```
string          buffer          =
sigDisplay.ConvertSignatureToImageBuffer(FF_BMP,  nWid,  nHgt,
true);

// encoding we've been switched to
Encoding unicode = Encoding.Unicode ;

// target encoding
Encoding extAscii = Encoding.GetEncoding(1252) ;

byte[] unicodeBytes = unicode.GetBytes(buffer) ;

// this will now contain the desired image data
byte[] extAsciiBytes = Encoding.Convert(unicode,  extAscii,
unicodeBytes) ;
```

## Notes



## 9. PIN Pad

---

### 9.1. General Information

---

Ingenico's implementation of the PIN Pad control follows the version 1.13 specifications for UnifiedPOS controls exactly.

### 9.2. Encryption Key Formats

---

#### 9.2.1. Master/Session Key Serial Number Format

Master/Session is a key management scheme in which a pre-shared Key Encrypting Key (called the "Master") is used to encrypt a randomly generated and insecurely communicated Working Key (called the "Session" key). The Working Key is then used to encrypting data to be exchanged.

This technique still finds widespread use in the financial industry, although its use in device communications is in decline given the advantages of techniques such as DUKPT.

#### 9.2.2. DUKPT Key Serial Number Format

During PIN transactions that use Derived Unique Key Per Transaction (DUKPT) key management, a key serial number (KSN) is returned from the PIN pad and stored in the **AdditionalSecurityInformation** property of the OPOS PINPad control when the Enter key is pressed. This property is a hex-formatted ASCII string 20 characters in length.

For example, if the KSN can be expressed in hexadecimal as 0xFFFF9876543210E000A, **AdditionalSecurityInformation** will report 'FFFF9876543210E000A'.

### 9.3. Usage Samples

---

#### 9.3.1. Handle PIN Entry

Confirm the key slot that has the PIN encryption key. Ingenico PIN pads only support the DUKPT encryption algorithm.

The application can specify the form to be used when PIN entry is enabled. The file name must be specified in the *IngenicoConfig.xml*.

```

- <Items>
  - <SignatureForm>
    <Name>Sigform</Name>
    <Value>SIGNATURE.HTM</Value>
  </SignatureForm>
</Items>
</SignatureCaptureForms>
- <PinEntryForms>
  - <Items>
    - <PinEntryForm>
      <Name>PINENTRY</Name>
      <Value>PINENTRY.HTML</Value>
    </PinEntryForm>
  </Items>
</PinEntryForms>
- <LineDisplayForms>
  - <Items>
    - <LineDisplayForm>
      <Name>Window0</Name>
      <Value>SCROLL.HTM</Value>
    </LineDisplayForm>
  </Items>
</LineDisplayForms>
- <IcgForms>
  - <Items>

```

The code sample below shows how to use PIN pad:

```

PINPad.SetAccountNumber("1234567890");
PINPad.SetAmount(COLECurrency(90,5000);

long lResponseCode =
PINPad.BeginEFTTransaction(csEncryptionType,nSlot);
if (lResponseCode != OPOS_SUCCESS)
{
    //handle error
}
else
{
    PINPad.SetDataEventEnabled(TRUE);
}

```

After BeginEFTTransaction returns successfully, call the following to start PIN entry:

```

lResponseCode = PINPad.EnablePINEntry();

```

### 9.3.2. Request Key ID or Key Slot Information

The following code sample shows how to request Key ID or Key Slot information:

```

long nCmd = 0; // using raw data format
LONG Data = 0;
long* pData = &Data;

long nResult;

```

```

WORD get_key_status[6] = {0x05, 0x06, 0xA3, 0x03, 0x00,
0xA3};

// 0x3 key slot number, this value could be 0x0 - 0x09
// 0x00 Master/Session Type
// 0x01 DUKPT Type
// 0xA3 Data Validity Check Value (this value will change
// when the key slot number changes. This is an exclusive
// OR of the previous 5 bytes of the message)

    m_PINPad.Open("Telium2");

m_PINPad.ClaimDevice (5000);
m_PINPad.SetDeviceEnabled(TRUE);
m_PINPad.SetBinaryConversion(OPOS_BC_NIBBLE) ;

WORD pFirst[13];

// (length of buffer to transmit * 2) + 1

EncodeBCNibble(6, get_key_status, pFirst) ;
BSTR pString = ::SysAllocString(pFirst);
nResult = m_PINPad.DirectIO(nCmd, pData, &pString);
m_PINPad.SetBinaryConversion(OPOS_BC_NONE) ;

// nResult == 0 (key slot contains key),
// nResult == 0xF7 (key slot does not contain key)

void EncodeBCNibble(int nOrigLen, WORD* pOrig, WORD* pDest)
{
    for (int j = 0; j < nOrigLen; j++)
    {
        pDest[2*j] = 0x30 + ( (pOrig[j] & (WORD)0xF0) >> 4);
        pDest[2*j+1] = 0x30 + ( pOrig[j] & (WORD)0x0F) ;
    }

    // terminate with NULL
    pDest[2*nOrigLen] = 0 ;
}

```

### 9.3.3. Retrieve PIN Block

The following code sample shows how to retrieve the PIN block:

```

// perform code under Handle PIN entry
// wait for PINPad Data or Error Event to be fired
CString PINString = m_PINPad.GetEncryptedPIN();

```

## Notes


## 10. OPOS Usage Samples

---

This chapter includes OPOS usage samples specific to the Ingenico OPOS driver. For general information on how to use UPOS Control Objects, refer to the UnifiedPOS version 1.13 specification. For all devices, Open, Claim, Enable are required before the following samples can be used.


### 10.1. Implementing a Terms and Conditions Screen

---

 Full implementation example planned for a later release.


### 10.2. Implementing a Survey Screen

---

 Full implementation example planned for a later release.

### 10.3. Simulating a Transaction

---

 Full implementation example planned for a later release.

### 10.4. Using OPOS for the .NET Application

---

.NET provides backward compatibility for the OPOS driver. There are two basic ways to use the OPOS driver in a .NET application: use PosExplorer or use .NET Interop. Refer to [http://monroecs.com/posfordotnet/opos\\_dotnet.htm](http://monroecs.com/posfordotnet/opos_dotnet.htm) for information.

#### 10.4.1. PosExplorer

First add reference a to Microsoft.PointOfService. Then add the following line to the source file:

```
"using Microsoft.PointOfService;"
```

The following is a code snippet:

```
//Create a PosExplorer object
PosExplorer = new PosExplorer(this);

//Retrieve a list of OPOS devices
DeviceCollection devices =
posExplorer.GetDevices((DeviceCompatibilities)Enum.Parse(typeof(D
eviceCompatibilities), DeviceCompatibilities. Opos.ToString(),
false));

//Find the device want to use
foreach(DeviceInfo device in devices)
{
    if (device.Type == "type wanted" &&
device.ServiceObjectName.Equals(Devicecename))
    {
        //found it. Break;
```

```

        break;
    }
}

//Create a PosDeviceTag
posInstance = (POSCommon)explorer.CreateInstance(device);
posInstance.Open();
posInstance.Claim(2000);
posInstance.DeviceEnabled = true;

//call OPOS methods.

//close the device
posInstance.Close();
posInstance = null;

```

### 10.4.2. .NET Interop

.NET Interop can be done through Visual Studio:

- Choose “Add Reference”. Select the COM tab from the property page and choose the OPOS control you want to use. It will create a wrapper class for the OPOS control you select.
- After that, create an instance of the wrapper class and call OPOS methods.

## 11. UIA Prompts

Prompts for various Ingenico PIN pad applications are found in the PROMPTS.xml and CUSTPROMPTS.xml files. All UIA prompts files must be signed by Ingenico before they can be loaded onto a PIN pad.

### 11.1. Prompts (PROMPTS.xml)

Prompts for various Ingenico PIN pad applications are found in the PROMPTS.xml file. The PROMPTS.xml file MUST be signed by Ingenico before being loaded on an Ingenico PIN pad.

The PROMPTS.xml file is subdivided into sections containing several different types of prompts. For the purposes of this User's Guide, we will only cover the prompts related to OPOS/UIA.

Prompt Type	Prompt ID	Default Value	Description
n/a	0	PROMPTINDEX	
UIA PIN Prompts	1	"Please enter PIN"	
		"Por favor, introduzca su PIN"	
		"Entrez votre NIP SVP"	
UIA PIN Prompts	3	"Enter valid PIN"	
		"PIN no válido. Vuelva a entrar"	
		"NIP erroné. Réessayez"	
Other PIN Prompts	21	"ENTER PIN"	
		"ENTRAR EL PIN"	
		"ENTRER PIN"	
Other PIN Prompts	22	"ENTER PIN & ENTER"	
		"ENTRAR EL PIN & ENTER"	
		"ENTRER PIN & ENTRER"	
Other PIN Prompts	23	"ENTER PIN & PRESS ENTER"	
		"ENTRAR EL PIN & PULSE ENTER"	
		"ENTRER PIN & PRESSE ENTRER"	

Prompt Type	Prompt ID	Default Value	Description
Other PIN Prompts	24	"ENTER PIN AND PRESS ENTER"	
		"ENTRAR EL PIN Y PULSE ENTER"	
		"ENTRER PIN ET APPUYEZ SUR ENTRER"	
Other PIN Prompts	25	"REENTER PIN"	
		"VUELVA A INTRODUCIR EL PIN"	
		"RETOURNER PIN"	
Other PIN Prompts	26	"REENTER PIN & ENTER"	
		"VUELVA A INTRODUCIR EL PIN & ENTER"	
		"RETOURNER PIN & ENTRER"	
Other PIN Prompts	27	"REENTER PIN & PRESS ENTER"	
		"VUELVA A INTRODUCIR EL PIN & PULSE ENTER"	
		"RETOURNER PIN & PRESSE ENTRER"	
Other PIN Prompts	28	"REENTER PIN AND PRESS ENTER"	
		"VUELVA A INTRODUCIR EL PIN y PULSE ENTER"	
		"RETOURNER PIN ET APPUYEZ SUR ENTRER"	
Other PIN Prompts	29	"PIN="	
		"PIN="	
		"PIN="	
Other PIN Prompts	30	"Please Enter Your PIN and Press &Enter&apos;"	
		"Introduzca su PIN y pulse &Enter&apos;"	
		"SVP entrer votre code PIN et appuyez sur &Enter&apos;"	
Other Clear Entry Prompts	201	"Enter Month and Day of Birth (MMDD)"	
		"Ingrese mes y día de nacimiento (MMDD)"	
		"Entrez le mois et le jour de naissance (MMDD)"	
Other Clear Entry Prompts	202	"Enter Home Phone Number"	
		"Ingrese el número de teléfono de la casa"	
		"Entrez le numéro de téléphone à domicile"	



Prompt Type	Prompt ID	Default Value	Description
Other Clear Entry Prompts	203	"Enter Last 4 Digits of Social Security #"	
		"Introduzca los 4 últimos dígitos de SSN"	
		"Entrez 4 derniers chiffres de la SSN"	
Other Clear Entry Prompts	204	"Annual Income (from all sources)"	
		"Ingreso anual (de todas las fuentes)"	
		"Revenu annuel (toutes sources confondues)"	
Other Clear Entry Prompts	205	"Enter Social Security #"	
		"Introduzca SSN"	
		"Entrez la SSN"	
Other Clear Entry Prompts	206	"Please enter your 10-digit HOME phone number"	
		"Introduzca su número de teléfono de la casa"	
		"Entrer votre numéro de téléphone"	
Other Clear Entry Prompts	207	"Social Security Number"	
		"Número de Seguro Social"	
		"Numéro de Sécurité Sociale"	
Other Clear Entry Prompts	208	"Annual Income Amount"	
		"Monto de ingresos anuales"	
		"Montant du revenu annuel"	
Other Clear Entry Prompts	209	"Years lived at Current Address"	
		"Años vivió en la dirección actual"	
		"Années vécues à l'adresse actuelle"	
Other Clear Entry Prompts	210	"Months lived at Current Address"	
		"Meses vivió en la dirección actual"	
		"Mois vécu à l'adresse actuelle"	
Other Clear Entry Prompts	211	"Mothers Maiden Name"	
		"Nombre de soltera de la madre"	
		"Mères Nom de jeune fille"	
Other Clear Entry Prompts	212	"Email Address"	
		"Dirección de correo electrónico"	
		"Adresse e-mail"	
Other Clear Entry Prompts	213	"Date of Birth"	
		"Fecha de Nacimiento"	
		"Date de naissance"	

Prompt Type	Prompt ID	Default Value	Description
Other Clear Entry Prompts	214	"Home Phone Number"	
		"Teléfono de la casa"	
		"Numéro de téléphone résidentiel"	
Other Clear Entry Prompts	215	"Zip Code"	
		"Código postal"	
		"Code postal"	
Other Clear Entry Prompts	216	"Use the attached pen to type mother's maiden name"	
		"Utilice el lápiz unido a las madres escriba el nombre de soltera"	
		"Utilisez le stylo attaché aux mères de type nom de jeune fille"	
Other Clear Entry Prompts	217	"Please enter your annual household income"	
		"Introduzca su ingreso familiar anual"	
		"S'il vous plaît entrer votre revenu annuel du ménage"	
Other Clear Entry Prompts	218	"Please enter your date of birth"	
		"Introduzca su fecha de nacimiento"	
		"S'il vous plaît entrer votre date de naissance"	
Other Clear Entry Prompts	219	"Please enter your social security number"	
		"Introduzca su número de seguro social"	
		"S'il vous plaît entrer votre numéro de sécurité sociale"	
Other Clear Entry Prompts	220	"Enter two-digit number of months"	
		"Introduzca el número de dos cifras de los meses"	
		"Entrez le numéro à deux chiffres du mois"	
Other Clear Entry Prompts	221	"Enter two-digit number of years"	
		"Introduzca el número de dos dígitos de años"	
		"Entrez le numéro à deux chiffres des années"	

Prompt Type	Prompt ID	Default Value	Description
Other Clear Entry Prompts	222	"Please enter your home phone number"	
		"Introduzca su número de teléfono de su casa"	
		"S'il vous plaît entrer votre numéro de téléphone à la maison"	

Below is an example of the initial portion of the default PROMPTS.XML file:

```

<?xml version="1.0" ?>
- <PromptDefinition>
  - <Pin>
    <Prompt id="0" message="PROMPTINDEX" />
    <!-- UIA PIN Prompts (1-13) -->
    <Prompt id="1" message="Please enter PIN" />
    <Prompt id="3" message="Enter Valid PIN" />
    <!-- RBA PIN Prompts (14-20) -->
    <Prompt id="14" message="Please enter your PIN:" />
    <Prompt id="15" message="Invalid PIN. Please re-enter" />
    <!-- Other PIN Prompts (21-49) -->
    <Prompt id="21" message="ENTER PIN" />
    <Prompt id="22" message="ENTER PIN & ENTER" />
    <Prompt id="23" message="ENTER PIN & PRESS ENTER" />
    <Prompt id="24" message="ENTER PIN AND PRESS ENTER" />
    <Prompt id="25" message="REENTER PIN" />
    <Prompt id="26" message="REENTER PIN & ENTER" />
    <Prompt id="27" message="REENTER PIN & PRESS ENTER" />
    <Prompt id="28" message="REENTER PIN AND PRESS ENTER" />
    <Prompt id="29" message="PIN=" />
    <Prompt id="30" message="Please Enter Your PIN and Press 'Enter'" />
  </Pin>
  - <Clear>
    <Prompt id="0" message="PROMPTINDEX" />
    <!-- RBA Clear Entry Prompts (1-200) -->
    <Prompt id="1" message="Enter home phone number" />
    <Prompt id="16" message="Please enter Cashback:" />
    <Prompt id="19" message="Please enter new amount:" />
  </Clear>
</PromptDefinition>

```

Figure 8 - the default PROMPTS.xml file.

## 11.2. Prompts (CUSTPROMPTS.xml)

The CUSTPROMPTS.xml file is an optional file which follows the same format as PROMPTS.xml (except it does not have a PIN section). This CUSTPROMPTS.xml file allows clients to define their own custom prompts for use with UIA.

CUSTPROMPTS.xml files must be packaged as signed PGZ files before they can be loaded on an Ingenico PIN pad.

## Notes

# Appendix A. Differences Between U32 OPOS and Telium OPOS

---

This Appendix lists differences between the latest U32 and Telium versions of the OPOS Integration Kit.

## A.1. At A Glance

---

Some key differences between the U32 OPOS and the Telium OPOS implementations are:

Feature	U32 OPOS	Telium OPOS
OCX control version(s)	v1.7 or v1.8	v1.13
# of devices supported in control panel	Single device	Multiple
Data signing	N/A	Required
EMV reader	Supported	Unsupported
Configuration	Registry-based	XML-based
Power Management	Supported	Unsupported
Form management	Through Form Control OCX control	Through Direct I/O
Firmware download	Enabled through Maintenance tab in control panel	Initiated through Direct I/O via SAVE_FILE
Form file format	.ICG	.K3Z
Form file packaging	N/A	Signed zip file (.PGZ)
Line display	Single scrolling window and DisplayTextAt	Multiple line display windows and DisplayTextAt

 Support for some of these features may be added in future releases.

## A.2. OPOS Control Support

---

### A.2.1. Form Control

The Telium implementation has removed support for the OPOS Form Control (IVICMForm.ocx).

### A.2.2. Direct I/O

The Telium implementation incorporates changes to the names and functions of commands recognized by the DirectIO OPOS control. For more information on current Direct I/O commands, see Direct I/O on page 43.

## A.3. OPOS Control Panel

---

Changes in the Telium implementation include the following:

- Added Connection tab to manage PIN pad communications type settings.
- Removed Maintenance tab.
- Removed Application Flow tab.
- Removed Form tab.

For more information on the configuration options available on each tab in the OPOS Control Panel, see OPOS Configuration on page 26.

### A.3.1. Installation

The new Telium OPOS silent install batch file requires modification in order to overwrite previously existing Ingenico configuration files for reinstallations or upgrades.

See Configuring a Silent Install on page 25 for more information.

### A.3.2. General Tab

Changes in the Telium implementation include the following:

- Added **Logical Device Names** section, which allows users to manage multiple Telium devices using the OPOS Control Panel.
- Removed **Device Connection** section (moved to Connection tab).
- Removed **Enable Backlight to Power Off During Inactivity** checkbox.
- Removed **Interval Inactivity** slider (moved to Connection tab).

### A.3.3. PIN Pad Tab

Changes in the Telium implementation include the following:

- Removed **Use form during PIN entry** checkbox.
- Removed **Check existence** checkbox.
- Removed **Cryptographic Key Management** section.
- Removed **Prompt Management** section.
- Added **PIN Entry min/max input digits** section.
- Added **Clear Screen after Pin Entry** checkbox.

### A.3.4. MSR Tab

Changes in the Telium implementation include the following:

- Added **Form name** field.
- Added **Enable form name** checkbox.
- Added **Card Prompt** field.

### A.3.5. Signature Tab

Changes in the Telium implementation include the following:

- Added **Form name** field.
- Removed **Signature Format** section.
- Removed **Check file existence during store form on device** checkbox.

### A.3.6. Line Display Tab

Changes in the Telium implementation include the following:

- Added **Form name** field.
- Added **Enable form name** checkbox.
- Added **Clear all windows during Claim** checkbox.
- Added **Add new line** checkbox.
- Removed **Device Model** section.
- Removed **Line Display** mode section.
- Removed **Behavior of LineDisplay** methods section.
- Removed **Default colors assigned at Claim()** section.

#### A.3.6.1. Debug Tab

Changes in the Telium implementation include the following:

- Removed **Use the following Polling Rate** checkbox.
- Removed **Interval Inactivity** slider.

## A.4. OPOS Test Application

---

The Telium release of the OPOS Integration Kit added the OPOS Test Application, which allows customers to drive PIN pad behavior using OPOS controls.

For more information on the OPOS Test Application, see OPOS Test Application on page 95.

## Notes



## Appendix B. U.S. Retail Telium Download Application (TDA)

---

The U.S. Retail Telium Download Application (TDA) is an Ingenico PIN pad application that handles the following functionality:

- Communication setting configuration
- Initial download and updates for the following PIN pad software:
  - Financial applications (for PIN pads that do not already have one loaded)
  - Telium operating system
  - Libraries
  - Telium Manager

### B.1. Accessing the Telium Download Application

---

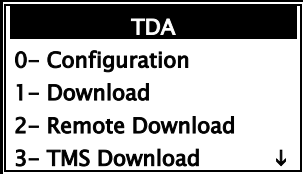
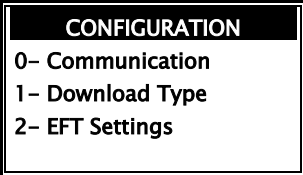
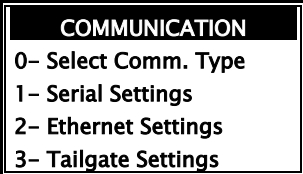
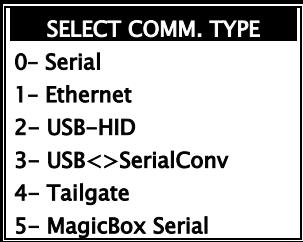

To access the Telium Download Application on your Ingenico PIN pad, follow these steps:

1. Power up the Ingenico PIN pad.
2. Wait for the gray UPOS Interface Application splash screen (see Figure 3 on page 11).
3. At the splash screen, press the key combination that corresponds to your PIN pad model:
  - **iSC350** - [2], [6], [3], [4], [Enter], [+]
  - **iPP3xx** - [2], [6], [3], [4], [Enter], [\*]
4. At FUNCTIONS screen, navigate to the TDA icon using your PIN pad's menu navigation keys:
  - **iSC350** – [-] and [+]
  - **iPP3xx** – [F2] and [F3]
5. **Once you have selected the TDA icon, press [Enter].** The main TDA menu displays.

## B.2. Configuring PIN Pad Communication Settings

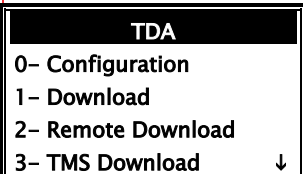
### B.2.1. Setting Communication Type

To configure a PIN pad to use a serial connection, follow these steps:

Step	PIN Pad Display	Merchant Action	Notes
1.		From the main TDA screen, press [0] -or- Press [Enter] for Configuration.	
2.		Press [0] -or- Press [Enter] for Communication.	
3.		Press [0] for Select Comm. Type.	
4.		Press the number corresponding to the desired communications type, then press [Cancel] or [Clear] three times.	Options 2 (USB-HID) not supported for the iSC250 PIN pad. Option 5 (MagicBox Serial) is only available for the iPP320 and the iPP350.
5.		Press [1] for Yes to save any updated settings and reboot the PIN pad.	

### B.2.2. Configuring Serial Connection Settings

To configure serial connection settings, follow these steps:

Step	PIN Pad Display	Merchant Action	Notes
1.		From the main TDA screen, press [0] -or- Press [Enter] for Configuration.	

Step	PIN Pad Display	Merchant Action	Notes
2.	<b>CONFIGURATION</b> 0- Communication 1- Download Type 2- EFT Settings	Press [0] -or- Press [Enter] for Communication.	
3.	<b>COMMUNICATION</b> 0- Select Comm. Type 1- Serial Settings 2- Ethernet Settings 3- Tailgate Settings	Press [1] for Serial Settings.	
4.	<b>SERIAL</b> 0- Baud Rate 1- Stop Bit 2- Bits Per Byte 3- Parity ↓	Press [0] for Baud Rate.	
5.	<b>SELECT BAUD RATE</b> 0- 115200 1- 57600 2- 38400 3- 19200	Select the desired baud rate, then press [Cancel] or [Clear] to return to the Serial Settings menu.	
6.	<b>SERIAL</b> 0- Baud Rate 1- Stop Bit 2- Bits Per Byte 3- Parity ↓	Press [1] for Stop Bit.	
7.	<b>SELECT STOP BIT</b> 0- 1 1- 2	Select the desired stop bit, then press [Cancel] or [Clear] to return to the Serial Settings menu.	
8.	<b>SERIAL</b> 0- Baud Rate 1- Stop Bit 2- Bits Per Byte 3- Parity ↓	Press [2] for Bits Per Byte.	
9.	<b>SELECT BITS PER BYTE</b> 0- 7 1- 8	Select the desired number of bits per byte, then press [Clear] to return to the Serial Settings menu.	
10.	<b>SERIAL</b> 0- Baud Rate 1- Stop Bit 2- Bits Per Byte 3- Parity ↓	Press [3] for Parity.	

Step	PIN Pad Display	Merchant Action	Notes
11.	<b>SELECT PARITY</b> 0- NONE 1- ODD 2- EVEN	Select the desired parity setting, then press [Cancel] or [Clear] to return to the Serial Settings menu.	
12.	<b>SERIAL</b> 0- Baud Rate 1- Stop Bit 2- Bits Per Byte 3- Parity ↓	Press [F2] or [-] four times -or- Press [4] for Flow Control.	
13.	<b>SELECT FLOW CONTROL</b> 0- Hardware 1- None	Select the desired flow control setting, then press [Cancel] or [Clear] four times.	
14.	<b>SAVE AND REBOOT?</b> 0- No 1- Yes	Press [1] for Yes to save any updated settings and reboot the PIN pad.	

### B.2.3. Configuring Ethernet Connection Settings

To configure Ethernet connection settings, follow these steps:

Step	PIN Pad Display	Merchant Action	Notes
1.	<b>TDA</b> 0- Configuration 1- Download 2- Remote Download 3- TMS Download ↓	From the main TDA screen, press [0] -or- Press [Enter] for Configuration.	
2.	<b>CONFIGURATION</b> 0- Communication 1- Download Type 2- EFT Settings	Press [0] -or- Press [Enter] for Communication.	
3.	<b>COMMUNICATION</b> 0- Select Comm. Type 1- Serial Settings 2- Ethernet Settings 3- Tailgate Settings	Press [2] for Ethernet Settings.	
4.	<b>ETHERNET</b> 0- Connection Type 1- DHCP 2- Host IP Address 3- IP Address ↓	Press [0] for Connection Type.	

Step	PIN Pad Display	Merchant Action	Notes
5.	<b>CONNECT AS</b> 0- Client 1- Server	Select the desired connection type, then press [Cancel] or [Clear] to return to the Ethernet Settings menu.	
6.	<b>ETHERNET</b> 0- Connection Type 1- DHCP 2- Host IP Address 3- IP Address ↓	Press [1] for DHCP.	
7.	<b>SELECT DHCP. TYPE</b> 0- Auto 1- Static	Select the desired DHCP type, then press [Cancel] or [Clear] to return to the Ethernet Settings menu.	
8.	<b>ETHERNET</b> 0- Connection Type 1- DHCP 2- Host IP Address 3- IP Address ↓	Press [2] for Host IP Address.	
9.	<b>Enter Host IP</b> 000 .000 .000 .000 -	Enter the host's IP address, then press [Enter] to return to the Ethernet Settings menu.	
10.	<b>ETHERNET</b> 0- Connection Type 1- DHCP 2- Host IP Address 3- IP Address ↓	Press [3] for IP Address.	
11.	<b>Enter IP Address</b> 192 .168 .002 .002 -	Enter the desired IP address, then press [Enter] to return to the Ethernet Settings menu.	
12.	<b>ETHERNET</b> 0- Connection Type 1- DHCP 2- Host IP Address 3- IP Address ↓	Press [F2] or [-] four times, then [Enter] -or- Press [4] for Subnet Mask.	
13.	<b>Enter Subnet Mask</b> 255 .255 .255 .000 -	Enter the desired subnet mask, then press [Enter] to return to the Ethernet settings menu.	

Step	PIN Pad Display	Merchant Action	Notes
14.	<b>ETHERNET</b> 0- Connection Type 1- DHCP 2- Host IP Address 3- IP Address ↓	Press [F2] or [-] five times, then [Enter] -or- Press [5] for Gateway.	
15.	<b>Enter Gateway</b> 192 .168 .001 .001 -	Enter the desired gateway IP, then press [Enter] to return to the Ethernet settings menu.	
16.	<b>ETHERNET</b> 0- Connection Type 1- DHCP 2- Host IP Address 3- IP Address ↓	Press [F2] or [-] six times, then [Enter] -or- Press [6] for IP Port.	
17.	<b>IP PORT</b> Current value= 12000	Enter the desired IP port, then press [Enter] to return to the Ethernet settings menu.	
18.	<b>ETHERNET</b> 0- Connection Type 1- DHCP 2- Host IP Address 3- IP Address ↓	Press [F2] or [-] seven times, then [Enter] -or- Press [7] for IP Display.	
19.	<b>DISPLAY IP INFO</b> 0- NO 1- YES	Select whether or not to display IP address information, then press [Cancel] or [Clear] to return to the Ethernet Settings menu.	
20.	<b>ETHERNET</b> 0- Connection Type 1- DHCP 2- Host IP Address 3- IP Address ↓	Press [Cancel] or [Clear] 3 times to exit TDA.	
21.	<b>SAVE AND REBOOT?</b> 0- No 1- Yes	Press [1] for Yes to save any updated settings and reboot the PIN pad.	

### B.2.4. Configuring Tailgate Connection Settings

To configure tailgate connection settings, follow these steps:

Step	PIN Pad Display	Merchant Action	Notes
1.	<b>TDA</b> 0– Configuration 1– Download 2– Remote Download 3– TMS Download ↓	From the main TDA screen, press [0] -or- Press [Enter] for Configuration.	
2.	<b>CONFIGURATION</b> 0– Communication 1– Download Type 2– EFT Settings	Press [0] -or- Press [Enter] for Communication.	
3.	<b>COMMUNICATION</b> 0– Select Comm. Type 1– Serial Settings 2– Ethernet Settings 3– Tailgate Settings	Press [3] for Tailgate Settings.	
4.	<b>TAILGATE</b> 0– Address	Press [0] for Address.	
5.	<b>SELECT ADDRESS</b> 0– 64h 1– 65h 2– 68h 3– 69h	Select the desired address, then press [Cancel] or [Clear] four times.	
6.	<b>SAVE AND REBOOT?</b> 0– No 1– Yes	Press [1] for Yes to save any updated settings and reboot the PIN pad.	

### B.3. Exiting the Telium Download Application

To exit the Telium Download Application, follow these steps:

- At the main TDA menu, press [Cancel] once, then press the key that corresponds to your PIN pad model to exit the application:
  - iSC350 – [ + ]
  - iPP350 – [ F ]
- If you...
  - ...made changes to PIN pad settings using TDA, press [1] for Yes to save your settings and reboot the PIN pad.

- b.** ... did not make changes, press [0] or [Enter] for No to return to the PIN pad applications menu.



## Appendix C. 5992 Mode

---

5992 Mode is a UIA operating mode enabled by setting the UIA variable `5992_MODE = 1`. This mode was implemented in order to allow UIA to support NCR 5992 functionality required for Telium terminals to integrate with NCR POS.

The message protocol between the Ingenico OPOS driver and Telium UIA is based on the NCR 5992 protocol.

### C.1. Special Functions

---

The NCR 5992 “Display Text at Row, Column” (0x34) command has been implemented as a DirectIO command for the OPOS SigCap control.

See Display Text At at page 53 for additional information.

### C.2. Line Displays

---

NCR systems display all text on line display screens using the NCR 5992 0x34 command. Multiple fonts can be used on the same screen.

When placed in 5992 mode, UIA uses the line display as the default screen in order to replicate the functionality available in NCR U32 systems. The 5992 line display form meets the following requirements:

- The form contains  $n$  label elements.
- Each label element represents one row of screen text.
- The labels are evenly spaced. The vertical distance between labels is defined by the `LDPRESETHEIGHT` variable.
- The width of each label element matches the width of the display, in pixels.
- The font family, weight, and size for each label are overridden by the font values specified in 0x34 messages.
- The ID value of each label is `LDn_Lm`, where  $n$  is the line display number and  $m$  is the row number (line display values are numbered 1 – 4, rows are numbered 1 –  $n$ ).

### C.3. Fonts

---

UIA has been modified to include a new font called Userfont3 (based on Bitstream Vera Sans Mono) which provides a similar appearance, shape, size, and spacing as the fonts used in NCR U32 systems:

- Userfont3 displayed at 14 pt – approximates the U32 6x8 font.
- Userfont3 displayed at 19 pt – approximates the U32 8x16 font.
- Userfont3 displayed at 30 pt – approximates the U32 Sans Serif 16 font.


## C.4. Required Scripting

A UIA startup script is required to establish full compatibility with NCR 5992 by enabling the required UIA variables. This script displays the 5992 line display form and sets the variables approximately 10 seconds after UIA completes its startup sequence.

The script must be set as a “Startup Script.”

The following variable values MUST be set by the script to ensure full NCR 5992 compatibility:

- CURRENTWINDOW = 1
- LDRPRESETHEIGHT
- MSR\_NO\_LIGHTS\_ON\_ENABLE = 1
- 5992\_MODE = 1

 See the Ingenico Script Builder Tool and corresponding online help documentation for more information on Telium scripting.

## C.5. 5992 Mode Font Tables

### C.5.1. Font Types

Font Family (bits 4-7)	Normal	Bold	Available Point Sizes
Monospace	1x	9x	7, 10, 13, 17, 20
Userfont2 (monospaced)	2x	Ax	9, 10, 12, 13, 17, 24, 36
Sans Serif (proportional)	3x	Bx	7, 10, 13, 17, 20
Serif (proportional)	4x	Cx	7, 10, 13, 17, 20
Userfont1 (proportional)	5x	Dx	6, 9, 10, 12, 13, 17, 24, 36, 48
Userfont3 (monospaced)	6x	Ex	14, 19, 30

### C.5.2. Point Size Values

Point Size (bits 0-3)	Value
6	1x
7	2x
9	3x
10	4x
12	5x
13	6x
14	7x
17	8x
19	9x
20	Ax

Point Size (bits 0-3)	Value
24	Bx
30	Cx
36	Dx
48	Ex

### C.5.3. Examples

To set a line to 24 point Monospace font, use 0x1B.

To set a line to 10 point Userfont2 bold, use 0xA4.

## Notes

## Appendix D. JPOS Test Application

The JPOS Test Application is an Ingenico application used to test JPOS functionality without processing real transactions.

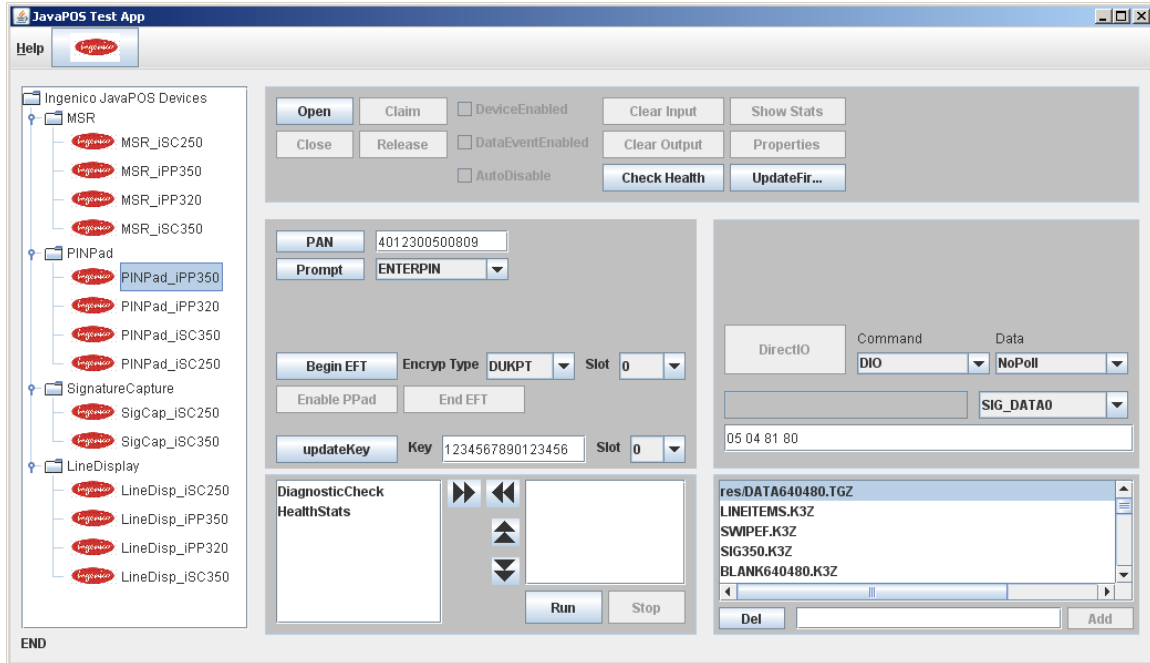


Figure 9 - The Ingenico JavaPOS test application.

### D.1. Getting Started

Launch the JPOS Test Application by executing the JavaPOSTest-RS232.bat file provided with your JPOS Integration Kit.

**Info** Note that the Test Application batch file must specify the correct path for the system's Java Installation in order to work properly. See Setting the Java Path on page 16 for more information on setting the JAVA\_HOME path.

### D.2. Updating JPOS Configuration

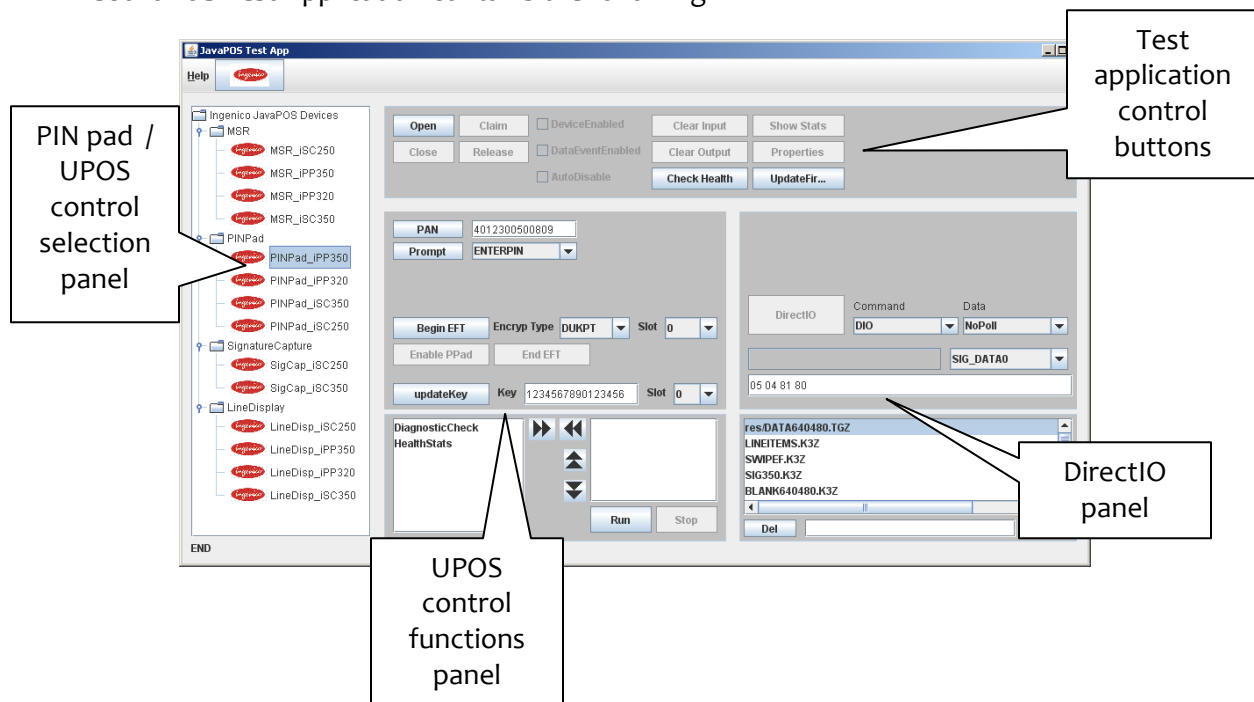
Open Jpos configuration file, ijpos.xml, with a notepad or xml editor. Use the following table to determine connection interface specific properties to be updated.

Interface Type	Properties
RS-232	<code>&lt;prop name="portName" type="String" value="COM1" /&gt;</code>
	<code>&lt;prop name="deviceBus" type="String" value="RS232" /&gt;</code>
	<code>&lt;prop name="baudRate" type="String" value="115200" /&gt;</code>

Interface Type	Properties
USB CDC	<pre>&lt;prop name="portName" type="String" value="COM5" /&gt;</pre> <p><b>info</b> This property must specify the Virtual COM port number as shown in the Device Manager. All other properties should be the same as those for RS-232 (serial).</p>
Ethernet	<pre>&lt;prop name="deviceBus" type="String" value="Ethernet" /&gt;</pre>
	<pre>&lt;prop name="ipaddress" type="String" value="192.168.1.74" /&gt;</pre>
	<pre>&lt;prop name="port" type="Integer" value="6780" /&gt;</pre>
USB-HID	<pre>&lt;prop name="deviceBus" type="String" value="HID" /&gt;</pre>
	<pre>&lt;prop name="portName" type="String" value="HID0" /&gt;</pre>

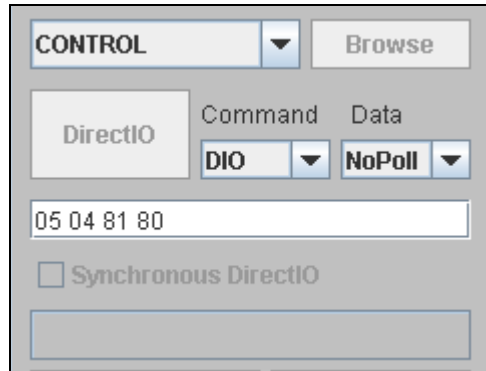
### D.3. The JavaPOS Test Application Interface

The JavaPOS Test Application contains the following:



## D.4. DirectIO

DirectIO functionality can be used by JavaPOS application to utilize capabilities in Telium terminals.

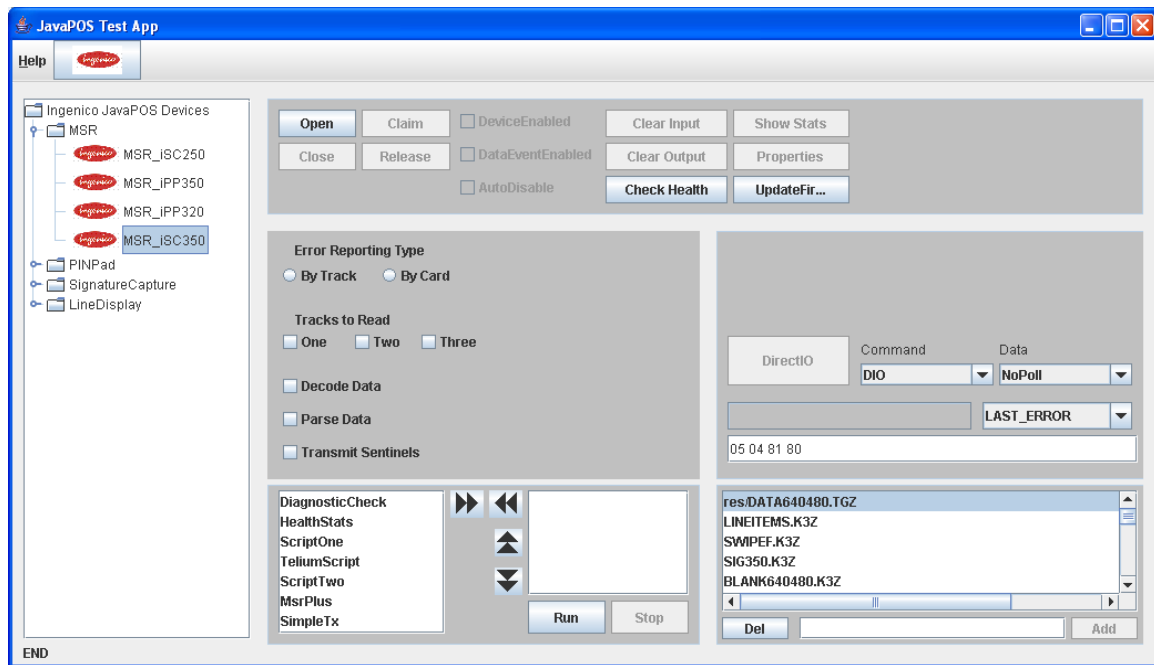


**Figure 10 - The JPOS test application's DirectIO panel.**

The following table lists the individual component of the test application's DirectIO panel and the associated functions.

Component	Function
CONTROL	A combo box that contains most commonly used DirectIO commands, pulled from the dio.properties file.
DirectIO	The DirectIO button is enabled when device is claimed.
Command	A combo box containing supported values for the DirectIO <b>Command</b> parameter.
Data	A combo box containing supported values for the DirectIO Data parameter.
Text Field	Text area used to display the five hexadecimal values (bytes) that make up the message associated with the selected command.
Progress Bar	Monitor progress during file download.

## D.5. Running the JPOS Test App



**Figure 11 - The JPOS test application interface.**

To run the JPOS Test Application, simply execute the JavaPOSTest-RS232.bat batch file.

Select the desired UPOS control and PIN pad from the list shown down the left side of the test application interface. The UPOS control functions panel changes to show settings for the selected UPOS control.

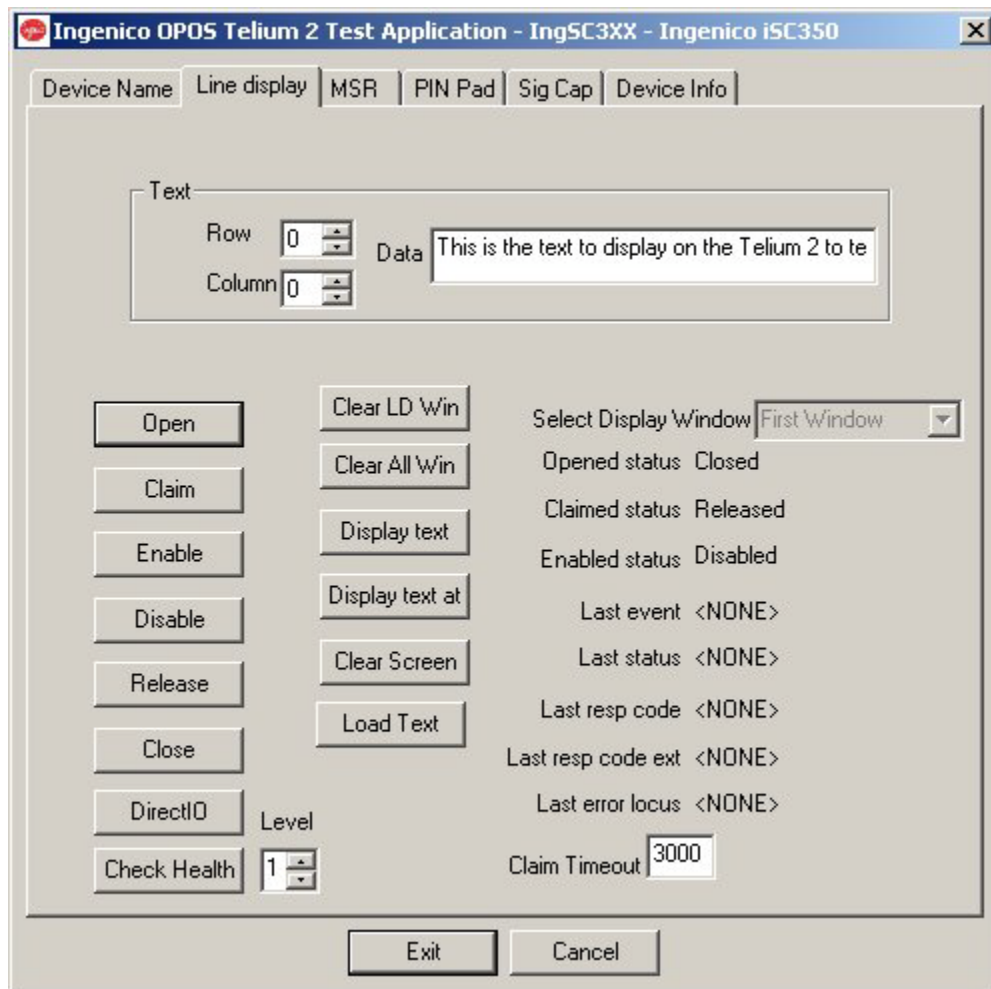
To activate a control, click the Open and Claim buttons.

DirectIO functionality can be used once the device has been claimed.



## Appendix E. OPOS Test Application

The OPOS Test Application (OPOS Test App.exe) is an application included with the OPOS Integration Kit that allows users to emulate a POS system connected to a UIA-equipped Ingenico PIN pad.



**Figure 12 - the OPOS Test Application.**

The test application contains a tab for each OPOS service objects as well as an additional Device Info tab, and it can be used to test all PIN pad functions controlled by OPOS service objects.

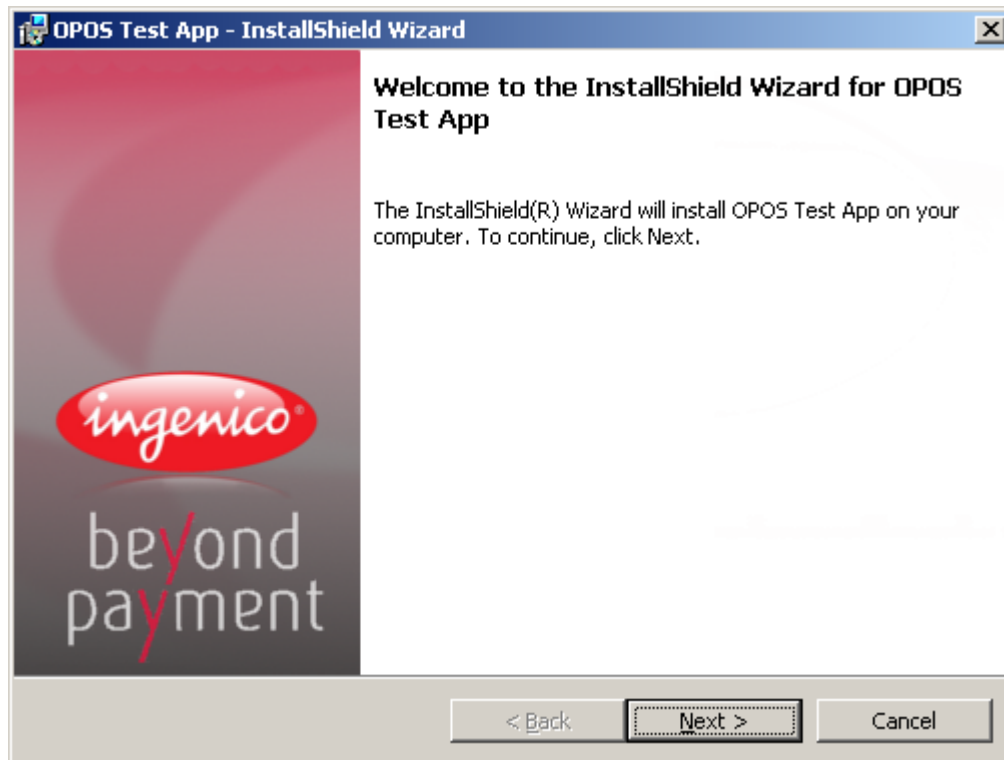
Users of the OPOS Integration Kit can find the OPOS Test Application in the Utilities folder included in the integration kit package.

## E.1. Installing the OPOS Test Application

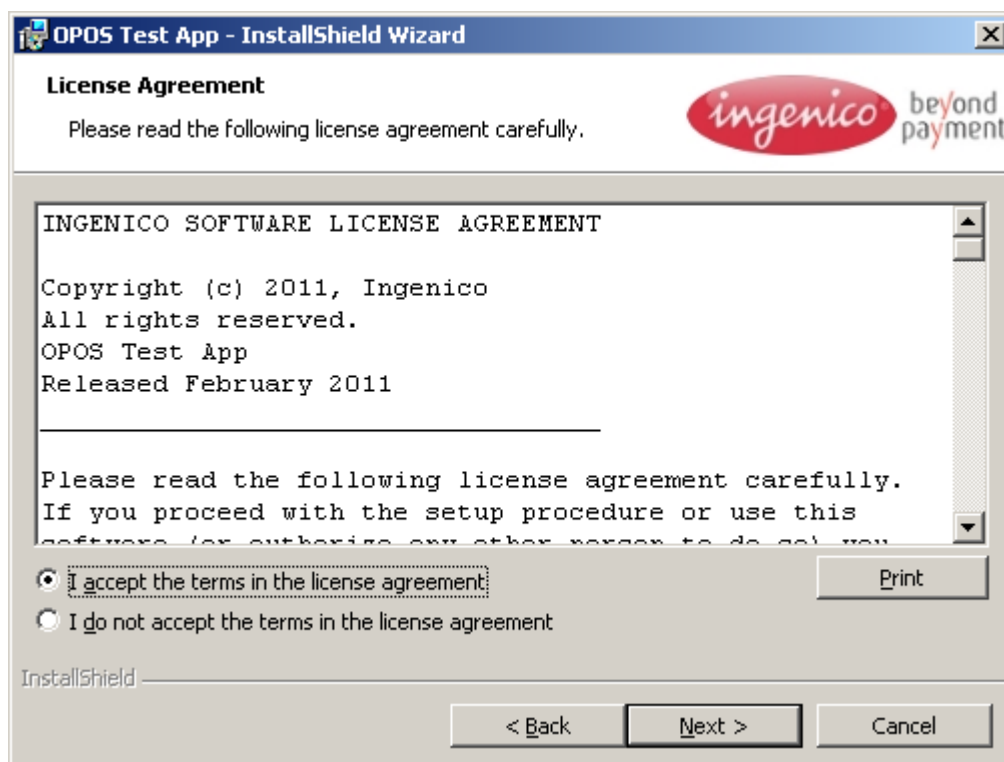
---

To install the OPOS Test Application, follow these steps:

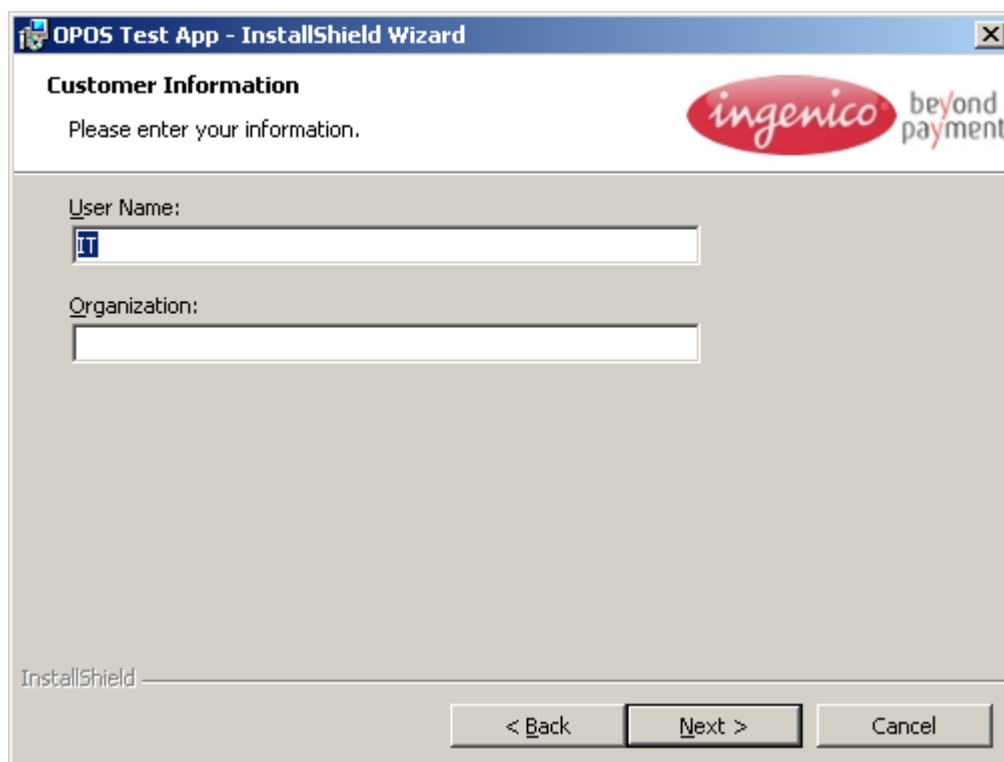
1. Start the OPOS Test Application installer by running the associated installer file.



2. Click **Next**.



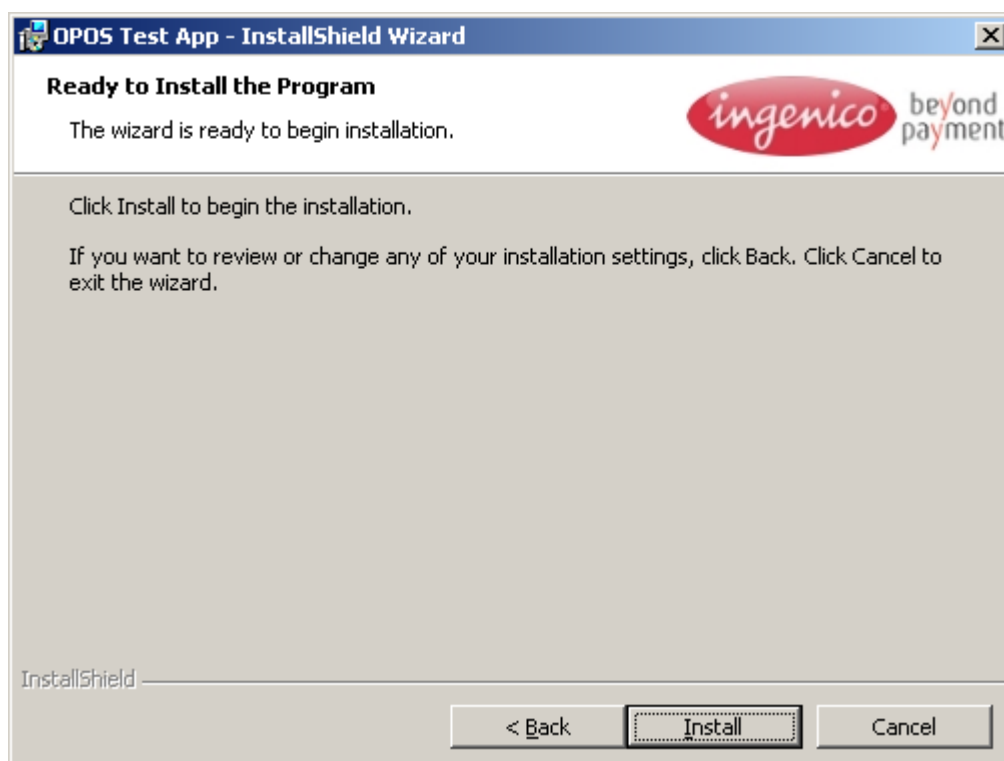
3. Select the radio button next to **I accept the terms in the license agreement** and click **Next**.



4. Type in the **User Name** and the name of the **Organization** that will be using the OPOS Test Application, then click **Next**.



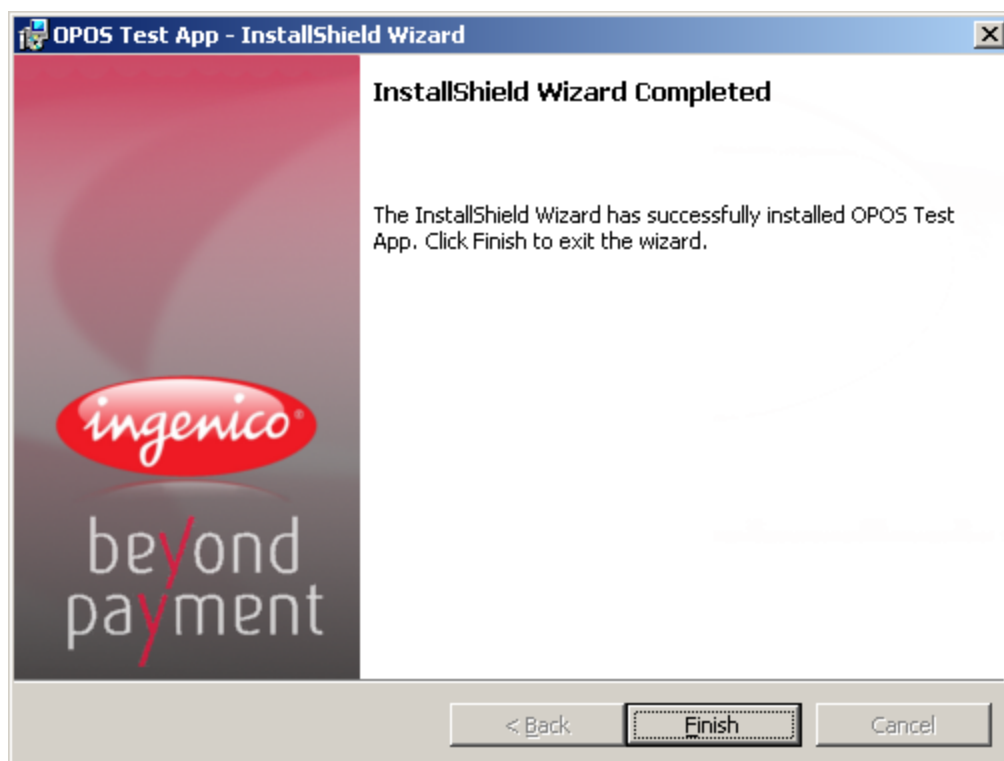
5. Click the radio button next to **Complete**, then click **Next**.



6. Click **Install** to begin the installation process.



7. A status window displays.



8. Click **Finish** once the installation is complete.

The OPOS Test Application is now available from the Start menu.

## E.2. Application Interface

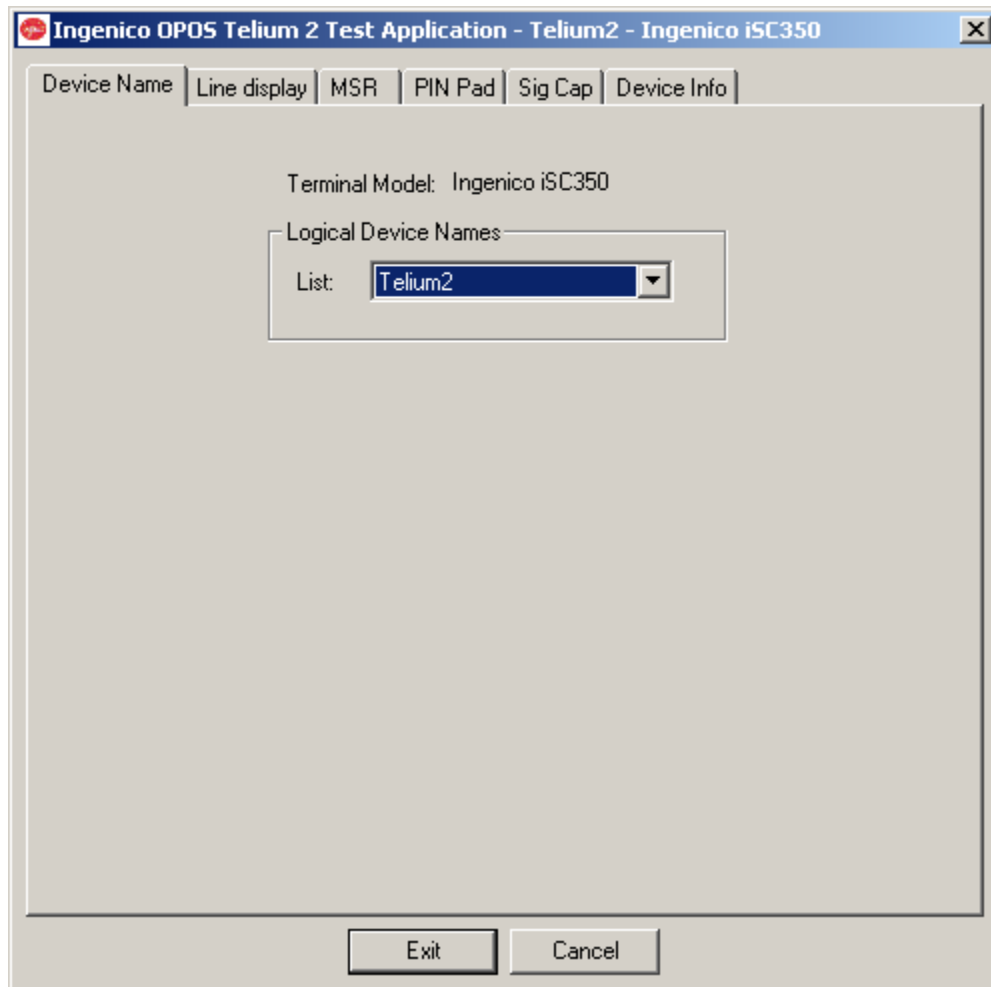
---

### E.2.1. Device Name Tab

The Device Name tab can be used to select the type of PIN pad that the OPOS Test Application will be interfacing with.

On the Device Name tab, select the **Logical Device Name** that corresponds to the desired type of Ingenico PIN pad from the drop-down list.

Device names can be defined using the OPOS Control Panel application (see OPOS Configuration on page 26 for more information).



### E.2.2. Line Display Tab

The Line Display tab in the OPOS Test Application can be used to send PIN pad commands using the Line Display control.

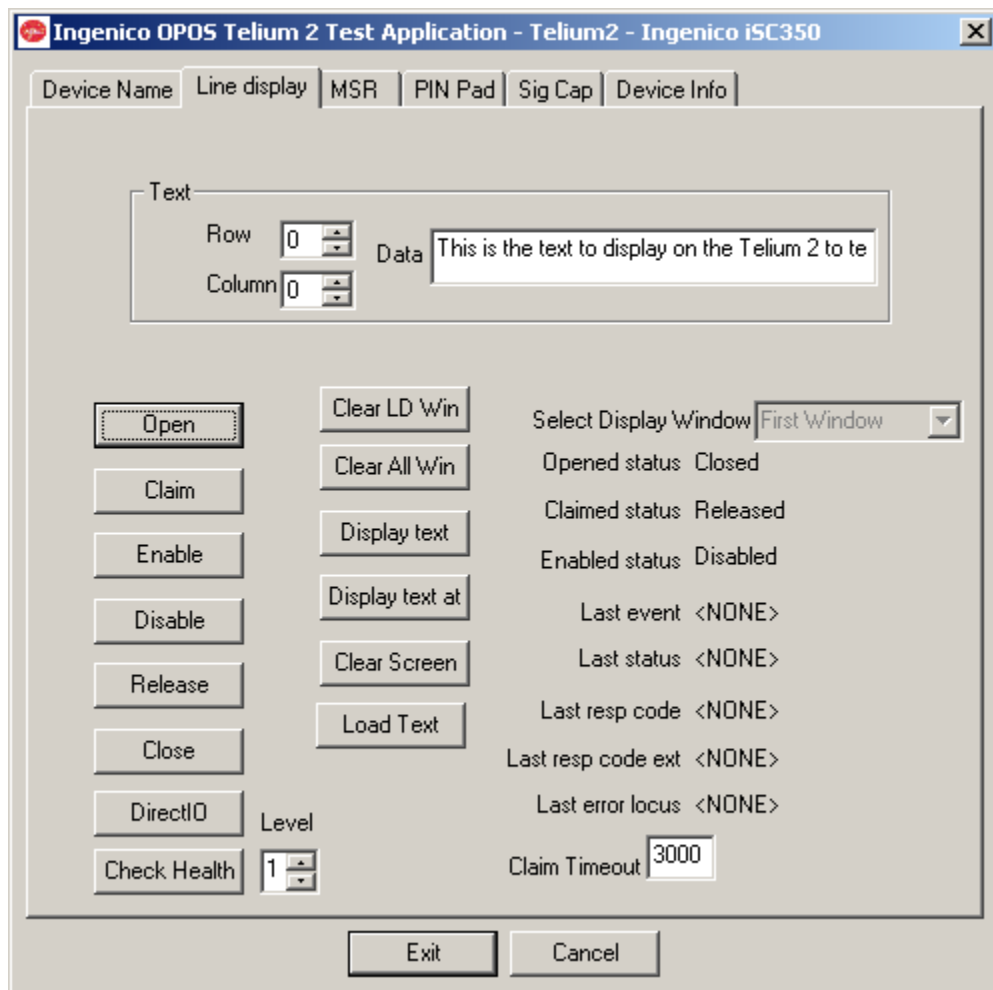


Figure 13 - the Line Display tab.

Once you have opened, claimed, and enabled the line display control, the Line Display displays on the PIN pad screen. Line Display control functionality can be tested using the buttons and controls found on the Line Display tab.

Control	Description
Text - Row	The number of the row at which to display the text specified in the Data field.
Text - Column	The number of the column at which to display the text specified in the Data field.
Text - Data	The text to display using the Line Display control.
Clear LD Win	Clears the line display for the active window.
Clear All Win	Clears the line display for all windows.
Display text	Displays the text in the Text – Data field on the PIN pad screen.

Control	Description
Display text at	Displays the text in the Text – Data field on the PIN pad screen at the location specified by the Row and Column boxes.
Clear Screen	Clears the screen.
Load Text	Allows you to load a text file to display using the Line Display control.

### E.2.3. MSR Tab

The MSR tab in the OPOS Test Application can be used to send PIN pad commands using the MSR control.

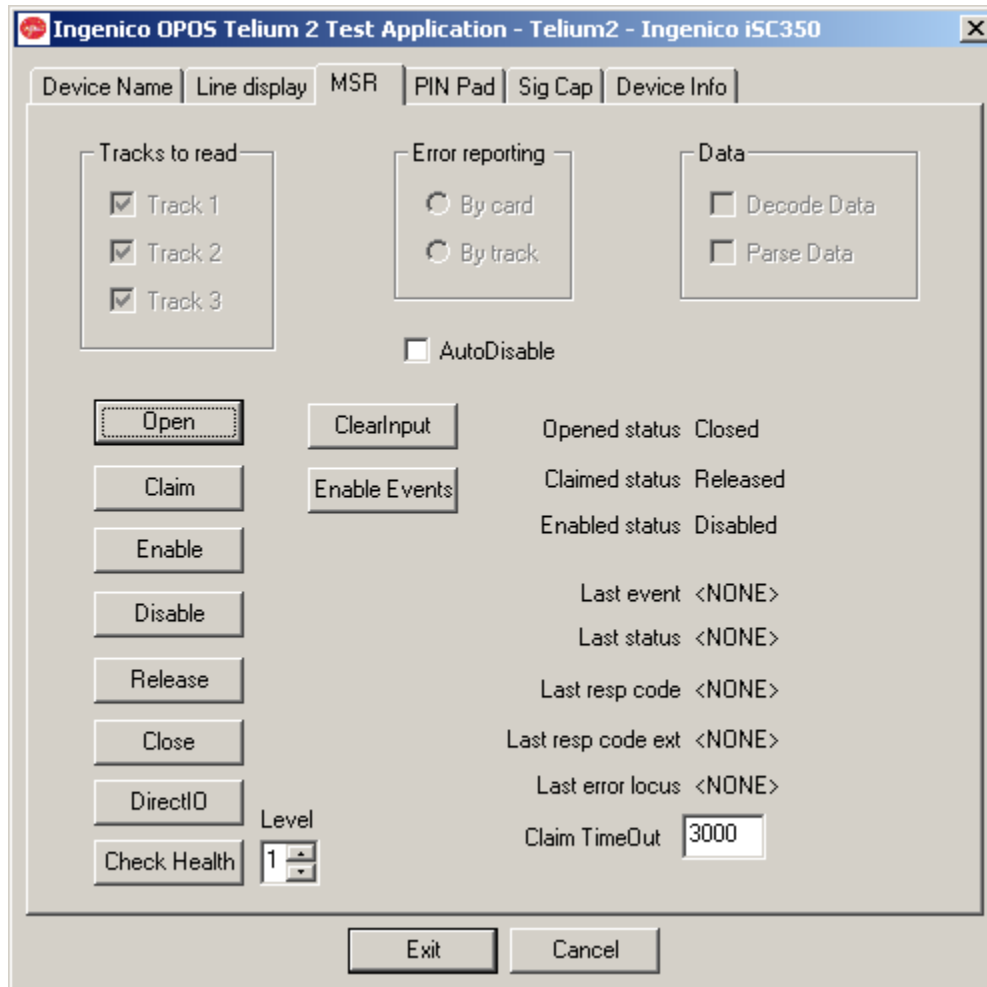


Figure 14 - the MSR tab.

Once you have opened, claimed, and enabled the MSR control, the MSR LEDs are activated and the MSR form displays on the PIN pad screen. MSR control functionality can be tested using the buttons and controls found on the MSR tab.

Control	Description
Tracks to read checkboxes	Determine which tracks of card data are scanned by the MSR.



Control	Description
Enable Events	Clicking Enable Events causes the test app to display MSR data received from the PIN pad.

#### E.2.4. PIN Pad Tab

The PIN Pad tab in the OPOS Test Application can be used to send PIN pad commands using the PIN Pad control.

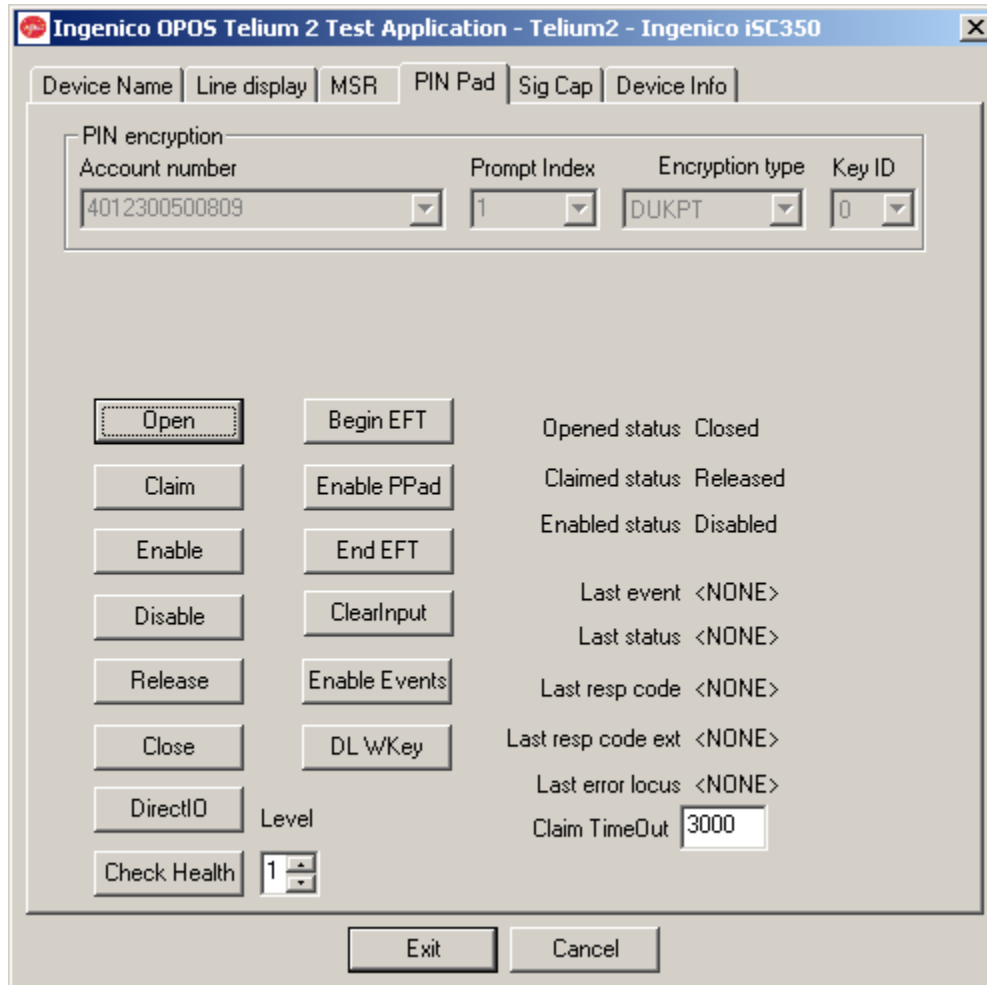


Figure 15 - the PIN Pad tab.

Once you have opened, claimed, and enabled the PIN pad control, you can test it using the buttons and controls found on the PIN pad tab.

Control	Description
Begin EFT	Initiates an EFT transaction.
Enable PPad	Displays the PIN entry form and allows for PIN input.
End EFT	Ends the EFT transaction.

### E.2.5. Sig Cap Tab

The Sig Cap tab in the OPOS Test Application can be used to send PIN pad commands using the Signature Capture control.

The screenshot shows the 'Sig Cap' tab of the 'Ingenico OPOS Telium 2 Test Application'. The interface includes a 'Signature preview' area with the text 'No Current Signature Data'. To the right, the 'Signature format' is set to 'Use OPOS'. Below this, the 'OPOS conversion' is set to 'OPOS\_BC\_NIBBLE' and the 'Form Value' field is empty. There are checkboxes for 'Show signature details' and 'Show number of sig points', both of which are unchecked. A series of buttons are arranged in two columns: 'Open', 'Claim', 'Enable', 'Disable', 'Release', 'Close', 'Direct IO', and 'Check Health' on the left; 'Begin capture', 'End capture', 'Get Data', and 'ClearInput' on the right. The 'Open' button is currently highlighted. On the right side, a status section displays various fields: 'Opened status' (Closed), 'Claimed status' (Released), 'Enabled status' (Disabled), 'Last event' (<NONE>), 'Last status' (<NONE>), 'Last resp code' (<NONE>), 'Last resp code ext' (<NONE>), 'Last error locus' (<NONE>), and 'Claim TimeOut' (3000). At the bottom, there are 'Exit' and 'Cancel' buttons.

Figure 16 - the Sig Cap tab.

Once you have opened, claimed, and enabled the signature capture control, you can test it using the buttons and controls found on the Sig Cap tab.

Control	Description
Begin capture	Displays the signature capture form and begins recording stylus contact with the screen.
End capture	Stops recording stylus contact with the screen and transmits signature data to host system.

### E.2.6. Device Info Tab

The Device Info in the OPOS Test Application can be used to query the properties of each of the listed OPOS controls.

The top part of the window displays the control's properties after the Open() command is sent, and the bottom displays the control's properties after the Claim() command is sent.

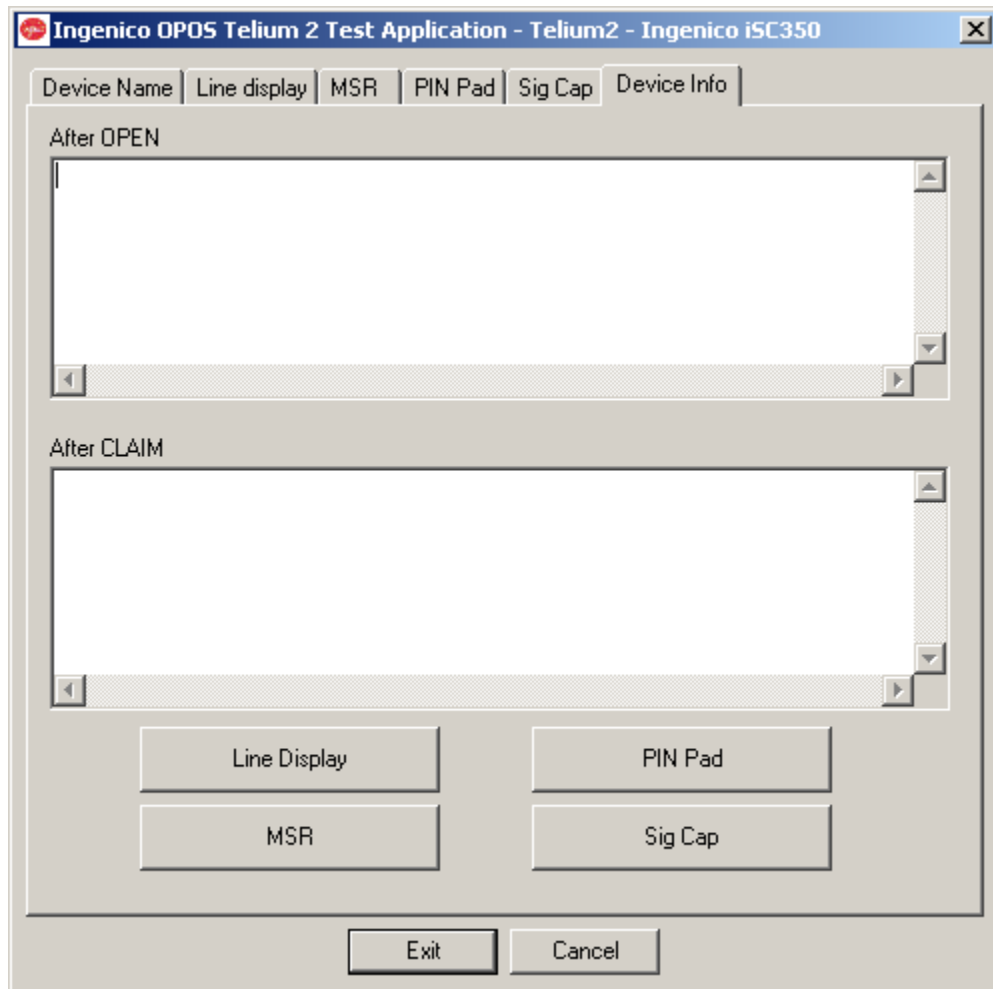


Figure 17 - the Device Info tab.

### E.2.7. Direct I/O

Clicking the DirectIO button found on the Test App's Line Display, MSR, PIN Pad or SigCap tabs launches the DirectIO panel, which can be used to issue DirectIO commands to a UIA-equipped Ingenico Telium PIN pad.

**Figure 18 - the DirectIO panel.**

To issue a Direct I/O command using the OPOS Test Application's DirectIO panel, follow these steps:

1. Select the desired DirectIO command from the **DirectIO** drop-down menu OR type the corresponding command number in the **Command** field.
2. Type in any data to send with the command in the **Data** field (if needed).
3. Select an **Object** by typing in the object path or clicking **Browse** and browsing to the desired file.
4. Specify any additional command-specific settings displayed at the bottom of the DirectIO panel.
5. Click **DirectIO** to send the command.

The OPOS Test Application's DirectIO panel displays any result data and/or object returned by the PIN pad.

## E.3. Working With the Test Application

### E.3.1. Opening, Claiming and Enabling a Control

Before using the OPOS test application to send a command to an UPOS control, you must first Open, Claim and Enable that control from the corresponding test application tab.

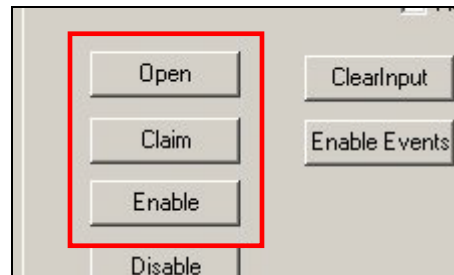


Figure 19 - Open, Claim, and Enable buttons on the MSR tab.

If you wish to use the test application to send a command to a different UPOS control, you must first Close the one that is currently open by clicking the Close button on the corresponding tab.

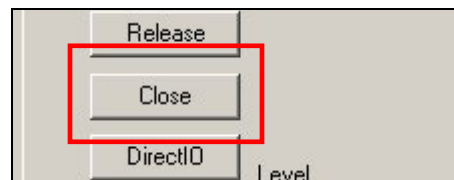


Figure 20 - the Close button on the MSR tab.

### E.3.2. Changing Variable Text With The Test Application

OPOS test application users can easily change Telium PIN pad variable text used within any of the PIN pad's \*.K3Z form files using the DirectIO "SET\_UIA\_VARIABLE" command.

To change the value of any Telium PIN pad variable, follow these steps:

1. From the OPOS Test Application, click the **DirectIO** button on the tab corresponding to any active UPOS control.
2. Select SET\_UIA\_VARIABLE from the **DirectIO** drop-down menu.
3. Type the variable update command in the **Object** field using the following format: `<variable1>=<variable1 value>%<variable2>=<variable2 value>`. Users can use the SET\_UIA\_VARIABLE command to update multiple variables by separating variable update commands using the "%" character.



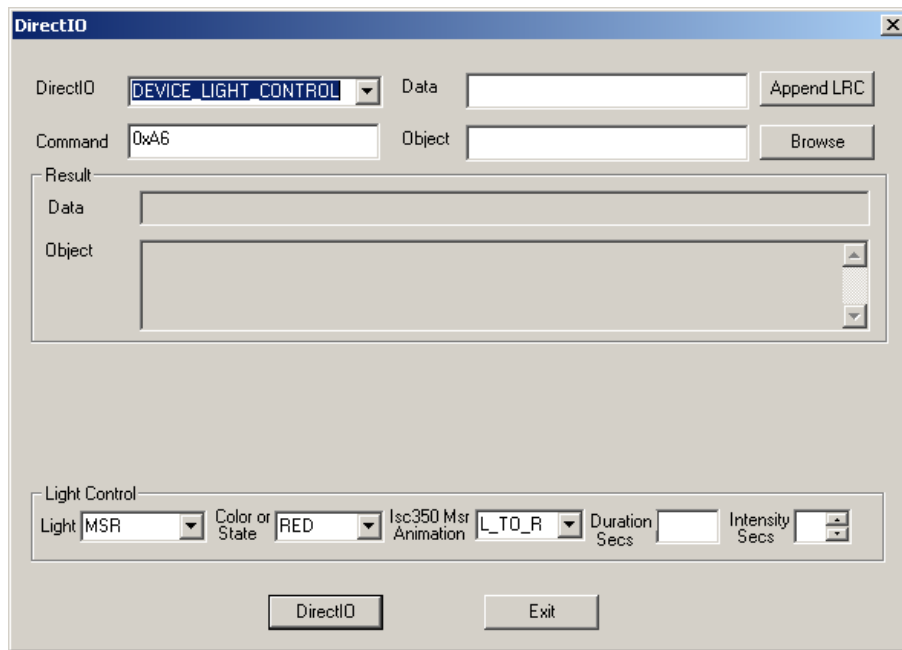
4. Click the **DirectIO** button to execute the command. The new variable values are stored in the PIN pad's memory until the device is rebooted. Any forms using the updated variables will display the new values.

### E.3.3. Configuring MSR LEDs

OPOS test application users can easily configure an iSC350's MSR LEDs using the DirectIO "DEVICE\_LIGHT\_CONTROL" command.

To change the value of any Telium PIN pad variable, follow these steps:

1. From the OPOS Test Application, click the **DirectIO** button on the tab corresponding to any active UPOS control.
2. Select DEVICE\_LIGHT\_CONTROL from the **DirectIO** drop-down menu.
3. Select the desired settings for **Light, Color or State, Isc350 MSR Animation, Duration Secs,** and **Intensity Secs** from the corresponding drop-down menus.



The image shows a software window titled "DirectIO". It contains several input fields and buttons. At the top, there is a "DirectIO" dropdown menu set to "DEVICE\_LIGHT\_CONTROL", a "Data" text box, and an "Append LRC" button. Below this is a "Command" text box containing "0xA6", an "Object" text box, and a "Browse" button. A "Result" section contains two large text boxes labeled "Data" and "Object". At the bottom, there is a "Light Control" section with a "Light" dropdown set to "MSR", a "Color or State" dropdown set to "RED", a "Lsc350 Msr Animation" dropdown set to "L\_TO\_R", a "Duration" text box, and an "Intensity" spinner box. At the very bottom are "DirectIO" and "Exit" buttons.

4. Click the **DirectIO** button to execute the command.

## Notes



## Appendix F. Telium PIN Pad USB Settings

The table below shows the VID and PID settings required for USB communication with Ingenico Telium PIN pads. In order to establish USB communications with an Ingenico PIN pad, a POS device must be configured to reference the device using the correct VID and PID settings.

### F.1. For HID Communication

**Table 6: Telium PIN Pad VID and PID Settings for HID Communication**

Device	String	Screen	Colors	MP4	VID	PID	Touch	Flash	Audio
iSC350	Ingenico iSC350	640x480	240k	Yes	0x0B00	0x0073	Yes	128m	Yes
iSC250	Ingenico iSC250	480x272	240k	Yes	0x0B00	0x0074	Yes	128m	Yes
iSC220	Ingenico iSC220	480x272	16 (Gray)	?	0x0B00	0x0075	Yes	128m	Buzzer
iPP350	Ingenico iPP350	320x240	4k	No	0x0B00	0x0072	No	128m	Buzzer
iPP320	Ingenico iPP320	128x64	Black/white	No	0x0B00	0x0071	No	128m	Buzzer

### F.2. For CDC Communication

**Table 7: Telium PIN Pad VID and PID Settings for CDC Communication**

Device	String	Screen	Colors	MP4	VID	PID	Touch	Flash	Audio
iSC350	Ingenico iSC350	640x480	240k	Yes	0x0B00	0x0061	Yes	128m	Yes
iSC250	Ingenico iSC250	480x272	240k	Yes	0x0B00	0x0062	Yes	128m	Yes
iSC220	Ingenico iSC220	480x272	16 (Gray)	?	0x0B00	0x0063	Yes	128m	Buzzer
iPP350	Ingenico iPP350	320x240	4k	No	0x0B00	0x0060	No	128m	Buzzer
iPP320	Ingenico iPP320	128x64	Black/white	No	0x0B00	0x0059	No	128m	Buzzer

## Notes