



beyond  
payment

**UPOS**

## **Developer's Guide**

**Telium Devices**

UPOS Developer's Guide  
Part Number DIV350825 Rev. K  
Released November, 2012.  
Copyright © 2012, Ingenico Corp. All rights reserved.

Ingenico Inc.	Ingenico Canada Ltd.
3025 Windward Plaza, Suite 600	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
3025 Windward Plaza, Suite 600	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 © 2012. All rights reserved.

# Contents

---

<b>Contents .....</b>	<b>i</b>
<b>Revision History .....</b>	<b>x</b>
<b>1. Introduction to UPOS .....</b>	<b>17</b>
1.1. About This Manual.....	17
1.2. Definitions.....	18
1.3. Supported Environments.....	18
1.4. UPOS Device Drivers .....	18
1.5. Compatible Devices.....	19
1.6. Connecting and Powering the PIN Pad .....	19
<b>2. Getting Started with the JPOS Integration Kit .....</b>	<b>21</b>
2.1. JPOS Overview .....	21
2.2. JPOS Integration Kit Contents.....	21
2.3. Installation .....	22
2.3.1. Installing JRE.....	22
2.3.2. Installing JavaCOMM API on Windows .....	22
2.3.3. Installing JavaCOMM within JRE .....	22
2.3.4. Installing JavaCOMM Outside of JRE .....	22
2.3.5. Installing the Java XML Parser .....	23
2.3.6. Installing Apache Log4J Logging API .....	23
2.3.7. Installing PayPal Authorization.....	24
2.3.8. Troubleshooting Installation Issues .....	24
2.4. JavaPOS Configuration .....	24
2.4.1. Configuring Beep Tones.....	25
2.5. JPOS Entry Editor .....	25
2.5.1. Setting the Java Path .....	26
2.5.2. Editing Standard Properties with the JPOS Entry Editor .....	26
2.5.3. Editing Bus Properties with the JPOS Entry Editor.....	27
2.5.4. Adding Vendor Properties with the JPOS Entry Editor .....	27
2.5.5. Editing Vendor Properties with the JPOS Entry Editor.....	27
2.5.6. Removing Vendor Properties with the JPOS Entry Editor .....	28
<b>3. Getting Started with the OPOS Integration Kit.....</b>	<b>30</b>
3.1. OPOS Overview .....	30
3.2. OPOS Integration Kit Contents.....	31
3.3. Installation .....	33
3.3.1. Typical Install .....	33

3.3.2. Silent Installation.....	36
3.3.3 Installing PayPal Authorization.....	38
3.4. OPOS Configuration .....	39
3.5. Uninstalling the Software .....	50
3.5.1. Typical Uninstall.....	50
3.5.2. Silent Uninstall.....	52
3.6. Using the OPOS Controls .....	52
3.7. Setting Up the System Registry.....	53
<b>4. Events.....</b>	<b>56</b>
4.1. Data Events.....	56
4.1.1. Event Object Format .....	56
4.1.2. Event Signature – JPOS.....	56
4.1.3. Event Signature – OPOS.....	56
4.1.4. Data Event Statuses .....	56
4.2. DirectIO Events.....	58
4.2.1. Event Object Format .....	58
4.2.2. Event Signature – JPOS.....	58
4.2.3. Event Signature – OPOS.....	59
4.2.4. DirectIO Event Numbers .....	59
4.3. Error Events .....	61
4.3.1. Event Object Format .....	62
4.3.2. Event Signature – JPOS.....	62
4.3.3. Event Signature – OPOS.....	63
4.3.4. Error Event Codes.....	63
4.4. Status Update Events (JPOS only).....	65
4.4.1. Event Object Format .....	65
4.4.2. Event Signature .....	65
4.4.3. Status Update Event Statuses .....	66
4.5. JPOS Event Handling - Samples .....	66
4.5.1. Data Event Handling.....	66
4.5.2. DirectIO Event Handling.....	66
4.5.3. Error Event Handling .....	66
4.5.4. Status Update Event Handling.....	67
4.6. OPOS Event Handling.....	67
<b>5. Direct I/O for JPOS .....</b>	<b>68</b>
5.1. The DirectIO Method.....	68
5.1.1. (0x00) Send Raw Data .....	69
5.1.2. (0x0D) Get Health Statistics .....	69
5.1.3. (0xA1) Set UIA Variable .....	73

5.1.4.	(0xA2) Get UIA Variable .....	75
5.1.5.	(0xA6) Configure PIN Pad Lights.....	75
5.1.6.	(0xEF) Trigger EFT/TMS Downloads.....	77
5.1.7.	(0x09) Reboot the PIN Pad.....	78
5.1.8.	(0x12) Transmit Check Box Data.....	78
5.1.9.	(0x13) Transmit Survey Data .....	78
5.1.10.	(0x30) Clear Line Display Element .....	78
5.1.11.	(0x31) Clear Screen .....	79
5.1.12.	(0x34) Display Text At.....	79
5.1.13.	(0x35) Delete Receipt Contents .....	81
5.1.14.	(0x36) Position Text Cursor .....	82
5.1.15.	(0x91) Save File .....	82
5.1.16.	(0x92) Run File.....	82
5.1.17.	(0x93) Delete File .....	83
5.1.18.	(0x94) Retrieve File .....	83
5.1.19.	(0x95) File Status.....	84
5.1.20.	(0x6A ) Activate Software .....	84
5.2.	Best Practices .....	84
5.3.	Usage Samples .....	84
5.3.1.	Set a Variable .....	85
5.3.2.	Clear Screen .....	85
5.3.3.	Store a Form .....	85
5.3.4.	Display a Form .....	85
<b>6.</b>	<b>Direct I/O for OPOS.....</b>	<b>86</b>
6.1.	The DirectIO Method.....	86
6.1.1.	(0x00) Send Raw Data .....	89
6.1.2.	(0x0D) Get Health Statistics .....	89
6.1.3.	(0xA1) Set UIA Variable .....	92
6.1.4.	(0xA2) Get UIA Variable .....	94
6.1.5.	(0xA6) Configure PIN Pad Lights.....	94
6.1.6.	(0x09) Reboot the PIN Pad.....	96
6.1.7.	(0x12) Transmit Check Box Data.....	98
6.1.8.	(0x13) (Transmit Survey (Radio Button) Data.....	98
6.1.9.	(0x30) Clear Line Display Element.....	98
6.1.10.	(0x31) Clear Screen .....	99
6.1.11.	(0x34) Display Text At.....	99
6.1.12.	(0x35) Delete Receipt Contents .....	100
6.1.13.	(0x36) Position Text Cursor .....	102
6.1.14.	(0x91) Save File .....	102

6.1.15. (0x92) Run File.....	102
6.1.16. (0x93) Delete File .....	102
6.1.17. (0x94) Retrieve File.....	103
6.1.18. (0x95) File Status.....	103
6.1.19. (0x6A) Activate Software .....	103
6.2. Best Practices .....	103
6.3. Usage Samples .....	103
6.3.1. Store a Form .....	103
6.3.2. Display a Form .....	104
<b>7. Format Specifiers.....</b>	<b>106</b>
7.1. General Attributes .....	106
7.2. Specific Attributes .....	107
7.3. Using Multiple Format Specifier Attributes .....	109
7.4. Unknown Format Specifiers .....	109
7.5. Examples of Format Specifiers .....	110
<b>8. Line Display.....</b>	<b>114</b>
8.1. General Information.....	114
8.2. Usage Samples .....	114
8.2.1. Clear a Window .....	114
8.2.2. Write Text To a Location (Row, Column).....	114
8.2.3. Write Text To a Variable.....	115
<b>9. MSR.....</b>	<b>116</b>
9.1. General Information.....	116
9.2. Usage Samples .....	116
9.2.1. Read All Tracks of Card Data.....	116
9.2.2. SetDeviceEnabled Error Reporting .....	116
9.2.3. Request Sentinels.....	117
9.2.4. Parse MSR Data (VB.NET) .....	117
<b>10. Signature Capture.....</b>	<b>120</b>
10.1. General Information.....	120
10.2. Signature Form.....	120
10.3. Usage Samples .....	120
10.3.1. Define the Signature Format .....	120
10.3.2. Define the OPOS Conversion Formats .....	120
10.3.3. Handle Signature Capture.....	120
<b>11. SigDisplay Control.....</b>	<b>124</b>
11.1. General Information.....	124
11.2. Summary.....	124
11.2.1. Properties .....	124

11.2.2. Methods.....	124
11.3. Properties .....	125
11.3.1. GetDrawBorder .....	125
11.3.2. SetDrawBorder .....	125
11.3.3. GetDrawBackground.....	125
11.3.4. SetDrawBackground .....	125
11.3.5. GetPenWidth .....	125
11.3.6. SetPenWidth.....	126
11.3.7. GetDisplayNumPoints .....	126
11.3.8. SetDisplayNumPoints.....	126
11.4. Methods.....	127
11.4.1. SetOPOSBCNIBBLESignatureData.....	127
11.4.2. SetOPOSBCNIBBLESignatureDataX .....	127
11.4.3. SetSignatureData .....	128
11.4.4. SetSignatureDataX .....	128
11.4.5. GetSignatureType.....	129
11.4.6. GetSignatureTypeString.....	129
11.4.7. WriteSignatureToFile .....	129
11.4.8. ConvertSignatureToImageBuffer .....	129
<b>12. PIN Pad.....</b>	<b>132</b>
12.1. General Information.....	132
12.2. Encryption Key Formats.....	132
12.2.1. Master/Session Key Serial Number Format .....	132
12.2.2. DUKPT Key Serial Number Format .....	132
12.3. RSA Encryption for PayPal PIN .....	132
12.4. Usage Samples .....	133
12.4.1. Handle PIN Entry .....	133
12.4.2. Request Key ID or Key Slot Information .....	134
12.4.3. Retrieve PIN Block.....	135
12.4.4. Set PIN Pad Prompt Language .....	135
12.5. Advertising Support .....	136
<b>13. Forms .....</b>	<b>137</b>
13.1. Post Splash Screen Custom Form Feature .....	137
<b>14. MSR Encryption.....</b>	<b>138</b>
14.1. Supported Encryption Methods.....	138
14.2. Loading Key Serial Number (KSN) Data .....	138
14.3. Enabling MSR Encryption in UIA.....	138
14.4. Signing Requirements for .DAT File Changes .....	138
14.5. Selecting Specific Cards to be Encrypted.....	141

14.6.	Manual Entry.....	141
14.6.1.	Manual Entry Variables.....	141
14.6.2.	Manual Card Entry Flow .....	142
14.7.	RSA-OAEP & TransArmor Encryption.....	146
14.7.1.	References .....	146
14.7.2.	Configuration - Prerequisites .....	146
14.7.3.	Configuration - General .....	147
14.7.4.	Usage .....	148
14.8.	.PGZ File Errors/Troubleshooting.....	149
<b>15.</b>	<b>DAT Files .....</b>	<b>150</b>
15.1.	Changes to security.dat or secbin.dat.....	150
15.2.	Security Parameters (security.dat).....	150
15.3.	Security BIN (secbin.dat) .....	154
15.3.1.	Enabling BIN Checking .....	155
<b>16.</b>	<b>OPOS Usage Samples with Microsoft Common Control .....</b>	<b>156</b>
16.1.	Implementing a Terms and Conditions Screen.....	156
16.2.	Implementing a Survey Screen.....	156
16.3.	Simulating a Transaction.....	156
16.4.	Using OPOS for the .NET Application.....	156
16.4.1.	PosExplorer.....	156
16.4.2.	.NET Interop.....	157
<b>17.</b>	<b>UIA Prompts .....</b>	<b>158</b>
17.1.	Security Prompts (SECURPROMPT.xml).....	158
17.2.	Transaction Prompts (PROMPT.xml).....	165
17.2.1.	Button Text.....	171
17.3.	Custom Prompts (CUSTPROMPTS.xml).....	173
<b>Appendix A.</b>	<b>Differences Between U32 UPOS and Telium UPOS.....</b>	<b>175</b>
A.1.	At A Glance.....	175
A.2.	UPOS Control Support .....	175
A.2.1.	Form Control.....	175
A.2.2.	Direct I/O (OPOS and JPOS).....	176
A.3.	Control Panels.....	176
A.3.1.	OPOS Control Panel.....	176
A.4.	Test Applications .....	177
<b>Appendix B.</b>	<b>U.S. Retail Telium Download Application (TDA) .....</b>	<b>179</b>
B.1.	Accessing the Telium Download Application.....	179
B.2.	Configuring PIN Pad Communication Settings .....	180
B.2.1.	Setting Communication Type .....	180
B.2.2.	Configuring Serial Connection Settings .....	180



B.2.3. Configuring Ethernet Connection Settings .....	182
B.2.4. Configuring Tailgate Connection Settings .....	185
B.3. Exiting the Telium Download Application.....	185
<b>Appendix C. 5992 Mode .....</b>	<b>187</b>
C.1. Special Functions.....	187
C.2. Line Displays .....	187
C.3. Fonts.....	187
C.4. Required Scripting.....	188
C.5. 5992 Mode Font Tables.....	188
C.5.1. Font Types.....	188
C.5.2. Point Size Values.....	188
C.5.3. Examples.....	189
<b>Appendix D. JPOS Test Application.....</b>	<b>191</b>
D.1. Getting Started .....	191
D.2. Updating JPOS Configuration.....	191
D.3. The JavaPOS Test Application Interface .....	192
D.4. Direct I/O .....	192
D.5. Running the JPOS Test App .....	194
D.6. Working With the Test Application .....	195
D.6.1. Opening, Claiming and Enabling a Control.....	195
D.6.2. Sending Direct I/O Commands.....	196
D.6.3. Using the Quick DirectIO Panel.....	197
<b>Appendix E. OPOS Test Application.....</b>	<b>199</b>
E.1. Installing the OPOS Test Application .....	200
E.2. Application Interface.....	204
E.2.1. Device Name Tab.....	204
E.2.2. Line Display Tab .....	205
E.2.3. MSR Tab .....	206
E.2.4. PIN Pad Tab.....	207
E.2.5. Sig Cap Tab.....	208
E.2.6. Device Info Tab .....	209
E.3. Working With the Test Application .....	210
E.3.1. Opening, Claiming and Enabling a Control.....	210
E.3.2. Sending Direct I/O Commands.....	210
E.3.3. Changing Variable Text .....	211
E.3.4. Changing Prompt Indices.....	212
E.3.5. Displaying Forms .....	213
E.3.6. Configuring MSR LEDs With The Test Application.....	213
<b>Appendix F. Telium PIN Pad USB Settings .....</b>	<b>215</b>

F.1.	For HID Communication.....	215
F.2.	For CDC Communication .....	215
<b>Appendix G.</b>	<b>Public Software License Acknowledgments.....</b>	<b>216</b>

## Notes

# Revision History

Manual Revision	Application Revision	Changes
K	OPOS 1.4.12 UIA 2.7.1 PK-UGENxx-0271	<p>3.6 – Forms</p> <ul style="list-style-type: none"> <li>Moved this content to another section of the document (is now section 13)</li> </ul> <p>5.1.18 – (0x94) Retrieve File (JPOS)</p> <ul style="list-style-type: none"> <li>Updated this section to include a new manifest file called ‘manifestcrc32’</li> </ul> <p>7 – Format Specifiers</p> <ul style="list-style-type: none"> <li>Updated this section to include additional examples</li> </ul> <p>11.1 – General Information (SigDisplay Control)</p> <ul style="list-style-type: none"> <li>Updated form control filename to EnFormSigDisplay.ocx</li> </ul> <p>12.5 – Advertising Support</p> <ul style="list-style-type: none"> <li>Added this new section</li> </ul> <p>13 – Forms</p> <ul style="list-style-type: none"> <li>Moved this general info section (from section 3.6 to section 13)</li> </ul> <p>13.1 – Post Splash Screen Custom Form Feature</p> <ul style="list-style-type: none"> <li>Added this new section</li> </ul> <p>14 – MSR Encryption</p> <ul style="list-style-type: none"> <li>Added this new section including TransArmor/RSA-OEAP Encryption types</li> </ul> <p>15 – DAT Files</p> <ul style="list-style-type: none"> <li>Added this new section including security.dat and secbin.dat</li> </ul>
J	OPOS 1.4.9.4 JPOS 1.13.250 UIA 2.5.0 PK-UGENxx-0250	<p>2.3.7 – Installing PayPal Authorization</p> <ul style="list-style-type: none"> <li>Removed note that PayPal Authorization for OPOS is not available (as it is now available).</li> </ul> <p>3.2 – OPOS Integration Kit Contents</p> <ul style="list-style-type: none"> <li>Removed note that PayPal Authorization for OPOS is not available (as it is now available).</li> </ul> <p>3.3.3 – Installing PayPal Authorization</p> <ul style="list-style-type: none"> <li>Added minimum installation requirements for PayPal Authorization for OPOS.</li> </ul> <p>4.3.4 – Error Event Codes</p> <ul style="list-style-type: none"> <li>Added note regarding application of the codes presented to OPOS.</li> </ul> <p>5.1.3 – (0xA1) Set UI Variable (JPOS)</p> <ul style="list-style-type: none"> <li>Updated PINENCRYPTION variable for PayPal</li> <li>Updated PINMAXKEY variable for PayPal</li> <li>Updated PINMINKEY variable for PayPal</li> <li>Created RSA_KEY variable for PayPal</li> </ul> <p>6.1 – The DirectIO Method (OPOS)</p> <ul style="list-style-type: none"> <li>Added corrective actions for return codes presented in this section’s table</li> </ul> <p>6.1.3 – (0xA1) Set UI Variable (OPOS)</p> <ul style="list-style-type: none"> <li>Updated PINENCRYPTION variable for PayPal</li> </ul>

Manual Revision	Application Revision	Changes
		<ul style="list-style-type: none"> <li>Updated PINMAXKEY variable for PayPal</li> <li>Updated PINMINKEY variable for PayPal</li> <li>Added RSA_KEY variable for PayPal</li> </ul> <p>12.2.2 – DUKPT Key Serial Number Format</p> <ul style="list-style-type: none"> <li>Added note that ‘AdditionalSecurityInformation’ property is not available for use with RSA PayPal PIN</li> </ul> <p>12.3 – RSA Encryption for PayPal PIN</p> <ul style="list-style-type: none"> <li>Added new section</li> </ul> <p>12.4.3.2 – Retrieve PIN Block (C++ Sample)</p> <ul style="list-style-type: none"> <li>Added note to use PINControl.GETEncryptedPIN() to retrieve PayPal’s PIN block</li> </ul>
I	JPOS 1.13.240 PK-UGENxx-0240	<p>2.2 – JPOS Integration Kit Contents</p> <ul style="list-style-type: none"> <li>Added Jungo driver to the list and noted a limitation with JPOS.</li> </ul> <p>2.3.7 – Installing PayPal Authorization</p> <ul style="list-style-type: none"> <li>Added minimum installation requirements for PayPal Authorization for JPOS.</li> </ul> <p>3.2 – OPOS Integration Kit Contents</p> <ul style="list-style-type: none"> <li>Added Jungo driver.</li> </ul> <p>3.4 – OPOS Configuration</p> <ul style="list-style-type: none"> <li>Added a description for the Current Prompt Language.</li> <li>Updated the PINPad tab’s image to include the language selection menu.</li> <li>Updated the Configuration Tool Form Configuration window’s Data Event Enable checkbox instruction.</li> </ul> <p>6.1.7 – Transmit Check Box Data</p> <ul style="list-style-type: none"> <li>Added a description for use with the Form Configuration window’s Data Event Enable checkbox in the configuration tool.</li> </ul> <p>6.1.8 – Transmit Survey (Radio Button) Data</p> <ul style="list-style-type: none"> <li>Added a description for use with the Form Configuration window’s Data Event Enable checkbox in the configuration tool.</li> </ul> <p>6.1.11 – (0x34) Display Text At</p> <ul style="list-style-type: none"> <li>Updated to exclude pixel display mode; provide sample code</li> </ul> <p>12.3.4 – Set PIN Pad Prompt Language</p> <ul style="list-style-type: none"> <li>Updated the C++ sample code.</li> </ul>
H	OPOS 1.4.8.3 JPOS 1.13.230 PK-UGENoX-0230Y	<p>2.4.1 – Configuring Beep Tones</p> <ul style="list-style-type: none"> <li>Added this new topic to JavaPOS Configuration.</li> </ul> <p>3.4 – OPOS Configuration</p> <ul style="list-style-type: none"> <li>Added a description of All/Any Available Readers reader type.</li> <li>Added General Tab’s Beep Tones configuration.</li> </ul> <p>4.2.4 – DirectIO Event Numbers</p> <ul style="list-style-type: none"> <li>Updated this table to include Hex Value 35 (ING_DIO_DELETE_RECEIPT_CONTENTS).</li> </ul> <p>5.1 – The DirectIO Method (for JPOS)</p> <ul style="list-style-type: none"> <li>Updated this table to include x35 (DIO_DELETE_RECEIPT_CONTENTS).</li> </ul>

Manual Revision	Application Revision	Changes
		<p>5.1.13 – Delete Receipt Contents (for JPOS)</p> <ul style="list-style-type: none"> <li>Added this new DIO command section.</li> </ul> <p>6.1 – The DirectIO Method (for OPOS)</p> <ul style="list-style-type: none"> <li>Updated this table to include x35 (DIO_DELETE_RECEIPT_CONTENTS).</li> </ul> <p>6.1.12 – Delete Receipt Contents (for OPOS)</p> <ul style="list-style-type: none"> <li>Added this new DIO command section.</li> </ul>
G	OPOS 1.4.7.11 JPOS 1.13.220 PK-UGENxx-0220	<p>1.3 –Supported Environments:</p> <ul style="list-style-type: none"> <li>Added this new section of supported and unsupported environments.</li> </ul> <p>3.4 –OPOS Configuration:</p> <ul style="list-style-type: none"> <li>Added new table of Readers to Enable Options.</li> </ul> <p>6.1.3 –Set UIA Variable:</p> <ul style="list-style-type: none"> <li>Added FormatSpecifier variable for the Clear Text Entry control to Table 5. Set UIA Variable Names.</li> </ul> <p>7 – Format Specifiers:</p> <ul style="list-style-type: none"> <li>Added this section describing the new FormatSpecifier variable’s attributes.</li> </ul> <p>13.1 – Security Prompts:</p> <ul style="list-style-type: none"> <li>Removed references to RBA; removed Prompt Type column from table.</li> <li>Added location note for SECURPROMPT.xml file.</li> </ul>
F	OPOS 1.4.6.4 JPOS1.13.160 TDA 2.0	<p>3.4 – OPOS Configuration</p> <ul style="list-style-type: none"> <li>Updated Control Panel screens.</li> <li>Added reference to UPOS spec.</li> </ul> <p>10.2 – Signature Form:</p> <ul style="list-style-type: none"> <li>Added new section.</li> </ul> <p>12.1 – General Information:</p> <ul style="list-style-type: none"> <li>Added note on DataEventEnabled property.</li> </ul> <p>12.4.4 – Set PIN Pad Prompt Language:</p> <ul style="list-style-type: none"> <li>Added new section.</li> </ul>
E	OPOS 1.4.6 JPOS 1.13.160 TDA 1.4.3	<p>1.1 – About This Manual:</p> <ul style="list-style-type: none"> <li>Added note for users updating POS applications from U32 to Telium.</li> </ul> <p>3.4 – OPOS Configuration</p> <ul style="list-style-type: none"> <li>Updated Control Panel screens.</li> </ul> <p>4 – Events:</p> <ul style="list-style-type: none"> <li>Added new chapter.</li> </ul> <p>5 – Direct I/O for JPOS:</p> <ul style="list-style-type: none"> <li>Renamed chapter.</li> </ul> <p>5.1 – The DirectIO Method:</p> <ul style="list-style-type: none"> <li>Removed Direct I/O commands.</li> <li>Removed Direct I/O result codes with labels and definitions.</li> </ul> <p>5.1.12 – Get Last Error:</p> <ul style="list-style-type: none"> <li>Removed section.</li> </ul>

Manual Revision	Application Revision	Changes
		<p>5.3.1 – Set a Variable:</p> <ul style="list-style-type: none"> <li>Added new code sample.</li> </ul> <p>5.3.2 – Clear Screen:</p> <ul style="list-style-type: none"> <li>Added new code sample.</li> </ul> <p>5.3.3 – Store a Form:</p> <ul style="list-style-type: none"> <li>Replaced code sample.</li> </ul> <p>5.3.4 – Display a Form:</p> <ul style="list-style-type: none"> <li>Replaced code sample.</li> </ul> <p>5.1.20 – (0x6A ) Activate Software:</p> <ul style="list-style-type: none"> <li>Added new section.</li> </ul> <p>5.1.6 – (0xEF) Trigger EFT/TMS Downloads:</p> <ul style="list-style-type: none"> <li>Added new section.</li> </ul> <p>6 – Direct I/O for OPOS:</p> <ul style="list-style-type: none"> <li>Added new chapter.</li> </ul> <p>8.2.1.1 – Java Sample:</p> <ul style="list-style-type: none"> <li>Added new code sample.</li> </ul> <p>8.2.2.1 – Java Sample:</p> <ul style="list-style-type: none"> <li>Added new code sample.</li> </ul> <p>8.2.3.1 – Java Sample:</p> <ul style="list-style-type: none"> <li>Added new code sample.</li> </ul> <p>9.2.1.1 – Java Sample:</p> <ul style="list-style-type: none"> <li>Added new code sample.</li> </ul> <p>9.2.2.1 – Java Sample:</p> <ul style="list-style-type: none"> <li>Added new code sample.</li> </ul> <p>9.2.3.1 – Java Sample:</p> <ul style="list-style-type: none"> <li>Added new code sample.</li> </ul> <p>10.3.3.1 – Java Sample:</p> <ul style="list-style-type: none"> <li>Added new code sample.</li> </ul> <p>12.4.1.1 – Java Sample:</p> <ul style="list-style-type: none"> <li>Added new code sample.</li> </ul> <p>12.4.3.1 – Java Sample:</p> <ul style="list-style-type: none"> <li>Added new code sample.</li> </ul> <p>E.2.5 – Sig Cap Tab:</p> <ul style="list-style-type: none"> <li>Updated Test App screens.</li> </ul>

Manual Revision	Application Revision	Changes
D	OPOS v1.4.5 JPOS v1.13.1 TDA 1.4.3	<p>2.2 – JPOS Integration Kit Contents:</p> <ul style="list-style-type: none"> <li>• Updated integration kit contents.</li> </ul> <p>5.1 – 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>5.1.17 – (0x93) Delete File:</p> <ul style="list-style-type: none"> <li>• Added new section.</li> </ul> <p>D.6 – Working With the Test Application:</p> <ul style="list-style-type: none"> <li>• Added new section.</li> </ul> <p>E.3.3 – Changing Prompt Indices :</p> <ul style="list-style-type: none"> <li>• Added new section.</li> </ul> <p>E.3.4 – Displaying Forms:</p> <ul style="list-style-type: none"> <li>• Added new section.</li> </ul>
C	OPOS v1.4.1.7 JPOS v1.13.1 TDA 1.4.3	<p>5.1 – 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>5.1.5– (0xA6) Configure PIN Pad Lights:</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 – Sending 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>



Manual Revision	Application Revision	Changes
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>5.1 – 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>5.1.3 – Set UIA Variable:</p> <ul style="list-style-type: none"> <li>• Added new variable.</li> </ul> <p>5.1.12 – (0x34) Display Text At:</p> <ul style="list-style-type: none"> <li>• Added new section.</li> </ul> <p>5.1.17 – Retrieve File:</p> <ul style="list-style-type: none"> <li>• Added new section.</li> </ul> <p>5.1.19 – (0x95) 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

---

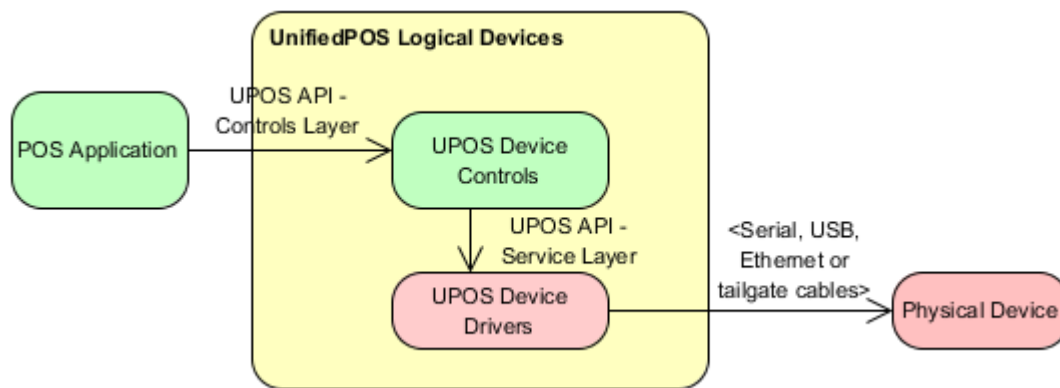
## 1.1. About This Manual

---

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

**Info** **IF YOU ARE UPDATING YOUR EXISTING U32-BASED POS APPLICATION TO WORK WITH TELIUM DEVICES**, please see Appendix A - Differences Between U32 UPOS and Telium UPOS on page 175.

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.

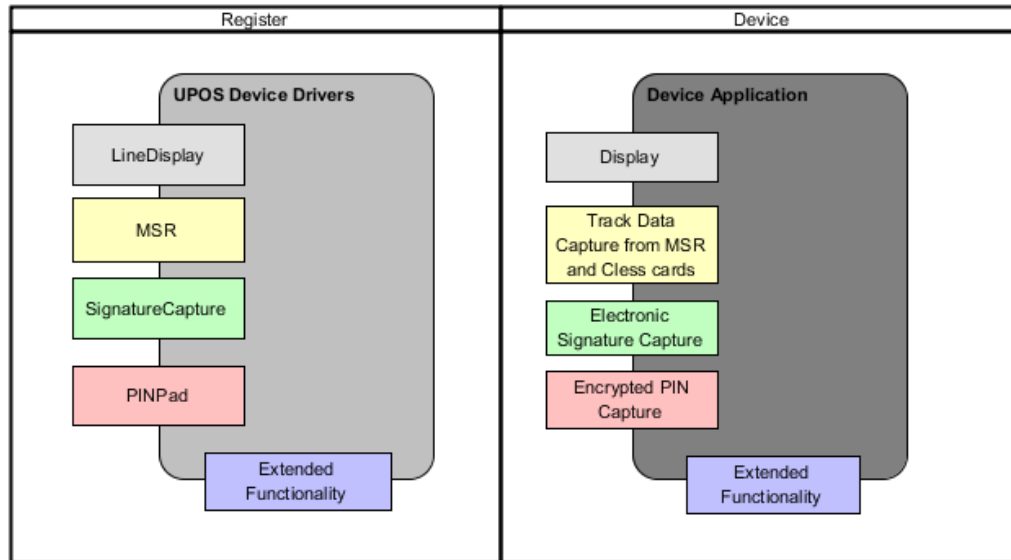


Figure 2 - Ingenico UPOS distributed application.

## 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. Supported Environments

Ingenico's UPOS-based solutions are supported by the following Windows environments:

Environment	JPOS	OPOS
WINXP	Supported	Supported
WIN7 32 bit	Supported	Supported
WIN7 64 bit	Unsupported	Supported

## 1.4. UPOS Device Drivers

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

- LineDisplay
- MSR – Magnetic Stripe Reader
- SignatureCapture

- PINPad

## 1.5. Compatible Devices

---

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

- iSC250
- iSC350
- iPP320
- iPP350

## 1.6. 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.

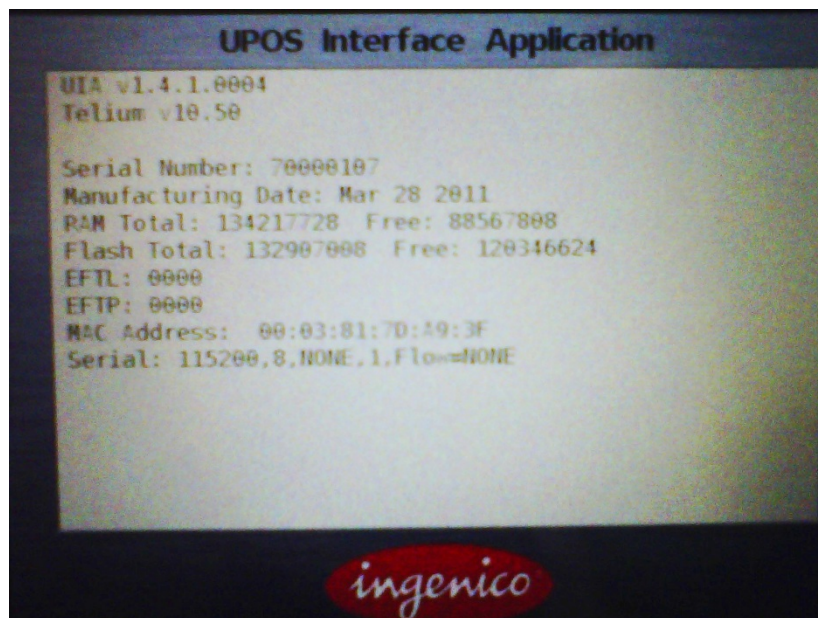


Figure 3 - UPOS interface application splash screen.

## 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

---

Ingenico's JPOS Integration Kit contains the following:

- Ingenico documentation
- JavaPOS drivers:
  - JavaPOS API
  - JavaPOS Implementation for Sun and IBM
  - JavaComm for Win 32 and USB-HID
  - Jungo\*
  - Log4J
  - Telium Sigdecoder v1.0
  - Xerces v1.0
- info

*\* It is recommended that when using JPOS and the Jungo CDC driver, the Sun Java comm driver is employed as it allows the creation of virtual comms without restriction in assignment of COM-numbers. Use of the Jungo CDC driver with the IBM Java comm driver is also acceptable, however, the following limitation exists: a virtual comm will need to be created where the COM-number range is limited to COM1, COM2... COM9 (e.g., COM10 and above will fail).*
- Telium utilities:
  - JavaPOS Test App
  - Telium LLT
  - Telium Tools:
    - Form Builder
    - Data Packaging Tool
    - Script Builder
    - SAT 1.5
- 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 24 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. Installing PayPal Authorization

Authorization using PayPal is included in the JPOS device driver installation. Other than the need to obtain and add a signed, encrypted PayPal public key, no additional configuration for JPOS is required.

#### 2.3.7.1 PayPal Minimum Requirements

Minimum requirements for using PayPal as a part of a JPOS production solution include the following:

- UIA version PK-UGENox-0240x or newer
- Access to GMT DateTime in addition to Local DateTime (provided by the device driver; no additional configuration is necessary). GMT- and Local- DateTimes are sent when an open connection is established with the PIN Pad device.
- Signed, encrypted PayPal public key (as with other keys, this is completed outside of the driver installation process) from PayPal or Ingenico. Follow the individual PIN Pad device's Operation and Product Support Guide for this requirement.

 *PayPal's encrypted pin block is significantly larger than those observed with other authorization types. PayPal's PIN block is Base64 encoded (thus, a 128 byte key is equal to a pin block of up to 210 bytes).*

### 2.3.8. 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`].

### 2.4.1. Configuring Beep Tones

To change the form's beep tones, you will need to alter the .XML file.

**Info** Unless otherwise set while creating the form in the FormBuilder application, the default attribute values for each of the Form Beep Tones are as follows:

Beep Tone Attribute	Description	Tone1 Defaults	Tone2 Defaults
Timer	Duration of time the tone is heard in milliseconds	15 milliseconds	30 milliseconds
Frequency	Pitch of the tone, measured in Hertz; a higher number is equivalent to a higher pitch	4200 Hz	2100 Hz

Sample JPOS code:

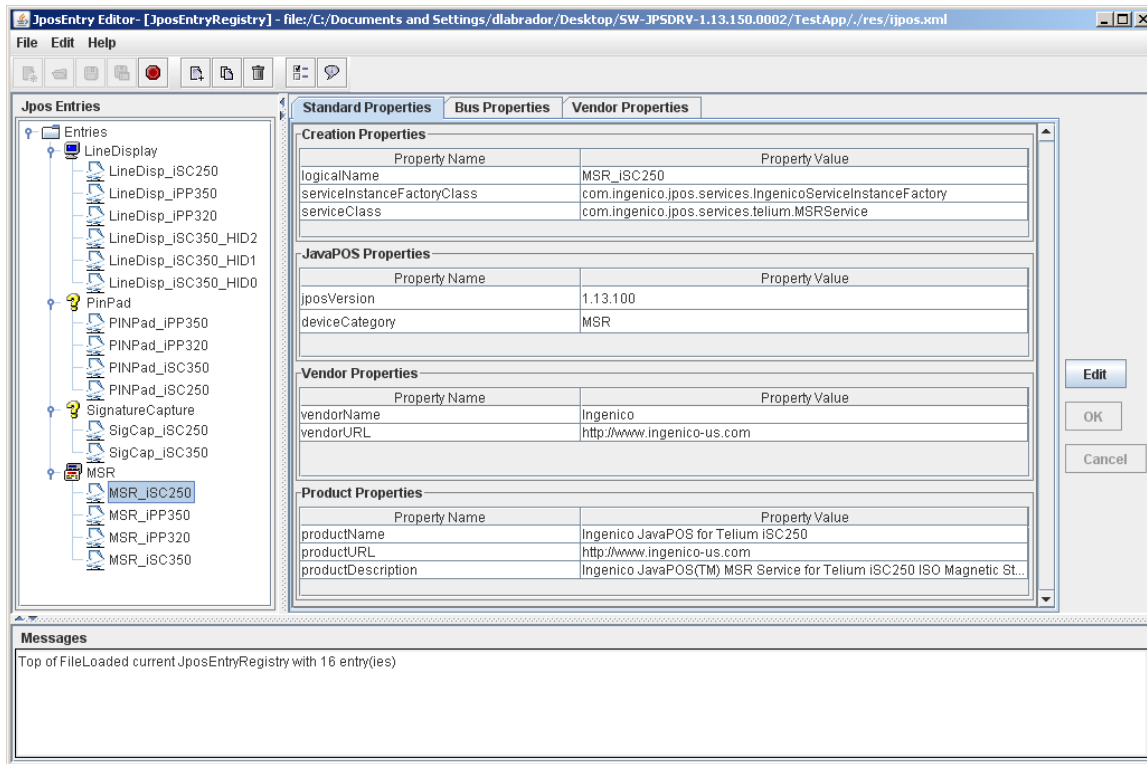
The JPOS configuration entries for tone can be added in 'ijpos.xml file' as follows under corresponding logical control:

```
<prop name="tone1Timer" type="String" value="15"/>
<prop name="tone1Frequency" type="String" value="4200"/>
<prop name="tone2Timer" type="String" value="30"/>
<prop name="tone2Frequency" type="String" value="2100"/>
```

The configured tone properties are set when the device is enabled in this call.  
`jpos.BaseJposControl.setDeviceEnabled(true);`

## 2.5. JPOS Entry Editor

The Java Configuration Loader comes with a Java-based GUI (`ConfigurationTool.bat`) 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, and 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.

## Notes

## 3. Getting Started with the OPOS Integration Kit

---

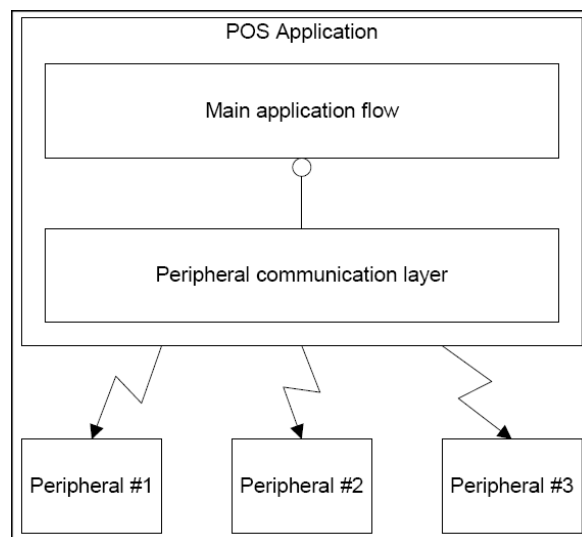
### 3.1. OPOS Overview

---

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).

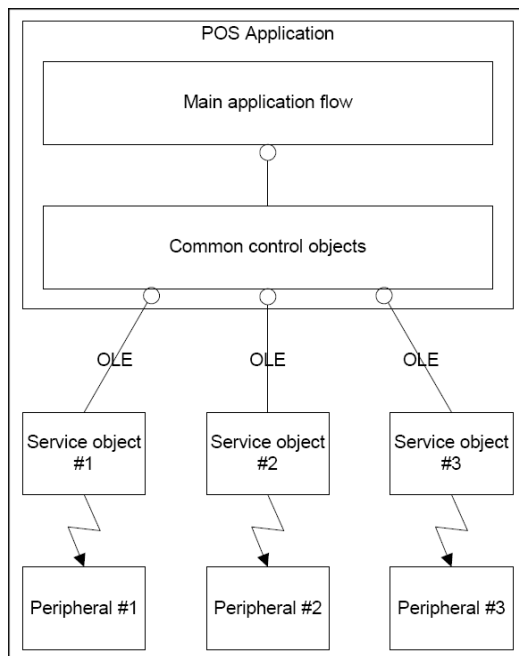


**Figure 5 - Architectural overview.**

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

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*)
- Jungo CDC driver
- *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 - UPOS Developer's Guide for Telium Devices)
  - 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 50 for more information.

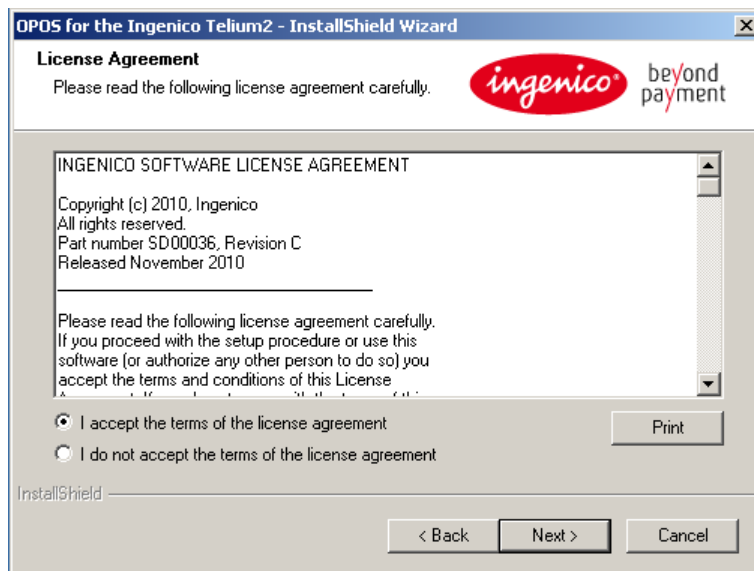
To install your OPOS Integration Kit, do the following:

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

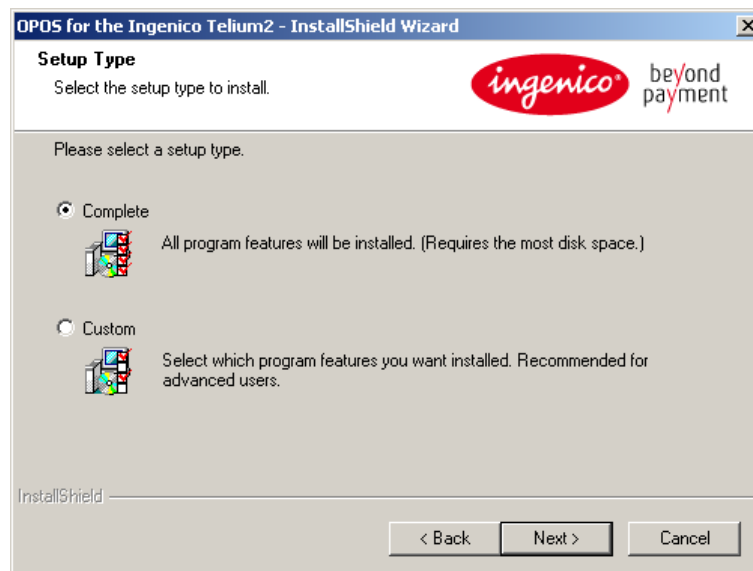
3. The Welcome window displays. Click **Next**.



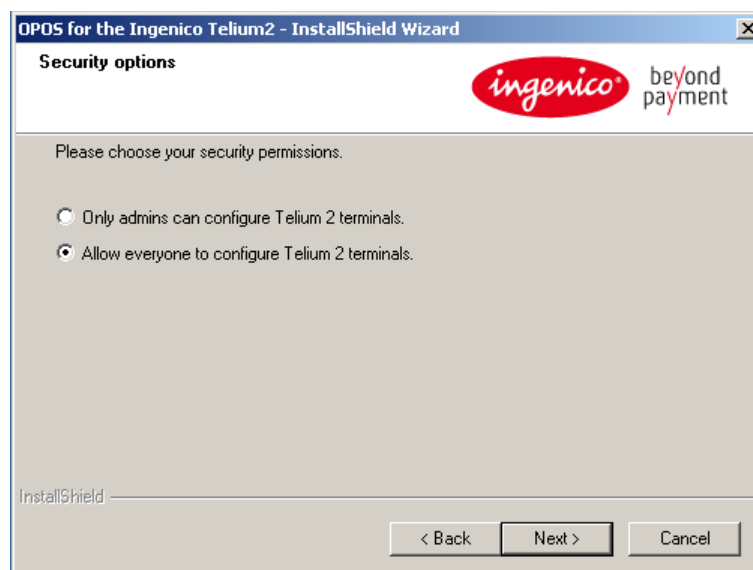
4. The License Agreement window displays. Read through the agreement and if you agree, click **I accept the terms of the license agreement** and click **Next** to accept it.



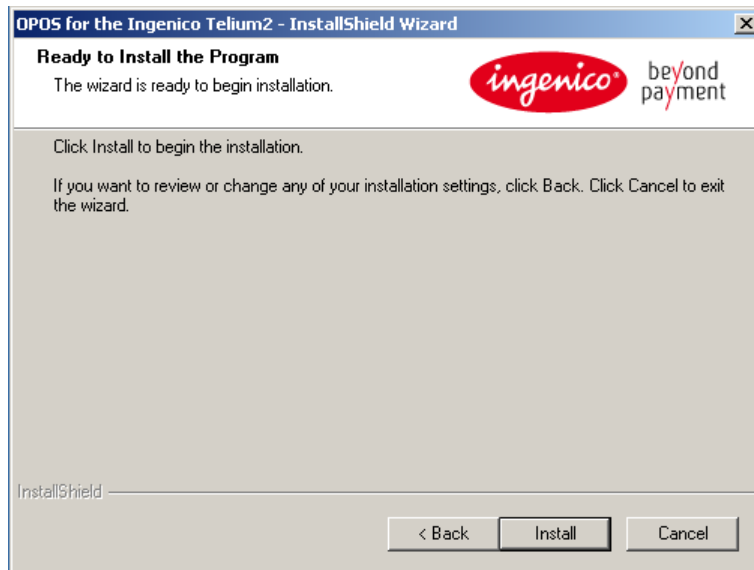
5. The Setup Type window displays. Select **Complete** to install all program features, and then click **Next** to continue.



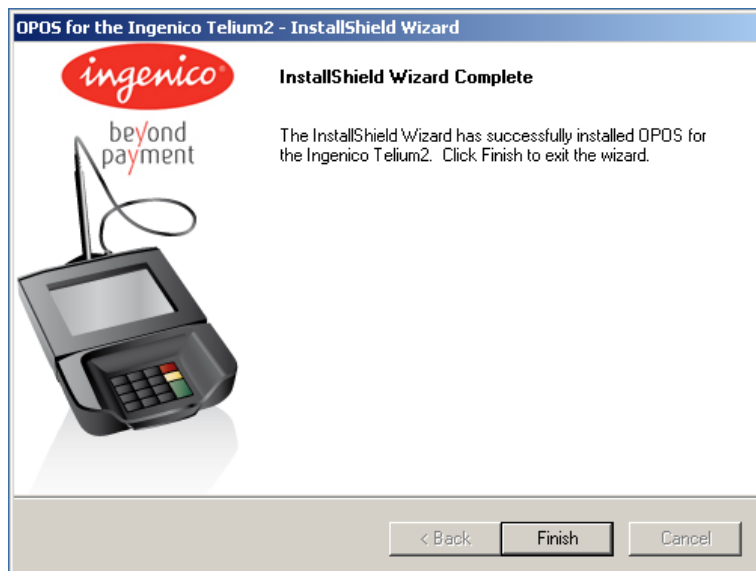
6. The Security options window displays. Select whether you only want administrators to configure Telium 2 PIN pads or if you want everyone to be able to configure Telium 2 PIN pads. Click **Next**.



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.

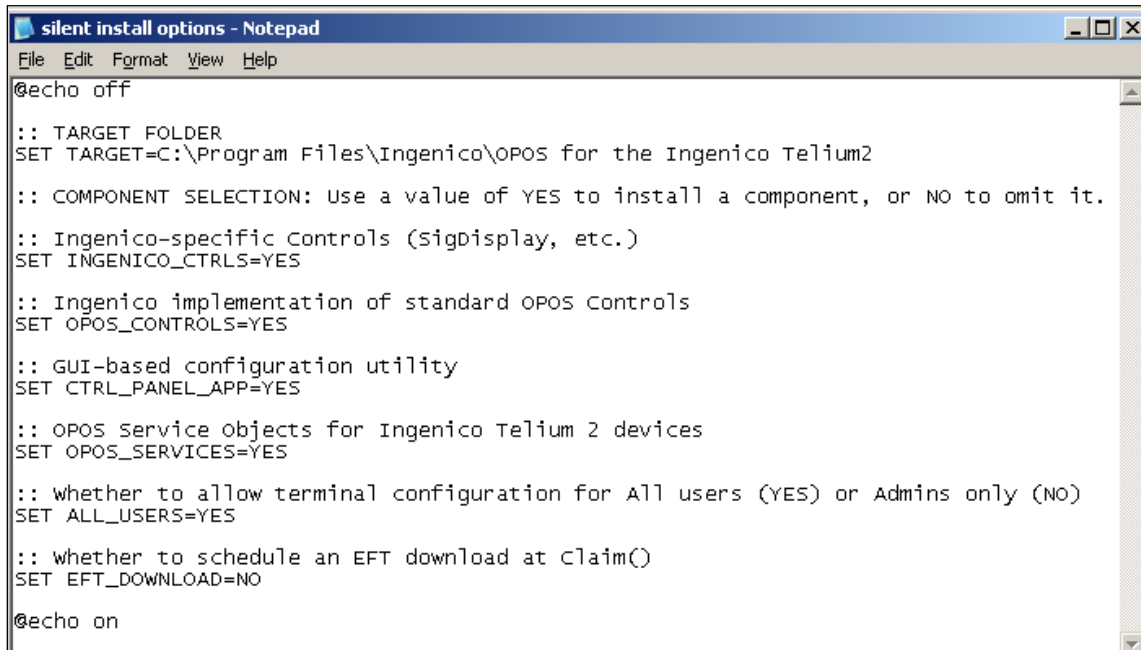


### 3.3.2. Silent Installation

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.

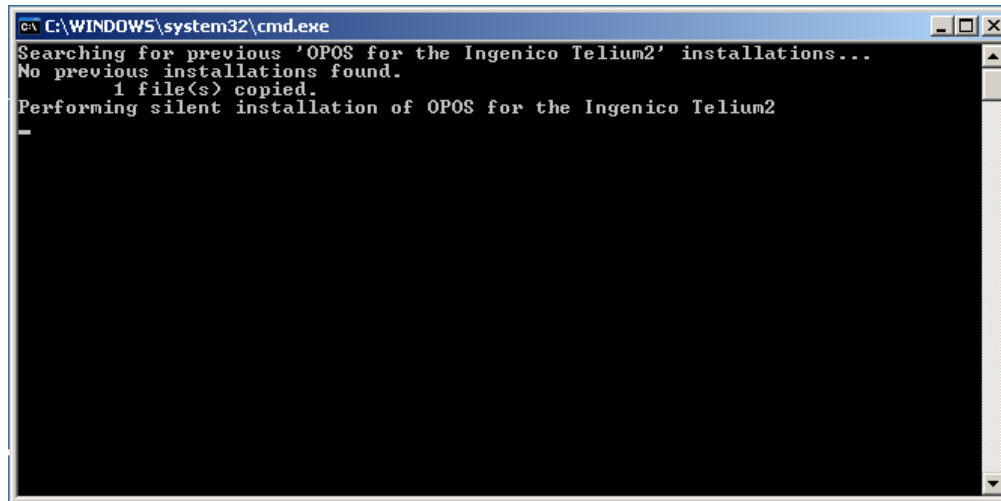
It is important to note that the OPOS Control Panel silent install WILL NOT overwrite an IngenicoConfig.xml file created by a previous installation. Users who wish to replace the IngenicoConfig.xml file during a silent install will need to modify the silent install options.bat file to add the following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


### 3.3.3 Installing PayPal Authorization

Authorization using PayPal is included in the OPOS device driver installation. Other than the need to obtain and add a signed, encrypted PayPal public key, no additional configuration for OPOS is required.

#### 3.3.3.1 PayPal Minimum Requirements

Minimum requirements for using PayPal as a part of an OPOS production solution include the following:

- UIA version PK-UGENox-0250x or newer
- Access to GMT DateTime in addition to Local DateTime (provided by the device driver; no additional configuration is necessary). GMT- and Local- DateTimes are sent when an open connection is established with the PIN Pad device.
- Signed, encrypted PayPal public key (as with other keys, this is completed outside of the driver installation process) from PayPal or Ingenico. Follow the individual PIN Pad device's Operation and Product Support Guide for this requirement. Use the Set UIA Variable [see section 6.1.3 (0xA1) Set UIA Variable] to set the RSA\_KEY variable (the public key name) to the PIN Pad.

 *PayPal's encrypted pin block is significantly larger than those observed with other authorization types. PayPal's PIN block is Base64 encoded (thus, a 128 byte key is equal to a pin block of up to 210 bytes).*



## 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.

The screenshot shows the 'General' tab of the Ingenico iSeriesTelium2 configuration window. It features several sections: 'Logical Device Names' with a list box and 'Edit', 'Save', 'Undo', 'Add', and 'Remove' buttons; 'Model Type' with a list box; 'Form Beep Tones' with fields for 'Tone1' and 'Tone2' including 'Duration' (with a '1/100 secnd' multiplier) and 'Frequency' (with a 'Hz' multiplier), each with an 'Edit' button; a '5992 Mode' checkbox; 'Claim Min Window Timeout Interval' and 'Save File Timeout Interval' fields with 'Static (millisec)' and 'Static (sec)' labels; and a 'Retrieve File' field with an 'Edit' button and a 'Browse Folder' button.


2. On the **General** tab:
  - a. You may choose to **Edit**, **Add**, or **Remove** a Logical Device if desired.
  - b. Pick the correct Logical Device **Model Type** from the corresponding drop-down list.
  - c. To change the form's beep tones, type your preferred values into the **Duration** (in centiseconds) and **Frequency** (in Hertz) fields for each of the tones assigned.

**Info** Unless otherwise set while creating the form in the FormBuilder application, the default attribute values for each of the Form Beep Tones are as follows:

Beep Tone Attribute	Description	Tone1 Defaults	Tone2 Defaults
Timer	Duration of time the tone is heard in centiseconds (1/100 <sup>th</sup> of a second)	15 centiseconds	30 centiseconds

Beep Tone Attribute	Description	Tone1 Defaults	Tone2 Defaults
Frequency	Pitch of the tone, measured in Hertz; a higher number is equivalent to a higher pitch	4200 Hz	2100 Hz

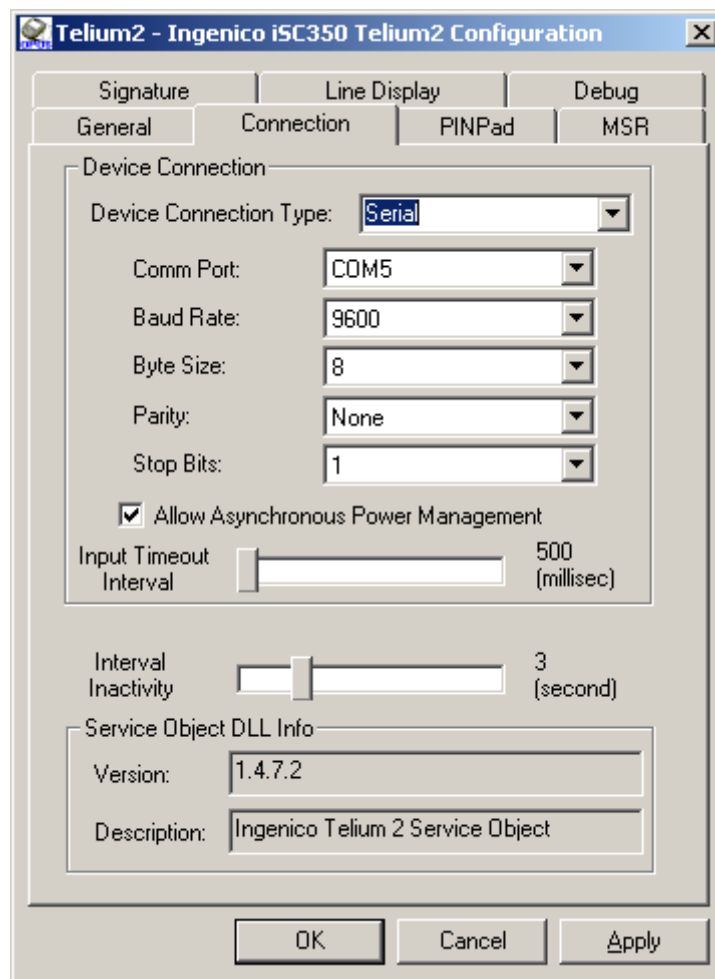
d. Check the **5992 Mode** checkbox to enable 5992 mode.

 See 5992 Mode on page 187 for more information.

e. Pick the **Claim Min Window Timeout Interval** using the corresponding slider.

f. Pick the desired **Save File Timeout Interval** using the corresponding slider.

g. Specify the location to save any files retrieved from the PIN pad in the **Retrieve File** field.




3. On the Connection tab:

a. Select the desired connection type in the **Device Connection Type** drop-down list:

i. 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.

- ii.** If using **Serial**, fill in the fields as follows:
    - In the **Comm Port** box, select the appropriate communications port.
    - In the **Baud Rate** box, select the desired baud rate.
    - In the **Byte Size** box, select **8**.
    - In the **Parity** box, select **None**.
    - In the **Stop Bits** box, select **1**.
  - b.** Specify Power Management options:
    - i.** Enable/disable **Asynchronous Power Management** using the corresponding check box.
    - ii.** Specify the **Input Timeout Interval** using the corresponding slider.
  - c.** Specify **Interval Inactivity** using the corresponding slider.
-  See the POS Power chapter in version 1.13 of the UnifiedPOS specification for more information on power management.

4. Click the PINPad tab.

The screenshot shows the 'ISC250 - Ingenico iSC250 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. The main area contains several sections: 'Pin Entry Form' with a 'Form name' field set to 'PINSC250.K3Z'; 'PIN Entry Timeouts (in seconds)' with 'First Key' and 'Inter Key' sliders both set to 30; 'PIN Entry min/max input digits' with 'Min input Key' set to 4 and 'Max input Key' set to 12; 'Configure Options' with a 'Current Prompt Language' dropdown menu showing 'English' and a 'Clear Screen after Pin E' checkbox; and 'Configure Forms' with a 'Form Config' button. At the bottom are 'OK', 'Cancel', and 'Apply' buttons.

- 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. Select the language [English (US), French (CAN) or Spanish (MEX)] for prompts displayed on the PIN Pad from the **Current Prompt Language** drop down list. The selected value will be stored in the Configuration.xml file.

**Info** If OPOS reads an invalid "Prompt Language" value from the Configuration.xml file, it will default to the English language. See Section 12.3.4 Set PIN Pad Prompt Language for a C++ code sample.

- e. Check the **Clear Screen after Pin Entry** check box to clear the screen after PIN entry.

- f. Click the **Form Config** button to enable/disable the display of specific PIN entry forms for various OPOS control events.

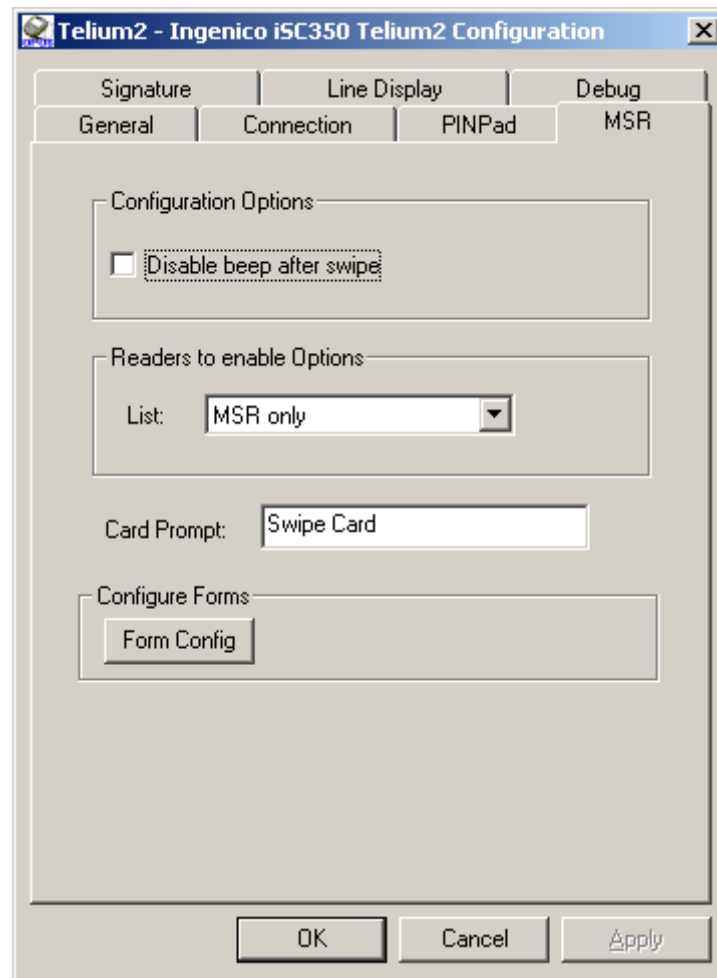
The image shows a 'Form Configuration' window with a blue title bar and a close button. The window contains several sections, each with an 'Enable' checkbox and a 'Name' text field. The sections are: 'Claim', 'Enable Device', 'Disable Device', 'Data Event', 'Error Event', and 'Release'. In all sections, the 'Enable' checkbox is unchecked and the 'Name' field contains the text 'PINSC350.K3Z'. At the bottom of the window, there 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.

***Info** If you are running the polling option and want to receive data back on a form that contains a checkbox or a set of radio buttons, leave the Data Event Enable checkbox unchecked. See also sections 6.1.7 Transmit Check Box Data and 6.1.8 Transmit Survey (Radio Button) Data. Applies to iSCxxx Telium2 devices only.*

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

5. 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.

Readers to Enable Options	Description
MSR Only	OPOS Driver will activate only the MSR reader type in the device.
CLESS Only	OPOS Driver will activate only the CLESS (Contactless) reader type in the device.
MSR and CLESS	OPOS Driver will activate both reader types (MSR and CLESS) in the device; if either reader type becomes unavailable, the driver will not allow the other type to function.
All/Any Available Readers	OPOS Driver will activate all/any reader types (MSR and CLESS) in the device; if either reader type becomes unavailable, the driver will allow the other type to function. See also, Info note, below.




*'All/Any Available Readers' is a generic, flexible reader type, allowing multiple reader types to be configured in a single environment. For example, suppose your company has*

implemented some devices with Contactless support, and others without Contactless support. If the intention is to configure all devices with a single OPOS configuration, select 'All/Any Available Readers'. Doing so will prevent the OPOS Driver from rejecting the Device Enable request on the device(s) with an unavailable or missing reader type. If 'MSR and CLESS' is selected, and only one of the reader types is present on the device, the OPOS Driver will reject the Device Enable request.

- 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. The dialog is organized into a 'Forms' section with several sub-sections, each containing an 'Enable' checkbox and a 'Name' text field. The 'Name' field in all sections is populated with 'SWIPE640.K3Z'. The sub-sections are: 'Claim' (checkbox unchecked), 'Enable Device' (checkbox checked), 'Disable Device' (checkbox unchecked), 'Data Event' (checkbox unchecked), 'Error Event' (checkbox unchecked), and 'Release' (checkbox unchecked). At the bottom of the dialog 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.

 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.

6. Click the Signature tab.

The screenshot shows the 'IngSC3XX - Ingenico iSC350 Telium2 Configuration' dialog box with the 'Signature' tab selected. The dialog has several tabs: 'General', 'Connection', 'PINPad', 'MSR', 'Signature', 'Line Display', and 'Debug'. The 'Signature' tab contains the following elements:

- Logical Form Names:** A section with a 'List:' dropdown menu showing 'Sigform', a 'Name:' text field also containing 'Sigform', and buttons for 'Edit', 'Save', 'Undo', 'Add', and 'Remove'.
- Form name:** A text field containing 'SIGSC350.K3Z'.
- Maximum Signature Size (in Kilobytes):** A slider control set to '8'.
- Configure Forms:** A button labeled 'Form Config'.
- Checkboxes:** Two unchecked checkboxes labeled 'Implicit Signature Retrieval' and 'Clear Screen after Signature'.
- Buttons:** 'OK', 'Cancel', and 'Apply' buttons at the bottom.

- 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 **Implicit Signature Retrieval** check box to allow the POS to retrieve signature data without checking the Signature control's `DataEventEnabled` property.
- e. Check the **Clear Screen after Signature** check box to clear the screen after signature.
- f. Click the **Form Config** button to enable/disable the display of specific PIN entry forms for various OPOS control events.

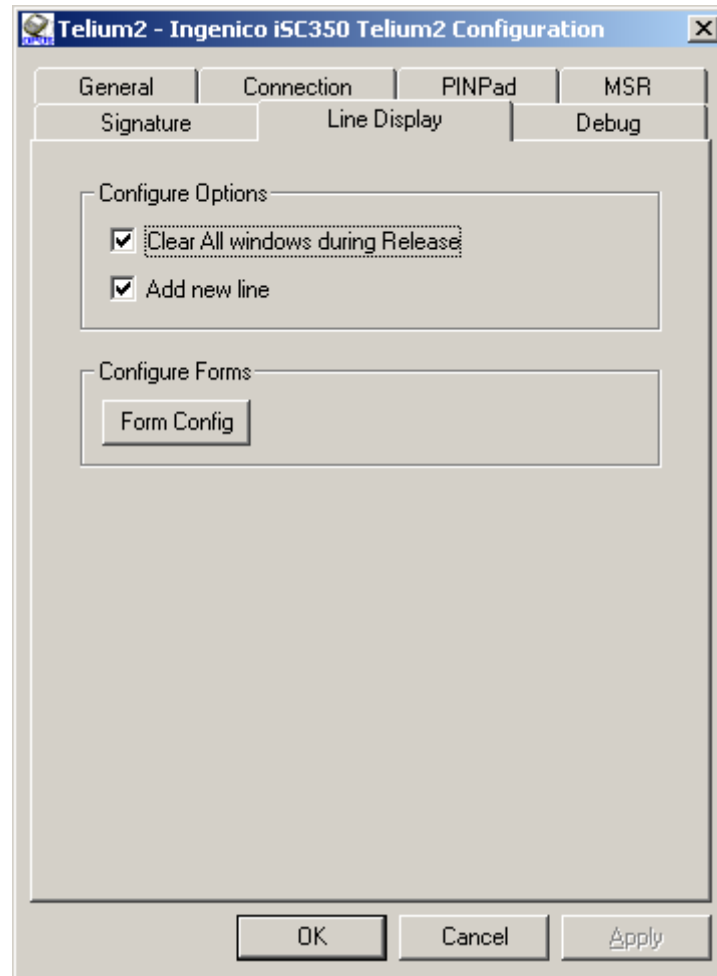


The image shows a 'Form Configuration' window with a title bar containing a close button. The window contains a 'Forms' section with six sub-sections: 'Claim', 'Enable Device', 'Disable Device', 'Data Event', 'Error Event', and 'Release'. Each sub-section has an 'Enable' checkbox and a 'Name' text field. All checkboxes are currently 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.

7. 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.

**Form Configuration**

Forms

Claim

☐ Enable Name: LDSC350.K3Z

Enable Device

☐ Enable Name: LDSC350.K3Z

Disable Device

☐ Enable Name: LDSC350.K3Z

Data Event

☐ Enable Name: LDSC350.K3Z

Error Event

☐ Enable Name: LDSC350.K3Z

Release

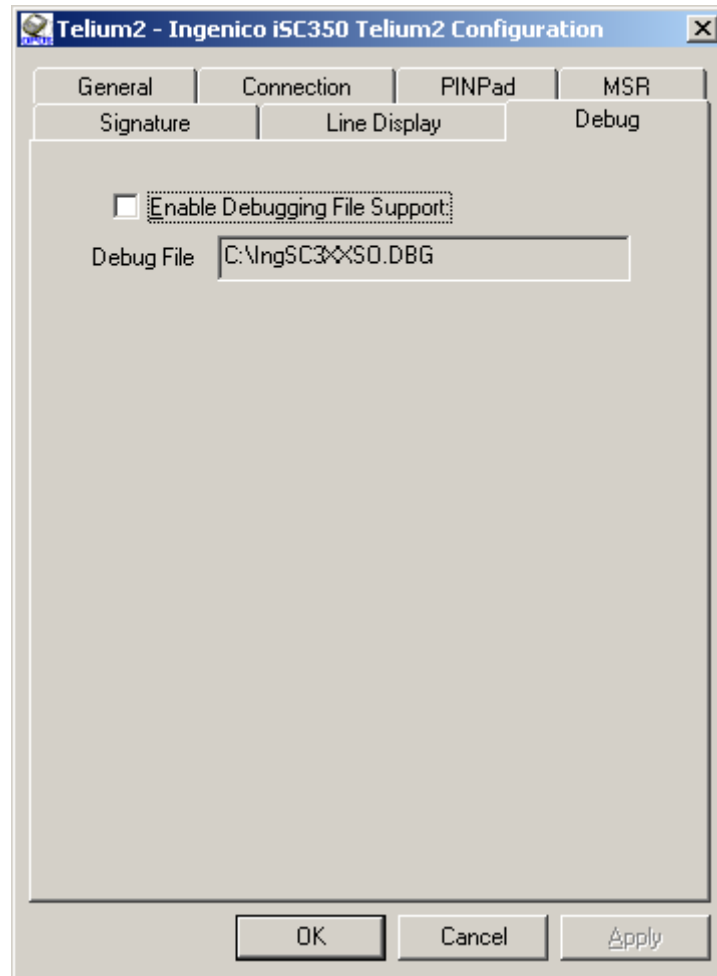
☐ Enable Name: LDSC350.K3Z

Save and Exit 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.

8. 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.
9. 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

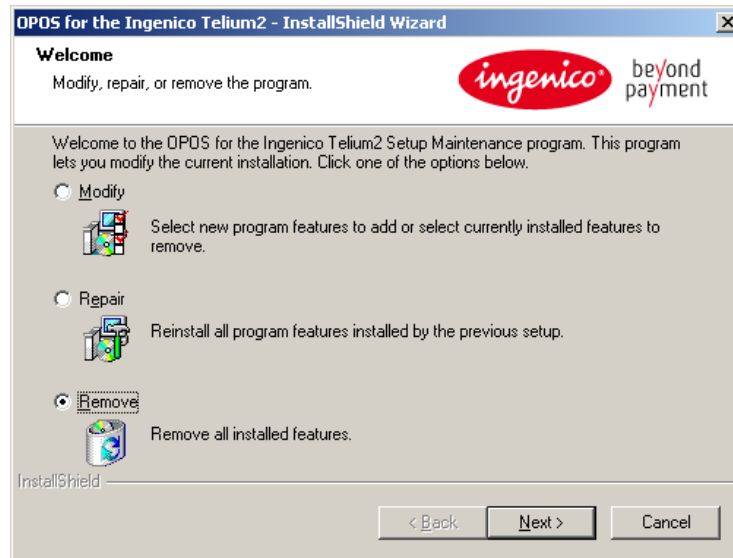
---

### 3.5.1. Typical Uninstall

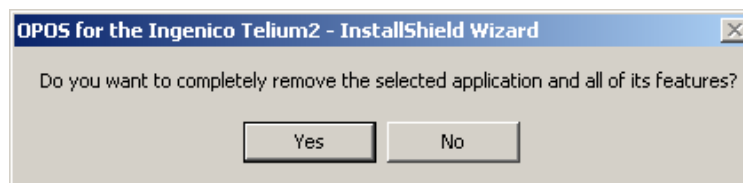
To uninstall the OPOS Integration Kit:

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.



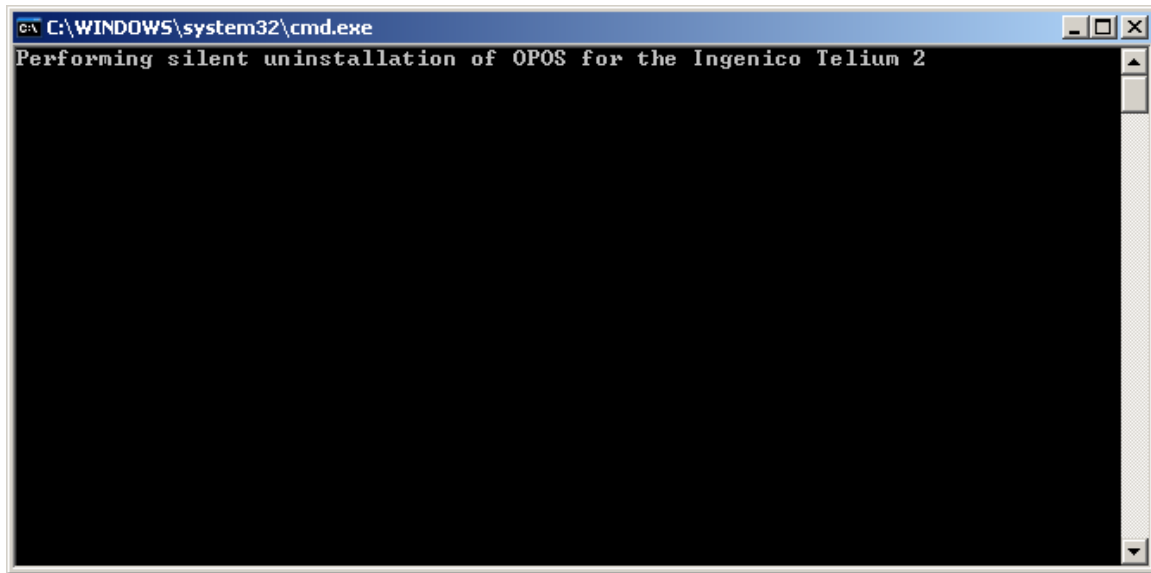
5. The Uninstall Complete window displays. Click **Finish** to close the screen.



### 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. 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.

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

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. Events

---

Telium OPOS/JPOS-equipped PIN pads use events to inform the POS about activities/changes that take place on the PIN pad. When a command is received from the POS, the OPOS/JPOS driver verifies the data and responds by firing the corresponding type of event.

### 4.1. Data Events

---

UIA/OPOS/JPOS fire data events to notify the POS about data entered and stored using PIN pad buttons or forms.

#### 4.1.1. Event Object Format

Ingenico's implementation of data event objects follows the UnifiedPOS 1.13 specification exactly.

#### 4.1.2. Event Signature – JPOS

<b>Formulation</b>	<b>public DataEvent(Object source, int status)</b>
<b>Source</b>	The <b>source</b> object is any of the UPOS controls (MSR, LineDisplay, SignatureCapture, and PINPad).
<b>Status</b>	The <b>status</b> value depends on the device category. It may describe the type or qualities of the input data.

#### 4.1.3. Event Signature – OPOS

<b>Formulation</b>	<b>void ClassObject::OnDataEvent(long status)</b>
<b>Source</b>	The <b>source</b> object is any of the UPOS controls (MSR, LineDisplay, SignatureCapture, and PINPad).
<b>Status</b>	The <b>status</b> value depends on the device category. It may describe the type or qualities of the input data.

#### 4.1.4. Data Event Statuses

Event codes for data events are listed in the table below:

Object	Status	Description
	1	PPAD_SUCCESS
	2	PPAD_CANCEL
PINPad	5	IPPAD_RSA_SUCCESS
	0x00	JPOS_RC_SUCCESS – default.
	0x05	JPOS_RC_SMC_INSERTED – SMART card inserted.
	0x06	JPOS_RC_SMC_REMOVED – SMART card removed.
	0x07	JPOS_RC_SMC_SSA_CLOSED – SMART card SSA closed.
	0x80	JPOS_RC_MORE_DATA – more data.
	0xE1	JPOS_RC_OBSOLETE – obsolete.

Object	Status	Description
	0xF0	JPOS_RC_NO_DATA – no data.
	0xF6	JPOS_RC_NOT_FOUND – not found.
	0xFA	JPOS_RC_NOT_PRESENT – not present.
	0x12	JPOS_RC_VERIFICATION_FAILED – verification failed.
	0xF1	JPOS_RC_NO_KEY – there is no key.
	0xF2	JPOS_RC_TIMEOUT – a timeout occurred.
	0xF4	JPOS_RC_BAD_PARAMETER – bad parameter.
	0xF7	JPOS_RC_SECURITY_FAIL – security check failed.
	0xF8	JPOS_RC_SYSTEM – system error.
	0xF9	JPOS_RC_SIGCAP_BUF_FULL – signature capture buffer is full.
	0xFC	JPOS_RC_BAD_STX_COMMAND – bad STX command.
	0xFD	JPOS_RC_UNKNOWN_COMMAND – unknown command.
	0xFF	JPOS_RC_BAD_CHECKSUM – bad checksum.
	0x01	JPOS_RC_FC2_SECURITY_ENABLE – security enabled.
	0x02	JPOS_RC_FC2_SECURITY_DISABLED – security disabled.
	0x00	JPOS_SBC_DEFAULT
	0x40	JPOS_SBC_F1 – F1: '@'
	0x41	JPOS_SBC_F2 – F2: 'A'
	0x42	JPOS_SBC_F3 – F3: 'B'
	0x43	JPOS_SBC_F4 – F4: 'C'
	0x3C	JPOS_SBC_DOWN – Down: '<'
	0x3F	JPOS_SBC_UP – Up: '?'
	0x3B	JPOS_SBC_CANCEL – Cancel
	0x3D	JPOS_SBC_CLEAR – Clear
	0x3A	JPOS_SBC_ENTER – Enter
	0x01	JPOS_MSC_STARTCMD_SEND_ENABLE_MSR
	0x02	JPOS_MSC_STARTCMD_SEND_ENABLE_CPEM – RFID
	0x01	JPOS_MSC_STARTCMD_RECV_ENABLE_MSR
	0x02	JPOS_MSC_STARTCMD_RECV_ENABLE_CPEM
	0x04	JPOS_MSC_STARTCMD_RECV_ENABLE_SMARTCARD
	0x00	JPOS_MSC_POLLCMD_RECV_SOURCE_MSR
	0x01	JPOS_MSC_POLLCMD_RECV_SOURCE_CPEM
	0x02	JPOS_MSC_POLLCMD_RECV_SOURCE_SMARTCARD
	0x03	JPOS_MSC_POLLCMD_RECV_SOURCE_MANUALENTY - manual entry
	0xFF	JPOS_DATA_START_FILE_INDICATOR – start file indicator

Object	Status	Description
	0xF1	JPOS_MSR_ERR_TRACK_1 – MSR track 1 start of error packet
	0xF2	JPOS_MSR_ERR_TRACK_2 – MSR track 2 start of error packet
	0x73	JPOS_MSR_ERR_TRACK_3 – MSR track 3 start of error packet
	0x61	JPOS_MSR_GOOD_TRACK_1 – MSR track 1 start of successful read packet
	0x25	JPOS_MSR_START_SENTINEL_TRACK_1 – MSR track 1 start of successful read packet
	0x5E	JPOS_MSR_FIELD_SEP_TRACK_1 – MSR track 1 start of successful read packet
	0x3F	JPOS_MSR_END_SENTINEL_TRACK_1 – MSR track 1 start of successful read packet
	0x62	JPOS_MSR_GOOD_TRACK_2 – MSR track 2 start of successful read packet
	0x3B	JPOS_MSR_START_SENTINEL_TRACK_2 – MSR track 1 start of successful read packet
	0x6F	JPOS_MSR_END_OF_DATA – MSR end packet
	0x00	JPOS_RC_GOOD_READ – good read
	0x01	JPOS_RC_READ_ERROR – read error
	0x02	JPOS_RC_PARITY_ERROR – parity error
	0x03	JPOS_RC_LRC_ERROR – LRC error
	0x04	JPOS_RC_NO_END_SENTINEL – no END sentinel found
	0x05	JPOS_RC_NO_START_SENTINEL – no START sentinel found
	0x06	JPOS_RC_NO_TRACK_DATA – no track data
	0x07	JPOS_RC_MISSING_FIELD_SEP – missing field separator
	0x08	JPOS_RC_NOT_BANKING – not banking
	0x09	JPOS_RC_TRACK_PARSE_ERROR – track parse error
	0x0A	JPOS_RC_TOO_MANY_CARDS – too many cards (RF)

## 4.2. DirectIO Events

OPOS/JPOS fire DirectIO events to confirm the receipt of DirectIO commands from the POS or when additional information is required by the POS after a data event is received.

### 4.2.1. Event Object Format

Ingenico's implementation of DirectIO event objects follows the UnifiedPOS 1.13 specification exactly.

### 4.2.2. Event Signature – JPOS

<b>Formulation</b>	<code>public DirectIOEvent(Object source, int eventNumber, int data, Object object)</code>
<b>Source</b>	The <b>source</b> object is any of the UPOS controls (MSR, LineDisplay, SignatureCapture, and PINPad).
<b>Event Number</b>	The <b>Event Number</b> values are specified by UPOS.
<b>Data</b>	The <b>data</b> parameter describes the result of the DirectIO command.

<b>Object</b>	The <b>object</b> parameter can be an Object or NULL. Additional data from PIN pad based on the event type.
---------------	---

### 4.2.3. Event Signature – OPOS

<b>Formulation</b>	<code>void ClassObject::OnDirectIOEvent(long EventNumber, long FAR* pData, BSTR FAR* pString)</code>
<b>Event Number</b>	The <b>Event Number</b> values are specified by UPOS.
<b>pData</b>	The <b>pData</b> parameter describes the result of the DirectIO command.
<b>pString</b>	The <b>pString</b> parameter can be an Object or NULL. Additional data from PIN pad based on the event type.

### 4.2.4. DirectIO Event Numbers

Event codes for Telium UIA DirectIO events are listed in the table below:

Number	Hex Value	Description	Data	Object
0	00	ING_DIO_SEND_RAW_DATA	No response.	Additional data.
1	01	MSR Control Reception. Used to indicate source of card read when both MSR and contactless are enabled. This event number is only sent when PIN pad is set to use both MSR and contactless.	Represents the card read source. Allowable values: <ul style="list-style-type: none"> <li>0 = MSR</li> <li>1 = Contactless reader</li> </ul>	NULL
9	09	ING_DIO_REBOOT	Allowable values: <ul style="list-style-type: none"> <li>-1 = a DirectIO error occurred</li> <li>5 = DirectIO success after resetting the PIN pad</li> </ul>	Additional data.
10	0A	ING_DIO_ABORT	No response.	Additional data.
13	0D	ING_DIO_GETSTATS	Allowable values: <ul style="list-style-type: none"> <li>-1 = a DirectIO error occurred</li> <li>0 = DirectIO succeeded</li> </ul>	Additional data.
18	12	ING_DIO_GET_CHECKBOX	Allowable values: <ul style="list-style-type: none"> <li>-1 = a DirectIO error occurred</li> <li>0 = DirectIO succeeded</li> </ul>	Additional data.
19	13	ING_DIO_GET_RADIO	Allowable values: <ul style="list-style-type: none"> <li>-1 = a DirectIO error occurred</li> <li>0 = DirectIO succeeded</li> </ul>	Additional data.

Number	Hex Value	Description	Data	Object
48	30	ING_DIO_CLEAR_LD	Allowable values: <ul style="list-style-type: none"> <li>-1 = a DirectIO error occurred</li> <li>0 = DirectIO succeeded</li> </ul>	Additional data.
49	31	ING_DIO_CLRSCRN	Allowable values: <ul style="list-style-type: none"> <li>-1 = a DirectIO error occurred</li> <li>0 = DirectIO succeeded</li> </ul>	Additional data.
50	32	ING_DIO_DISPLAY_TEXT	Allowable values: <ul style="list-style-type: none"> <li>-1 = a DirectIO error occurred</li> <li>0 = DirectIO succeeded</li> </ul>	Additional data.
52	34	ING_DIO_DISPLAY_TEXT_AT	Allowable values: <ul style="list-style-type: none"> <li>-1 = a DirectIO error occurred</li> <li>0 = DirectIO succeeded</li> </ul>	Additional data.
53	35	ING_DIO_DELETE_RECEIPT_CONTENTS	Allowable values: <ul style="list-style-type: none"> <li>-1 = a DirectIO error occurred</li> <li>0 = DirectIO succeeded</li> </ul>	Additional data.
54	36	ING_DIO_POSITION_TEXT_CURSOR	Allowable values: <ul style="list-style-type: none"> <li>-1 = a DirectIO error occurred</li> <li>0 = DirectIO succeeded</li> </ul>	Additional data.
106	6A	ING_DIO_INSTALL_FIRMWARE	Allowable values: <ul style="list-style-type: none"> <li>-1 = a DirectIO error occurred</li> <li>4 = DirectIO success after activation</li> </ul>	Additional data.
145	91	ING_DIO_SAVE_FILE	Allowable values: <ul style="list-style-type: none"> <li>-1 = a DirectIO error occurred</li> <li>0 = DirectIO succeeded</li> <li>1 = DirectIO file(s) download progress when folder name was specified</li> <li>2 = DirectIO file download progress</li> <li>3 = DirectIO notification to client to activate firmware</li> </ul>	Additional data.

Number	Hex Value	Description	Data	Object
146	92	ING_DIO_RUN_FILE	Allowable values: <ul style="list-style-type: none"> <li>-1 = a DirectIO error occurred</li> <li>0 = DirectIO succeeded</li> </ul>	Additional data.
147	93	ING_DIO_DELETE_FILE	Allowable values: <ul style="list-style-type: none"> <li>-1 = a DirectIO error occurred</li> <li>0 = DirectIO succeeded</li> </ul>	Additional data.
148	94	ING_DIO_RETRIEVE_FILE	Allowable values: <ul style="list-style-type: none"> <li>-1 = a DirectIO error occurred</li> <li>0 = DirectIO succeeded</li> </ul>	Additional data.
149	95	ING_DIO_FILE_STATUS	Allowable values: <ul style="list-style-type: none"> <li>-1 = a DirectIO error occurred</li> <li>0 = DirectIO succeeded</li> </ul>	Additional data.
161	A1	ING_DIO_SETVAR	Allowable values: <ul style="list-style-type: none"> <li>-1 = a DirectIO error occurred</li> <li>0 = DirectIO succeeded</li> </ul>	Additional data.
162	A2	ING_DIO_GET_UIA_VARIABLE	Allowable values: <ul style="list-style-type: none"> <li>-1 = a DirectIO error occurred</li> <li>0 = DirectIO succeeded</li> </ul>	Additional data.
166	A6	ING_DIO_LIGHT_CONTROL	Allowable values: <ul style="list-style-type: none"> <li>-1 = a DirectIO error occurred</li> <li>0 = DirectIO succeeded</li> </ul>	Additional data.
239	EF	ING_DIO_TMS_TRIGGER	Allowable values: <ul style="list-style-type: none"> <li>-1 = a DirectIO error occurred</li> <li>6 = DirectIO success after EFT/TMS download trigger command issued</li> </ul>	Additional data.
512	200	ING_DIO_UPDATE_POLL_STATE	No response.	Additional data.

### 4.3. Error Events

UIA/OPOS/JPOS fire error events to notify the POS anytime an error occurs.

### 4.3.1. Event Object Format

Ingenico's implementation of error event objects follows the UnifiedPOS 1.13 specification exactly.

### 4.3.2. Event Signature – JPOS

Formulation	<code>public ErrorEvent(Object source, int errorCode, int errorCodeExtended, int errorLocus, int errorResponse)</code>
Source	The <b>source</b> object is any of the UPOS controls (MSR, LineDisplay, SignatureCapture, and PINPad).
Error Code	The <b>Error Code</b> value represents the error that caused the event.
Error Code Extended	The <b>Error Code Extended</b> value represents the error that caused the event.
Error Locus	The <b>Error Locus</b> value is the location of the error.
Error Response	The <b>Error Response</b> parameter is the error response or default value.



### 4.3.3. Event Signature – OPOS

<b>Formulation</b>	<code>void ClassObject:OnErrorEvent(long ResultCode, long ResultCodeExtended, long ErrorLocus, long FAR* pErrorResponse)</code>
<b>Result Code</b>	The <b>Result Code</b> value represents the error that caused the event.
<b>Result Code Extended</b>	The <b>Result Code Extended</b> value represents the error that caused the event.
<b>Error Locus</b>	The <b>Error Locus</b> value is the location of the error.
<b>pErrorResponse</b>	The <b>pErrorResponse</b> parameter is the error response or default value.

### 4.3.4. Error Event Codes

Event codes for error events in both JPOS and OPOS are listed in the table below.

**Info** For use with OPOS, replace the prefix 'JPOS\_' with 'OPOS\_'. One exception to this list is that an equivalent to Error Locus '10 = IJPOS\_EL\_IO' does not exist for OPOS.

Source	Error Code	Error Code Extended	Error Locus	Error Response
Line Display	Allowable values: <ul style="list-style-type: none"> <li>100 = JPOS_SUCCESS</li> <li>101 = JPOS_E_CLOSED</li> <li>102 = JPOS_E_CLAIMED</li> <li>103 = JPOS_E_NOTCLAIMED</li> <li>104 = JPOS_E_NOSERVICE</li> <li>105 = JPOS_E_DISABLED</li> <li>106 = JPOS_E_ILLEGAL</li> <li>107 = JPOS_E_NOHARDWARE</li> <li>108 = JPOS_E_OFFLINE</li> <li>109 = JPOS_E_NOEXIST</li> <li>110 = JPOS_E_EXISTS</li> <li>111 = JPOS_E_FAILURE</li> <li>112 = JPOS_E_TIMEOUT</li> <li>113 = JPOS_E_BUSY</li> <li>114 = JPOS_E_EXTENDED</li> <li>115 = JPOS_E_DEPRECATED</li> </ul>	Allowable values: <ul style="list-style-type: none"> <li>201 = JPOS_EDISP_TOOBIG</li> <li>202 = JPOS_EDISP_BADFORMAT</li> </ul>	Allowable values: <ul style="list-style-type: none"> <li>1 = JPOS_EL_OUTPUT</li> <li>2 = JPOS_EL_INPUT</li> <li>3 = JPOS_EL_INPUT_DATA</li> <li>10 = IJPOS_EL_IO</li> </ul>	Allowable values: <ul style="list-style-type: none"> <li>11 = JPOS_ER_RETRY</li> <li>12 = JPOS_ER_CLEAR</li> <li>13 = JPOS_ER_CONTINUEINPUT</li> </ul>

Source	Error Code	Error Code Extended	Error Locus	Error Response
MSR	Allowable values: <ul style="list-style-type: none"> <li>• 100 = JPOS_SUCCESS</li> <li>• 101 = JPOS_E_CLOSED</li> <li>• 102 = JPOS_E_CLAIMED</li> <li>• 103 = JPOS_E_NOTCLAIMED</li> <li>• 104 = JPOS_E_NOSERVICE</li> <li>• 105 = JPOS_E_DISABLED</li> <li>• 106 = JPOS_E_ILLEGAL</li> <li>• 107 = JPOS_E_NOHARDWARE</li> <li>• 108 = JPOS_E_OFFLINE</li> <li>• 109 = JPOS_E_NOEXIST</li> <li>• 110 = JPOS_E_EXISTS</li> <li>• 111 = JPOS_E_FAILURE</li> <li>• 112 = JPOS_E_TIMEOUT</li> <li>• 113 = JPOS_E_BUSY</li> <li>• 114 = JPOS_E_EXTENDED</li> <li>• 115 = JPOS_E_DEPRECATED</li> </ul>	Allowable values: <ul style="list-style-type: none"> <li>• 201 = JPOS_EMSR_START</li> <li>• 202 = JPOS_EMSR_END</li> <li>• 203 = JPOS_EMSR_PARITY</li> <li>• 204 = JPOS_EMSR_LRC</li> <li>• 205 = JPOS_EMSR_DEVICE_AUTHENTICATION_FAILED</li> <li>• 206 = JPOS_EMSR_DEVICE_DEAUTHENTICATION_FAILED</li> </ul>	Allowable values: <ul style="list-style-type: none"> <li>• 1 = JPOS_EL_OUTPUT</li> <li>• 2 = JPOS_EL_INPUT</li> <li>• 3 = JPOS_EL_INPUT_DATA</li> <li>• 10 = JPOS_EL_IO</li> </ul>	Allowable values: <ul style="list-style-type: none"> <li>• 11 = JPOS_ER_RETRY</li> <li>• 12 = JPOS_ER_CLEAR</li> <li>• 13 = JPOS_ER_CONTINUEINPUT</li> </ul>
PINPad	Allowable values: <ul style="list-style-type: none"> <li>• 100 = JPOS_SUCCESS</li> <li>• 101 = JPOS_E_CLOSED</li> <li>• 102 = JPOS_E_CLAIMED</li> <li>• 103 = JPOS_E_NOTCLAIMED</li> <li>• 104 = JPOS_E_NOSERVICE</li> <li>• 105 = JPOS_E_DISABLED</li> <li>• 106 = JPOS_E_ILLEGAL</li> <li>• 107 = JPOS_E_NOHARDWARE</li> <li>• 108 = JPOS_E_OFFLINE</li> <li>• 109 = JPOS_E_NOEXIST</li> <li>• 110 = JPOS_E_EXISTS</li> <li>• 111 = JPOS_E_FAILURE</li> <li>• 112 = JPOS_E_TIMEOUT</li> <li>• 113 = JPOS_E_BUSY</li> <li>• 114 = JPOS_E_EXTENDED</li> <li>• 115 = JPOS_E_DEPRECATED</li> </ul>	Allowable values: <ul style="list-style-type: none"> <li>• 201 = JPOS_EPPAD_BAD_KEY</li> </ul>	Allowable values: <ul style="list-style-type: none"> <li>• 1 = JPOS_EL_OUTPUT</li> <li>• 2 = JPOS_EL_INPUT</li> <li>• 3 = JPOS_EL_INPUT_DATA</li> <li>• 10 = JPOS_EL_IO</li> </ul>	Allowable values: <ul style="list-style-type: none"> <li>• 11 = JPOS_ER_RETRY</li> <li>• 12 = JPOS_ER_CLEAR</li> <li>• 13 = JPOS_ER_CONTINUEINPUT</li> </ul>

Source	Error Code	Error Code Extended	Error Locus	Error Response
All other sources.	Allowable values: <ul style="list-style-type: none"> <li>• 100 = JPOS_SUCCESS</li> <li>• 101 = JPOS_E_CLOSED</li> <li>• 102 = JPOS_E_CLAIMED</li> <li>• 103 = JPOS_E_NOTCLAIMED</li> <li>• 104 = JPOS_E_NOSERVICE</li> <li>• 105 = JPOS_E_DISABLED</li> <li>• 106 = JPOS_E_ILLEGAL</li> <li>• 107 = JPOS_E_NOHARDWARE</li> <li>• 108 = JPOS_E_OFFLINE</li> <li>• 109 = JPOS_E_NOEXIST</li> <li>• 110 = JPOS_E_EXISTS</li> <li>• 111 = JPOS_E_FAILURE</li> <li>• 112 = JPOS_E_TIMEOUT</li> <li>• 113 = JPOS_E_BUSY</li> <li>• 114 = JPOS_E_EXTENDED</li> <li>• 115 = JPOS_E_DEPRECATED</li> </ul>	Allowable values: <ul style="list-style-type: none"> <li>• 280 = JPOS_ESTATS_ERROR</li> <li>• 281 = JPOS_EFIRMWARE_BAD_FILE</li> <li>• 282 = JPOS_ESTATS_DEPENDENCY</li> </ul>	Allowable values: <ul style="list-style-type: none"> <li>• 1 = JPOS_EL_OUTPUT</li> <li>• 2 = JPOS_EL_INPUT</li> <li>• 3 = JPOS_EL_INPUT_DATA</li> <li>• 10 = JPOS_EL_IO</li> </ul>	Allowable values: <ul style="list-style-type: none"> <li>• 11 = JPOS_ER_RETRY</li> <li>• 12 = JPOS_ER_CLEAR</li> <li>• 13 = JPOS_ER_CONTINUEINPUT</li> </ul>

## 4.4. Status Update Events (JPOS only)

JPOS fires status update events in response to scenarios like file downloads and firmware updates.

### 4.4.1. Event Object Format

Ingenico's implementation of status update event objects follows the UnifiedPOS 1.13 specification exactly.

### 4.4.2. Event Signature

Formulation	<code>public StatusUpdateEvent(Object source, int status)</code>
Source	The <b>source</b> object is any of the UPOS controls (MSR, LineDisplay, SignatureCapture, and PINPad).
Status	The <b>status</b> value represents the error that caused the event.

### 4.4.3. Status Update Event Statuses

Event codes for status update events are listed in the table below:

Status	Description
2001	JPOS_SUE_POWER_ONLINE
2002	JPOS_SUE_POWER_OFF
2003	JPOS_SUE_POWER_OFFLINE
2004	JPOS_SUE_POWER_OFF_OFFLINE
2100	JPOS_SUE_UF_PROGRESS
2200	JPOS_SUE_UF_COMPLETE
2201	JPOS_SUE_UF_FAILED_DEV_OK
2202	JPOS_SUE_UF_FAILED_DEV_UNRECOVERABLE
2203	JPOS_SUE_UF_FAILED_DEV_NEEDS_FIRMWARE
2204	JPOS_SUE_UF_FAILED_DEV_UNKNOWN
2205	JPOS_SUE_UF_COMPLETE_DEV_NOT_RESTORED

## 4.5. JPOS Event Handling - Samples

---

### 4.5.1. Data Event Handling

The following code sample in Java shows how the POS handles data events:

```
public void dataOccurred(DataEvent de) {
    s_Logger.info("DataEvent status: " + de.getStatus());
}
```

### 4.5.2. DirectIO Event Handling

The following code sample in Java shows how the POS handles DirectIO events:

```
public void directIOOccurred(DirectIOEvent e) {
    s_Logger.info("Event number: " + e.getEventNumber());
    s_Logger.info("Event object: " + e.getObject().toString());
}
```

### 4.5.3. Error Event Handling

The following code sample in Java shows how the POS handles Error events:

```
public void errorOccurred(ErrorEvent ee) {
    MSR oMSR = (MSR) ee.getSource();
    if (oMSR.getErrorReportingType() == MSRConst.MSR_ERT_CARD)
    {
        s_Logger.info("ERROR REPORTING BY CARD");
        s_Logger.info(String.format("ErrorCode: %d", ee.getErrorCode()));
        s_Logger.info(String.format("ErrorCodeExtended: %d" +
            ee.getErrorCodeExtended()));
    }
}
```

```
s_Logger.info(String.format("ErrorResponse: %d" +
ee.getResponse()));
}
}
```

#### 4.5.4. Status Update Event Handling

The following code sample in Java shows how the POS handles Status Update events:

```
public void statusUpdateOccurred(StatusUpdateEvent e) {
s_Logger.info("Source: "+e.getSource());
s_Logger.info("Status: "+e.getStatus());
}
```

### 4.6. OPOS Event Handling

---

In OPOS, event information is not encapsulated in an object; therefore no special code is needed to retrieve event information.

## 5. Direct I/O for JPOS

Several features of the Telium PIN pads that are not covered by the UnifiedPOS 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.

### 5.1. The DirectIO Method

Syntax	<b>directIO</b> (int <i>command</i> , int[] <i>pData</i> , Object <i>object</i> );																																								
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, listed in the <code>res/dio.properties</code> file:</p> <table><tbody><tr><td>DIO_SEND_RAW_DATA</td><td>= x00</td></tr><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>DIO_LIGHT_CONTROL</td><td>= xA6</td></tr><tr><td>DIO_EFT_TMS_TRIGGER</td><td>= xEF</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_DELETE_RECEIPT_CONTENTS</td><td>= x35</td></tr><tr><td>DIO_TEXT_CURSOR</td><td>= x36</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_DELETE_FILE</td><td>= x93</td></tr><tr><td>DIO_RETRIEVE_FILE</td><td>= x94</td></tr><tr><td>DIO_FILE_STATUS</td><td>= x95</td></tr><tr><td>DIO_ACTIVATE_SOFTWARE</td><td>= x6A</td></tr></tbody></table> <p>To have JPOS 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>JPOS returns a DirectIO event each time a DirectIO command is received.</p>	DIO_SEND_RAW_DATA	= x00	DIO_HEALTH_STATS	= x0D	DIO_SET_VARIABLE	= xA1	DIO_GET_VARIABLE	= xA2	DIO_LIGHT_CONTROL	= xA6	DIO_EFT_TMS_TRIGGER	= xEF	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_DELETE_RECEIPT_CONTENTS	= x35	DIO_TEXT_CURSOR	= x36	DIO_SAVE_FILE	= x91	DIO_RUN_FILE	= x92	DIO_DELETE_FILE	= x93	DIO_RETRIEVE_FILE	= x94	DIO_FILE_STATUS	= x95	DIO_ACTIVATE_SOFTWARE	= x6A
DIO_SEND_RAW_DATA	= x00																																								
DIO_HEALTH_STATS	= x0D																																								
DIO_SET_VARIABLE	= xA1																																								
DIO_GET_VARIABLE	= xA2																																								
DIO_LIGHT_CONTROL	= xA6																																								
DIO_EFT_TMS_TRIGGER	= xEF																																								
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_DELETE_RECEIPT_CONTENTS	= x35																																								
DIO_TEXT_CURSOR	= x36																																								
DIO_SAVE_FILE	= x91																																								
DIO_RUN_FILE	= x92																																								
DIO_DELETE_FILE	= x93																																								
DIO_RETRIEVE_FILE	= x94																																								
DIO_FILE_STATUS	= x95																																								
DIO_ACTIVATE_SOFTWARE	= x6A																																								

Return	JPOS does not return any values in response to DirectIO function calls. JPOS will throw a JposException if unexpected behavior occurs.
--------	---

### 5.1.1. (0x00) Send Raw Data

Parameter	Description
Command	DIO_SEND_RAW_DATA
pData	User-defined data.
pString	Varies.

The host sends this command to transfer raw data to the PIN pad.

### 5.1.2. (0x0D) 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
APP	Internal application name.
APP_BIN_NAME	Application numeric Telium name.
APP_VERS	Application version.
BAD_BLOCK	Accumulated number of nand bad block found (some nand flash blocks may be bads without problem).
BOOSTER_VERS	Version of the Booster (security) processor.
BUILD	Build number.
CLESS	Is terminal capable of reading contactless cards? <ul style="list-style-type: none"> <li>0 = No</li> <li>1 = Yes</li> </ul>
CLESS_ERRORS	Total number of contactless errors.
CLESS_READS	Incremented when contactless card entry was enabled ("0x41" command) and a contactless card was read.
DEVICE	Name of the PIN pad device.

Variable Name	Variable Description
ENTER_KEY	Incremented when [ENTER] button was pressed while key entry task is enabled on PIN pad.
FIN_KEY	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>
FLASH_REFRESH	Accumulated number of nand flash block refresh (1 bit ECC error had been found in NAND flash. Data had been corrected.)
FREE_DFS	Amount of free data file space in bytes.
FREE_RAM	Amount of free program memory in bytes.
GARBAGE_COLLECT	Accumulated number of nand flash garbage collection (physically erasing nand flash).
HOST_CONFIG	Details about the host port configuration.
HOST_PORT	Host communications port type. Possible values are: Ethernet, Serial, USB, Tailgate.
IP_ADDR	IP address of PIN pad. Display of this item can be disabled via configuration file.
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.



Variable Name	Variable Description
MSR_ERRORS <sub>2</sub>	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_SWIPES <sub>3</sub>	Accumulated number of ISO3 magnetic cards swiped.
MSR_ERRORS <sub>3</sub>	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.
SIG	Is PIN pad capable of capturing signatures? <ul style="list-style-type: none"> <li>0 = No</li> <li>1 = Yes</li> </ul>
SIG_CANCEL	Incremented when a signature is cancelled on the PIN pad.
SIG_TOTAL	Incremented when a signature is captured on PIN pad. This includes partial signatures captured before a CANCEL keypress.
STARTUPS	Accumulated number of Main processor (Thunder) Operating System starts.
SUBNET_MASK	IP subnet mask of the PIN pad. Display of this item can be disabled via configuration file.
TELIUM_VERS	Version number of the Telium system.
TERM_SERIAL#	Ingenico serial number, generated at manufacture.
THUNDER_VERS	Version number of the Thunder (main) processor.
TOUCH	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>
UP_TIME	Total accumulated running time for the device, in seconds.


Variable Name	Variable Description
USAGE	Incremented when backlight is on. Total use time. <b>Currently out of scope.</b>

### 5.1.3. (0xA1) 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.

 For PayPal authorization, see the following variable names: PINENCRYPTION, PINMAXKEY, PINMINKEY, RSA\_KEY.

**Table 3: Set UIA Variable Names**

Variable Name	Variable Description
ACCOUNT_NAME	Cardholder's name. User for manual credit card entry.
BACKLIGHT	Set display backlight brightness (0 – 100 percent).
CVV2	Credit verification value. Used for manual credit card entry.
CLESSDELAY	Amount of time (in 10 ms increments) to delay for a MSR swipe after a contactless card is detected. Default is 750 ms.
CLESSMODE	Current contactless mode: <ul style="list-style-type: none"><li>• 1 = PPSE only</li><li>• 2 = PPSE first</li><li>• 3 = Mifare first</li></ul>
CURRENTLANGUAGE	Current prompt language: <ul style="list-style-type: none"><li>• 1 = English</li><li>• 2 = Spanish</li><li>• 3 = French</li></ul>
CURRENTWINDOW	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).
EXP_DATE	Expiration date. Used for manual credit card entry. Format: mmyy.
FORMATSPECIFIER	This variable can be used to define a format specifier dynamically. The format specifier is a string that specifies rules regarding data entry for the Clear Text Entry control in the specified form. See also, Section 7 Format Specifiers, for details on format specifier strings. Note that this variable is only used when FORMATSPECIFIERINDEX is zero (0), and the form, itself, has its format property set to zero (0), as well.
FORMATSPECIFIERINDEX	Specifies the index of the secure format specifier for Clear Text Entry. Must be set to zero (0) for the FORMATSPECIFIER variable to be used.

Variable Name	Variable Description
KEYBOARDLIGHT	Set keyboard backlight brightness (0 – 100 percent).
KEYBEEP	Beep on keypress: <ul style="list-style-type: none"> <li>• 0 = Off</li> <li>• 1 = On</li> </ul>
KEYINDEX	PIN entry encryption key index (slot).
LDPRESETHEIGHT	Sets the vertical distance between label elements in UIA's NCR 5992 line display form.
MSRBADBEEP	Beeps on bad magnetic stripe card swipe: <ul style="list-style-type: none"> <li>• 0 = Off</li> <li>• 1 = On</li> </ul>
MSRGOODBEEP	Beeps on good magnetic stripe card swipe: <ul style="list-style-type: none"> <li>• 0 = Off</li> <li>• 1 = On</li> </ul>
MSR_NO_LIGHTS_ON_ENABLE	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>
MSR_NO_LIGHTS_ON_ERROR	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>
PAN	Primary account number. Used for manual credit card entry.
pIDLEBACKLIGHTLEVEL	Set idle screen backlight brightness (0 – 100 percent).
pIDLEBACKLIGHTTIMEOUT	Amount of idle time (in seconds) before idle screen or script is displayed. Must be greater than 60 seconds.
pIDLESCREEN	Form or script to be displayed on idle.
pIDLESCREENTIMEOUT	Amount of idle time (in seconds) before idle screen or script is displayed. Must be greater than 60 seconds.
PINCARDNUMBER	The account number to be associated with the entered PIN.
PINENCRIPTION	PIN entry encryption type: <ul style="list-style-type: none"> <li>• 0 = Master/Session</li> <li>• 1 = DUKPT</li> <li>• 2 = RSA Encryption (for PayPal)</li> </ul>
PINERRORBEEP	Beeps on PIN entry error: <ul style="list-style-type: none"> <li>• 0 = Off</li> <li>• 1 = On</li> </ul>
PINFIRSTTIMEOUT	Timeout for 1 <sup>st</sup> digit input. Default is 30 seconds. Set to 0 for infinite timeout.
PININTERKEYTIMEOUT	Inter-digit timeout. Default is 30 seconds. Set to 0 for infinite timeout.

Variable Name	Variable Description
<b>PINMAXKEY</b>	Maximum number of digits in PIN. Must be 12 or less. Default is 12. Maximum for PayPal = 8.
<b>PINMINKEY</b>	Minimum number of digits in PIN. Must be 4 or greater. Default is 4. Minimum for PayPal = 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. Only used if the prompt index specified in the form, itself, is zero (0); otherwise, the index in the form is used.
<b>RSA_KEY</b>	Specifies a public key name and is specific to PayPal authorization. This variable is set in order to encrypt the end user's PayPal PIN entered into the PIN Pad. This variable must be set into the PIN Pad before attempting any BeginEFTTransactions(). If the RSA_KEY is not found by the PIN Pad, then the Set UIA Variable function will return an error. See also section 12.3 RSA Encryption for PayPal PIN.
<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\\nTEXT_ID=5,Ha";
BSTR pString2 = ::SysAllocString(csVarName);
nResult = m_Form.DirectIO(nCmd,pData,&pString2);
```

#### 5.1.4. (0xA2) 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.

#### 5.1.5. (0xA6) Configure PIN Pad Lights

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

Parameter	Description
Command	DIO_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
MSR;OFF;;;	Turns off MSR LEDs.
MSR;RED;L_TO_R;20;;	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>
MSR;OFF;;;:MSR;GREEN;L_TO_R;;;	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>
MSR;OFF;;;:MSR;GREEN;L_TO_R;15;; :MSR;RED;L_TO_R;15;;	Turns off MSR LEDs, turns on setting color to green with left-to-right animation for 15 seconds, and 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

#### 5.1.6. (0xEF) Trigger EFT/TMS Downloads

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

The host sends this command to trigger downloads through EFT or TMS.

 The `DIO_EFT_TMS_TRIGGER` command will only trigger downloads to Ethernet-connected PIN pads.

### 5.1.7. (0x09) Reboot the PIN Pad

Parameter	Description
Command	<code>DIO_DEVICE_RESET</code>
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 "

### 5.1.8. (0x12) Transmit Check Box Data

Parameter	Description
Command	<code>DIO_TRANSMIT_CHECKBOX_DATA</code>
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.

### 5.1.9. (0x13) Transmit Survey Data

Parameter	Description
Command	<code>DIO_TRANSMIT_SURVEY_DATA</code>
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.

### 5.1.10. (0x30) Clear Line Display Element

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



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.

### 5.1.11. (0x31) Clear Screen

Parameter	Description
Command	DIO_CLEAR
pData	Ignored
pString	<ul style="list-style-type: none"> <li>• <i>Input:</i> Ignored</li> <li>• <i>Output:</i> 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 68). 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);
```

### 5.1.12. (0x34) Display Text At

Parameter	Description
Command	DIO_DISPLAY_TEXT_AT
pData	<ul style="list-style-type: none"> <li>• 0 = for Row/Column display mode without font selection.</li> <li>• 2 = for Row/Column 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 = 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> </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. Valid values for row and column offsets will vary as per the Telium device's individual specifications; each of these values is represented by a byte. Use this command with the font tables and rules outlined in Appendix C 5992 Mode.



Note that row and column numbers begin with row 0 (zero) and column 0 (zero).

#### Code Samples

if pData = 0 then:

```
csData = strRow + 0x0A + strCol + 0x0A + strDisplayText
```

OR csData = oxRowByteoxoAoxColByteoxoAHexValOfString;  
 e.g Row = 3, Col = 4 & strDisplay = "This is text to display"  
 Hex rep of pString = 33 0A 34 0A 54 68 69 73 20 69 73 20 74 68 65 20 74 65 78 74 20 74 6F  
 20 64 69 73 70 6C 61 79 2E

if pData = 2 then:

csData = strRow + 0x0A + strCol + 0x0A + strFont + 0x0A + strDisplayText

OR csData = oxRowByteoxoAoxColByteoxooxFontByteoxoAAHexValOfString;

*Font value can be selected from the Font table below*

e.g Row = 3, Col = 4, strFont = "12" 1=>monospace(Normal) & 2=>7(size)

& String = "This is text to display"

then Hex rep of pString = 33 0A 34 0A 31 32 0A 54 68 69 73 20 69 73 20 74 68 65 20 74 65 78  
 74

20 74 6F 20 64 69 73 70 6C 61 79 2E

long Cmd = 0x34

long pData = 0 or 2

BStr bstr = csData.AllocString()

UPOSControl.DirectIO(Cmd, &pData, &bstr);

### 5.1.13. (0x35) Delete Receipt Contents

This command deletes lines from the current line display. On successful execution, the desired line(s) will be deleted from the Terminal's line display.

Parameter	Description
Command	DIO_DELETE_RECEIPT_CONTENTS
pData	(0xbStartRow << 24)   (0xbNumRows << 16); //long is 4 bytes, use byte [4] for starting row and byte[3] for number of rows number
pString	NULL; //Not used

#### Example Scenario and JPOS code:

Five lines of text exist on the display, and you wish to delete the third and fourth lines.

BEFORE:

This is the text to display on the	Row 0 (user sees as Line 1)
Telium 2 to test	Row 1 (Line 2)
the OPOS line	Row 2 (Line 3)
display control	Row 3 (Line 4)
	Row 4 (Line 5)

AFTER:

This is the text to display on the	Row 0 (user sees as Line 1)
	Row 1 (Line 2)
	Row 2 (Line 3)
	Row 3 (Line 4)
display control	Row 4 (Line 5)

#### Example JPOS code:

```
Integer nStartRow = 2;
Integer nRowsToDelete = 2;

Long oDeleteRows = 0L;

oDeleteRows = (nStartRow.longValue() << 24) | (nRowsToDelete.longValue()
<< 16);

jpos.LineDisplay ld = new jpos.LineDisplay();
try {
```

```
ld.directIO(53, //DELETE_RECEIPT_CONTENTS=0x35
            new int[] {0},
            oDeleteRows);
} catch (JposException e) {
    e.printStackTrace();
}
```



Note that this instruction accommodates text wrapping.



Row numbering begins at zero (0). The maximum number of lines possible in a byte is 255.

#### 5.1.14. (0x36) Position Text Cursor

Parameter	Description
Command	DIO_TEXT_CURSOR
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.

#### 5.1.15. (0x91) 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.

#### 5.1.16. (0x92) 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	File name (e.g. sigcap.k3z) passed as an array of bytes.

Displays a form on the PIN pad.

### 5.1.17. (0x93) Delete File

Parameter	Description
Command	DIO_DELETE_FILE
pData	Ignored.
pString	File name (e.g. sigcap.k3z) passed as an array of bytes.

The host sends this command to delete a file from the PIN pad.

### 5.1.18. (0x94) Retrieve File

Parameter	Description
Command	DIO_RETRIEVE_FILE
pData	Ignored.
pString	The word “manifest” or “manifestcrc32” 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.

Depending on your company’s needs, if a comparison is made against the files stored locally on the register, a choice can be made as to whether to download all files, or just those that are different, resulting in a significant reduction in download time (in some cases from 45 minutes to 10 minutes or less).

#### 5.1.18.1. Retrieving “manifest”

This manifest returns a list of files in the terminal separated by byte ‘0x1C’.

For ‘n’ number of files in the terminal, the format of the bytes returned will be as follows:

- <file name<sub>1</sub>> 0x1C <time> 0x1C <date> 0x1C <size> 0x1C <file name<sub>2</sub>> 0x1C <time> 0x1C <date> 0x1C <size> ... 0x1C <file name<sub>n</sub>> 0x1C <time> 0x1C <date> 0x1C <size>

Where: (Example: SECURPROMPT.XML14:18:1206/28/1226933)

The ‘manifest’ file includes:

- <file name<sub>i</sub>> = the name of a file #i. Example: SECURPROMPT.XML
- <time> = the time (hh:mm:ss) the file was saved in the terminal. Example: 14:18:12
- <date> = the date (mm/dd/yy) the file was saved in the terminal. Example: 06/28/12
- <size> = the size of the file in bytes. Example: 26933

#### 5.1.18.2. Retrieving “manifestcrc32”

This manifest returns a list of files in the terminal separated by byte ‘0x1C’, with the addition of CRC32 checksum information.

For 'n' number of files in the terminal, the format of the bytes returned will be as follows:

- <file name<sub>1</sub>> 0x1C <time> 0x1C <date> 0x1C <size> 0x1C <crc32> 0x1C <file name<sub>2</sub>> 0x1C <time> 0x1C <date> 0x1C <size> 0x1C <crc32> ... 0x1C <file name<sub>n</sub>> 0x1C <time> 0x1C <date> 0x1C <size> 0x1C <crc32>

Where: (Example: SECURPROMPT.XML14:18:1206/28/12269335E493499)

The 'manifestcrc32' file includes:

- <file name<sub>i</sub>> = the name of a file #i. Example: SECURPROMPT.XML
- <time> = the time (hh:mm:ss) the file was saved in the terminal. Example: 14:18:12
- <date> = the date (mm/dd/yy) the file was saved in the terminal. Example: 06/28/12
- <size> = the size of the file in bytes. Example: 26933
- <crc32> = the crc32 of the file in HEX. Example: 5E493499

**Info** Because an 8 character hex value is required, if the value is 7 characters in length, a leading '0' will be added (i.e., a CRC32 value of '1234567' will be sent as '01234567'), therefore a string comparison is not required. If it is determined that a comparison is needed, the Hex value will need to be converted to a long prior to conducting the comparison.

### 5.1.19. (0x95) 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.

### 5.1.20. (0x6A ) Activate Software

The Activate Software command is achieved by sending the DIO\_SAVE\_FILE command with pData = 1. See (0x91) Save File on page 82 for more information.

## 5.2. 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.

## 5.3. 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);
```

### 5.3.1. Set a Variable

The following code sample in Java shows how to set a variable:

```
int nCmd = IngenicoConst.ING_DIO_SETVAR; //  
ING_DIO_SETVAR=161(0xA1)  
int[] nData = new int[] {0};  
String sVarName = "BUTTON_ID=01,255\nTEXT_ID=5,Ha";  
JPOSControl.directIO(nCmd, nData, sVarName);
```

### 5.3.2. Clear Screen

The following code sample in Java shows how to clear the screen:

```
int nCmd = IngenicoConst.ING_DIO_CLRSCRN; // ING_DIO_CLRSCRN  
=49(0x31)  
int[] nData = new int[] {0};  
JPOSControl.directIO(nCmd, nData, null);
```

### 5.3.3. Store a Form

The following code sample in Java shows how to store a form:

```
m_BaseJposControl.open("Telium2");  
m_BaseJposControl.claim(5000);  
m_BaseJposControl.setDeviceEnabled(true);  
  
String sFormName = "INGENICO.PGZ";  
m_BaseJposControl.directIO(IngenicoConst.ING_DIO_SAVE_FILE, new  
int[] {0}, sFormName);
```

### 5.3.4. Display a Form

The following code sample in Java shows how to display a form:

```
m_BaseJposControl.open("Telium2");  
m_BaseJposControl.claim(5000);  
m_BaseJposControl.setDeviceEnabled(true);  
  
String sFormName = "INGENICO.K3Z";  
m_BaseJposControl.directIO(IngenicoConst.ING_DIO_RUN_FILE, new  
int[] {0}, sFormName);
```

## 6. Direct I/O for OPOS

Several features of the Telium PIN pads that are not covered by the UnifiedPOS 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.

### 6.1. 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_SEND_RAW_DATA</td><td>= x00</td></tr><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>DIO_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_DELETE_RECEIPT_CONTENTS</td><td>= x35</td></tr><tr><td>DIO_TEXT_CURSOR</td><td>= x36</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_DELETE_FILE</td><td>= x93</td></tr><tr><td>DIO_RETRIEVE_FILE</td><td>= x94</td></tr><tr><td>DIO_FILE_STATUS</td><td>= x95</td></tr><tr><td>DIO_ACTIVATE_SOFTWARE</td><td>= x6A</td></tr></table> <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 pString parameter of this</p>	DIO_SEND_RAW_DATA	= x00	DIO_HEALTH_STATS	= x0D	DIO_SET_VARIABLE	= xA1	DIO_GET_VARIABLE	= xA2	DIO_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_DELETE_RECEIPT_CONTENTS	= x35	DIO_TEXT_CURSOR	= x36	DIO_SAVE_FILE	= x91	DIO_RUN_FILE	= x92	DIO_DELETE_FILE	= x93	DIO_RETRIEVE_FILE	= x94	DIO_FILE_STATUS	= x95	DIO_ACTIVATE_SOFTWARE	= x6A
DIO_SEND_RAW_DATA	= x00																																						
DIO_HEALTH_STATS	= x0D																																						
DIO_SET_VARIABLE	= xA1																																						
DIO_GET_VARIABLE	= xA2																																						
DIO_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_DELETE_RECEIPT_CONTENTS	= x35																																						
DIO_TEXT_CURSOR	= x36																																						
DIO_SAVE_FILE	= x91																																						
DIO_RUN_FILE	= x92																																						
DIO_DELETE_FILE	= x93																																						
DIO_RETRIEVE_FILE	= x94																																						
DIO_FILE_STATUS	= x95																																						
DIO_ACTIVATE_SOFTWARE	= x6A																																						



	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.			
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	Corrective Action
	00	SUCCESS	The method was successfully executed.	N/A
	113	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.	It is unlikely that this error message will be encountered, however, if it is, continue to wait for the previous request to finish its processing. The next request will follow.
	101	E_CLOSED	An attempt was made to access a closed device.	Open and claim the device.
	102	E_CLAIMED	An attempt was made to access a device that is claimed by another control.	The process claiming the device must release it before another process can use it.
	115	E_DEPRECATED	The requested operation cannot be performed since it has been deprecated.	Recommend not using this operation and/or to find and use an operation that may have replaced the deprecated operation.
	105	E_DISABLED	Cannot perform the requested operation while the device is disabled.	The device must always be enabled to access it.
	110	E_EXISTS	The file name (or other specified value) already exists. This may be caused by a spelling error, or when the system finds that a file it wants to load already exists.	Check spelling. If caused by a file that already exists, a decision must be made on a case by case basis re: loading the newest file.
	114	E_EXTENDED	A device category-specific error condition occurred. The error condition code is held in an extended error code.	Refer to section 4.3.4 Error Event Codes for additional information.
	111	E_FAILURE	The device cannot perform the requested procedure, even though the physical device is connected to the system, powered on, and on-line. OPOS will also send this error message when it does not receive a UIA response for a DIO call.	Check to ensure the set variable's value is a valid one. Can also be caused by an attempt to set a character to an invalid location; check locations defined.
	106	E_ILLEGAL	An attempt was made to perform an illegal or unsupported operation (such as download of an unsigned file, or attempt to send no data from a radio button or checkbox) with the device, or an invalid parameter value was used.	Obtain signed files.

109	E_NOEXIST	The file name (or other specified value) does not exist. File not found in the PIN Pad.	Ensure file names are entered correctly, and/or that the file exists.
107	E_NOHARDWARE	The physical device is not connected to the system or is not powered on.	Check physical cord connections, power outlets, communication (COM) settings. Turn the device on.
104	E_NOSERVICE	The control cannot communicate with the service due to a configuration error.	Most likely a setup or configuration error. This could also be caused by using different COs than those supplied by Ingenico.
103	E_NOTCLAIMED	An attempt was made to access an exclusive-use device that must be claimed before the method can be used.	The process must claim a device before using it.
108	E_OFFLINE	The physical device is off-line.	Check physical cord connections, power outlets, communication (COM) settings. Turn the device on.
112	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.	Increase the time out value.
F4	BAD_PARAMETER	General failure code for DirectIO commands. Causes include an incorrectly typed or missing filename, or attempt to download an unsupported file type.	Ensure file names are entered correctly and the file type/extension is supported.
F7	ERROR/SECURITY FAILURE	Error. An unknown security error. A possible cause is that a financial key (DUKPT or Session Key) does not exist in the slot and an attempt is made to use an index that does not contain a key. Can also appear when a file has an invalid signature.	Use a valid index or inject keys. Obtain files with valid signature.
Default	UNKNOWN RESULT CODE	No return code received.	Contact Ingenico for instructions on how to correct this error.

### 6.1.1. (0x00) Send Raw Data

Parameter	Description
Command	DIO_SEND_RAW_DATA
pData	User-defined data.
pString	Varies.

The host sends this command to transfer raw data to the PIN pad.

### 6.1.2. (0x0D) 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 4: Health Statistics Variable Names**

Variable Name	Variable Description
APP	Internal application name.
APP_BIN_NAME	Application numeric Telium name.
APP_VERS	Application version.
BAD_BLOCK	Accumulated number of nand bad block found (some nand flash blocks may be bads without problem).
BOOSTER_VERS	Version of the Booster (security) processor.
BUILD	Build number.
CLESS	Is terminal capable of reading contactless cards? <ul style="list-style-type: none"><li>• 0 = No</li><li>• 1 = Yes</li></ul>
CLESS_ERRORS	Total number of contactless errors.
CLESS_READS	Incremented when contactless card entry was enabled ("0x41" command) and a contactless card was read.
DEVICE	Name of the PIN pad device.
ENTER_KEY	Incremented when [ENTER] button was pressed while key entry task is enabled on PIN pad.

Variable Name	Variable Description
FIN_KEY	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>
FLASH_REFRESH	Accumulated number of nand flash block refresh (1 bit ECC error had been found in NAND flash. Data had been corrected.)
FREE_DFS	Amount of free data file space in bytes.
FREE_RAM	Amount of free program memory in bytes.
GARBAGE_COLLECT	Accumulated number of nand flash garbage collection (physically erasing nand flash).
HOST_CONFIG	Details about the host port configuration.
HOST_PORT	Host communications port type. Possible values are: Ethernet, Serial, USB, Tailgate.
IP_ADDR	IP address of PIN pad. Display of this item can be disabled via configuration file.
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.


Variable Name	Variable Description
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.
SIG	Is PIN pad capable of capturing signatures? <ul style="list-style-type: none"> <li>0 = No</li> <li>1 = Yes</li> </ul>
SIG_CANCEL	Incremented when a signature is cancelled on the PIN pad.
SIG_TOTAL	Incremented when a signature is captured on PIN pad. This includes partial signatures captured before a CANCEL keypress.
STARTUPS	Accumulated number of Main processor (Thunder) Operating System starts.
SUBNET_MASK	IP subnet mask of the PIN pad. Display of this item can be disabled via configuration file.
TELIUM_VERS	Version number of the Telium system.
TERM_SERIAL#	Ingenico serial number, generated at manufacture.
THUNDER_VERS	Version number of the Thunder (main) processor.
TOUCH	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>
UP_TIME	Total accumulated running time for the device, in seconds.
USAGE	Incremented when backlight is on. Total use time. <b>Currently out of scope.</b>

### 6.1.3. (0xA1) 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.

 For PayPal authorization, see the following variable names: PINENCRYPTION, PINMAXKEY, PINMINKEY, RSA\_KEY.

**Table 5: Set UIA Variable Names**

Variable Name	Variable Description
ACCOUNT_NAME	Cardholder's name. User for manual credit card entry.
BACKLIGHT	Set display backlight brightness (0 – 100 percent).
CVV2	Credit verification value. Used for manual credit card entry.
CLESSDELAY	Amount of time (in 10 ms increments) to delay for a MSR swipe after a contactless card is detected. Default is 750 ms.
CLESSMODE	Current contactless mode: <ul style="list-style-type: none"><li>• 1 = PPSE only</li><li>• 2 = PPSE first</li><li>• 3 = Mifare first</li></ul>
CURRENTLANGUAGE	Current prompt language: <ul style="list-style-type: none"><li>• 1 = English</li><li>• 2 = Spanish</li><li>• 3 = French</li></ul>
CURRENTWINDOW	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).
EXP_DATE	Expiration date. Used for manual credit card entry. Format: mmyy.
FORMATSPECIFIER	This variable can be used to define a format specifier dynamically. The format specifier is a string that specifies rules regarding data entry for the Clear Text Entry control in the specified form. See also, Section 7 Format Specifiers, for details on format specifier strings. Note that this variable is only used when FORMATSPECIFIERINDEX is zero (0), and the form, itself, has its format property set to zero (0), as well.
FORMATSPECIFIERINDEX	Specifies the index of the secure format specifier for Clear Text Entry. Must be set to zero (0) for the FORMATSPECIFIER variable to be used.

Variable Name	Variable Description
KEYBOARDLIGHT	Set keyboard backlight brightness (0 – 100 percent).
KEYBEEP	Beep on keypress: <ul style="list-style-type: none"> <li>0 = Off</li> <li>1 = On</li> </ul>
KEYINDEX	PIN entry encryption key index (slot).
LDPRESETHEIGHT	Sets the vertical distance between label elements in UIA's NCR 5992 line display form.
MSRBADBEEP	Beeps on bad magnetic stripe card swipe: <ul style="list-style-type: none"> <li>0 = Off</li> <li>1 = On</li> </ul>
MSRGOODBEEP	Beeps on good magnetic stripe card swipe: <ul style="list-style-type: none"> <li>0 = Off</li> <li>1 = On</li> </ul>
MSR_NO_LIGHTS_ON_ENABLE	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>
MSR_NO_LIGHTS_ON_ERROR	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>
PAN	Primary account number. Used for manual credit card entry.
pIDLEBACKLIGHTLEVEL	Set idle screen backlight brightness (0 – 100 percent).
pIDLEBACKLIGHTTIMEOUT	Amount of idle time (in seconds) before idle screen or script is displayed. Must be greater than 60 seconds.
pIDLESCREEN	Form or script to be displayed on idle.
pIDLESCREENTIMEOUT	Amount of idle time (in seconds) before idle screen or script is displayed. Must be greater than 60 seconds.
PINCARDNUMBER	The account number to be associated with the entered PIN.
PINENCRIPTION	PIN entry encryption type: <ul style="list-style-type: none"> <li>0 = Master/Session</li> <li>1 = DUKPT</li> <li>2 = RSA Encryption (for PayPal)</li> </ul>
PINERRORBEEP	Beeps on PIN entry error: <ul style="list-style-type: none"> <li>0 = Off</li> <li>1 = On</li> </ul>
PINFIRSTTIMEOUT	Timeout for 1 <sup>st</sup> digit input. Default is 30 seconds. Set to 0 for infinite timeout.
PININTERKEYTIMEOUT	Inter-digit timeout. Default is 30 seconds. Set to 0 for infinite timeout.

Variable Name	Variable Description
<b>PINMAXKEY</b>	Maximum number of digits in PIN. Must be 12 or less. Default is 12. Maximum for PayPal = 8.
<b>PINMINKEY</b>	Minimum number of digits in PIN. Must be 4 or greater. Default is 4. Minimum for PayPal = 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. Only used if the prompt index specified in the form, itself, is zero (0); otherwise, the index in the form is used.
<b>RSA_KEY</b>	Specifies a public key name and is specific to PayPal authorization. This variable is set in order to encrypt the end user's PayPal PIN entered into the PIN Pad. This variable must be set into the PIN Pad before attempting any BeginEFTTransactions(). If the RSA_KEY is not found by the PIN Pad, then the Set UIA Variable function will return an error. See also section 12.3 RSA Encryption for PayPal PIN.
<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\\nTEXT_ID=5,Ha";
BSTR pString2 = ::SysAllocString(csVarName);
nResult = m_Form.DirectIO(nCmd,pData,&pString2);
```

#### 6.1.4. (0xA2) 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.

#### 6.1.5. (0xA6) Configure PIN Pad Lights

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



Parameter	Description
Command	DIO_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:

- 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.

6. 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.
7. For best results, always specify a color when turning on MSR LEDs, i.e., use GREEN or RED rather than ON.
8. 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
MSR;OFF;;;	Turns off MSR LEDs.
MSR;RED;L_TO_R;20;;	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>
MSR;OFF;;;:MSR;GREEN;L_TO_R;;;:	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>
MSR;OFF;;;:MSR;GREEN;L_TO_R;15;;:MSR;RED;L_TO_R;15;;	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

#### 6.1.6. (0x09) Reboot the PIN Pad

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 "

### 6.1.7. (0x12) 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.

If you are running the polling option and want to receive data back on a form that contains a checkbox, leave the Data Event Enable checkbox unchecked (this checkbox is located in the Telium2 Configuration Tool's Form Configuration panel). The reason for this is that checkbox or survey data is held by the PIN Pad until the next anticipated command (DIO\_GET\_CHECKBOX in this case) to send the information back to the POS via the OPOS Driver. If the next command is RunFile (where the Form Configuration's DataEvent checkbox is enabled) then the checkbox data is no longer available. Applies to iSCxxx Telium2 devices only.



See also section 3.4 OPOS Configuration.

### 6.1.8. (0x13) (Transmit Survey (Radio Button) 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.

As with the checkboxes, if you are running the polling option and want to receive radio button data back, leave the Data Event Enable checkbox unchecked (this checkbox is located in the Telium2 Configuration Tool's Form Configuration panel). The reason for this is that radio button data is held by the PIN Pad until the next anticipated command (DIO\_GET\_RADIO in this case) to send the information back to the POS via the OPOS Driver. If the next command is RunFile (where the Form Configuration's DataEvent checkbox is enabled) then the PIN Pad will serve the RunFile command and radio button data will be lost. Applies to iSCxxx Telium2 devices only.



See also section 3.4 OPOS Configuration.

### 6.1.9. (0x30) 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.

### 6.1.10. (0x31) 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 68). 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);
```

### 6.1.11. (0x34) Display Text At

Parameter	Description
Command	DIO_DISPLAY_TEXT_AT
pData	<ul style="list-style-type: none"> <li>0 = for Row/Column display mode without font selection.</li> <li>2 = for Row/Column 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 = 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> </ul> <p><i>Info</i> This command may only be used with the font tables and rules outlined in Appendix C 5992 Mode.</p>

This DirectIO command is used to display text on the PIN pad screen in a specific font at a specified row and column offset. Valid values for row and column offsets will vary as per the Telium device's individual specifications; each of these values is represented by a byte. Use this command with the font tables and rules outlined in Appendix C 5992 Mode.

*Info* Note that row and column numbers begin with row 0 (zero) and column 0 (zero).

#### Code Samples

**if pData = 0 then:**

```
csData = strRow + 0x0A + strCol + 0x0A + strDisplayText
```

OR csData = 0xRowByte0xA0xColByte0xAHexValOfString;

e.g Row = 3, Col = 4 & strDisplay = "This is text to display"

Hex rep of pString = 33 0A 34 0A 54 68 69 73 20 69 73 20 74 68 65 20 74 65 78 74 20 74 6F

20 64 69 73 70 6C 61 79 2E

**if pData = 2 then:**

csData = strRow + 0x0A + strCol + 0x0A + strFont + 0x0A + strDisplayText

OR csData = oxRowByteoxoAoxColByteoxooxFontByteoxoAAHexValOfString;

Font value can be selected from the Font table below

e.g Row = 3, Col = 4, strFont = "12" 1=>monospace(Normal) & 2=>7(size)

& String = "This is text to display"

then Hex rep of pString = 33 0A 34 0A 31 32 0A 54 68 69 73 20 69 73 20 74 68 65 20 74 65 78  
74

20 74 6F 20 64 69 73 70 6C 61 79 2E

long Cmd = 0x34

long pData = 0 or 2

BStr bstr = csData.AllocString()

UPOSControl.DirectIO(Cmd, &pData, &bstr);

### 6.1.12. (0x35) Delete Receipt Contents

This command deletes lines from the current line display. On successful execution, the desired line(s) will be deleted from the Terminal's line display.

Parameter	Description
Command	DIO_DELETE_RECEIPT_CONTENTS
pData	(oxbStartRow << 24)   (oxbNumRows << 16); //long is 4 bytes, use byte [4] for starting row and byte[3] for number of rows number
pString	NULL

### Example Scenario and OPOS code:

Five lines of text exist on the display, and you wish to delete the third and fourth lines.

BEFORE:

This is the text	Row 0 (user sees as Line 1)
to display on the	Row 1 (Line 2)
Telium 2 to test	Row 2 (Line 3)
the OPOS line	Row 3 (Line 4)
display control	Row 4 (Line 5)

AFTER:

This is the text	Row 0 (user sees as Line 1)
to display on the	Row 1 (Line 2)
	Row 2 (Line 3)
	Row 3 (Line 4)
display control	Row 4 (Line 5)

### Example OPOS code:

```
BYTE bStartRow = (BYTE) nStartRowValue; //Desired start Row to be deleted - as per  
the above scenario, this value will be '2'.
```


```
BYTE bNumRows = (BYTE) nNumofRowsValue; //Number of Rows to be deleted, including  
the StartRow - as per the above scenario, this value will also be '2'.
```


```
long Cmd = 0x35 //DIO_DELETE_RECIEPT_CONTENTS Command
```

```
long pData = (0xbStartRow << 24) | (0xbNumRows << 16); //long is 4 bytes, use  
byte[4] for starting row and byte[3] for number of rows number.
```

```
BSTR pString = NULL //Not used
```

```
OPOSDevice.DirectIO(Cmd, &nData, &pString);
```

 Note that this instruction accommodates text wrapping.

 Row numbering begins at zero (0). The maximum number of lines possible in a byte is 255.

### 6.1.13. (0x36) Position Text Cursor

Parameter	Description
Command	DIO_TEXT_CURSOR
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.

### 6.1.14. (0x91) 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.

### 6.1.15. (0x92) 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	File name (e.g. sigcap.k3z) passed as an array of bytes.

Displays a form on the PIN pad.

### 6.1.16. (0x93) Delete File

Parameter	Description
Command	DIO_DELETE_FILE
pData	Ignored.
pString	File name (e.g. sigcap.k3z) passed as an array of bytes.



The host sends this command to delete a file from the PIN pad.

### 6.1.17. (0x94) Retrieve File

Parameter	Description
Command	DIO_RETRIEVE_FILE
pData	Ignored.
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.

### 6.1.18. (0x95) 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.

### 6.1.19. (0x6A) Activate Software

The Activate Software command is achieved by sending the DIO\_SAVE\_FILE command with pData = 1. See (0x91) Save File on page 82 for more information.

## 6.2. 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.

## 6.3. 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);
```

### 6.3.1. Store a Form

The following code sample in C++ 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);

```

### 6.3.2. Display a Form

The following code sample in C++ 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);

```

## Notes

## 7. Format Specifiers

---

This section explains the display attributes and provides examples on how to use the attributes in an FS string using the Clear Text Entry control.

Ingenico PIN pad format specifiers and their corresponding ID numbers are defined in the SECURPROMPT.XML files.

A format specifier allows you to customize the display of key data entered by the user during the clear text entry process. By default, the format specifier is an empty string, and numbers are displayed on the screen with no additional formatting.

FS strings contain display attributes that tell the PIN pad how to display the data on the screen. Display attributes are separated within the FS string by the percent sign (%).

- The percent character (‘%’) \*MAY\* be displayed as a fixed, hidden, or overwrite character by repeating it twice (e.g., “%o100%f%%”).

There are two kinds of display attributes: general and specific.

- General attributes apply to the entire data entry process.
- Specific attributes apply to one or more display positions used by the data entry process.

### 7.1. General Attributes

---

General attributes follow the format:

```
%  
General attribute  
Data
```

The general attributes are:

**m:** The ‘%m’ attribute specifies the minimum number of digits to be entered by the user. The value following this attribute is interpreted as the minimum number of digits.

If the ‘%m’ attribute is not defined in the format specifier, itself, two alternatives exist:

1. a global maximum may be set using the 21.x message (Offset 5) – or –
2. the default value for the ‘%m’ attribute will be used, which is zero (0).

The range for this attribute is 0 – MAX. If ENTER is pressed before typing the minimum number of digits, an invalid beep indicates that the entered input has not been accepted. If the number specified cannot be accommodated, the ‘%m’ attribute is adjusted internally.

 *MAX is the maximum number of display positions available within the data entry field.*

**M:** The ‘%M’ attribute specifies the maximum number of digits to be entered by the user. The value following this attribute is interpreted as the maximum number of digits.

If the ‘%M’ attribute is not defined in the format specifier, itself, two alternatives exist:

1. a global maximum may be set using the 21.x message (Offset 7) – or –

2. the default value for the '`%M`' attribute will be used, which will be ONLY ONE OF THE FOLLOWING:
  - MAX – or –
  - the maximum of the total number of specified overwrite characters ('`%o`') in the format specifier string – or –
  - a maximum of 20 (for backwards compatibility with U32 format specifiers).

The range for this attribute is 1 – MAX. If digits are pressed after the maximum number has been reached, an invalid beep indicates that those digits will not be displayed on the screen, nor will the digits be recorded. If the number specified cannot be accommodated, the '`%M`' attribute is adjusted internally.

**p:** The '`%p`' attribute password-protects and changes the appearance of all characters entered by the user. The ASCII character following this attribute is interpreted as the password character and is displayed on the screen in place of the characters entered by the user. For example, when the user enters a PIN, the '`%p`' attribute can specify that asterisks appear on the screen instead of numbers.

**P:** The '`%P`' attribute password-protects and changes the appearance of all characters entered by the user on a delayed basis. The ASCII character following this attribute is interpreted as the password character and is displayed on the screen in place of the characters entered by the user *as the next character is entered or after one second of time passes*. In other words, you will see the last character entered for up to one second.

As an example, when entering a PIN of '1 2 3 4', if the '`%P`' attribute has been specified that asterisks appear on the screen instead of numbers, an asterisk will replace the number 1 as the number 2 is entered, an asterisk will replace the number 2 as the number 3 is entered, etc. As an additional precaution, if a number 1 is entered and one second of time passes without another number also being entered, the number just entered will be replaced by an asterisk.

**z:** The z attribute forces UIA to recognize leading zeros, which are otherwise ignored by default.

## 7.2. Specific Attributes

---

Specific attributes follow the format:

```
%
Specific attribute
Display string
```

The display string appears on the screen. The specific attributes are:

**f:** Fixed characters. The '`%f`' attribute defines the corresponding positions to be displayed at all times. The '`%f`' attribute cannot be modified during the data entry process.


**h:** Hidden characters. The '`%h`' attribute causes the specific display positions to show only when each hidden character from the right is passed by the shifting text (text being entered by user). From that moment on, these positions are fixed and cannot be modified for the rest of the data entry process.

**o:** Overwriting characters. The '`%o`' attribute defines the corresponding positions to be displayed at the beginning of the data entry process, but allows shifting text to overwrite them.

If the total number of specified overwrite characters ( ' %o ' ) in the format specifier string is less than the maximum number of digits that the user can enter, then an additional number of overwrite characters (equal to the difference in the total number of specified overwrite characters and the maximum number of digits) are added as blank spaces into the format specifier.

Any additional white space overwrite characters are appended after all other overwrite/fixed characters.

**s:** Shifting characters. The ' %s ' attribute defines the corresponding positions to be displayed at the beginning of the data entry process, and then shifted one position at a time for each digit entered (or cleared) when the first one from the right is passed by shifting text.

 *Shift characters are (currently) only implemented for default direction ' %dr ' and are ignored (i.e., \*NOT\* included in the display) for ' %dl '.*

**dl:** Direction left. The ' %dl ' attribute defines the direction in which characters are added to a text field. When ' %dl ' is set, characters are added to the left side of the screen, and the text fills the screen from left to right.

**dr:** Direction right. The ' %dr ' attribute defines the direction in which characters are added to a text field. When ' %dr ' is set, characters are added to the right side of the screen, and the text fills the screen from right to left.

' %dr ' is set by default if neither ' %dl ' or ' %dr ' is specified in the format specifier string.

### 7.3. Using Multiple Format Specifier Attributes

If more than one of each of the following format specifier attributes are present in the format specifier string, then the last (e.g., first from the right end of the format specifier) of each specific attribute is used and all other identical attributes are ignored:

- `'%m'` minimum number of digits to enter
- `'%M'` maximum number of digits to enter
- `'%d'` direction to enter digits

If more than one of each of the following format specifier attributes is present in the format specifier string, then the first (e.g., from the left end of the format specifier) of each specific attribute is used and all other identical attributes are ignored:

- `'%s'` shift characters

### 7.4. Unknown Format Specifiers



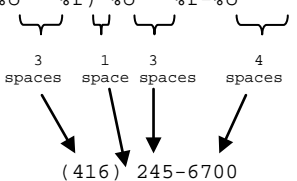
Unknown format specifier characters are now ignored. In the following examples, the format specifier characters enclosed in top-and-bottom asterisks are ignored:

Format Specifier Key	Description
***** " 123% %o %f.%o " *****	Initial unknown characters before first known format specifier attribute are ignored
***** " 123%% %o %f.%o " *****	Initial unknown characters before first known format specifier attribute are ignored including escaped delimiter character
**** "%o %q% %f " ****	All characters following an unknown format specifier attribute but before next known format specifier attribute are ignored
**** "%o %q%%%f " ****	All characters following an unknown format specifier attribute but before next known format specifier attribute are ignored including escaped delimiter character

## 7.5. Examples of Format Specifiers

Format Specifier	Key Pressed	Display	Explanation
<pre> "%o"   {   4 spaces   ↓ 1234 </pre> <p>** OR **</p> <pre> "%dr%o"   {   4 spaces   ↓ 1234 </pre>	<pre> &lt;none&gt; 'Clear' '0' '1' '2' '3' '4' '5' 'Clear' '5' 'Clear' 'Clear' 'Clear' 'Clear' 'Clear' </pre>	<pre> "  " "  " "  " " 1" " 12" " 123" "1234" "1234" " 123" "1235" " 123" " 12" "  1" "   " "   " </pre>	<p>long/bad beep because no digits entered to clear</p> <p>long/bad beep because leading zeros not specified (i.e., no 'z')</p> <p>long/bad beep because maximum digits entered</p> <p>long/bad beep because no digits remaining to clear</p>
<pre> "%dl%o"   {   4 spaces </pre>	<pre> &lt;none&gt; 'Clear' '1' '2' '3' '4' '5' </pre>	<pre> "  " "  " "1  " "12 " "123 " "1234" "1234" </pre>	<p>long/bad beep because no digits entered to clear</p> <p>long/bad beep because maximum digits entered</p>
<pre> "%z%o"   {   4 spaces </pre> <p>** OR **</p> <pre> "%dr%z%o"   {   4 spaces </pre>	<pre> &lt;none&gt; 'Clear' '0' '1' '2' '3' '4' </pre>	<pre> "  " "  " " 0" " 01" " 012" "0123" "0123" </pre>	<p>long/bad beep because no digits entered to clear</p> <p>long/bad beep because maximum digits entered</p>



Format Specifier	Key Pressed	Display	Explanation
<pre>"%m2%M4%o"</pre> <p style="text-align: center;">  </p> <p>** OR **</p> <pre>"%m2%M4%dr%o"</pre> <p style="text-align: center;">  </p>	<p>&lt;none&gt;</p> <p>'Clear'</p> <p>'Enter'</p> <p>'1'</p> <p>'Enter'</p> <p>'2'</p> <p>'3'</p> <p>'4'</p> <p>'5'</p> <p>'Clear'</p> <p>'Enter'</p>	<p>" "</p> <p>" "</p> <p>" "</p> <p>" 1"</p> <p>" 1"</p> <p>" 12"</p> <p>" 123"</p> <p>"1234"</p> <p>"1234"</p> <p>" 123"</p> <p>" 123"</p>	<p>long/bad beep because no digits entered to clear</p> <p>silently ignored because minimum number of digits NOT entered</p> <p>silently ignored because minimum number of digits NOT entered</p> <p>long/bad beep because maximum number digits entered</p> <p>silently accepted because minimum number digits entered</p>
<pre>"%m10%M10%dl%f(%o %f) %o %f-%o"</pre> <p style="text-align: center;">  </p>	<p>&lt;none&gt;</p> <p>'Clear'</p> <p>'4'</p> <p>'1'</p> <p>'6'</p> <p>'2'</p> <p>'4'</p> <p>'5'</p> <p>'Enter'</p> <p>'6'</p> <p>'7'</p> <p>'0'</p> <p>'0'</p> <p>'1'</p>	<p>" ( ) - "</p> <p>" ( ) - "</p> <p>" ( 4 ) - "</p> <p>" ( 41 ) - "</p> <p>" ( 416 ) - "</p> <p>" ( 416 ) 2 - "</p> <p>" ( 416 ) 24 - "</p> <p>" ( 416 ) 245 - "</p> <p>" ( 416 ) 245 - "</p> <p>" ( 416 ) 245-6 "</p> <p>" ( 416 ) 245-67 "</p> <p>" ( 416 ) 245-670 "</p> <p>" ( 416 ) 245-6700 "</p> <p>" ( 416 ) 245-6700 "</p>	<p>long/bad beep because no digits entered to clear telephone prompt</p> <p>silently ignored because minimum number digits NOT entered</p> <p>long/bad beep because maximum number digits entered</p>

Format Specifier	Key Pressed	Display	Explanation
<pre>"%m9%M9%dl%o    %f-%o    %f-%o    "</pre> <p>3 spaces    2 spaces    4 spaces</p> <p>123-45-6789</p>	<p>&lt;none&gt;</p> <p>'Clear'</p> <p>'1'</p> <p>'2'</p> <p>'3'</p> <p>'4'</p> <p>'5'</p> <p>'Enter'</p> <p>'6'</p> <p>'7'</p> <p>'8'</p> <p>'9'</p> <p>'0'</p>	<p>" - - "</p> <p>" - - "</p> <p>" - - "</p> <p>"12 - - "</p> <p>"123- - "</p> <p>"123-4 - "</p> <p>"123-45- "</p> <p>"123-45- "</p> <p>"123-45-6 "</p> <p>"123-45-67 "</p> <p>"123-45-678 "</p> <p>"123-45-6789 "</p> <p>"123-45-6789 "</p>	<p>long/bad beep because no digits entered to clear SSN prompt</p> <p>silently ignored because minimum number digits NOT entered</p> <p>long/bad beep because maximum number digits entered</p>
<pre>"%m8%M8%dl%z%omm%f/%odd%f/%oyyyy"</pre> <p>mm/dd/yyyy</p> <p>06/15/2012</p>	<p>&lt;none&gt;</p> <p>'Clear'</p> <p>'0'</p> <p>'6'</p> <p>'1'</p> <p>'5'</p> <p>'2'</p> <p>'0'</p> <p>'Enter'</p> <p>'1'</p> <p>'2'</p> <p>'0'</p>	<p>"mm/dd/yyyy"</p> <p>"mm/dd/yyyy"</p> <p>"0m/dd/yyyy"</p> <p>"06/dd/yyyy"</p> <p>"06/1d/yyyy"</p> <p>"06/15/yyyy"</p> <p>"06/15/2yyy"</p> <p>"06/15/20yy"</p> <p>"06/15/20yy"</p> <p>"06/15/201y"</p> <p>"06/15/2012"</p> <p>"06/15/2012"</p>	<p>long/bad beep because no digits entered to clear</p> <p>silently ignored because minimum number digits NOT entered</p> <p>long/bad beep because maximum number digits entered</p>
<pre>"%m2%M6%h,%o    0%f.%o00"</pre> <p>1 space (implied)    comma    2 spaces    1<sup>st</sup> zero    last 2 zeros</p> <p>0.00</p> <p>1 digit comma 3 digits dot 2 digits (comma is a hidden character)</p>	<p>&lt;none&gt;</p> <p>'1'</p> <p>'2'</p> <p>'3'</p> <p>'4'</p> <p>'5'</p> <p>'6'</p> <p>'7'</p>	<p>" 0.00 "</p> <p>" 0.01 "</p> <p>" 0.12 "</p> <p>" 1.23 "</p> <p>" 12.34 "</p> <p>" 123.45 "</p> <p>"1,234.56 "</p> <p>"1,234.56 "</p>	<p>hidden comma appears</p> <p>long/bad beep because maximum number digits entered</p>

Format Specifier	Key Pressed	Display	Explanation
"%m0%M6%o %h,%o %s\$%o0%f.%o00"	<none>	" \$0.00"	
1 space (explicit)	'5'	" \$0.05"	
2 spaces	'1'	" \$0.51"	
	'2'	" \$5.12"	
	'3'	" \$51.23"	
	'9'	" \$512.39"	
	'7'	"\$5,123.97"	hidden comma appears
	'6'	"\$5,123.97"	long/bad beep because maximum number digits entered

**Info** Note that the following format specifiers have been updated/corrected:

Format Specifier Key	Description
"%d1%o" / "%d1%z%o"	20 digits may be entered, unless sent down in a 21.x message with the '%M' attribute set
"%m0%M3%o %f%%"	Three (3) digits may be entered to the left of the displayed percent sign

## 8. Line Display

---

### 8.1. General Information

---

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

### 8.2. Usage Samples

---

#### 8.2.1. Clear a Window

##### 8.2.1.1. Java Sample

The following sample code in Java shows how to clear a window:

```
m_LineDisplay.open("Telium2");  
m_LineDisplay.claim(5000);  
m_LineDisplay.setDeviceEnabled(true);  
m_LineDisplay.clearText();
```

##### 8.2.1.2. C++ Sample

The following code sample in C++ shows how to clear a window:

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

#### 8.2.2. Write Text To a Location (Row, Column)

##### 8.2.2.1. Java Sample

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

```
int row = 0, col = 3;  
m_LineDisplay.open("Telium2");  
m_LineDisplay.claim(5000);  
m_LineDisplay.setDeviceEnabled(true);  
m_LineDisplay.displayTextAt(row, col, "Hello World",  
DISP_DT_NORMAL);
```

##### 8.2.2.2. C++ Sample

The following code sample in C++ 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));
```

## 8.2.3. Write Text To a Variable

### 8.2.3.1. Java Sample

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

```
int nCmd = IngenicoConst.ING_DIO_SETVAR; //
ING_DIO_SETVAR=161(0xA1)
int[] nData = new int[] {0};
String sVarName = "TOTAL=$12.99";
m_LineDisplay.open("Telium2");
m_LineDisplay.claim (5000);
m_LineDisplay.setDeviceEnabled(true);
m_LineDisplay.directIO(nCmd, nData, sVarName);
```

### 8.2.3.2. C++ Sample

The following code sample in C++ 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);
```

## 9. MSR

---

### 9.1. General Information

---

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

### 9.2. Usage Samples

---

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

##### 9.2.1.1. Java Sample

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

```
String strTrack1, strTrack2, strTrack3;
m_MSR.open("Telium2");
m_MSR.claim(5000);
m_MSR.setDeviceEnabled(true);
m_MSR.setDataEventEnabled(true);
// Now check each track for data after Data Event happens
strTrack1 = new String(m_MSR.getTrack1Data());
strTrack2 = new String(m_MSR.getTrack2Data());
strTrack3 = new String(m_MSR.getTrack3Data());
```

##### 9.2.1.2. C++ Sample

The following code sample in C++ 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();
```

#### 9.2.2. SetDeviceEnabled Error Reporting

##### 9.2.2.1. Java Sample

The following code sample in Java shows how to enable error reporting:

```
m_MSR.open("Telium2");
m_MSR.claim(5000);
m_MSR.setDeviceEnabled(true);
// use one of the error methods below but not both
// error reporting by card
m_MSR.setErrorReportingType(jpos.MSRConst.MSR_ERT_CARD);
// error reporting by track
```

```
m_MSR.setErrorReportingType(jpos.MSRConst.MSR_ERT_TRACK);
```

#### 9.2.2.2. C++ Sample

The following code sample in C++ 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);
```

### 9.2.3. Request Sentinels

#### 9.2.3.1. Java Sample

The following code sample in Java shows how to request sentinels:

```
m_MSR.open("Telium2");
m_MSR.claim(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);
```

#### 9.2.3.2. C++ Sample

The following code sample in C++ 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);
```

### 9.2.4. Parse MSR Data (VB.NET)

The following 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
```



## Notes

## 10. Signature Capture

---

### 10.1. General Information

---

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

### 10.2. Signature Form

---

The UIA signature form allows users to define additional buttons beyond the standard Enter, Clear and Cancel buttons.

When a user-defined button is pressed, the OPOS/JPOS driver sends a data event with the corresponding button ID.

### 10.3. Usage Samples

---

#### 10.3.1. 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.

#### 10.3.2. 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 ;
```

#### 10.3.3. 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*.

#### 10.3.3.1. Java Sample

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

```

m_SignatureCapture.open("Telium2");
m_SignatureCapture.claim(5000);
m_SignatureCapture.setDeviceEnabled(true);
m_SignatureCapture.setDataEventEnabled(true);
m_SignatureCapture.beginCapture("Sigform");

```

#### 10.3.3.2. C Sample

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
}

```

#### 10.3.3.3. Visual Basic Sample

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
```

## Notes

## 11. SigDisplay Control

---

### 11.1. General Information

---

Ingenico wrote its own form control, **EnFormSigDisplay.ocx**, as an extension to the OPOS specifications in order to display or save signature capture data from the Telium PIN pads.

### 11.2. Summary

---

This section contains precise usage prerequisites for each property.

#### 11.2.1. Properties

**Table 6: 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

#### 11.2.2. Methods

**Table 7: Specific Methods**

Name	May Use After
SetOPOSBCNIBBLESignedData	Acquiring signature data as LPCSTR
SetOPOSBCNIBBLESignedDataX	Acquiring signature data as LPCSTR
SetSignatureData	Creating VARIANT with signature data
SetSignatureDataX	Creating VARIANT with signature data
GetSignatureType	Methods
GetSignatureTypeString	Methods

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

## 11.3. Properties

### 11.3.1. 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.

### 11.3.2. 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.

### 11.3.3. GetDrawBackground

Syntax	BOOL GetDrawBackground;
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.

### 11.3.4. SetDrawBackground

Syntax	BOOL SetDrawBackground;
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.

### 11.3.5. GetPenWidth

Syntax	LONG GetPenWidth[Printer];
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> </ul>

	<ul style="list-style-type: none"> <li>• Default pen width on printer: 2</li> </ul>
--	---

### 11.3.6. SetPenWidth

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

### 11.3.7. GetDisplayNumPoints

Syntax	<b>BOOL GetDisplayNumPoints;</b>
Remarks	<p><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:</p> <pre> if (m_bDisplayNumPoints) {     CString csTmp;     csTmp.Format("%ld", m_lTotalDisplayedPoints);     pdc-&gt;TextOut(0, 0, csTmp); } </pre>

### 11.3.8. SetDisplayNumPoints

Syntax	<b>BOOL SetDisplayNumPoints;</b>
Remarks	<p><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:</p> <pre> if (m_bDisplayNumPoints) {     CString csTmp;     csTmp.Format("%ld", m_lTotalDisplayedPoints);     pdc-&gt;TextOut(0, 0, csTmp); } </pre>



## 11.4. Methods

### 11.4.1. 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>

### 11.4.2. 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>

### 11.4.3. 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>

### 11.4.4. 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>

### 11.4.5. GetSignatureType

Syntax	short <b>GetSignatureType</b>
Remarks	<p>Once <b>SetSignatureData[X]</b> or <b>SetOPOSBCNIBBLESignedData[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>

### 11.4.6. GetSignatureTypeString

Syntax	BSTR <b>GetSignatureTypeString</b>
Remarks	<p>Once <b>SetSignatureData[X]</b> or <b>SetOPOSBCNIBBLESignedData[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>

### 11.4.7. 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>

### 11.4.8. 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

## 12. PIN Pad

---

### 12.1. General Information

---

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



*The PIN Pad control's **DataEvent** property can be always be updated, even after the **BeginEFT()** function has been called.*

### 12.2. Encryption Key Formats

---

#### 12.2.1. 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 encrypt 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.

#### 12.2.2. 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'.



*The **AdditionalSecurityInformation** property is not available for the RSA PayPal PIN.*

### 12.3. RSA Encryption for PayPal PIN

---

The RSA Key is a public key name provided by either PayPal or Ingenico to encrypt the PayPal PIN entered to the PIN Pad device by the end user.

This public key name variable must be set and loaded to the PIN Pad device using the **RSA\_KEY** variable [see section 6.1.3 (0xA1) Set UIA Variable command] prior to allowing any **BeginEFTTransactions()**. The reason for loading the public key, first, is that the OPOS driver needs to be notified as to the encryption type via a parameter in the function **PinCtrl.BeginEFTTransaction (RSA, nKeySlot)** prior to allowing transactions.

Once the POS has enabled the **PINEntry()** function, the OPOS driver will begin polling for the PIN block.

When the end user hits the Enter key on the PIN Pad, the RSA Key encrypts and sends the (now encrypted) PIN to the OPOS driver. The OPOS driver then encodes the PIN block (using Base64 encoding).

## 12.4. Usage Samples

---

### 12.4.1. Handle PIN Entry

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>
```

#### 12.4.1.1. Java Sample

The following code sample in Java shows how to handle PIN entry:

```
m_PINPad.setAccountNumber("1234567890");
m_PINPad.setAmount(90);
m_PINPad.beginEFTTransaction(JposDeviceServiceConst.SVC_ENCRY
PTTYPE_DUKPT, 0);
m_PINPad.setDataEventEnabled(true);
m_PINPad.enablePINEntry();
```

#### 12.4.1.2. C++ Sample

The following code sample in C++ shows how to handle PIN entry:

```
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();
```

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

The following code sample in C++ 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 ;
}

```

### 12.4.3. Retrieve PIN Block

#### 12.4.3.1. Java Sample

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

```
String sEncryptedPin = m_PINPad.getEncryptedPIN();
```

#### 12.4.3.2. C++ Sample

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

```
CString PINString = m_PINPad.GetEncryptedPIN();
```



Use the `PINControl.GETEncryptedPIN()` function to retrieve PayPal's PIN block.

### 12.4.4. Set PIN Pad Prompt Language

Ingenico supports three languages following UPOS standards for the North American region (English, Spanish and French).

The Microsoft macro `MAKELANGID`(Primary language identifier, Secondary language identifier) returns the language identifier for the supported language(s).

- English = 1033
- Spanish = 2058
- French = 3084

#### 12.4.4.1. Java Sample

The following code sample in Java shows how to set the PIN Pad prompt language:

```
jpos.PINPad m_PINPad = new PINPad();
m_PINPad.setPrompt(1);
```

#### 12.4.4.2. C++ Sample

The following code samples in C++ shows how to set the PIN pad prompt language:

```
COPOSPINPad m_PIN;
long m_PromptLang;

if(strPromptLang.CompareNoCase("French") == 0)
{


```


```

        m_PromptLang =
        MAKELANGID( LANG_FRENCH, SUBLANG_FRENCH_CANADIAN );
    }
    else if( strPromptLang.CompareNoCase( "Spanish" ) == 0 )
    {
        m_PromptLang =
        MAKELANGID( LANG_SPANISH, SUBLANG_SPANISH_MEXICAN );
    }
    else
    {
        m_PromptLang = MAKELANGID( LANG_ENGLISH, SUBLANG_ENGLISH_US );
    }

    m_PIN.SetPromptLanguage( m_PromptLang );

```

 Current UIA languages are English (1), Spanish (2), and French (3).

 The default language is English. If OPOS is unable to read a valid “Prompt Language” value from the Configuration.xml file, English will be used.

## 12.5. Advertising Support

---

UIA supports server-based advertising on Telium devices. Please contact your Ingenico Representative for additional information.

## 13. 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.

### 13.1. Post Splash Screen Custom Form Feature

---

UIA has the ability to automatically display a form five seconds after the boot process completes. This feature provides the following features:

- Quickly hides the UIA splash screen and displays a screen that is of interest to a PIN pad user.
- Limits access to the PIN pad's configuration menu provided by the splash screen. The configuration menu is only available during the five seconds the splash screen is displayed.
- Automatically displays a screen that can be used for displaying text, images, image looping, and videos until the POS sends a message to the PIN pad.

**To enable** this feature, follow these steps:

1. Create a form (.K3Z file) and give it the reserved name "UIAPOSTBOOT.K3Z". NOTE: The form must be a display only form (e.g., no user input).
2. Download the form along with any associated images and videos to the PIN pad.
3. Reboot the PIN pad.

UIA will display the form five seconds after the splash screen displays if the form exists.

If the form does not exist, UIA will continue the splash screen until a message is received the POS.

**To disable** this feature, delete the "UIAPOSTBOOT.K3Z" form from the PIN pad.

## 14. MSR Encryption

---

### 14.1. Supported Encryption Methods

---

The following encryption methods are supported by the UIA Application:

Encryption Method	Supported	Not Yet Supported	0091_0001 Enable Encryption Value (“n”)
RSA-OAEP		✓	9
TransArmor	✓		10

### 14.2. Loading Key Serial Number (KSN) Data

---

Applicable to encryption types that use DUKPT keys, before MSR Encryption processing can begin, the PIN pad needs to be loaded with test keys. During the loading process, an administrator injects key serial number (KSN) data into the device.

**Info** PIN pad KSN data is visible next to KSN indexes 0 and 6 in the PIN pad’s TSA application. See your PIN pad’s Operations Guide for information on accessing terminal applications other than UIA.

### 14.3. Enabling MSR Encryption in UIA

---

UIA users can enable MSR encryption by adjusting the following config.dfs parameters found in the **security.dat** file (see also section 15.2 Security Parameters (security.dat)):

- 0091\_0001 – set value to “n” as noted, above in the table in section 14.1 Supported Encryption Methods.

**Info** Please note that to set these, users will have to edit config.dfs (the security.dat file) directly, and obtain signature and a new .PGZ file prior to implementation. (See also section 14.4 Signing Requirements for .DAT File Changes for details.)

### 14.4. Signing Requirements for .DAT File Changes

---

If your organization requires changes to the **config.dfs file**, you must follow the procedure outlined in this section; a change to the config.dfs file requires that Ingenico generate and sign data files to be downloaded to the terminal for your implementation. The generated and newly signed files are security.dat and secbin.dat.

- **Security.dat** - Security parameters - NOTE! Contains a binary version of the contents in the 0091\_xxxx section of config.dfs. These parameters must be signed and cannot be changed by messages! (See also section 15.2 Security Parameters (security.dat))

- **Secbin.dat** - Security BIN table - NOTE! Contains binary version of 0092\_xxxx section of config.dfs. These parameters must be signed and cannot be changed by messages! (See also section 15.3 Security BIN (secbin.dat))



After approval and signature of any changes to your security.dat and/or secbin.dat files, you will receive a new .PGZ file from Ingenico.

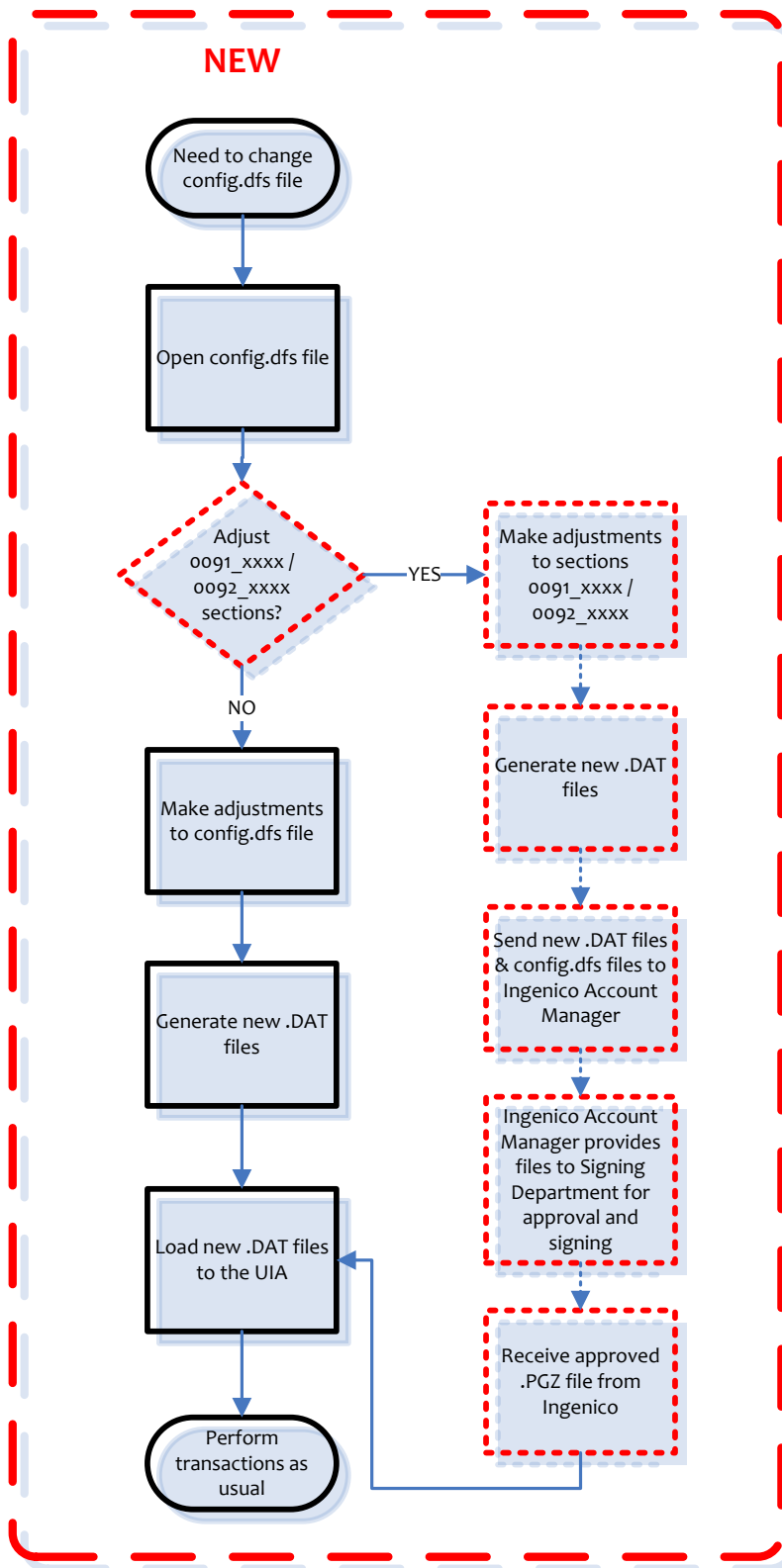


NOTE: You will be unable to implement your changes to the security.dat and secbin.dat files without a signed .PGZ file from Ingenico. If you implement your changes prior to receipt of the new .PGZ file, your Telium devices may appear to run properly, however, your devices will actually be running as previously configured, without your changes. See the process diagram on the following page for approval process information.



The POS app must download the .PGZ file and activate it using DirectIO with option 1 ('Save and Activate').


Contact your Ingenico Account Manager with any questions you may have about the signing process.



**Figure 8. MSR Encryption Configuration Process for UIA**

## 14.5. Selecting Specific Cards to be Encrypted


By default, MSR encryption encrypts all cards. If you need some cards to be encrypted and others not, you need to configure the appropriate bin ranges.

 Please also refer to sections 15.3.1 Enabling BIN Checking and 15.3 Security BIN (secbin.dat) for more information.

## 14.6. Manual Entry

UIA supports manual entry when using encryption. This section discusses configuration for this need, and is applicable to all encryption types supported by UIA.

### 14.6.1. Manual Entry Variables

Variable Name	Variable Description
pTERMINALID	<p>A variable length alphanumeric value that allows customers to uniquely identify each of their terminals. This variable is currently used only for TransArmor encryption. (The lowercase 'p' at the beginning of the variable name indicates that the value persists over terminal reboots.)</p> <p> TransArmor encryption requires that pTERMINALID be set prior to setting MANUAENTRY variables, else an error will occur, prohibiting movement forward.</p>
MANUAENTRY	<p>Used to set the following manual entry variables.</p> <ul style="list-style-type: none"><li>• ON – Begins the Manual Entry session. MANUAENTRY=ON.</li><li>• PAN – Any value entered during a clear text entry will be stored as the manual entry account number. Subsequent clear text entries while MANUAENTRY=PAN will replace the stored account number. Can only be set during a Manual Entry session.</li><li>• EXP – Any value entered during a clear text entry will be stored as the manual entry expiration date. Subsequent clear text entries while MANUAENTRY=EXP will replace the stored expiration date. Can only be set during a Manual Entry session.</li><li>• CVV2 – Any value entered during a clear text entry will be stored as the manual entry CVV2. Subsequent clear text entries while MANUAENTRY=CVV2 will replace the stored CVV2. Can only be set during a Manual Entry session.</li><li>• CREATETRACKS – Used to retrieve the encrypted manual entry result. MANUAENTRY=CREATETRACKS. Can only be set during a Manual Entry session. Should call only after PAN, EXP and CVV2 are collected.</li><li>• OFF – Ends the Manual Entry session. MANUAENTRY=OFF. This action disables the MSR device.</li><li>• UIA will not return the entered value in a Get Input (0x16) Poll. It will only return the ENTER or CANCEL keys.</li></ul>
EXP_DATE	<p>Used to set the manual entry expiration date from the POS as an alternative to prompting for user entry. This can be set only during a Manual Entry session.</p> <ul style="list-style-type: none"><li>• This variable cannot be used to retrieve an expiration date value entered from the terminal keypad.</li></ul>

Variable Name	Variable Description
CVV2	<p>Used to set the manual entry CVV2 from the POS as an alternative to prompting for user entry. This can be set only during a Manual Entry session.</p> <ul style="list-style-type: none"> <li><i>This variable cannot be used to retrieve a CVV2 value entered from the terminal keypad.</i></li> </ul>
MANUALENTRYSTATUS	<p>To retrieve Manual Entry data status at any time, the POS App must retrieve the value of the variable “MANUALENTRYSTATUS”. Can be called at any time.</p> <p>The valid value returned is a bit wise combination of the PAN, Expiration Date and CVV2.</p> <p><b>Status values:</b></p> <ul style="list-style-type: none"> <li>0 = No data set</li> <li>1 = Only PAN set</li> <li>2 = Only Expiration Date set</li> <li>3 = PAN and Expiration Date set</li> <li>4 = Only CVV2 set</li> <li>5 = PAN and CVV2 set</li> <li>6 = Expiration Date and CVV2 set</li> <li>7 = PAN, Expiration Date and CVV2 set</li> </ul>

### 14.6.2. Manual Card Entry Flow

The diagram below illustrates the flow for UIA’s encrypted manual card entry process:

- The POS opens, claims and enables the MSR control.
- The POS sets the MANUALENTRY variable to ON (MANUALENTRY=ON). This begins the Manual Entry session.
- UIA returns no data (0xF0) to MSR Polls if all information (e.g., PAN, expiration date and CVV2) is not available.
- Once UIA has the values for account number, expiration date and CVV2, the POS must set the MANUALENTRY variable to CREATETRACKS. The terminal then builds and encrypts the track data.
- The driver will retrieve the encrypted data in the next MSR Poll. UIA returns track1, track2 and track3 with the data source identified as “manual entry” (value=3).
- The UPOS driver fires two events: data event back to the POS, and Direct IO event (value=3).
- POS sets MANUALENTRY=OFF. This ends the Manual Entry session.



*At any time during a Manual Entry session, MANUALENTRYSTATUS may be used to determine which values have been entered.*

Additional detail follows the diagram.



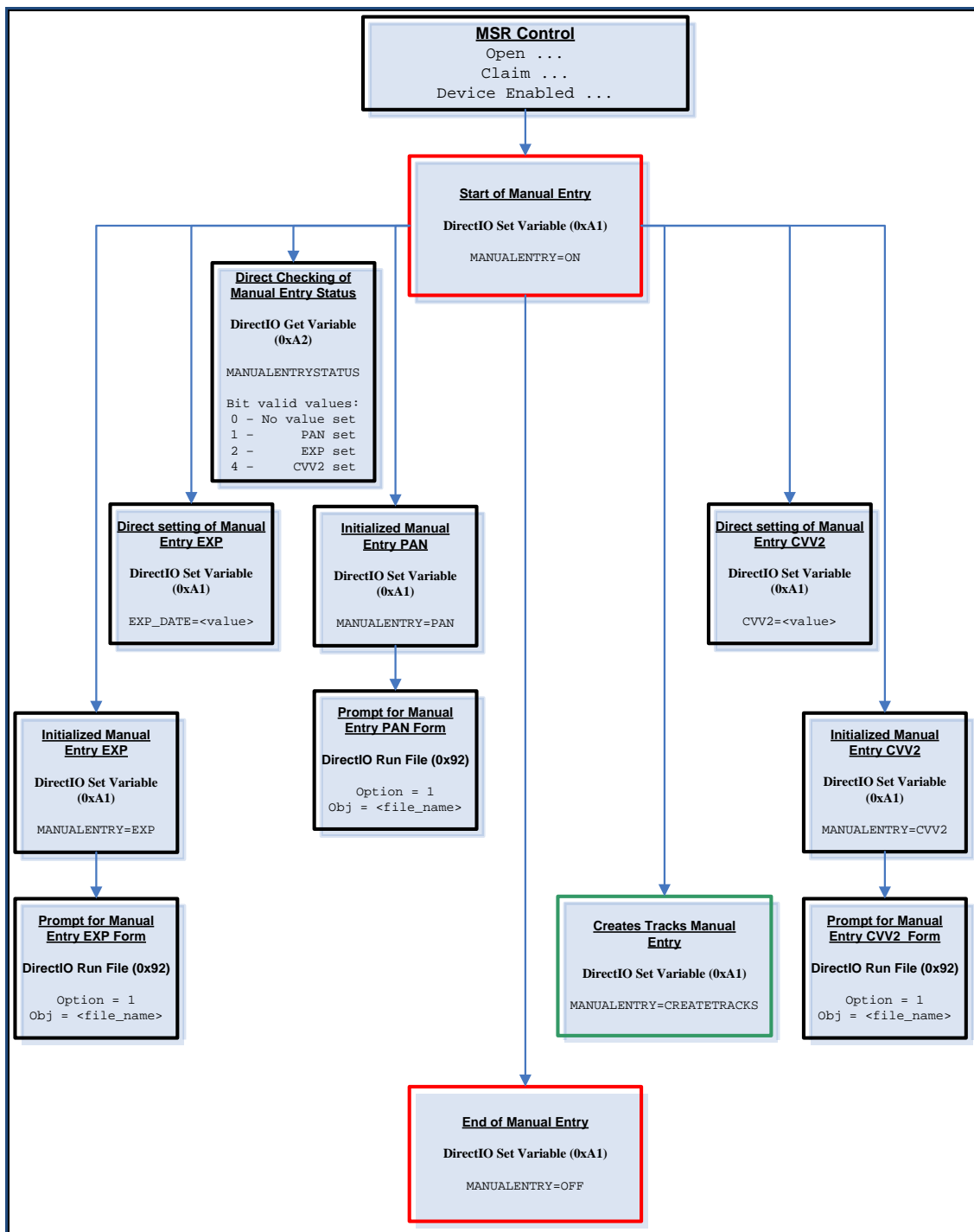


Figure 9. Manual Card Entry Flow for encryption type

#### 14.6.2.1. Terminal ID

To set a Terminal ID, the POS App needs to set a persistent variable (i.e., one that survives a terminal reboot). The persistent variable in this case is “pTERMINALID”.

#### 14.6.2.2. Starting/Stopping Manual Entry session

To Start a Manual Entry session, the POS App needs to set the following variable:

- MANUALENTY=ON.
- In this case, UIA turns off all readers and positions itself to receive Manual Entry info.

To Stop a Manual Entry session, the POS App needs to set the following variable:

- MANUALENTY=OFF.
- In this case, UIA clears all Manual Entry variables (PAN, EXP DATE and CVV2). The OPOS driver disables the MSR Control, and there will be no more polling for MSR data.

#### 14.6.2.3. Setting Manual Entry data (PAN, EXP DATE, CVV2)

These three values may be entered or set in any order.

As soon as UIA has collected all three values, UIA will create the track data.

The track data will be masked and encrypted based on the current encryption setting. For example, for TransArmor (see section 14.7), Tracks 1 & 2 will be masked, while Track 3 will be encrypted and Base64 encoded.

##### 14.6.2.3.1. Setting an Account Number (PAN)

There is only one way to set the PAN - Prompt from a Form.

###### **Prompt from a form:**

1. POS sets MANUALENTY=PAN.
2. POS sets prompt requesting account number.
3. POS runs file using Direct IO with option '1' for Polling. Object=<file\_name>.
4. User enters account number on terminal and presses ENTER key or presses CANCEL key.
5. UI returns ENTER or CANCEL key via UPOS driver.
6. POS may use MANUALENTYSTATUS to determine which values have been entered.

##### 14.6.2.3.2. Setting an Expiration Date (EXP)

Use one of the following two ways to set the Expiration Date:

###### **Prompt from a form:**

1. POS sets MANUALENTY=EXP.
2. POS sets prompt requesting account number.
3. POS runs file using Direct IO with option '1' for Polling. Object=<file\_name>.
4. User enters account number on terminal and presses ENTER key or presses CANCEL key.
5. UI returns ENTER or CANCEL key via UPOS driver.

6. POS may use MANUAENTRYSTATUS to determine which values have been entered.

OR –

**Direct Setting:**

1. Set the variable (EXP\_DATE=<value>).



The POS sends the expiration date to UIA in the EXP\_DATE variable instead of user entry.

**14.6.2.3.3. Setting the CVV2**

Use one of the following two ways to set the CVV2:

**Prompt from a form:**

1. POS sets MANUAENTRY=CVV2.
2. POS sets prompt requesting account number.
3. POS runs file using Direct IO with option '1' for Polling. Object=<file\_name>.
4. User enters account number on terminal and presses ENTER key or presses CANCEL key.
5. UI returns ENTER or CANCEL key via UPOS driver.
6. POS may use MANUAENTRYSTATUS to determine which values have been entered.

OR –

**Direct Setting:**

1. Set the variable (CVV2=<value>).



The POS sends the CVV2 to UIA in CVV2 variable instead of user entry.

**14.6.2.4. Retrieving Manual Entry Data Status (PAN, EXP DATE, CVV2) by POS**

To retrieve Manual Entry data status at any time, the POS App can retrieve the value of the variable "MANUAENTRYSTATUS".

The valid value returned is a bit wise combination of the PAN, Expiration Date and CVV2.

**Status values:**

- 0 = No data set
- 1 = Only PAN set
- 2 = Only Expiration Date set
- 3 = PAN and Expiration Date set
- 4 = Only CVV2 set
- 5 = PAN and CVV2 set
- 6 = Expiration Date and CVV2 set
- 7 = PAN, Expiration Date and CVV2 set

**14.6.2.5. Obtaining Encrypted Manual Entry Result by POS**

To obtain the encrypted Manual entry result, the POS App must explicitly request it by setting the variable "MANUAENTRY=CREATETRACKS". An example of encrypted data can be found in the table in section 14.7.4.2 Data Returned to the POS Application.



Note: Multiple calls to “MANUALENTY=CREATETRACKS” in the same session will generate new data based on the currently entered PAN, EXP and CVV2. Any change to any of these values will result in new track data.

## 14.7. RSA-OAEP & TransArmor Encryption

---

RSA-OAEP Encryption is a feature of Telium OPOS and UIA.

RSA-OAEP encryption is RSA public key encryption with a key length of 2048 bits, and OAEP padding. TransArmor encryption is a special case of RSA-OAEP encryption, which uses the same encryption algorithm but differs in other details. The following information applies to both unless noted otherwise.



See [http://en.wikipedia.org/wiki/RSA\\_\(algorithm\)](http://en.wikipedia.org/wiki/RSA_(algorithm)) and [ftp://ftp.rsasecurity.com/pub/rsalabs/rsa\\_algorithm/rsa-oaep\\_spec.pdf](ftp://ftp.rsasecurity.com/pub/rsalabs/rsa_algorithm/rsa-oaep_spec.pdf) for details of the algorithm.

The current Telium implementation can be used with MSR, contactless and manual entry data.

For both encryption types, the public key consists of a 2048-bit modulus and an exponent which is normally ‘65537’. These values need to be configured in the security.dat section of config.dfs. The resulting security.dat must be signed and downloaded to the terminal to enable the encryption.

### 14.7.1. References

Please refer to the following documents for additional information:

- UnifiedPOS v1.13 specification, available in the ARTS Standards section of the Association for Retail Technology Standards website (<http://www.nrf-arts.org>)
- UPOS Developer’s Guide, Ingenico Part Number DIV350825 Rev. J

### 14.7.2. Configuration - Prerequisites

1. As discussed in section 14.7.3 Configuration, when changes are made to the config.dfs file (due to changes in the security parameters), a newly signed config.dfs file must be downloaded to the terminal.
2. TransArmor encryption requires a unique Terminal ID to be included in the encrypted data block. This ID should be numeric and no longer than 8 digits. This ID is configured by setting the UIA persistent (i.e., one that survives a terminal reboot) variable **pTERMINALID**.




UIA will return a security error (0xF7) if the POS App start Manual Entry (MANUALENTY=ON) and there is no Terminal ID set.




The value for pTERMINALID will persist across terminal reboots, but must be set again after an EFT download.

### 14.7.3. Configuration - General

Configuration information for RSA-OAEP is contained in two sections of the config.dfs file: **SECURITY.DAT** and **SECBIN.DAT**. See also Chapter 15 DAT Files for additional details on these sections.

 *Note: Changes to the SECURITY.DAT and SECBIN.DAT sections must be signed by Ingenico in a new config.dfs file and downloaded to the terminal. This prevents an attacker from turning off encryption or modifying the settings.*

The specific security parameters (in SECURITY.DAT) relevant to TransArmor/RSA-OAEP encryption include the following:

Parameter Name (in security.dat)	DFS Data Index	Default Value	Description
Enable Encryption	0091_0001	0	Specify this value as 9 for RSA-OAEP, 10 for TransArmor.
Specify Encryption Key Slot (Key Index)	0091_0002	6	RSA-OAEP/TransArmor ignores the value of this parameter. Encryption key slots are not used.
Configure Leading PAN Digits in the Clear	0091_0003	6	RSA-OAEP/TransArmor ignores the value of this parameter. Specifies the number of leading digits to be displayed in the clear (Maximum = 6). The default value of 6 is hardcoded for RSA-OAEP/TransArmor
Configure Trailing PAN Digits in the Clear	0091_0004	4	RSA-OAEP/TransArmor ignores the value of this parameter. Specifies the number of trailing digits to be displayed in the clear (Maximum = 4). The default value of 4 is hardcoded for RSA-OAEP/TransArmor.
Masking the PAN	0091_0012	0	RSA-OAEP/TransArmor ignores the value of this parameter. Specifies the character to use for masking the PAN. The default value of 0 (zero) is hardcoded for RSA-OAEP/TransArmor.
Public Key encoded in Base64	0091_0013	(Base64 encoded string value of 392 characters)	Made up of a 2048-bit modulus and an exponent (normally '65537'). See Web page links noted above for more information. This Public Key should be encoded in ASN.1 Base64 format, which will result in a 392-character string value for this parameter.
Exponent Value for RSA-OAEP/TransArmor	0091_0014	010001	Specify this value as the default value = 010001.  <i>This overrides the exponent value from the public key in parameter 0091_0013. This value is in binary format and should generally be set to the default (where 010001 = 65537), but may need to be modified – check with your key authority if you are unsure.</i>
Key ID	0091_0015	(11-byte Key ID)	For TransArmor, the 11-byte Key ID corresponding to the Public Key. Not needed for RSA-OAEP.
Encryption Target	0091_0016	2	For TransArmor, set to 1 or 2 to indicate whether Track 1 or Track 2 data should be used in the encrypted data block. Currently ignored for RSA-OAEP. 1 = Track 1 2 = Track 2 (Default)

## 14.7.4. Usage

### 14.7.4.1. Determining the Encryption Configuration

As noted above, the encryption settings are configured on the terminal and cannot be changed by the application. The application can find out how the terminal is configured by querying certain UIA variables corresponding to the configuration parameters.

The general rule is that the variable **DFS\_xxxx\_yyyy** contains the value of the parameter with DFS Data Index xxxx\_yyyy, as described in the previous section. Only the variables needed by the POS application are provided. Currently, the following variables are supported:

Variable Name	Variable Description
<b>DFS_0091_0001</b>	A read-only variable containing the encryption type: 0 for no encryption, 9 for RSA-OAEP encryption, 10 for TransArmor encryption.
<b>DFS_0091_0015</b>	A read-only variable containing the Key ID corresponding to the public key. May be empty for Generic RSA encryption.
<b>DFS_0091_0017</b>	A read-only variable indicating whether Track 1 or 2 should be returned encrypted for a card swipe or tap (Encryption Target). For Generic RSA encryption this will be set to 4.
<b>pTERMINALID</b>	The Terminal ID (for TransArmor encryption only). This value can be written and read.

### 14.7.4.2. Data Returned to the POS Application

The input to the encryption process depends on the encryption type:

- For generic RSA-OAEP encryption, the input consists of the concatenated raw Track 1, Track 2, and Track 3 data.
- For TransArmor encryption, the input consists of the Terminal ID (padded to 8 bytes with leading 0's), followed by either the raw Track 1, raw Track 2, or manually-entered PAN, depending on the data available for the current transaction.

Once a card has been read, the following OPOS properties are set:

OPOS Property	Contents when using RSA-OAEP Encryption
<b>AccountNumber</b>	Masked account number (middle digits are o)
<b>EXP_DATE</b>	Expiration date (in the clear)
<b>FirstName</b>	First name (in the clear)
<b>MiddleInitial</b>	Middle initial (in the clear)
<b>ServiceCode</b>	Service code (in the clear)
<b>Suffix</b>	Suffix (in the clear)
<b>Surname</b>	Surname (in the clear)
<b>Title</b>	Title (in the clear)
<b>Track1Data</b>	Masked Track 1 data
<b>Track1DiscretionaryData</b>	Masked Track 1 discretionary data
<b>Track2Data</b>	Masked Track 2 data

OPOS Property	Contents when using RSA-OAEP Encryption
Track2DiscretionaryData	Masked Track 2 discretionary data
Track3Data	<p>The encrypted data (2048 bits or 256 bytes) are returned in Track 3 to the POS device. The encrypted data are Base64-encoded, resulting in a 344-byte string.</p> <p>For <b>RSA-OAEP</b>, the input to the encryption process consists of the concatenated raw Track 1, Track 2, and Track 3 data. The resulting 344-byte Base64-encoded string of encrypted data is sent as Track 3 to the POS.</p> <p>For <b>TransArmor</b> encryption, the input consists of the Terminal ID (padded to 8 bytes with leading 0's), followed by either the raw Track 1, raw Track 2, or manually-entered PAN, depending on the data available for the current transaction. The Track 3 data sent to the POS consists of three items separated by colons (":"): </p> <ul style="list-style-type: none"> <li>• The 344-byte Base64 string of encrypted data</li> <li>• One digit indicating which data were encrypted: 1=Track 1, 2=Track 2, 3=PAN for manually-entered data.</li> <li>• KEY ID from the security.dat file.</li> </ul> <p>Example of Track3Data for TransArmor encryption, where Track 2 data were encrypted with the Key ID of 12345678901:</p> <pre>h7S2Qv71zutAc/6my+V3XaKQv62sQowIhmv2yhogDKyINchR28kv26ZfRrQCq yTkne7nTFjxiES5jonFJRax3xhOoEKwlohpDikEi4roStHvF8osY9KwJ+5Uguo XC+YfubQacSKtZ2ic5ATLwqooWhNkjgTBtooyZNhiDRVWok7LGNMx9plqO XIG5nvzONkzLak72hbxjRH452QYN+qC+XcJKgSsQdxziMhNSygdUY7HcfQ1K QogkkZtwz5Ei+HFrVPKhheAivhJkOwrBa6w6humyvg+2A1VATGIZUkgXwYq Rxf0/iRSSgH29lHUXxmCn/MAa2/Ui34diQUnaolMLg==:2:12345678901</pre>



Note: If the TransmitSentinels property is true, then Track1Data, Track2Data and Track3Data will each begin with a start sentinel and end with an end sentinel.



Note: Currently, Track 3 data either in the clear, or masked, is not available to the OPOS application. Track 3 data will be included in the encrypted data block if generic RSA encryption is supported.

## 14.8. .PGZ File Errors/Troubleshooting

If you experience errors from your authorizer stating decryption cannot occur, you may need to obtain a new, signed .PGZ file from Ingenico.

- Contact your Ingenico Account Manager with any outstanding questions.

## 15. DAT Files

---

DAT files are files that reside in an Ingenico PIN pad. The name and extension of these files are determined at data entry. Each DAT file will consist of two parts: a header and data.

- The header contains basic information about file size, data offset, data format, and revision information, followed by the data. All data is contiguous. The format field determines the format of the data.

The file size and data offset will each be 4 bytes in length. The format will be ASCII hex. The format field is 5 bytes in length. The revision information is 12 bytes in length.

- The data is delimited with the binary number zero (0x0).


The DAT file should not be edited since it is difficult to determine the meaning of the data therein. Instead, edit the DFS file and run the translator from the DFS to DAT format.

### 15.1. Changes to security.dat or secbin.dat

---

The following policy applies to all MSR encryption methods.

If your organization requires changes to either the **security.dat** or **secbin.dat** sections of the config.dfs file, you must follow the procedure to obtain approval and signature from Ingenico prior to your implementation. See section 14.4 Signing Requirements for .DAT File Changes for more information.


 You will be unable to implement your changes to security.dat and secbin.dat without a signed .PGZ file from Ingenico. If you implement your changes prior to receipt of the new .PGZ file, your Telium devices may appear to run properly, however, your devices will actually be running as previously configured, without your changes.


Contact your Ingenico Account Manager with any questions you may have about the signing process.

### 15.2. Security Parameters (security.dat)

---

This section describes the security parameters in the security.dat file.

 If your organization requires changes to the security.dat file, you must follow the procedure to obtain approval and signature from Ingenico prior to your implementation. See section 14.4 Signing Requirements for .DAT File Changes.

 This policy applies to all MSR encryption methods.




Parameter Name	DFS Data Index	Default Value	Description
Enable Track Data Encryption	0091_0001	0	<p>Encrypt track data sent in messages.</p> <ul style="list-style-type: none"> <li>0 = Disabled</li> <li>9 = RSA-OAEP</li> <li>10 = TransArmor</li> </ul>
Track Data Encryption Key Index	0091_0002	6	<p>Encryption key index for encrypting track data in messages.</p> <ul style="list-style-type: none"> <li>Only valid if 0091_0001 is set to 1, 2, 3, 7 or 8.</li> </ul> <hr/> <p>RSA-OAEP/TransArmor ignores the value of this parameter. Encryption key slots are not used.</p>
Unmasked leading digits	0091_0003	6	<p>Number of leading digits not masked (e.g., displayed in the clear) when an encryption method is set to encrypt the account number. (Maximum = 6)</p> <hr/> <p>RSA-OAEP/TransArmor ignores the value of this parameter. The default value of 6 is hardcoded for RSA-OAEP/TransArmor</p>
Unmasked trailing digits	0091_0004	4	<p>Number of trailing digits not masked (e.g., displayed in the clear) when an encryption method is set to encrypt the account number. (Maximum = 4)</p> <hr/> <p>RSA-OAEP/TransArmor ignores the value of this parameter. The default value of 4 is hardcoded for RSA-OAEP/TransArmor.</p>
<b>Voltage-Specific Settings (0091_0005 through 0091_0011)</b>			
Max Number of Transactions with Same Key	0091_0005	0	<p>Maximum number of transactions with the same key.</p> <ul style="list-style-type: none"> <li>0 = Don't change keys based on transaction count</li> <li>1-65000 = Change key after this many transactions with the same key</li> </ul>

Parameter Name	DFS Data Index	Default Value	Description
Periodically Change Keys	0091_0006	0	<p>Periodically change keys (Requires setting the device's date and time).</p> <ul style="list-style-type: none"> <li>• All letters must be entered in UPPER CASE</li> <li>• 0 = Disabled</li> <li>• D = Daily</li> <li>• SU = Change every Sunday</li> <li>• MO = Change every Monday</li> <li>• TU = Change every Tuesday</li> <li>• WE = Change every Wednesday</li> <li>• TH = Change every Thursday</li> <li>• FR = Change every Friday</li> <li>• SA = Change every Saturday</li> <li>• 01-31 = Change on the XXth day of the month</li> </ul>
Preserve Keys During Power Failure	0091_0007	0	<p>Preserve keys during power failure</p> <ul style="list-style-type: none"> <li>• 0 = A new key is generated at power up</li> <li>• 1 = Keys are saved when generated and restored at power up</li> </ul>
Return ETB with Track Data	0091_0008	0	<p>Append ETB to Authorization Request</p> <ul style="list-style-type: none"> <li>• 0 = Do not append</li> <li>• 1 = Append</li> </ul>
Identity String	0091_0009	<a href="#">id@sample.com</a>	<p>Identity String</p> <ul style="list-style-type: none"> <li>• Provided by authorizer</li> </ul> <p>“<a href="#">id@sample.com</a>” is sample data – not for production.</p>
Identity Date	0091_0010	*	<p>Identity date</p> <ul style="list-style-type: none"> <li>• Use format mmddyyyy</li> <li>• If set to “*” the device's current date will be used.</li> </ul>


Parameter Name	DFS Data Index	Default Value	Description
Parameter Data Encoded in base64	0091_0011		<p>Sample data – not for production (actual data must be obtained from your authorizer):</p> <p>"AkkI5lUTQBRABHW+Q8WeFmGOQOXIe6Av2WvRhsSVLm1BWKza4u/1JcN56dPW5L1pC8xd6H/7UcovAWH8Na07Ge5lsfCW3wbPRDouotjVd7QJoODLQe1CGQpNJSN/Wbx5j3E5iYnWuEhSifDqy9M2zZzXWAHdzd6vkRrFkLRyavYGUsaJa4fzg1y/BEOYWaURa9l4uUDoexrG4VWrRdQ5GWJGQ3oNQC/U9Bd6FaOxfnAKttNGiidLN5mNB6ZxjuBGMj8jM3WxJKvmceyyIW18t4ddUllrTptQ2ZU1BkoZjlbJvLuggdGv2IDZP4CMtznrvCoc5lI8QaoRD7+flpfYZZR+789TqSo3YeI5BB5rdv9iR/uloc/+DOUJSUERhVjTps2N5sJ5BX4OJy3dnD/BDiLulQK2VBLtiJ1QNjYpmfZPUBGT5xGIMrbXw4e5WX87bmc9jAkJZkgoP5pJxm/MXuD1vtCcNVCrFnZ/Qi/5SI/PEvxxYnZkfCfzWGM/PgwA1L2T3VFBfTQeTBLBLTV8us5rsGUce/nyzuD54ZFRbOeqVcWEdrnZT/GWSUC97YAoAFsxT6GkbFR1qv7OAPIzeyw1W6WpciGC4bgATdelxSD76mm+kPy4DwckTFdVKiDdo4N6rIBUjph1eV5/StpNXQ4vmwTMhbo2v9b+5WgcGboiWcAcc8AiuDzeYxZL2iJUYo77U3dYKpTsYLZuWaLFYiLuwgdBQ+lsZrTYxPEjx/bp4JHvxMI8tBtU5l3DRjo rk4H7H/syGDtb7z//////////v//////////fPml+YrYeuVmH8NJpSgBYtAlj7VuzyppnWI65coCPZRCWIYxkCtTybOIgej5cYKTxLFyP12yWno17VRN+NBnozJlF6nMoOEw+GWdSzo7kpi+Et8YytoLEA+BPu7vGvUFUmD67v21PL7i+UYjHMN7erBhbGIB+e8ywafDBEQqaIOFza3libHVLnZvbHRhZ2uUy29t"</p>
<b>End Voltage-Specific Settings (above)</b>			
Masking Character	0091_0012	0	<p>Defines the character to use for masking the PAN.</p> <p><b>Info</b> For 0091_0001 only 'o' (zero) and '*' (asterisk) are valid.</p> <hr/> <p>RSA-OAEP/TransArmor ignores the value of this parameter. The default value of 0 (zero) is hardcoded for RSA-OAEP/TransArmor.</p>
Public Key encoded in Base64 (in security.dat)	0091_0013	(392-character string)	<p>Made up of a 2048-bit modulus and an exponent (normally '65537'). See Web page links noted above for more information.</p> <p>This Public Key should be encoded in ASN.1 Base64 format, which will result in a 392-character string value for this parameter.</p>
Exponent Value for RSA-OAEP/TransArmor (in security.dat)	0091_0014	010001	<p>Specify this value as the default value = 010001.</p> <p><b>Info</b> This overrides the exponent value from the public key in parameter 0091_0013. This value is in binary format and should generally be set to the default (where 010001 = 65537), but may need to be modified – check with your key authority if you are unsure.</p>
Key ID (in security.dat)	0091_0015	12345678901	For TransArmor, the 11-byte Key ID corresponding to the Public Key. Not needed for RSA-OAEP.


Parameter Name	DFS Data Index	Default Value	Description
Encryption Target (in security.dat)	0091_0016	2	For TransArmor, set to 1 or 2 to indicate whether Track 1 or Track 2 data should be used in the encrypted data block. Track 2 data should be used in encrypted data block. Currently ignored for RSA-OAEP. 1 = Track 1 2 = Track 2 (Default)

 See also, section 14.7 RSA-OAEP & TransArmor Encryption for additional information specific to RSA-OAEP and TransArmor encryption types.


### 15.3. Security BIN (secbin.dat)

This section describes the security parameters in the secbin.dat file.

 If your organization requires changes to the secbin.dat file, you must follow the procedure to obtain approval and signature from Ingenico prior to your implementation. See section 14.4 Signing Requirements for .DAT File Changes.


 This policy applies to all MSR encryption methods.


Descriptions for the various card types are same as prior versions.

 **NOTE!** The start and end BINS are compared up to the length of the entry.

The first parameter (0092\_0001) enables the Security BIN Table, itself. The default setting is 0 = Off/Disabled. Use 1 = On/Enabled to enable the table.

Parameter Name	DFS Data Index	Start BIN Range	End BIN Range	Acct. Num Min Length	Acct. Num Max Length	Encrypt (0=no, 1=yes)
Discover 1 card	0092_0002 (bin1.dat)	601100	601199	14	16	1
Discover 2 card	0092_0003 (bin2.dat)	622126	622925	14	16	1
Discover 3 card	0092_0004 (bin3.dat)	644000	649999	14	16	1
Discover 4 card	0092_0005 (bin4.dat)	650000	659999	14	16	1
MasterCard card	0092_0006 (bin5.dat)	510000	559999	14	16	1
VISA card	0092_0007 (bin6.dat)	400000	499999	13	16	1
AMEX – Range 1	0092_0008 (bin7.dat)	340000	349999	14	15	1
AMEX – Range 2	0092_0009 (bin8.dat)	370000	379999	14	15	1
All Other Cards	0092_0010 (bin10.dat)	000000	999999	10	20	0

 0092\_0011 through 0092\_0030 are reserved for expansion (up to bin30.dat). When assigning, follow the same format as the others.

 **NOTE:** The card bins listed in the above table are example bin assignments. Your company may want to adjust the assigned bins as per the variety of cards accepted by your company.

### 15.3.1. Enabling BIN Checking

UIA has the ability to automatically establish a payment type for the customer card. This can be done by using values from the UIA local configuration (internal Spin the BIN - STB), or by asking the POS to select the payment type (external STB).

If BIN range checking is enabled, UIA compares the cardholder account with data from all binX.dat files. If there is no match, UIA checks to see if Spin the BIN (STB) is enabled.

- If enabled, the application goes to STB.
- If it is not enabled, UIA displays the payment screen with the payment buttons, and prompts the cardholder to select the payment type.

As noted in the section above, the first parameter (0092\_0001) enables the Security BIN Table, itself.

- 0 = Off/Disabled (default setting)
- 1 = On/Enabled

## 16. OPOS Usage Samples with Microsoft Common Control

---

This chapter includes OPOS usage samples where Microsoft Common Control in a .NET environment is used to communicate with the driver instead of the standard OPOS controls.

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.

### 16.1. Implementing a Terms and Conditions Screen

---



*Full implementation example planned for a later release.*

### 16.2. Implementing a Survey Screen

---



*Full implementation example planned for a later release.*

### 16.3. Simulating a Transaction

---



*Full implementation example planned for a later release.*

### 16.4. 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://monroeecs.com/posfordotnet/opos\\_dotnet.htm](http://monroeecs.com/posfordotnet/opos_dotnet.htm) for information.

#### 16.4.1. PosExplorer

First add reference 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(DeviceName))
    {
        //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;

```

### 16.4.2. .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.

## 17. UIA Prompts


Prompts for various Ingenico PIN pad applications are found in the PROMPT.xml, SECURPROMPT.xml and CUSTPROMPT.xml files. All prompt files must be signed by Ingenico before they can be loaded onto a PIN pad.

To comply with PCI-DSS requirements, PROMPT.xml and CUSTPROMPT.xml are subject to security restrictions that prohibit the display of prompts containing character combinations representing the words “PIN” and “NIP.”

### 17.1. Security Prompts (SECURPROMPT.xml)

Prompts for various Ingenico PIN pad applications are found in the SECURPROMPT.xml file. The SECURPROMPT.xml file includes prompts in English, Spanish, and French. The SECURPROMPT.xml file MUST be signed by Ingenico before being loaded on an Ingenico PIN pad.

Each prompts XML file is subdivided into sections containing several different types of prompts. All of the following security prompts are available for use in the UIA application.

 **NOTE:** The SECURPROMPT.xml file should be installed on your PC as a part of the Integration Kit. You will find it in the Telium Tools folder.

Prompt ID	Default Value	Description
0	PROMPTINDEX	Prompt ID 0 (zero) indicates that the prompt number should be taken from the PROMPTINDEX variable.
<b>PIN Prompts</b>		
1	"Please enter PIN"	This prompt may be used during PIN entry.
	"Por favor marque PIN"	
	"Entrez votre NIP SVP"	
3	"Enter Valid PIN"	This prompt may be used during PIN entry.
	"Marque un PIN actuel"	
	"NIP erroné. Réessayez"	
14	"Please enter your PIN:"	This prompt may be used during PIN entry.
	"Por favor marque su PIN: "	
	"Entrez votre NIP SVP:"	
15	"Invalid PIN. Please re-enter"	This prompt may be used during PIN entry.
	"PIN incorrecto. Favor de marcar de nuevo"	
	"NIP erroné. Réessayez"	
21	"ENTER PIN"	This prompt may be used during PIN entry.
	"MARQUE PIN"	
	"ENTRER PIN"	



Prompt ID	Default Value	Description
22	"ENTER PIN & ENTER"	This prompt may be used during PIN entry.
	"MARQUE PIN & ENTER"	
	"ENTRER PIN & ENTRER"	
23	"ENTER PIN & PRESS ENTER"	This prompt may be used during PIN entry.
	"MARQUE PIN & OPRIMA ENTER"	
	"ENTRER PIN & PRESSE ENTRER"	
24	"ENTER PIN AND PRESS ENTER"	This prompt may be used during PIN entry.
	"MARQUE PIN Y OPRIMA ENTER"	
	"ENTRER PIN ET APPUYEZ SUR ENTRER"	
25	"REENTER PIN"	This prompt may be used during PIN entry.
	"VUELVA A MARCAR PIN"	
	"RETOURNER PIN"	
26	"REENTER PIN & ENTER"	This prompt may be used during PIN entry.
	"VUELVA A MARCAR PIN & ENTER"	
	"RETOURNER PIN & ENTRER"	
27	"REENTER PIN & PRESS ENTER"	This prompt may be used during PIN entry.
	"VUELVA A MARCAR PIN & OPRIMA ENTER"	
	"RETOURNER PIN & PRESSE ENTRER"	
28	"REENTER PIN AND PRESS ENTER"	This prompt may be used during PIN entry.
	"VUELVA A MARCAR PIN Y OPRIMA ENTER"	
	"RETOURNER PIN ET APPUYEZ SUR ENTRER"	
29	"PIN="	This prompt may be used during PIN entry.
	"PIN="	
	"PIN="	
30	"Please Enter Your PIN and Press &apos;Enter&apos;"	This prompt may be used during PIN entry.
	"Por Favor Marque Su PIN y Oprima &apos;Enter&apos; "	
	"SVP entrer votre code PIN et appuyez sur &apos;Enter&apos;"	
Clear Entry Prompts		
1	"Enter home phone number"	This prompt may be used during user input entry.
	"Marque su número de teléfono"	
	"Entrez le numéro de téléphone à la maison"	
16	"Please enter Cashback:"	This prompt may be used during user input entry.
	Por favor marque Cashback: "	

Prompt ID	Default Value	Description
	"Entrez montant du retrait d'argent SVP:"	
19	"Please enter new amount:"	This prompt may be used during user input entry.
	"Por favor marque la cantidad nueva: "	
	"Entrez le nouveau montant:"	
39	"Enter Cashback in \$&lt;?ivCB_INC?&gt; increments"	This prompt may be used during user input entry.
	"Marque Cashback en incrementos de \$&lt;?ivCB_INC?&gt;"	
	"Retrait d'argent: multiples de \$&lt;?ivCB_INCS?&gt;"	
151	"Enter Card Number"	This prompt may be used during user input entry.
	"Marque Numero de Tarjeta"	
	"Entrer le Nombre de Carte"	
153	"Enter Expiration Date"	This prompt may be used during user input entry.
	"Marque Fecha de Expiración"	
	"Entrer la Date d'Expiration"	
155	"Enter CVV or CID from card"	This prompt may be used during user input entry.
	"Marque CVV o CID de su tarjeta"	
	"Entrer le Code de Sécurité"	
201	"Enter Month and Day of Birth (MMDD)"	This prompt may be used during user input entry.
	"Marque Mes y Día de Nacimiento (MMDD) "	
	"Entrez le mois et le jour de naissance (MMDD)"	
202	"Enter Home Phone Number"	This prompt may be used during user input entry.
	"Marque teléfono de su Hogar"	
	"Entrez le numéro de téléphone à domicile"	
203	"Enter Last 4 Digits of Social Security #"	This prompt may be used during user input entry.
	"Marque Últimos 4 Dígitos de Seguro Social"	
	"Entrez 4 derniers chiffres de la SSN"	
204	"Annual Income (from all sources)"	This prompt may be used during user input entry.
	"Ingreso Anual (de todas fuentes) "	
	"Revenu annuel (toutes sources confondues)"	
205	"Enter Social Security #"	This prompt may be used during user input entry.
	"Marque # de Seguro Social"	
	"Entrez la SSN"	

Prompt ID	Default Value	Description
206	"Please enter your 10-digit HOME phone number"	This prompt may be used during user input entry.
	"Por favor marque el numero de teléfono de 10-dígitos de SU HOGAR"	
	"Entrer votre numéro de téléphone"	
207	"Social Security Number"	This prompt may be used during user input entry.
	"Numero de Seguro Social"	
	"Numéro de Sécurité Sociale"	
208	"Annual Income Amount"	This prompt may be used during user input entry.
	"Cantidad de Ingreso Anual"	
	"Montant du revenu annuel"	
209	"Years lived at Current Address"	This prompt may be used during user input entry.
	"Años Vividos en la Dirección Actual"	
	"Années vécues à l'adresse actuelle"	
210	"Months lived at Current Address"	This prompt may be used during user input entry.
	"Meses Vividos en la Dirección Actual"	
	"Mois vécu à l'adresse actuelle"	
211	"Mothers Maiden Name"	This prompt may be used during user input entry.
	"Apellido Materno"	
	"Mères Nom de jeune fille"	
212	"Email Address"	This prompt may be used during user input entry.
	"Dirección de Correo Electrónico"	
	"Adresse e-mail"	
213	"Date of Birth"	This prompt may be used during user input entry.
	"Fecha de Nacimiento"	
	"Date de naissance"	
214	"Home Phone Number"	This prompt may be used during user input entry.
	"Numero de teléfono"	
	"Numéro de téléphone résidentiel"	
215	"Zip Code"	This prompt may be used during user input entry.
	"Código Postal"	
	"Code postal"	
216	"Use the attached pen to type mother's maiden name"	This prompt may be used during user input entry.
	"Use el lápiz para teclear su apellido materno"	
	"Utilisez le stylo attaché aux mères de type nom de jeune fille"	

Prompt ID	Default Value	Description
217	"Please enter your annual household income"	This prompt may be used during user input entry.
	"Por favor marque su cantidad de ingreso anual"	
	"S'il vous plaît entrer votre revenu annuel du ménage"	
218	"Please enter your date of birth"	This prompt may be used during user input entry.
	"Por favor marque su fecha de nacimiento"	
	"S'il vous plaît entrer votre date de naissance"	
219	"Please enter your social security number"	This prompt may be used during user input entry.
	"Por favor marque su numero de seguro social"	
	"S'il vous plaît entrer votre numéro de sécurité sociale"	
220	"Enter two-digit number of months"	This prompt may be used during user input entry.
	"Marque numero de meses de dos dígitos"	
	"Entrez le numéro à deux chiffres du mois"	
221	"Enter two-digit number of years"	This prompt may be used during user input entry.
	"Marque numero de años de dos dígitos"	
	"Entrez le numéro à deux chiffres des années"	
222	"Please enter your home phone number"	This prompt may be used during user input entry.
	"Por favor marque su numero de teléfono"	
	"S'il vous plaît entrer votre numéro de téléphone à la maison"	
224	"Enter CVN"	This prompt may be used during user input entry.
225	"Please Enter the Check Amount"	This prompt may be used during user input entry.
226	"Please Enter the Check Number"	This prompt may be used during user input entry.
227	"Please Enter Billing Zip Code"	This prompt may be used during user input entry.
228	"Please Enter CID"	This prompt may be used during user input entry.

Prompt ID	Default Value	Description
229	“Please Enter CID Code”	This prompt may be used during user input entry.
230	“Please Enter CVV2”	This prompt may be used during user input entry.
231	“Please Enter CVV2 Code”	This prompt may be used during user input entry.
232	“Please Enter Customer Identification Code”	This prompt may be used during user input entry.
233	“Please Enter Security Code”	This prompt may be used during user input entry.
234	“Please Enter the Last Four digits of your Card Number”	This prompt may be used during user input entry.
235	“Please Enter Invoice Number”	This prompt may be used during user input entry.
236	“Please Enter Driver ID”	This prompt may be used during user input entry.
237	“Please Enter Vehicle Number”	This prompt may be used during user input entry.
238	“Please Enter Vehicle ID”	This prompt may be used during user input entry.
239	“Please Enter Odometer”	This prompt may be used during user input entry.
240	“Please Enter Prompt Code”	This prompt may be used during user input

Prompt ID	Default Value	Description
		entry.
241	“Please Enter Vehicle Card Number”	This prompt may be used during user input entry.
242	“Please Enter ID Number”	This prompt may be used during user input entry.
243	“Please Enter Reference Number”	This prompt may be used during user input entry.
244	“Please Enter Approval Code”	This prompt may be used during user input entry.
245	“Please Enter Customer Code”	This prompt may be used during user input entry.
246	“Please Enter EBT Voucher Number”	This prompt may be used during user input entry.
247	“Is this a Cash Benefit Card”	This prompt may be used during user input entry.
248	“Enter PD Seq # (Vehicle Number, 5 digits)”	This prompt may be used during user input entry.
249	“Please Enter ZIP Code”	This prompt may be used during user input entry.
	“Por favor marque Código Postal”	
	“Entrez votre Code postal”	

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

```
-<PromptDefinition>
-<Lang1>
  <!-- English -->
  -<Pin>
    <Prompt id="0" message="PROMPTINDEX" shortmessage="PROMPTINDEX"/>
    <!-- UIA PIN Prompts (1-13) -->
    <Prompt id="1" message="Please enter PIN" shortmessage="Please enter PIN"/>
    <Prompt id="3" message="Enter Valid PIN" shortmessage="Enter Valid PIN"/>
    <!-- RBA PIN Prompts (14-20) -->
    <Prompt id="14" message="Please enter your PIN:" shortmessage="Please enter your PIN:"/>
    <Prompt id="15" message="Invalid PIN. Please re-enter" shortmessage="Invalid PIN. Please re-enter"/>
    <!-- Other PIN Prompts (21-49) -->
    <Prompt id="21" message="ENTER PIN" shortmessage="ENTER PIN"/>
    <Prompt id="22" message="ENTER PIN & ENTER" shortmessage="ENTER PIN & ENTER"/>
    <Prompt id="23" message="ENTER PIN & PRESS ENTER" shortmessage="ENTER PIN & PRESS ENTER"/>
    <Prompt id="24" message="ENTER PIN AND PRESS ENTER" shortmessage="ENTER PIN AND PRESS ENTER"/>
    <Prompt id="25" message="REENTER PIN" shortmessage="REENTER PIN"/>
    <Prompt id="26" message="REENTER PIN & ENTER" shortmessage="REENTER PIN & ENTER"/>
    <Prompt id="27" message="REENTER PIN & PRESS ENTER" shortmessage="REENTER PIN & PRESS ENTER"/>
    <Prompt id="28" message="REENTER PIN AND PRESS ENTER" shortmessage="REENTER PIN AND PRESS ENTER"/>
    <Prompt id="29" message="PIN=" shortmessage="PIN="/>
    <Prompt id="30" message="Please Enter Your PIN and Press 'Enter'" shortmessage="Please Enter Your PIN and Press 'Enter'"/>
  </Pin>
</Lang1>
</PromptDefinition>
```

Figure 10 - the default SECURPROMPT.xml file.

## 17.2. Transaction Prompts (PROMPT.xml)

Prompts and other display strings are also found in the PROMPT.xml file. The RBA is delivered with prompts in English, Spanish, and French.

Prompt text display properties (such as font type, font color or text justification) are described by the text element used to call the prompt in a given form.

To comply with PCI-DSS requirements, PROMPT.xml is subject to security restrictions that prohibit the display of prompts containing character combinations representing the words “PIN” and “NIP.”

Prompt	Prompt ID	Default Value	Description
Prompt Index	0	“PROMPTINDEX”	Prompt ID 0 (zero) indicates that the prompt number should be taken from the PROMPTINDEX variable.
Language Choice Prompt	1	"Please select language"	If more than one language is specified in the Language Count parameter, this prompt may be displayed at the beginning of each transaction. A button for each available language will appear on the screen for customer selection.
		"Por favor seleccione idioma"	
		"Choisissez la langue SVP"	
Processing BIN Lookup Message	2	"Processing... please wait"	The Processing BIN Lookup message displays while the application is using the BIN lookup function to select the payment type for the swiped card.
		"Procesando... favor de esperar"	
		"En traitement... Un moment SVP"	
Slide Card Prompt	3	"Please slide card"	This parameter prompts the customer to swipe a payment card.
		"Por favor deslice su tarjeta"	

Prompt	Prompt ID	Default Value	Description
		"Glissez la carte SVP"	
Expired Card Prompt	4	"Expired card. Please use another."	This parameter displays when date checking is enabled and the customer used a card with an expiration date that is before today's date. Another form of payment must be submitted in order to complete the transaction.
		"Tarjeta expirada. Favor de usar otra. "	
		"Carte expirée"	
Bad Card Read Message	5	"Card read error. Try again."	The Bad Card Read message informs the customer that PIN pad could not read the card that was swiped.
		"Error de lectura de tarjeta. Intente de nuevo. "	
		"Erreur de lecture. Réessayez"	
Select Payment Prompt	6	"Please select payment type"	This parameter prompts the customer to select the desired payment type.
		"Seleccione tipo de pago"	
		"Choisissez le type de paiement"	
Wait for Amount Message	7	"Please wait for the cashier"	The Wait for Amount Message is displayed when waiting for the purchase amount from the POS.
		"Por favor espere por el cajero(a) "	
		"Attendez le cassier SVP"	
Cash Back Correct Prompt	8	"Cashback correct? \$&lt;?ivCASHBACK?&gt;"	This prompts the customer to verify the cash back amount.
		"Cashback correcto? \$&lt;?ivCASHBACK?&gt; "	
		"Retrait d'argent? \$&lt;?ivCASHBACK?&gt;"	
Ignored Cash Back Message	9	"Not a sale. Cashback cancelled."	This informs the customer that the selected cash back amount will be ignored due to a transaction type limitation. This parameter only applies to void, return, and void return transactions.
		"No es una venta. Cashback cancelado."	
		"Retrait d'argent annulé"	
Amount OK Prompt	10	"Amount OK? \$&lt;?ivTOTAL?&gt;"	This prompts the customer to verify the purchase amount.
		"Cantidad correcta? \$&lt;?ivTOTAL?&gt;"	
		"Montant OK? \$&lt;?ivTOTAL?&gt;"	
Processing Authorization Prompt	11	"Processing... please wait"	This displays while the transaction is being authorized.
		"Procesando... favor de esperar"	
		"En traitement... Un moment SVP"	
Invalid Card	12	"Invalid card for payment type"	The Invalid Card prompt informs the customer that the swiped card is not an accepted form of payment.
		"Tarjeta no permitida para ese tipo de pago"	
		"Carte erronée pour ce type de paiement"	



Prompt	Prompt ID	Default Value	Description
Bad PIN Size Message	14	"PIN must be 4 to 12 digits"	This informs the customer that the PIN entered was not either less than 4 or more than 12 digits.
		"PIN debe ser 4 a 12 dígitos"	
		"Un NIP va de 4 à 12 chiffres"	
Cash Back Prompt	15	"Cashback?"	This asks the customer if he would like to receive cash back.
		"Cashback? "	
		"Retrait d'argent?"	
Cash Back Limit Exceeded Message	17	"Cashback limit is \$&lt;?ivCB_MAX_TEXT?&gt;,"	This informs the customer that the amount requested is greater than the maximum allowed.
		"Limite de cashback es \$&lt;?ivCB_MAX_TEXT?&gt;,"	
		"Retrait d'argent limité à \$&lt;?ivCB_MAX_TEXT?&gt;,"	
Invalid Cash Back Amount	18	"Invalid amount: \$&lt;?ivCASHBACK?&gt;,"	This informs the customer that the amount requested is not a multiple of the cash back increment amount (0002_0009).
		"Cantidad no permitida: \$&lt;?ivCASHBACK?&gt;,"	
		"Montant erroné: \$&lt;?ivCASHBACK?&gt;,"	
Partial Payment Amount Error Prompt	20	"Amount must be less than \$&lt;?ivAMOUNT? &gt;,"	This is an error message that displays if the customer enters a payment amount greater than the purchase amount.
		"Cantidad debe ser menos que \$&lt;?ivAMOUNT?&gt; "	
		"Montant dépasse le total \$&lt;?ivAMOUNT?&gt;,"	
Approved Message	21	"Approved"	This informs the customer that the transaction has been approved. If the authorization message includes a prompt, the prompt from the authorization message is used.
		"Aprobado"	
		"Approuvé"	
Declined Message	22	"Declined"	This informs the customer that the transaction has been declined. If the authorization message includes a prompt, the prompt from the authorization message is used.
		"Denegado"	
		"Refusé"	
Transaction Cancellation Prompt	23	"Transaction cancelled"	This prompt informs the customer that the transaction has been cancelled.
		"Transacción cancelada"	
		"Transaccion Cancelada"	
Invalid Payment Prompt	24	"Invalid payment type"	This prompt informs the customer that they attempted to use an invalid payment type.
		"Tipo de pago no permitido"	
		"Le type nul de Paiement"	
Card Not Accepted	25	"Card not accepted"	This prompt informs the customer that the card was not accepted.
		"Tarjeta no aceptada"	

Prompt	Prompt ID	Default Value	Description
Prompt		"Cette carte n'est pas acceptée"	
Encrypting Message	26	"Encrypting... please wait..."	This is displayed while a PIN is being encrypted after entry.
		"Codificando... favor de esperar..."	
		"Chiffrement... patientez SVP..."	
CPEM Test Card Prompt	27	"CPEM test card read"	This prompt indicates a CPEM test card read.
		"Lectura de tarjeta de prueba CPEM"	
		"Lecture de la carte test CPEM"	
Select EBT Type Message	28	"Please select benefit type"	This is displayed while waiting for the user to select the EBT message type.
		"Por favor seleccione tipo de beneficio"	
		"Choisissez le type d'allocation SVP"	
Card Read Cancelled Message	32	"Card read cancelled"	This prompt is displayed when the cash register cancels the card swipe on demand.
		"Lectura de tarjeta cancelada"	
		"Lecture de la carte annulée"	
Input Request Cancelled Prompt	33	"Input cancelled"	This informs the customer that the POS has cancelled the request to read a card.
		"Entrada cancelada"	
		"Entrée annulée"	
Signature Cancelled Message	34	"Signature cancelled"	This prompt is displayed when the cash register cancels the signature request.
		"Firma cancelada"	
		"Signature annulée"	
Show Card to Cashier Prompt	35	"Please show card to cashier"	This asks the customer to hand the payment card to the cashier so the signature or account number may be verified.
		"Por favor muestre su tarjeta al cajero(a) "	
		"Montrez carte au cassier SVP"	
Void Amount OK Prompt	36	"Void OK? \$&lt;?ivTOTAL?&gt;,"	This asks the customer to verify the void amount.
		"Confirma anulación? \$&lt;?ivTOTAL?&gt; "	
		"Annulation OK? \$&lt;?ivTOTAL?&gt;,"	
Return Amount OK Prompt	37	"Return OK? \$&lt;?ivTOTAL?&gt;,"	This asks the customer to verify the return amount.
		"Confirma devolución? \$&lt;?ivTOTAL?&gt; "	
		"Remboursement OK? \$&lt;?ivTOTAL?&gt;,"	
Void Return Amount OK Prompt	38	"Void return OK? \$&lt;?ivTOTAL?&gt;,"	This asks the customer to verify the void return amount.
		"Confirma anulación de devolución? \$&lt;?ivTOTAL?&gt; "	
		"Annuler remboursement OK? \$&lt;?ivTOTAL?&gt;,"	
Wait for	90	"Please wait..."	This prompt displays after a signature

Prompt	Prompt ID	Default Value	Description
Signature Approval		"Favor de esperar... "	is entered, if the RBA is configured to wait for the cashier to verify the signature (if 0009_0008 = 1, Display Signature Until Download parameter is enabled).
		"Un moment SVP..."	
Approval Failure Message	91	"Unable to authorize"	This prompt is displayed if the PIN pad is unable to send an authorization request to the POS.
		"Incapaz de autorizar"	
		"Incapable d'autorise"	
Signature Accepted	92	"Signature accepted"	This prompt informs the customer that the signature was accepted.
		"Firma aceptada"	
		"Signature acceptée"	
Input Accepted Prompt	93	"Input accepted"	This informs the customer that the extra input just entered was accepted.
		"Entrada aceptada"	
		"Entrée acceptée"	
Card Accepted	94	"Card accepted"	This prompt informs the customer that the card swipe was accepted.
		"Tarjeta aceptada"	
		"Carte acceptée"	
Terms Accepted Prompt	95	"Terms accepted"	This prompt confirms that the customer has accepted terms and conditions.
		"Condiciones aceptadas"	
		"Conditions acceptées"	
Terms Declined Prompt	96	"Terms declined"	This prompt confirms that the customer has declined terms and conditions.
		"Condiciones negadas"	
		"Conditions refusées"	
More Prompt	97	"- More -"	This prompt is displayed to indicate there is more available.
		"- Mas -"	
		"- Plus -"	
Card Read Error Prompt	98	"Card read error"	This prompt is displayed when a card read error occurs.
		"Error de lectura de tarjeta"	
		"Erreur de lecture"	
Accept Prompt	120	"Accept"	This prompt is displayed to confirm the customer's acceptance.
		"Aceptar"	
		"Approuvé"	
Decline Prompt	121	"Decline"	This prompt is displayed to confirm the customer's denial.
		"Denegar"	
		"Refusé"	

Prompt	Prompt ID	Default Value	Description
Hand Card to Cashier Prompt	126	"Please hand card to cashier"	This prompt asks the customer to hand the payment card to the cashier. The cashier may then enter the card number manually. It is displayed when the PIN pad is unable to read the card after the specified number of allowed bad read attempts has been reached (parameter 0003_0001, msr.dat file).
		"Por favor pase tarjeta al cajero(a) "	
		"Donnez carte au cassier SVP"	
Too Many PIN Entries	127	"Too many PIN entry errors"	This informs the customer that the transaction is being cancelled because the customer is having trouble entering a valid PIN.
		"Demasiados errores de entrada de PIN"	
		"Trop d'essais - NIP erroné"	
Card Type A – Debit	130	"Debit"	This prompt is displayed to confirm the selected payment type. The payment may be selected through a POS message, BIN range checking, or customer screen selection.
		"Debito"	
		"Débit"	
Card Type B – Credit	131	"Credit"	This prompt is displayed to confirm the selected payment type. The payment may be selected through a POS message, BIN range checking, or customer screen selection.
		"Crédito"	
		"Crédit"	
Card Type C - EBT Cash Benefit	132	"EBT Cash"	This prompt is displayed to confirm the selected payment type. The payment may be selected through a POS message, BIN range checking, or customer screen selection.
		"Efectivo EBT"	
		"Comptant - EBT"	
Card Type D - EBT Food Stamps	133	"EBT Foodstamps"	This prompt is displayed to confirm the selected payment type. The payment may be selected through a POS message, BIN range checking, or customer screen selection.
		"Estampillas EBT"	
		"Bon alimentaire - EBT"	
Card Type E - Store Charge	134	"Store Charge"	This prompt is displayed to confirm the selected payment type. The payment may be selected through a POS message, BIN range checking, or customer screen selection.
		"Cargo de la Tienda"	
		"Carte magasin"	
Card Type F – Loyalty	135	"Thank you for your loyalty"	This prompt is displayed to confirm the selected payment type. The payment may be selected through a POS message, BIN range checking, or customer screen selection.
		"Gracias por su lealtad"	
		"Merci de votre fidélité"	
Invalid Date Prompt	154	"Invalid Date"	This prompt is displayed when an invalid date is entered.
		"Fecha No Permitida"	
		"Date Nulle"	

Prompt	Prompt ID	Default Value	Description
Small Security Code Prompt	156	"Security Code too small"	This prompt is displayed when the user enters a security code that is too small.
		"Código de Seguridad demasiado pequeño"	
		"Code de Sécurité trop petit"	
Ask for Assistance	164	"Please ask for assistance"	This prompt displays when the PIN pad is unable to read the card after the specified number of allowed bad read attempts has been reached (parameter 0003_0001, msr.dat file). The cashier may then enter the card number manually.
		"Por favor pida ayuda"	
		"Demandez de l'aide SVP"	
Signature Prompt	165	"Please sign and tap Ok with pen"	This prompts the customer to enter a signature, using the electronic pen attached to the PIN pad.
		"Por favor firme y toque OK con el lápiz"	
		"Signez avec le stylo SVP"	
Slide Card Prompt	166	"Please slide card or Tap"	This prompts the customer to slide or tap their card.
		"Por favor deslice o toque su tarjeta"	
		"Glissez la carte SVP or TAP"	

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

```

<PromptDefinition>
  <Lang1>
    <!-- English -->
    <NonEntry>
      <Prompt id="0" message="PROMPTINDEX" shortmessage="PROMPTINDEX"/>
      <Prompt id="1" message="Please select language" shortmessage="Please select language"/>
      <Prompt id="2" message="Processing... please wait" shortmessage="Processing... please wait"/>
      <Prompt id="3" message="Please slide card" shortmessage="Please slide card"/>
      <Prompt id="4" message="Expired card. Please use another." shortmessage="Expired card. Use another."/>
      <Prompt id="5" message="Card read error. Try again." shortmessage="Card read error. Try again."/>
      <Prompt id="6" message="Please select payment type" shortmessage="Please select payment type"/>
      <Prompt id="7" message="Please wait for the cashier" shortmessage="Please wait for the cashier"/>
      <Prompt id="8" message="Cashback correct? $<?vCASHBACK?>" shortmessage="Cashback correct? $<?vCASHBACK?>"/>
      <Prompt id="9" message="Not a sale. Cashback cancelled." shortmessage="Not a sale. Cashback cancelled."/>
      <Prompt id="10" message="Amount OK? $<?vTOTAL?>" shortmessage="Amount OK? $<?vTOTAL?>"/>
      <Prompt id="11" message="Processing... please wait" shortmessage="Processing... please wait"/>
      <Prompt id="12" message="Invalid card for payment type" shortmessage="Invalid card payment type"/>
      <Prompt id="14" message="PIN must be 4 to 12 digits" shortmessage="PIN must be 4 to 12 digits"/>
      <Prompt id="15" message="Cashback?" shortmessage="Cashback?"/>
      <Prompt id="17" message="Cashback limit is $<?vCB_MAX_TEXT?>" shortmessage="Cashback limit is $<?vCB_MAX_TEXT?>"/>
      <Prompt id="18" message="Invalid amount. $<?vCASHBACK?>" shortmessage="Invalid amount. $<?vCASHBACK?>"/>
      <Prompt id="19" message="Amount must be less than $<?vAMOUNT?>" shortmessage="Amount must be less than $<?vAMOUNT?>"/>
    
```

Figure 11 - the default PROMPT.xml file.

### 17.2.1. Button Text

Text displayed on dynamically generated buttons is specified in the PROMPT.XML file. The button text is cross referenced with the Key Press Values in section 5.2 using the Button ID value.

Prompt	Prompt ID	Default Value	Description
Accept	101	"Accept"	
		"Aceptar"	
		"Approuvé"	

Prompt	Prompt ID	Default Value	Description
Cashback	102	“Cashback”	
		“Cashback”	
		“Cashback”	
Clear	103	“Clear”	
		“Borrar”	
		“Claire”	
Credit	104	“Credit”	
		“Crédito”	
		“Crédit”	
Debit	105	“Debit”	
		“Débito”	
		“Débit”	
Decline	106	“Decline”	
		“Negada”	
		“Refusé”	
EBT Cash	107	“EBT Cash”	
		“Efectivo”	
		“Trésorerie EBT”	
EBT Food	108	“EBT Food”	
		“Comida”	
		“Alimentaire EBT”	
English	109	“English”	
		“English”	
		“English”	
Enter Card	110	“Enter Card”	
		“Entre Tarjeta”	
		“Entrer Carte”	
Español	111	“Español”	
		“Español”	
		“Español”	
Français	112	“Français”	
		“Français”	
		“Français”	
No	113	“No”	
		“No”	
		“Pas”	

Prompt	Prompt ID	Default Value	Description
Ok	114	“Ok”	
		“Ok”	
		“Ok”	
Other	115	“Other”	
		“Otro”	
		“D'autres”	
Partial Payment	116	“Partial Payment”	
		“Pague Menos”	
		“Paiement Partiel”	
Store	117	“Store”	
		“Tienda”	
		“Magasin”	
Yes	118	“Yes”	
		“Si”	
		“Oui”	

### 17.3. Custom Prompts (CUSTPROMPTS.xml)

Ingenico may provide custom financial application prompts in English, Spanish and French in an optional CUSTPROMPT.xml file based on customer requests. Each customer's CUSTPROMPT.xml file is unique. The CUSTPROMPT.xml file must be packaged as a signed PGZ file before it can be loaded on an Ingenico PIN pad.

Prompt text display properties (such as font type, font color or text justification) are described by the text element used to call the prompt in a given form.

**Info** To comply with PCI-DSS requirements, CUSTPROMPT.xml is subject to security restrictions that prohibit the display of prompts containing character combinations representing the words “PIN” and “NIP.”

## Notes



# Appendix A. Differences Between U32 UPOS and Telium UPOS

This Appendix lists differences between the latest versions of the U32 and Telium UPOS controls and the associated OPOS and JPOS Integration Kits.

## A.1. At A Glance

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

Feature	U32 UPOS	Telium UPOS
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 (OPOS) or interfaces file (JPOS)	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. UPOS Control Support

### A.2.1. Form Control

The Telium implementation has removed support for the Form Control (IVICMForm.ocx) in OPOS and the interfaces file in JPOS (ijpos113.jar or ijpos113\_ext.jar).

### A.2.2. Direct I/O (OPOS and JPOS)

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 the corresponding chapters for OPOS and JPOS.

## A.3. Control Panels

---

### A.3.1. 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 3.4 OPOS Configuration.

#### A.3.1.1. 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 37 for more information.

#### A.3.1.2. 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.3.1.3. 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.3.1.4. MSR Tab

Changes in the Telium implementation include the following:

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

#### A.3.1.5. 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.3.1.6. 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.3.1.7. Debug Tab

Changes in the Telium implementation include the following:

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

### A.4. Test Applications

---

The Telium release of the UPOS Integration Kit added the OPOS and JPOS Test Applications, which allow customers to drive PIN pad behavior using UPOS controls.

For more information on the JPOS Test Application, see JPOS Test Application on page 191.

For more information on the OPOS Test Application, see OPOS Test Application on page 199.

## 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

---

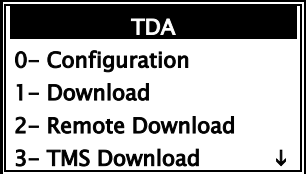
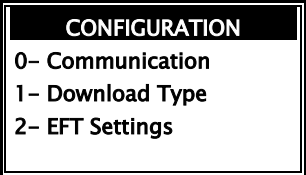
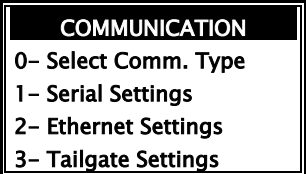
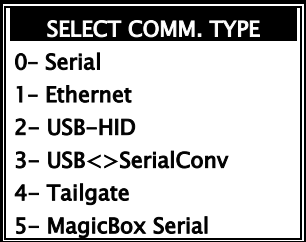

To access the Telium Download Application on your Ingenico PIN pad, follow these steps:

1. Power up the Ingenico PIN pad.
2. Wait for the gray UPOS Interface Application splash screen (see Figure 3 on page 19).
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

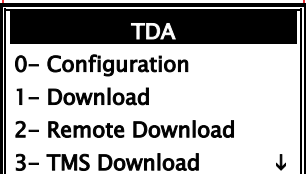
### B.2.1. Setting Communication Type

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 [o] 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 (0x34) Display Text At on page 79 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.

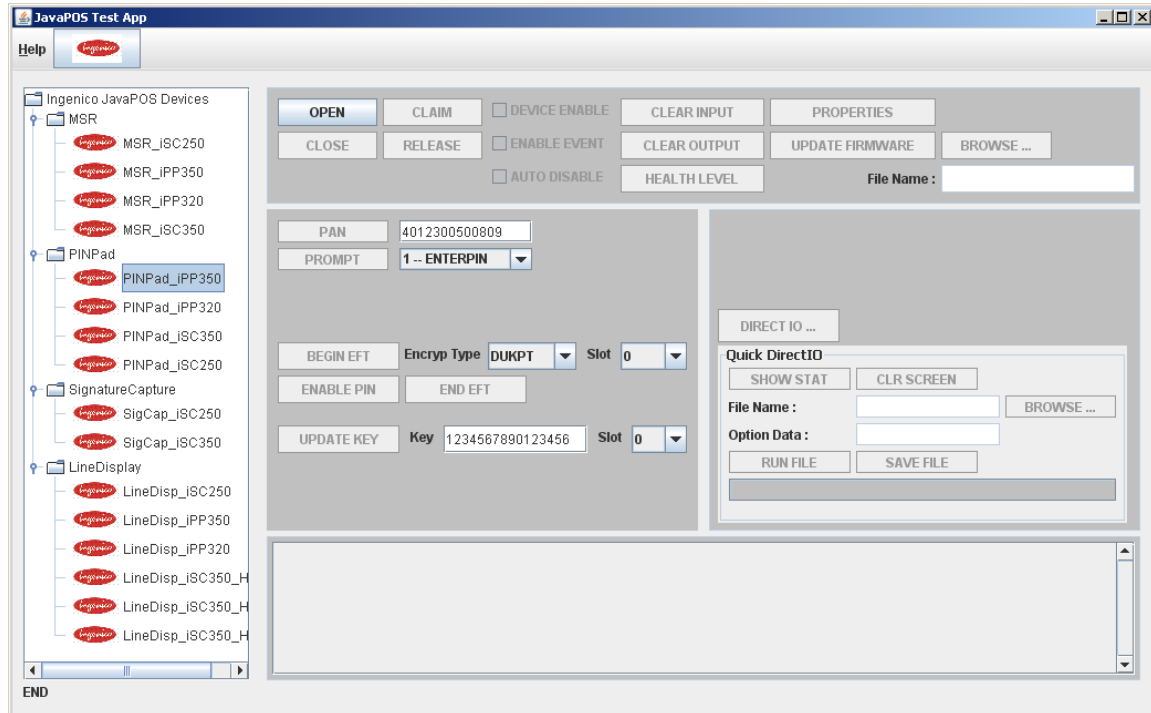


Figure 12 - The Ingenico JavaPOS test application.

### 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 26 for more information on setting the JAVA\_HOME path.

### D.2. Updating JPOS Configuration

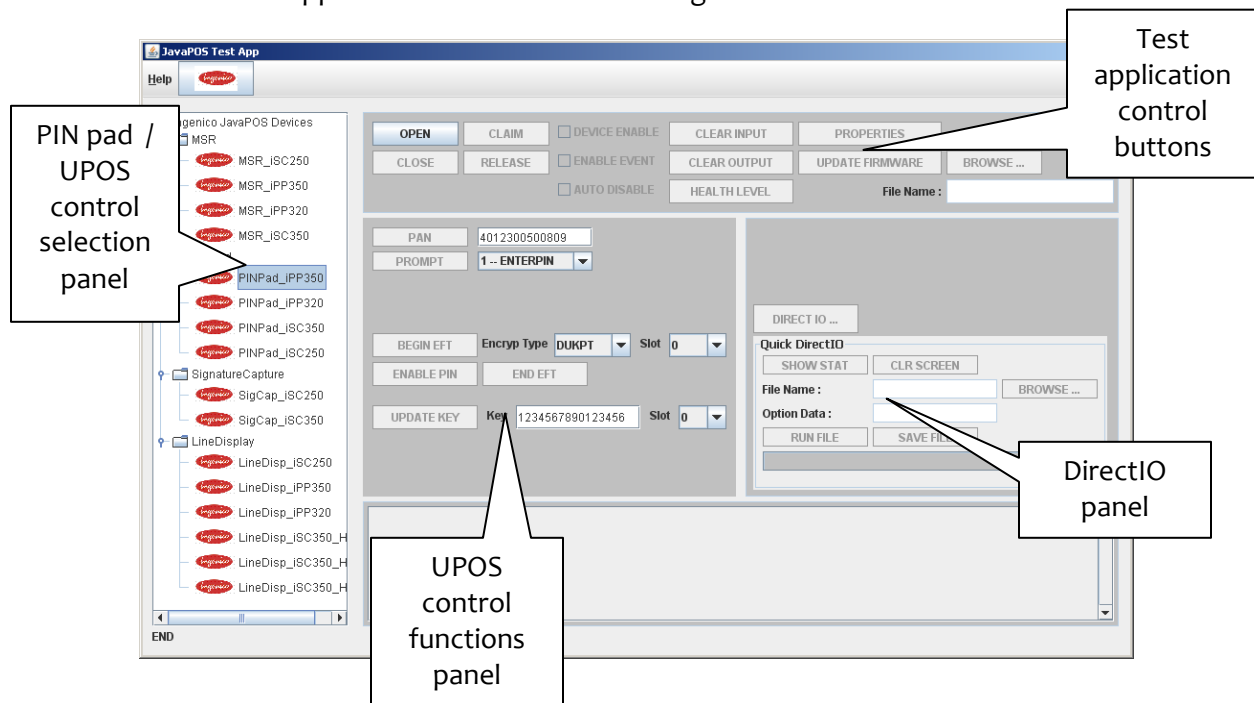
Open Jpos configuration file, ijpos.xml, with a notepad or xml editor. Use the following table to determine connection interface specific properties to be updated.

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>

Interface Type	Properties
	<pre>&lt;prop name="baudRate" type="String" value="115200" /&gt;</pre>
USB CDC	<pre>&lt;prop name="portName" type="String" value="COM5" /&gt;</pre> <p><i>info 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).</i></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>

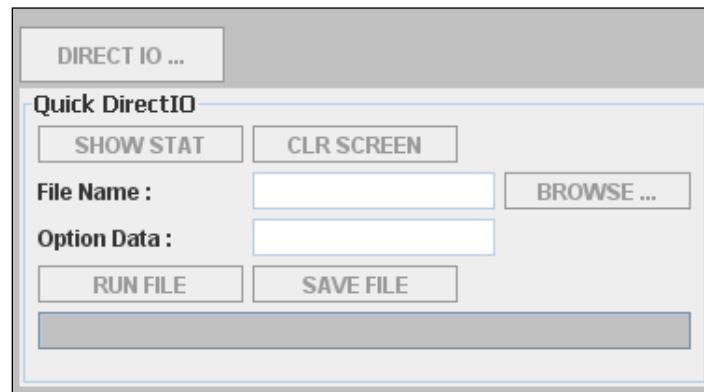
### D.3. The JavaPOS Test Application Interface

The JavaPOS Test Application contains the following:



### D.4. Direct I/O

The JavaPOS test application allows users to send DirectIO commands to UIA-equipped Telium PIN pads.

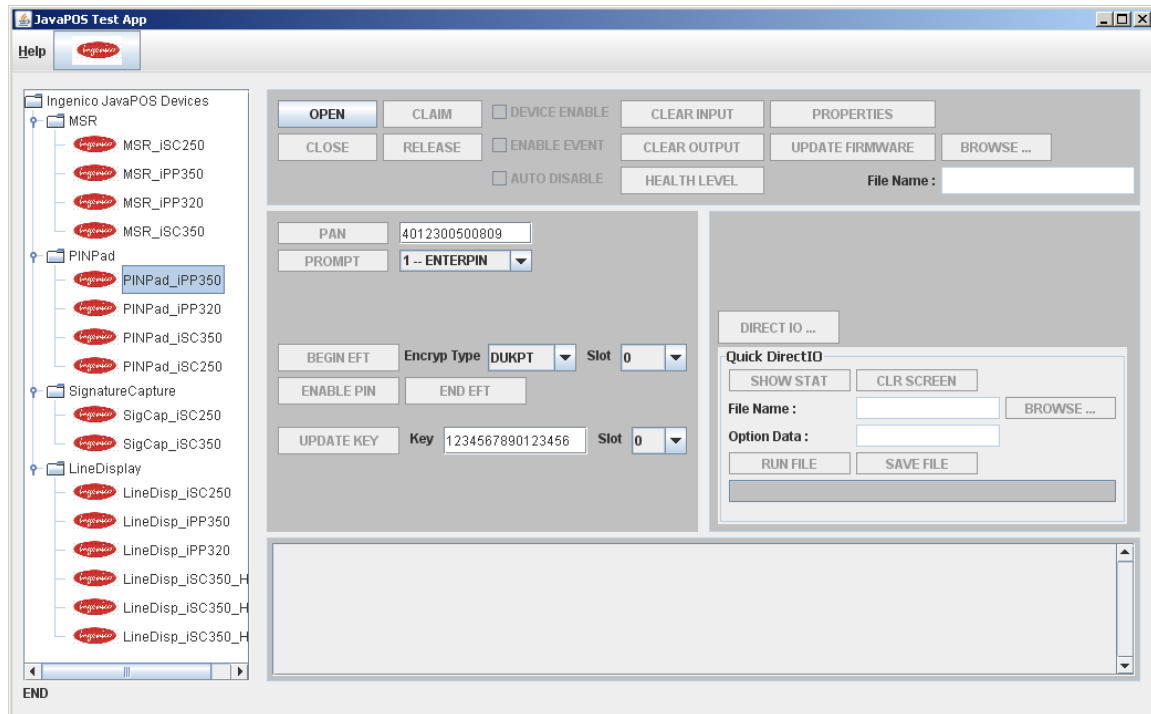


**Figure 13 - The JPOS test application's DirectIO panel.**

The following DirectIO commands are supported by the JPOS Test Application:

- CLEAR\_SCREEN=0x31
- CLEAR\_LD=0x30
- GET\_HEALTH\_STAT=0x0D
- GET\_UIA\_VARIABLE=0xA2
- RUN\_FILE=0x92
- SAVE\_FILE=0x91
- SET\_UIA\_VARIABLE=0xA1
- POSITION\_TEXT\_CURSOR=0x36
- GET\_CHECKBOX=0x12
- GET\_RADIO=0x13
- SEND\_RAW\_DATA=0x00
- DEVICE\_RESET=0x09
- RETRIEVE\_FILE=0x94
- FILE\_STATUS=0x95
- DISPLAY\_TEXT\_AT=0x34
- DEVICE\_LIGHT\_CONTROL=0xA6
- DISPLAY\_TEXT=0x32
- DELETE\_FILE=0x93

## D.5. Running the JPOS Test App



**Figure 14 - 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.

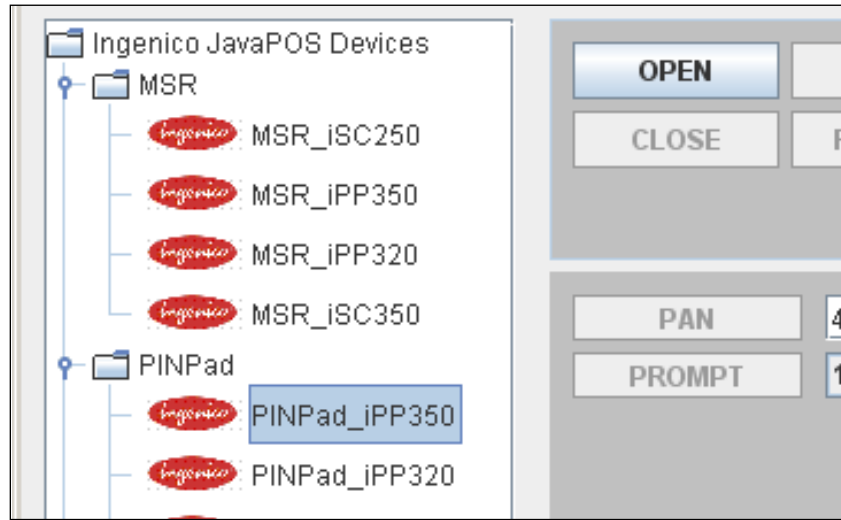
## D.6. Working With the Test Application

### D.6.1. Opening, Claiming and Enabling a Control

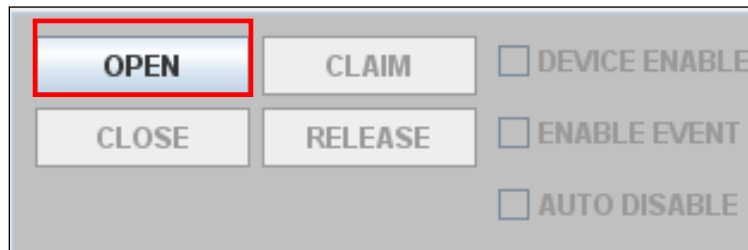
Before using the JPOS test application to send a command, you must first Open, Claim and enable the associated device.

To open, claim and enable a JPOS control, follow these steps:

1. Select the desired control for the desired PIN pad from the panel on the left side of the test application.



2. Click the **OPEN** button. This opens the control and enables the CLAIM button.



3. Click the **CLAIM** button. This claims the control and enables the remaining checkboxes and buttons.
4. Check the box labeled **DEVICE ENABLE**.

The JPOS control is activated and the PIN pad is now ready to receive commands.

To close a control that is currently open, click the CLOSE button in the test app's control button section.



Figure 15 - the Close button.

### D.6.2. Sending Direct I/O Commands

The JavaPOS test application can be used to issue DirectIO commands in order to drive the corresponding functionality in Telium terminals.

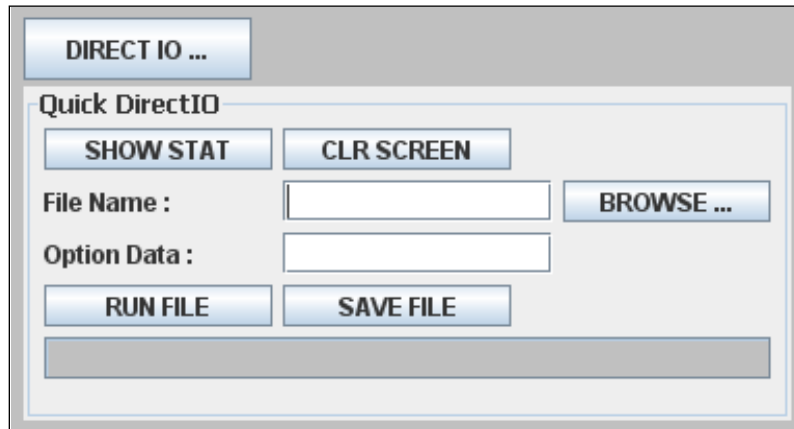
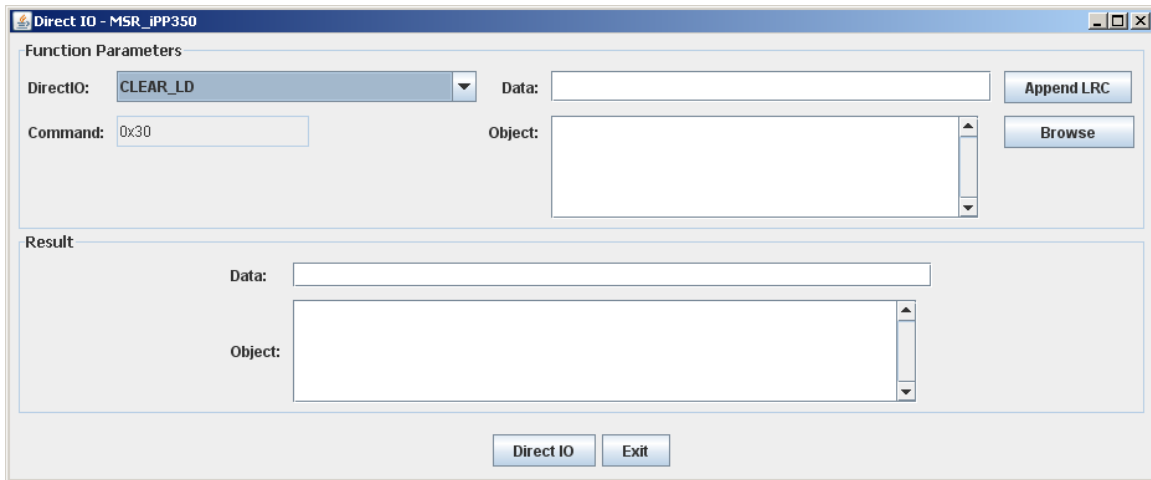


Figure 16 - The JPOS test application's DirectIO panel.

To send a DirectIO command from the JavaPOS Test Application, follow these steps:

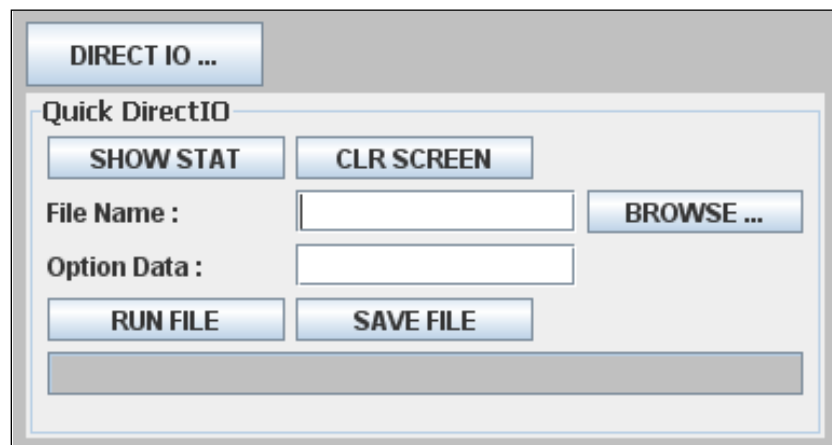
1. Make sure you have opened, claimed, and enabled the desired control (see Opening, Claiming and Enabling a Control on page 195 for more information).
2. If you wish to view the PIN pad's Health Stats, clear the screen, run a file or save a file, see Using the Quick DirectIO Panel on page 197. Otherwise, continue to the next step.
3. Click the **Direct IO...** button in the test application's DirectIO panel. The Direct IO window displays.



4. Select the desired command from the **DirectIO** drop-down menu.
5. If needed, type in numeric data to send with the command in the **Data** field.
6. If needed, select an **Object** by typing in the object path or clicking **Browse** and browsing to the desired file.
7. Specify any additional command-specific settings displayed at the bottom of the DirectIO panel.
8. Click **DirectIO** to send the command.

### D.6.3. Using the Quick DirectIO Panel

JavaPOS test application users also have access to the **Quick DirectIO** panel which provides shortcuts to frequently-used commands.



*Figure 17 - The JPOS test application's DirectIO panel.*

From the Quick DirectIO panel, users can:

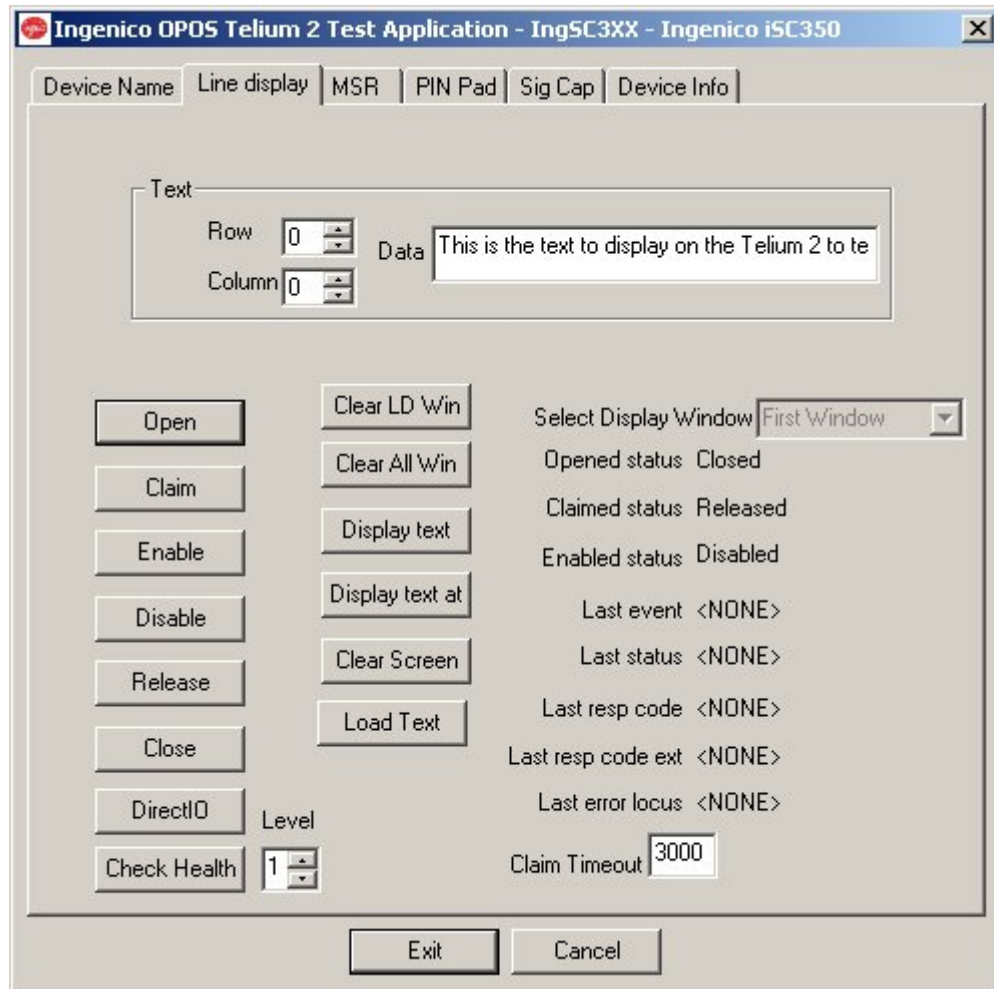
- Request the PIN pad's Health Statistics by pressing the **SHOW STAT** button.
- Clear the PIN pad's screen by pressing the **CLR SCREEN** button.

- Load a file onto the PIN pad by clicking **BROWSE...** to navigate to the file path and then clicking the **SAVE FILE** button.
- Run a file found on the PIN pad by typing the path in the **File Name** field and clicking the **RUN FILE** button.



## 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 18 - 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.

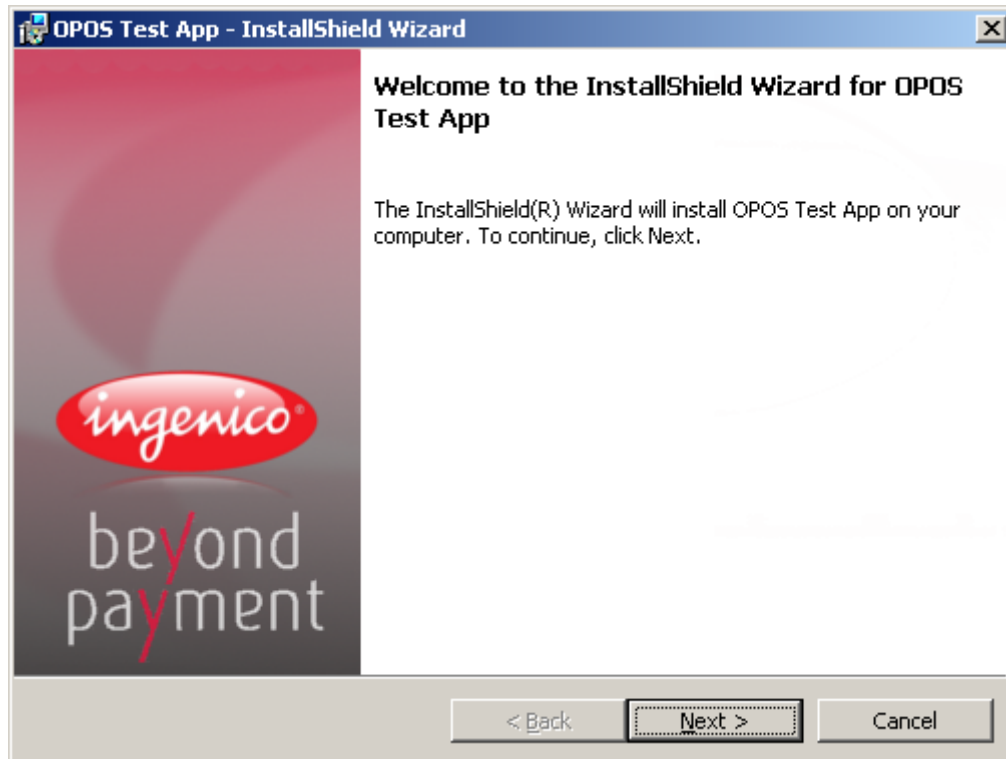
Users of the OPOS Integration Kit can find the OPOS Test Application in the Utilities folder included in the integration kit package.

## 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.



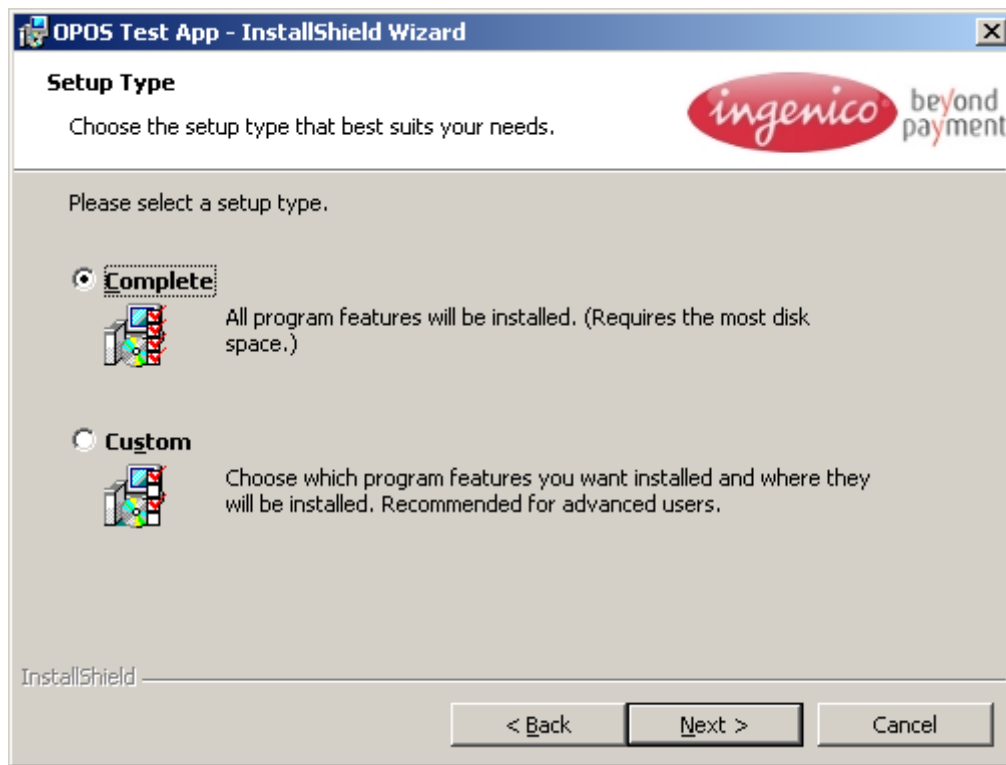
2. Click **Next**.



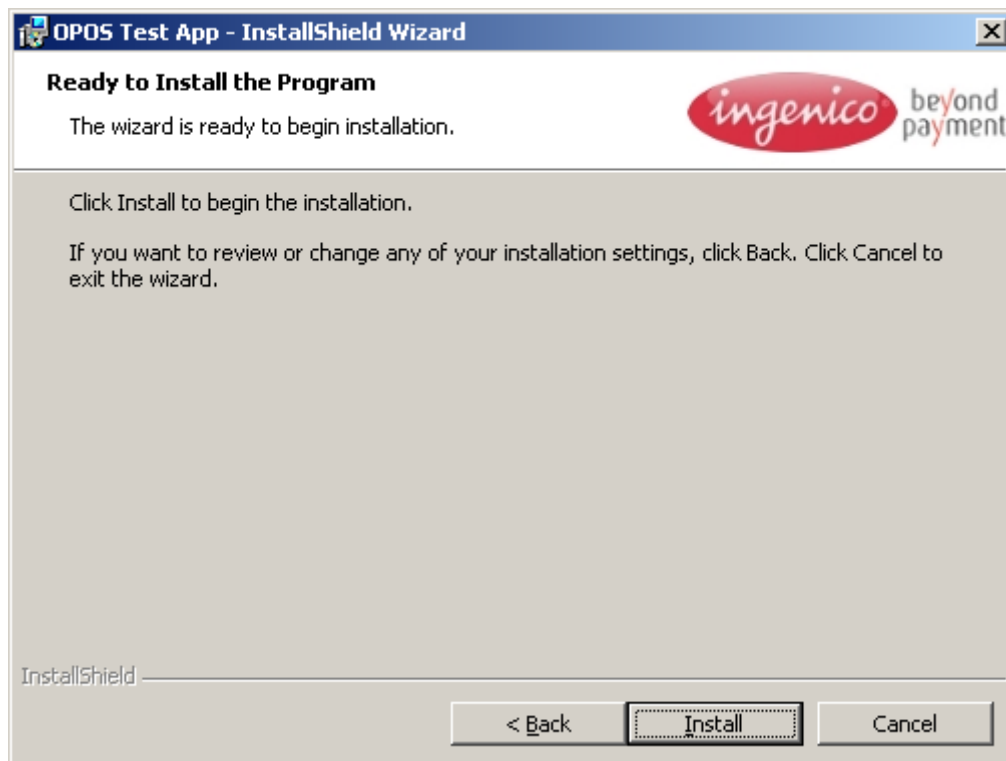
3. Select the radio button next to **I accept the terms in the license agreement** and click **Next**.



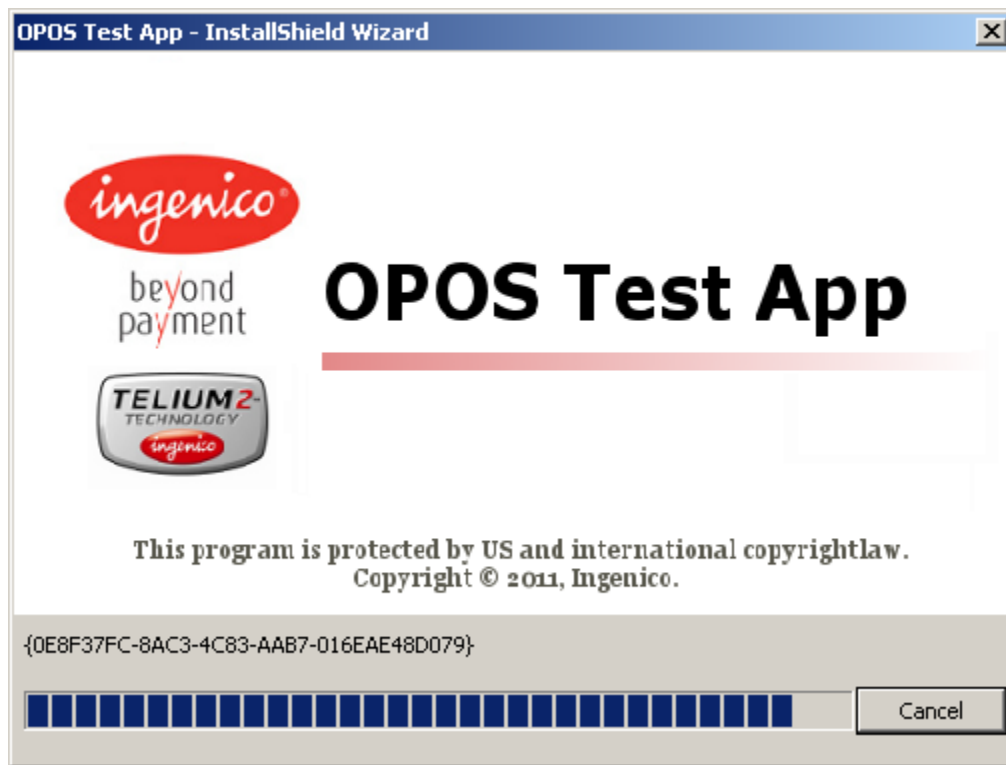
4. Type in the **User Name** and the name of the **Organization** that will be using the OPOS Test Application, then click **Next**.



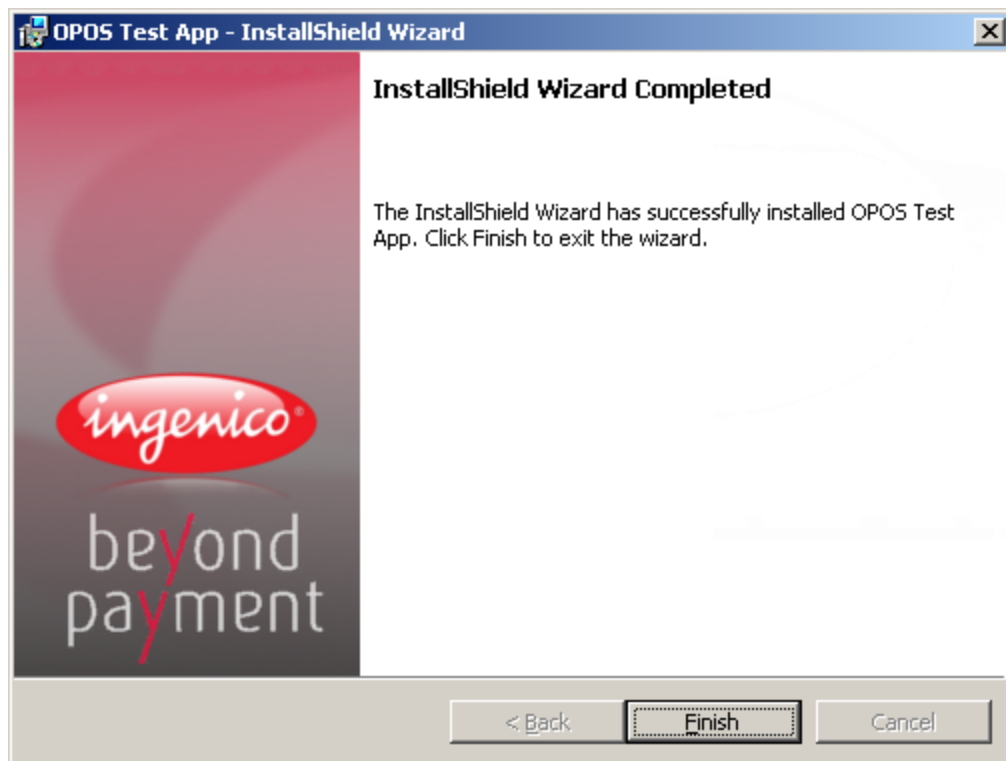
5. Click the radio button next to **Complete**, then click **Next**.



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

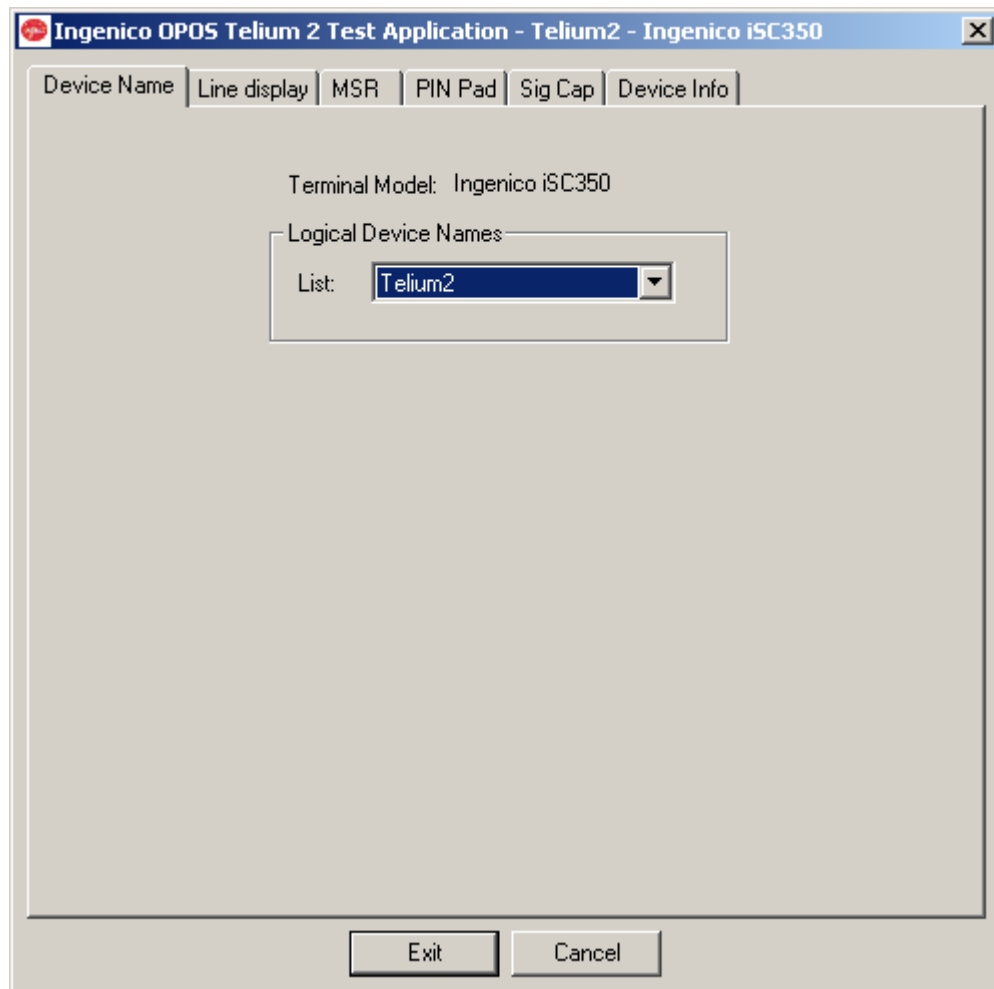
---

### 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 38 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.

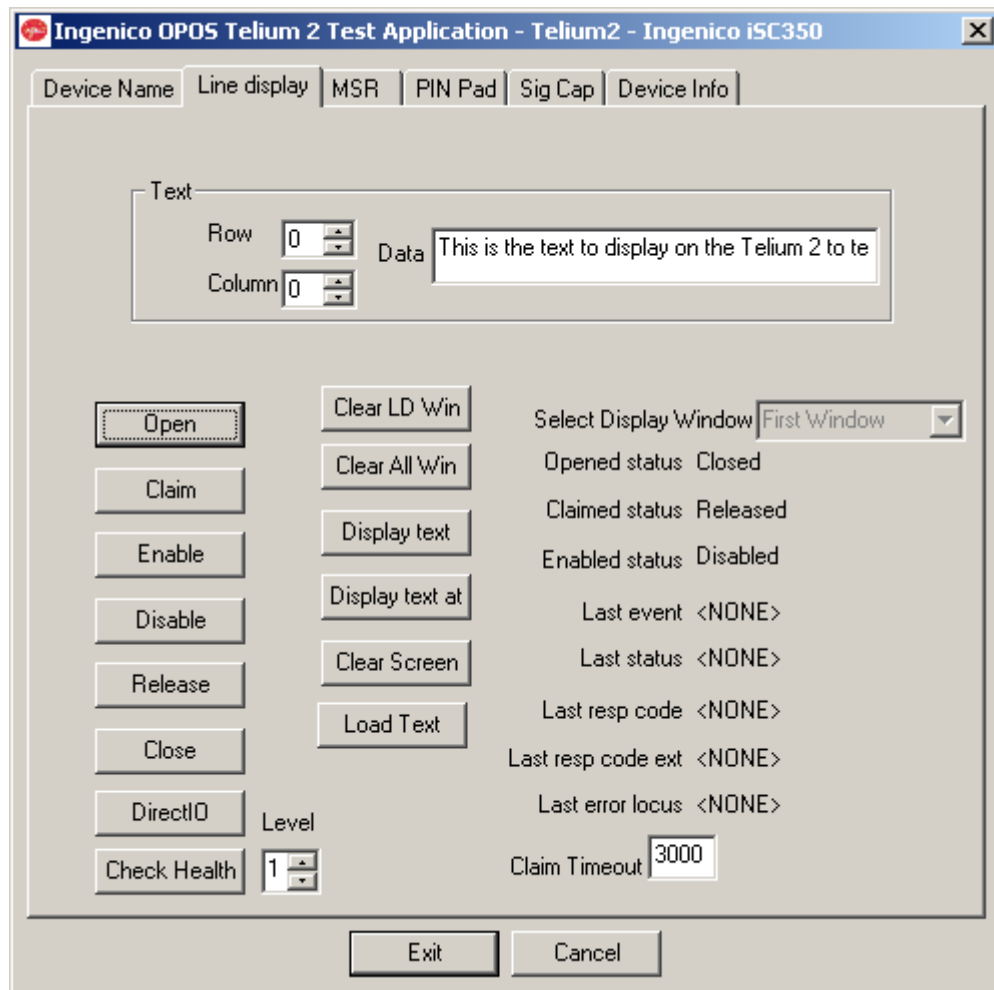


Figure 19 - the Line Display tab.

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 20 - 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.

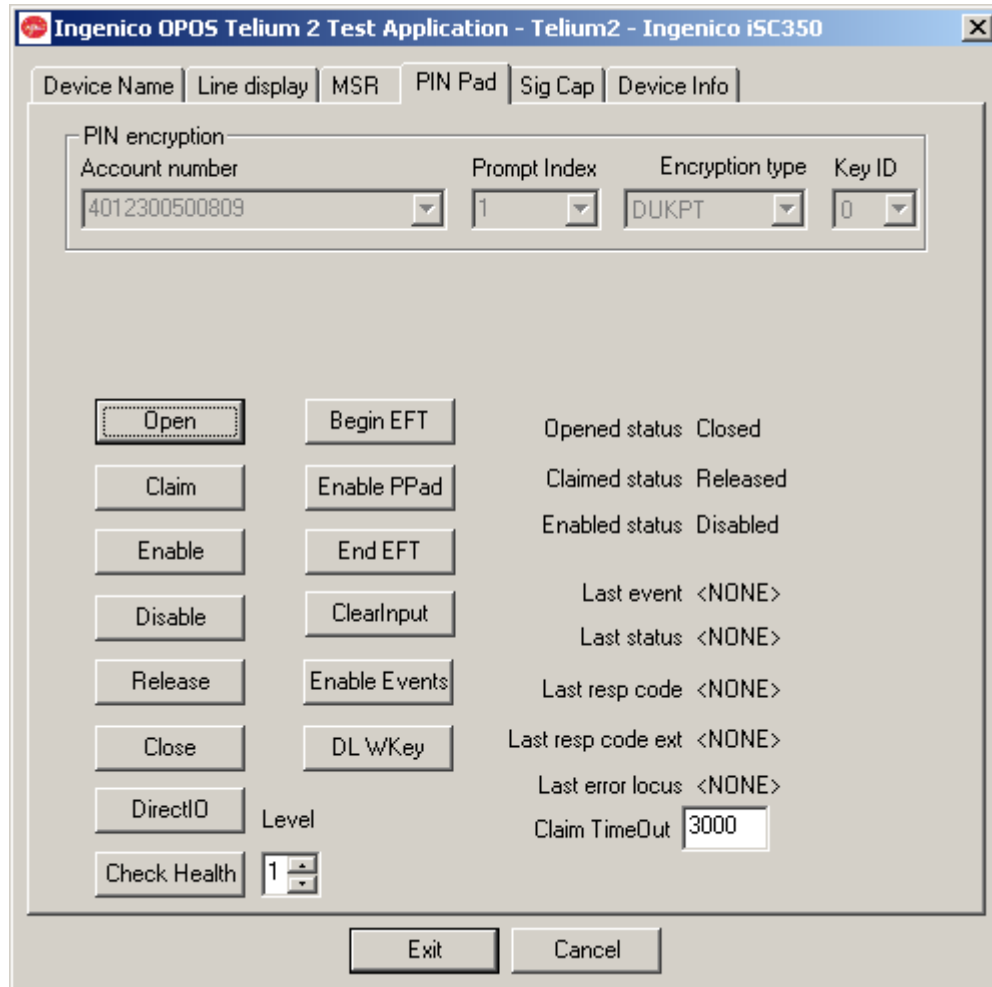


Figure 21 - the PIN Pad tab.

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 'Ingenico OPOS Test App Version 1.4.7.2 - Telium2 - Ingenico iSC350' window. The 'Sig Cap' tab is selected. The interface includes a 'Signature preview' area, a 'Signature format' section with 'Use OPOS' selected, a 'Form Value' text box, an 'OPOS conversion' dropdown set to 'OPOS\_BC\_NIBBLE', and a 'SigCap ButtonId' section with various status fields. On the left, there are checkboxes for 'Show signature details', 'Show number of sig points', and 'Real Time Signature Capture', followed by buttons for 'Open', 'Claim', 'Begin capture', 'End capture', 'Get Data', 'ClearInput', 'Disable', 'Release', 'Close', 'Direct IO', 'Check Health', and a 'Level' spinner set to 1. At the bottom are 'Exit' and 'Cancel' buttons.

Figure 22 - 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
Enabled DataEvent	Check prior to starting signature capture to get the signature data to draw signature in the signature box. The signature control must be in OPEN state to select this check box.
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.

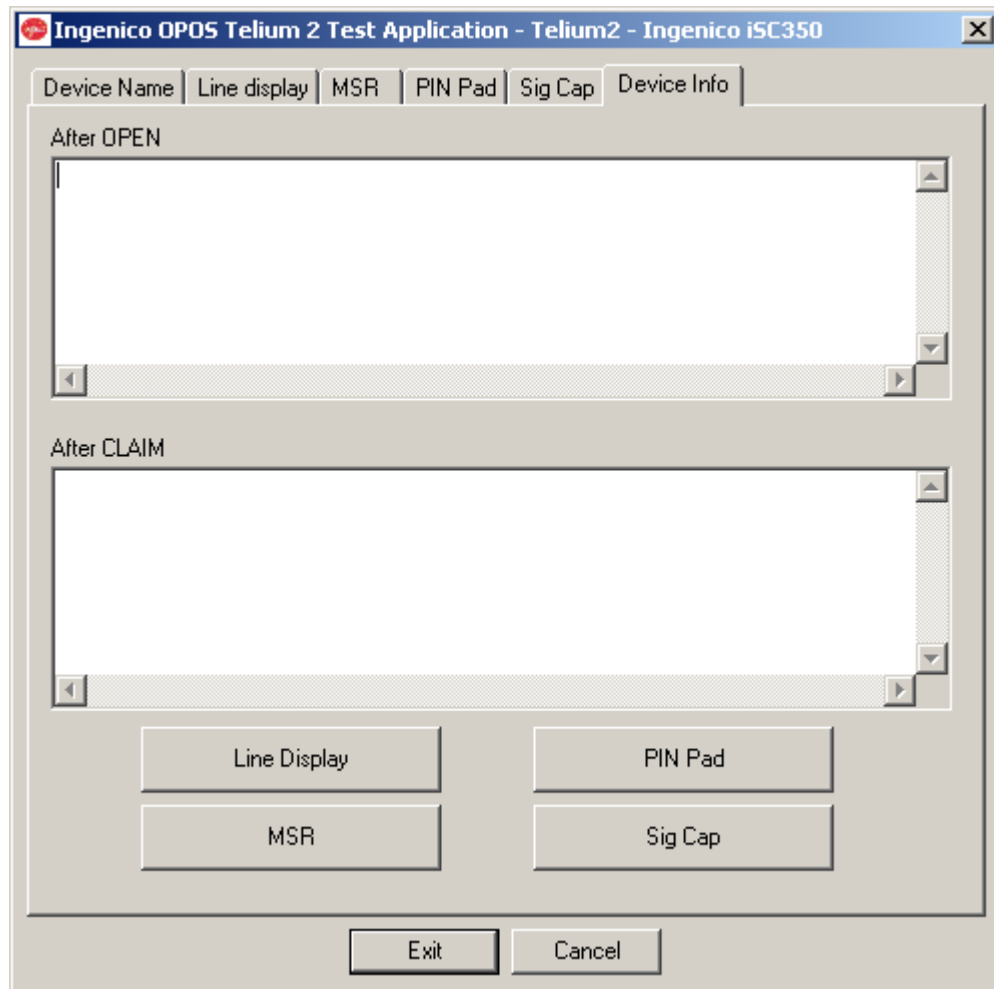


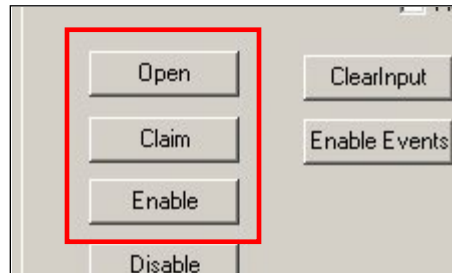
Figure 23 - the Device Info tab.

## E.3. Working With the Test Application

---

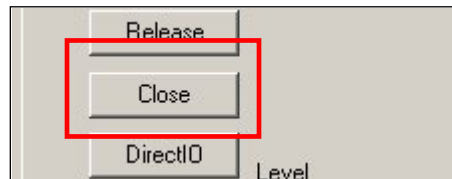
### E.3.1. Opening, Claiming and Enabling a Control

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 24 - Open, Claim, and Enable buttons on the MSR tab.*

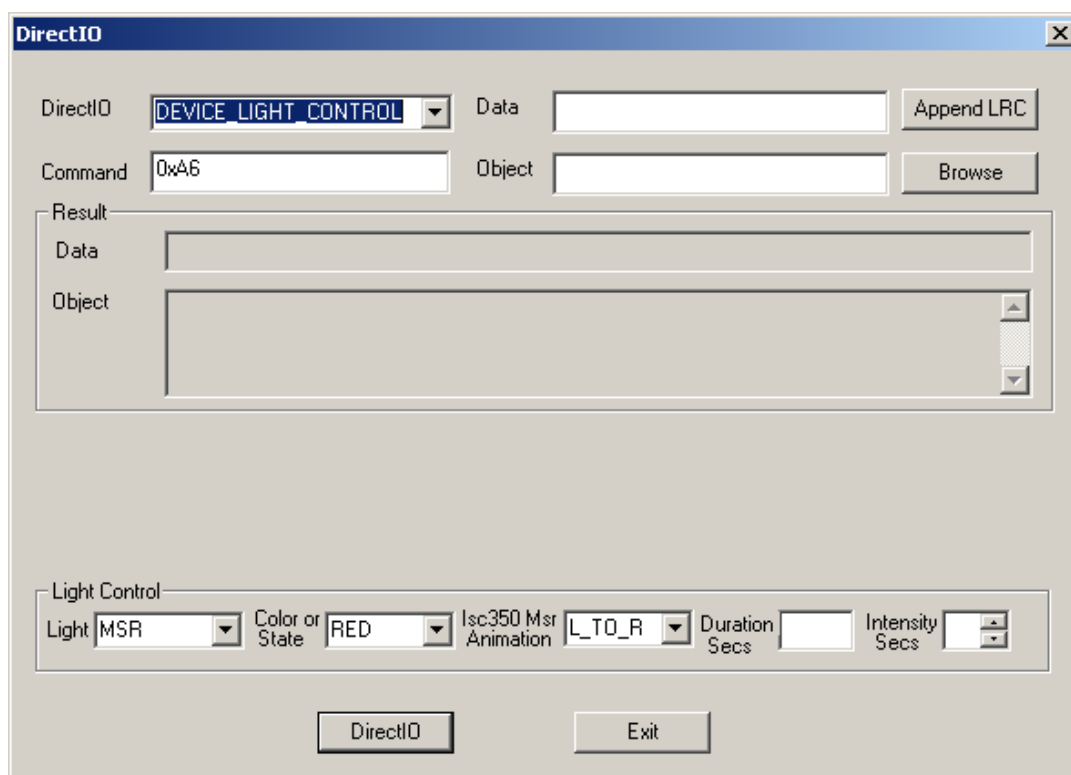
If you wish to use the test application to send a command to a different UPOS control, you must first Close the one that is currently open by clicking the Close button on the corresponding tab.



*Figure 25 - the Close button on the MSR tab.*

### E.3.2. Sending Direct I/O Commands

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 26 - 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. If needed, type in data to send with the command in the **Data** field.
3. If needed, 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.

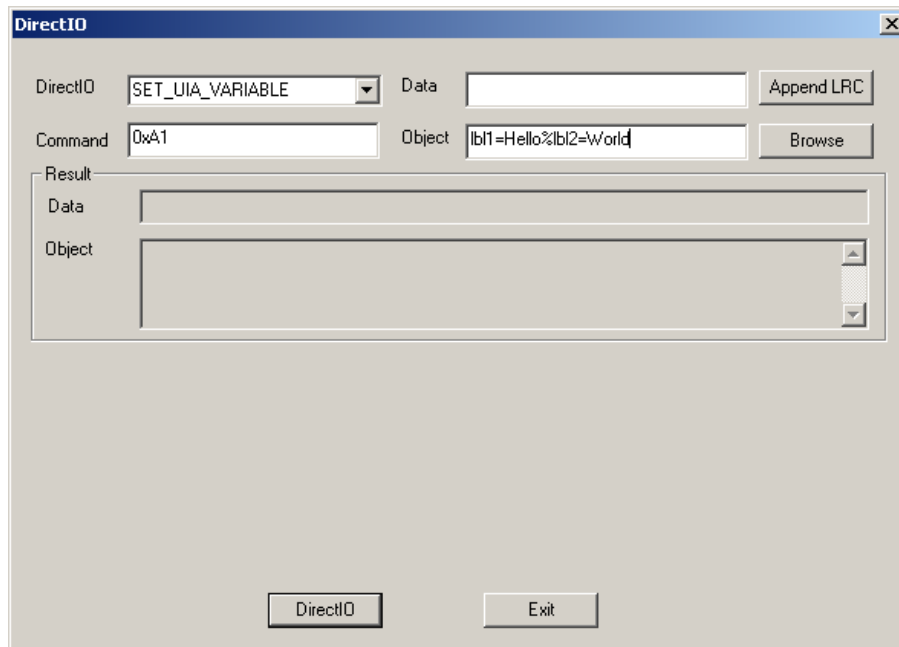
### **E.3.3. Changing Variable Text**

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.3.4. Changing Prompt Indices

OPOS test application users can easily change Telium PIN pad prompt indices 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 prompt index, 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: `<prompt1>=<prompt_index1>%<prompt2>=<prompt_index2>`, where the prompt indices correspond to pre-defined PIN pad prompts defined in PROMPTS.xml. Users can use the SET\_UIA\_VARIABLE command to update multiple prompts variables by separating variable update commands using the “%” character.
4. Click the **DirectIO** button to execute the command. The new prompt indices are stored in the PIN pad’s memory until the device is rebooted. Any forms using the updated prompt indices will display the corresponding prompts.

### E.3.5. Displaying Forms

OPOS test application users can easily display any standard or custom \*.K3Z form files stored on the PIN pad using the DirectIO “RUN\_FILE” command.

To display a form, follow these steps:

1. From the OPOS Test application, click the DirectIO button on the tab corresponding to any active UPOS control.
2. Select RUN\_FILE from the DirectIO drop-down menu.
3. Type the number “2” in the Data field.
4. Type the file name of the form you wish to display in the Object field.
5. Click the DirectIO button to execute the command. The PIN pad displays the requested form.

### E.3.6. Configuring MSR LEDs With The Test Application

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

To configure an iSC350’s MSR LEDs, 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 screenshot shows the DirectIO dialog box with the following configuration:

- DirectIO** dropdown: DEVICE\_LIGHT\_CONTROL
- Data** field: (empty)
- Append LRC** button: (disabled)
- Command** field: 0xA6
- Object** field: (empty)
- Browse** button: (disabled)
- Result** section:
  - Data** field: (empty)
  - Object** field: (empty)
- Light Control** section:
  - Light** dropdown: MSR
  - Color or State** dropdown: RED
  - Isc350 Msr Animation** dropdown: L\_TO\_R
  - Duration Secs** field: (empty)
  - Intensity Secs** field: (empty)
- Buttons**: DirectIO, Exit

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 8: 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 9: 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

# Appendix G. Public Software License

## Acknowledgments

---

Copyright (c) 1999, 2003, 2005 Kungliga Tekniska Högskolan  
(Royal Institute of Technology, Stockholm, Sweden).  
All rights reserved.

*Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:*

1. *Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.*
2. *Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.*
3. *Neither the name of KTH nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.*

*THIS SOFTWARE IS PROVIDED BY KTH AND ITS CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL KTH OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.*

## Notes