ZEBRA SCANNER OPOS DRIVER DEVELOPER'S GUIDE

ZEBRA SCANNER OPOS DRIVER DEVELOPER'S GUIDE

72E-149783-06

Revision B

January 2020

Revision History

Changes to the original guide are listed below:

Change	Date	Description
-01 Rev A	5/2011	Initial release.
-02 Rev A	2/2012	64 Bit Updates.
-03 Rev A	4/2014	Added scale related information.
-04 Rev A	3/2015	Zebra Re-branding
-05 Rev A	5/2016	Software Re-branding
-06 Rev A	2/2019	Added: - IBM Table-top - Direct IO, CheckHealth and Registry key information.
-06 Rev B	1/2020	Replaced the IBM USB Hand-held with Full Scan Disable and IBM USB Table-top barcodes on page 2-2.

TABLE OF CONTENTS

ABOUT THIS GUIDE

Introduction

This guide provides information about the Zebra OPOS Driver which enables bar code data communication between any scanner and an OPOS compliant POS application via either a USB (IBM Hand-held and SNAPI) or RS-232 (Wincor-Nixdorf Mode B and SSI) connection.

The Zebra OPOS Driver also enables weight data communication between an MP6200 scanner and an OPOS compliant POS application via either a USB (IBM Hand-held, IBM Table-top and SNAPI) or RS-232 (SSI) connection.

Chapter Descriptions

Topics covered in this guide are as follows:

- *Chapter 1, INTRODUCTION TO THE ZEBRA SCANNER OPOS DRIVER* provides an overview of the Zebra OPOS Driver.
- *Chapter 2, INSTALLATION & CONFIGURATION* describes specific installation instructions and settings to configure the Zebra Scanner OPOS Driver on a host computer.
- *Chapter 3, OPOS PROPERTIES, METHODS, EVENTS* provides information about the Zebra OPOS Driver properties, methods, and events.
- *Chapter 4, SCANNER OPOS SAMPLE APPLICATION* provides information about the Zebra OPOS Driver properties, methods, and events.
- *Chapter 5, SCALE OPOS SAMPLE APPLICATION* demonstrates all the OPOS operations available with a connected Zebra scale.
- *Chapter 6, SUPPORTED SYMBOLOGY TYPES VS. SCANNER MODE* provides information about the sample application in the Zebra Scanner OPOS Driver suite.

Notational Conventions

The following conventions are used in this document:

- Courier New font is used for code segments.
- *Italics* are used to highlight:
  - Chapters and sections in this and related documents
  - Dialog box, window and screen names
  - Drop-down list and list box names
  - Screen field names
  - Check box and radio button names
  - File names
  - Directory names.
- **Bold** text is used to highlight:
  - Parameter and option names
  - Icons on a screen
  - Key names on a keypad
  - Button names on a screen.
- bullets (•) indicate:
  - Action items
  - Lists of alternatives
  - Lists of required steps that are not necessarily sequential
- Sequential lists (e.g., those that describe step-by-step procedures) appear as numbered lists.
- Notes, caution and warning statements appear as follows:

✓ ***NOTE*** This symbol indicates something of special interest or importance to the reader. Failure to read the note will not result in physical harm to the reader, equipment or data.

⚠ ***CAUTION*** This symbol indicates that if this information is ignored, the possibility of data or material damage may occur.

⚠ ***WARNING!*** **This symbol indicates that if this information is ignored the possibility that serious personal injury may occur.**

Service Information

If you have a problem using the equipment, contact your facility's technical or systems support. If there is a problem with the equipment, they will contact the Zebra Global Customer Support Center at: www.zebra.com/support.

CHAPTER 1 INTRODUCTION TO THE ZEBRA SCANNER OPOS DRIVER

Overview

The Zebra OPOS Driver enables bar code data communication between any scanner and an OPOS compliant POS application via either a USB (IBM Hand-held, IBM Table-top, and SNAPI) or RS-232 (Wincor-Nixdorf Mode B and SSI) connection. This driver provides an interface for reading bar code data, receiving events, opening, claiming, and enabling/disabling the device in accordance with the UPOS committee's version 1.12 specification. The OPOS specification defines a two-layer open-driver software architecture between a POS application running on a Microsoft Windows operating system and the physical POS hardware device.

- The upper layer, known as the Common Control Object (CCO), is an ActiveX Control that the POS application uses to interact indirectly with the Zebra scanner. The CCO is unique to each POS device class (e.g., scanners, scales, printers) and is provided by the UPOS committee. To simplify development and integration, the Zebra Scanner SDK includes a Monroe CCO (a vendor-independent scanner CCO). The control object (.ocx) file is located in the corresponding scanner or scale \Bin folder.
- The lower layer, known as the Service Object (SO), is an in-process OLE Automation Server (a DLL) and is used to interact directly with the POS device, in this case the scanner. The Service Object is unique to each specific vendor's POS device. The Zebra Scanner SDK contains the Zebra specific SO.

The Zebra OPOS Driver also enables weight data communication between a scale equipped MP6200 scanner and an OPOS compliant POS application via either a USB (IBM Table-top and SNAPI) or RS-232 (SSI) connection. This driver provides an interface for reading weight data, receiving events, opening, claiming, and enabling/disabling the device in accordance with the UPOS committee's version 1.13 specification.

Zebra Scanner OPOS Driver Architecture

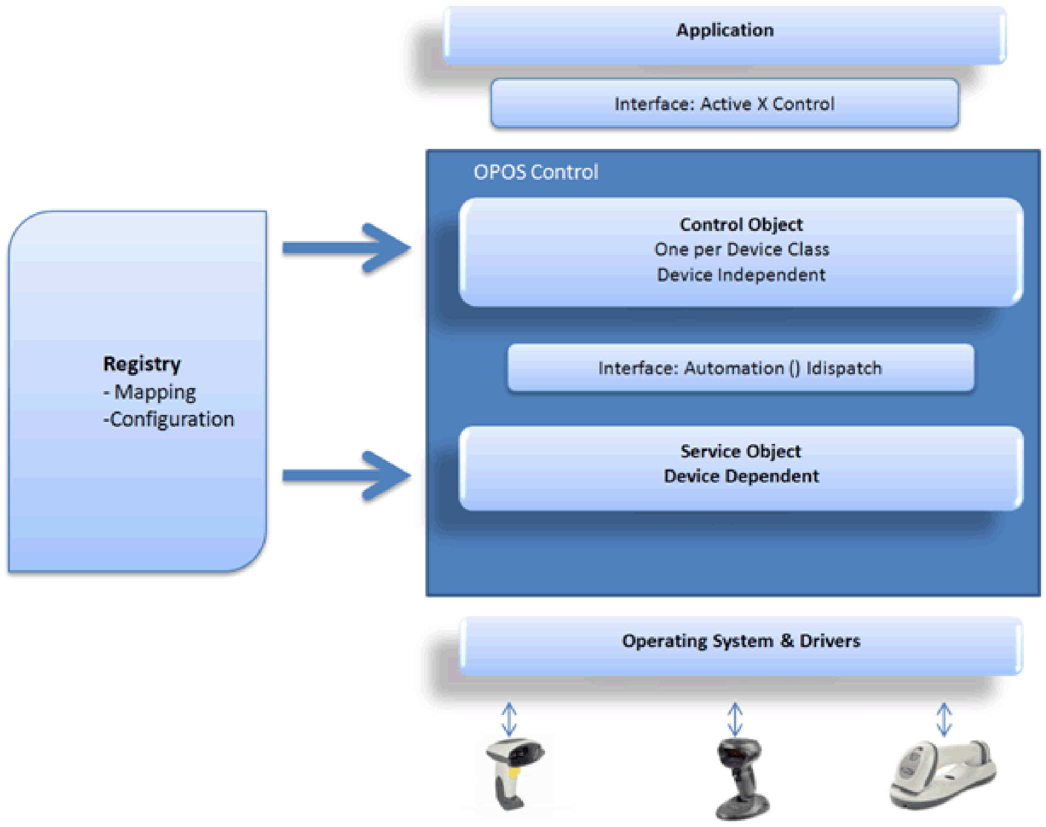


Figure 1-1 *Zebra Scanner OPOS Driver Architecture*

For more information about OPOS, OPOS architecture, terminology, and programmer's guides, refer to:

- UPOS Home Page (http://www.nrf-arts.org).
- Monroe Consulting Services, Inc. (http://www.monroecs.com/opos.htm).

CHAPTER 2 INSTALLATION & CONFIGURATION

Overview

This chapter describes installation instructions and settings to configure the Zebra Scanner OPOS Driver on a host computer.

For custom installation instructions, refer to the *Zebra Scanner SDK for Windows Developer's Guide* (p/n 72E-149784-xx).

✓ ***NOTE*** OPOS components are installed by default with the standard Scanner SDK installation. If a custom Scanner SDK installation is performed, the OPOS option must be selected to install the OPOS components.

Configuration

After a successful installation of the Zebra Scanner SDK with the OPOS driver, the following system registry entries are created at:

HKEY_LOCAL_MACHINE\SOFTWARE\OLEforRetail\ServiceOPOS\Scanner\ZEBRA_SCANNER

or

HKEY_LOCAL_MACHINE\SOFTWARE\OLEforRetail\ServiceOPOS\Scale\ZEBRA_SCALE

The Zebra Scanner OPOS Driver reads the registry entry above to retrieve required configurations, for example baud rate for serial scanners, and scanner filtering rules to form the logical scanner defined by the user.

See *Appendix A, WINDOWS REGISTRY KEYS for OPOS DRIVER* for descriptions of supported registry keys.

Scanner Configuration Bar Codes

Scan the **Set All Defaults** bar code below to return all parameters to the scanner's default values. Refer to the scanner's Product Reference Guide for default values.

Set All Defaults

Scan the appropriate bar code below to configure the scanner for either USB or RS-232 communication protocols.

USB Communication Protocol

IBM USB Hand-held with Full Scan Disable

IBM USB Table-top

RS-232 Communication Protocol

SSI

Wincor-Nixdorf RS-232 Mode B (Scanner only)

✓ ***NOTE*** To configure serial communication settings refer to the *Zebra Scanner SDK for Windows Developer’s Guide* (p/n 72E-149784-xx).

CHAPTER 3 OPOS PROPERTIES, METHODS, EVENTS

Overview

The following steps depict the behavioral model of the OPOS driver and scanner.

1. The scanner reads encoded data from a label.
2. When the Control receives input, it queues a DataEvent.
3. If the AutoDisable property is TRUE, the Control is disabled when a DataEvent is queued.
4. The Control can deliver a queued DataEvent to the application when the DataEventEnabled property is TRUE. Just before delivering this event, the Control copies the data into properties and disables further data events by setting the DataEventEnabled property to FALSE. This causes the Control to queue subsequent input data while the application processes the current input and associated properties. When the application finishes the current input and is ready for more data, it re-enables events by setting DataEventEnabled to TRUE.
5. The Control queues an ErrorEvent (or events) if it encounters an error while gathering or processing input, and delivers this to the application when the DataEventEnabled property is TRUE.
6. The DataCount property contains the number of DataEvents queued by the Control.
7. Call the ClearInput method to delete all input that the Control queued.

Scanned data is placed into the property ScanData. If the application sets the property DecodeData to TRUE, the data is decoded into ScanDataLabel and ScanDataType.

The following steps depict the behavioral model of the OPOS driver and scale.

1. The user/OPOS application performs read weight operation scale.
2. When the *ReadWeight* method successfully returns a value, scale control returns the weight data to the POS application.

Deviations from OPOS Specifications

The Zebra Scanner OPOS Driver includes several deviations from the OPOS specification for more flexibility. For example, the Claim method and Claimed property have significant deviations. According to the OPOS model, one control accesses only one physical device. The Zebra Scanner OPOS Driver allows access to multiple scanners simultaneously. Hence, a claim succeeds with one or more scanners. Also, several applications can share one scanner.

In addition to the Zebra Scanner OPOS Driver's architectural aspects, the following special behaviors occur:

- When there is no scanner connected to a cordless base, Zebra Scanner OPOS Driver considers the cordless base a scanner. Therefore a claim succeeds with a cordless base.
- In serial mode, a claim succeeds even when no scanner is connected to the port. In this case, it indicates the success of the port opening.
- Since the Zebra Scanner OPOS Driver supports multiple scanners, it implements the OPOS retrieveStatistics method call with certain deviations.
  - When claiming multiple scanners, scanner details appear sequentially with comma separation. The order is the same for all scanner statistics.
  - When there are multiple scanners, OPOS schema validation can fail because some date fields contain comma-separated multiple dates.
  - Non-RSM scanners, including serial scanners in Wincor-Nixdorf RS-232 Mode B, do not provide Model Number and Serial number. These values appear as empty strings.

Supported Feature Set

This section describes the supported feature set per the OPOS specification.

Properties

Table 3-1 *Common Properties*

Property	Version	Type	Access	May Use After	Comments on Zebra Scanner Support
AutoDisable	1.2	Boolean	R/W	Open	Supported
BinaryConversion	1.2	Int32	R/W	Open	Supported
CapCompareFirmwareVersion	1.9	Boolean	R	Open	Not supported
CapPowerReporting	1.3	Int32	R	Open	Not supported
CapStatisticsReporting	1.8	Boolean	R	Open	Supported
CapUpdateFirmware	1.9	Boolean	R	Open	Not supported
CapUpdateStatistics	1.8	Boolean	R	Open	Supported
CheckHealthText	1.0	String	R	Open	Not supported. Always returns OPOS_E_ILLEGAL
Claimed	1.0	Boolean	R/W	Open	Supported (see *Deviations from OPOS Specifications on page 3-2*)
DataCount	1.2	Int32	R	Open	Supported
DataEventEnabled	1.0	Boolean	R/W	Open	Supported
DeviceEnabled	1.0	Boolean	R/W	Open & Claim	Supported
FreezeEvents	1.0	Boolean	R/W	Open	Supported
OpenResult	1.5	Int32	R	n/a	Supported
PowerNotify	1.3	Int32	R/W	Open	Not supported. Always returns OPOS_E_ILLEGAL
PowerState	1.3	Int32	R	Open	Supported
ResultCode	1.0	Int32	R	n/a	Supported
ResultCodeExtended	1.0	Int32	R	Open	Supported
State	1.0	Int32	R	n/a	Supported
ControlObjectDescription	1.0	Int32	R	n/a	Supported
ControlObjectVersion	1.0	Int32	R	n/a	Supported
ServiceObjectDescription	1.0	String	R	Open	Supported

Table 3-1 *Common Properties (Continued)*

Property	Version	Type	Access	May Use After	Comments on Zebra Scanner Support
ServiceObjectVersion	1.0	Int32	R	Open	Supported
DeviceDescription	1.0	String	R	Open	Supported
DeviceName	1.0	String	R	Open	Supported

Table 3-2 *Scanner Specific Properties*

Property	Version	Type	Access	May Use After	Comments on Zebra Scanner Support
DecodeData	1.2	Boolean	R/W	Open	Supported
ScanData	1.0	BSTR	R	Open	Supported
ScanDataLabel	1.2	BSTR	R	Open	Supported
ScanDataType	1.2	Int32	R	Open	Supported

Table 3-3 *Scale Specific Properties*

Property	Version	Type	Access	May Use After	Comments on Zebra Scanner Support
MaximumWeight	1.2	INT32	R	Open	Supported
WeightUnits	1.0	INT32	R	Open	Supported
AsyncMode	1.3	Boolean	R/W	Open	Supported
MaxDisplayTextChars	1.13	INT32	R	Open	Not supported
TareWeight	1.3	INT32	R/W	Open & Claim	Not supported
ScaleLiveWeight	1.9	INT32	R	Open	Not supported
StatusNotify	1.9	INT32	R/W	Open	Not supported
ZeroValid	1.13	Boolean	R/W	Open	Supported
CapDisplay	1.2	Boolean	R	Open	Not supported
CapDisplayText	1.3	Boolean	R	Open	Not supported
CapPriceCalculating	1.3	Boolean	R	Open	Not supported
CapTareWeight	1.3	Boolean	R	Open	Not supported
CapZeroScale	1.3	Boolean	R	Open	Supported
CapStatusUpdate	1.9	Boolean	R	Open	Not supported

Methods

Table 3-4 *Common Methods*

Method	Version	May Use After	Comments on Zebra Scanner Support
Open	1.0	n/a	Supported
Close	1.0	Open	Supported
ClaimDevice	1.0	Open	Supported (see *Deviations from OPOS Specifications on page 3-2*).
ReleaseDevice	1.0	Open & Claim	Supported
CheckHealth	1.0	Open, Claim & Enable	Supported
ClearInput	1.0	Open & Claim	Supported
ClearInputProperties	1.10	Open & Claim	Supported
DirectIO	1.11	Open	Supported. See *Zebra OPOS Driver Direct IO API on page 3-6*.
compareFirmwareVersion	1.9	Open, Claim & Enable	Not supported
resetStatistic	1.8	Open, Claim & Enable	Supported
retrieveStatistics	1.8	Open, Claim & Enable	Supported
updateFirmware	1.9	Open, Claim & Enable	Not supported
updateStatistics	1.8	Open, Claim & Enable	Supported

Table 3-5 *Scale Specific Methods*

Method	Version	May Use After	Comments on Zebra Scanner Support
DisplayText	1.3	Open, Claim & Enable	Not supported
ReadWeightReadWeight	1.3	Open, Claim & Enable	Supported
ZeroScale	1.3	Open, Claim & Enable	Supported

Events

Table 3-6 *Events*

Event	Version	May Use After	Comments on Scale Support	Comments on Scanner Support
DataEvent	1.0	Open, Claim & Enable	Supported	Supported
DirectIOEvent	1.0	Open & Claim	Supported	Not supported
ErrorEvent	1.0	Open, Claim & Enable	Supported	Not supported
StatusUpdateEvent	1.3	Open, Claim & Enable	Not supported	Not supported

Zebra OPOS Driver Direct IO API

The Zebra OPOS driver provides direct access to Zebra devices connected to a host PC through the OPOS Direct IO API. It is possible for an application developer to configure all the vendor specific configurations of Zebra devices, like beeper tone, beeper volume, enable/disable symbologies, or even rebooting the device to factory defaults using Direct I/O functionality.

The Zebra OPOS Direct IO API is called in a manner similar to calling the CoreScanner API with op-codes and XML based IN and OUT parameters (refer to the Zebra Scanner SDK for Windows Developer Guide for more detail). The Zebra OPOS Direct IO API provides the following functions.

- Discover all connected Motorola devices (op-code: GET_SCANNERS)
  - Op-code: GET_SCANNERS
  - Input XML Arg: None
  - Scanner ID values retrieved by executing this command are used in the value for <scannerID> tags in next opcodes described below.
- Retrieve all attributes supported by a device (op-code: RSM_ATTR_GETALL)
  - Op-Code: RSM_ATTR_GETALL
  - Input XML Arg:

```
<inArgs>
 <scannerID>1</scannerID>
</inArgs>
```

- Retrieve value of one or set of attributes (op-code: RSM_ATTR_GET)
  - Op-Code: RSM_ATTR_GET
  - Input XML Arg:

```
<inArgs>
  <scannerID>1</scannerID>
  <cmdArgs>
    <arg-xml>
      <attrib_list>1</attrib_list>
    </arg-xml>
  </cmdArgs>
</inArgs>
```

- Retrieve value of the attribute next to the given attribute (op-code: RSM_ATTR_GETNEXT)
  - Op-Code: RSM_ATTR_GETNEXT
  - Input XML Arg:

```
<inArgs>
  <scannerID>1</scannerID>
  <cmdArgs>
   <arg-xml>
    <attrib_list>1</attrib_list>
   </arg-xml>
  </cmdArgs>
</inArgs>
```

- Temporarily set the value of an attribute (op-code: RSM_ATTR_SET)
  - Op-Code: RSM_ATTR_GETNEXT
  - Input XML Arg:

```
<inArgs>
  <scannerID>1</scannerID>
  <cmdArgs>
   <arg-xml>
     <attrib_list>
      <attribute>
        <id>6000</id>
        <datatype>X</datatype>
        <value>2</value>
      </attribute>
     </attrib_list>
   </arg-xml>
  </cmdArgs>
</inArgs>
```

  - Values for <id> and <value> can be found in the table below
- Permanently set the value of an attribute (op-code: RSM_ATTR_STORE)
  - Op-Code: RSM_ATTR_STORE
  - Input XML Arg:

```
<inArgs>
  <scannerID>1</scannerID>
  <cmdArgs>
   <arg-xml>
    <attrib_list>
      <attribute>
        <id>1</id>
        <datatype>F</datatype>
        <value>True</value>
      </attribute>
     </attrib_list>
    </arg-xml>
   </cmdArgs>
</inArgs>
```

Action Attributes and Values For Use With RSM_ATTR_SET

Table 3-7 *Action Attributes and Values*

Attribute Number	Attribute Name	Description	Data Type	Values	
				Beep/LED Action	Value
6000	Beeper/LED	Triggers the beeper/LED via command.	'X'	1 high short beep	0
				2 high short beeps	1
				3 high short beeps	2
				4 high short beeps	3
				5 high short beeps	4
				1 low short beep	5
				2 low short beeps	6
				3 low short beeps	7
				4 low short beeps	8
				5 low short beeps	9
				1 high long beep	10
				2 high long beeps	11
				3 high long beeps	12
				4 high long beeps	13
				5 high long beeps	14
				1 low long beep	15
				2 low long beeps	16
				3 low long beeps	17
				4 low long beeps	18
				5 low long beeps	19
				Fast warble beep	20
				Slow warble beep	21
				High-low beep	22
				Low-high beep	23
				High-low-high beep	24
				Low-high-low beep	25
				High-high-low-low beep	26
				Green LED off	42
				Green LED on	43
				Yellow LED on	45
				Yellow LED off	46
				Red LED on	47
				Red LED off	48
6001	ParameterDefaults	Initiates a parameter defaults command.	'X'	0 – Restore Defaults 1 – Restore Factory Defaults 2 – Write Custom Defaults	
6003	BeepOnNextBootup	Controls whether or not the boot up / power up beep is suppressed on the next power up.	'X'	0 – Disable beep on next bootup 1 – Enable beep on next bootup	

Table 3-7 *Action Attributes and Values (Continued)*

Attribute Number	Attribute Name	Description	Data Type	Values	
				Beep/LED Action	Value
6004	Reboot	Remote reboot command	‘X’		
6005	HostTriggerSession	Triggers the scanner to start scanning via command.	‘X’	0 – start Host Trigger Session 1 – stop Host Trigger Session	
6011	StatsReset	Reset/default a specific statistic	‘X’	The specific statistic attribute to reset. Range 15002-19999	
6013	StatsResetAll	Reset/default all statistics	‘X’		
6017	ScaleReadWeight	Read Weight from scale	‘A’	Byte[0] status: 0=scaleNotEnabled 1=scaleNotReady 2=stableWeightOverLimit 3=stableWeightUnderZero 4=nonStableWeight 5=stableZeroWeight 6=stableNonZeroWeight Byte[1] units: 0=kgs, 1=lbs Bytes[2-5] weight in thousandths of units	
6018	ScaleZero	Zeros the scale	‘X’		
6019	ScaleReset	Resets the scale	‘X’		
6022	ChangeAllCodeTypes	Enables/Disables all code types	‘X’	0 = Disable All Code types 1 = Enable All Code Types	

CHAPTER 4 SCANNER OPOS SAMPLE APPLICATION

Overview

The Zebra Scanner OPOS Driver suite ships with a sample application that demonstrates all the OPOS operations on a connected Zebra scanner.

OPOS Sample Application (Scanner OPOS Test Utility)

The Scanner OPOS Test Utility allows you to simulate an application communicating with the Zebra Scanner OPOS Driver. This utility displays scanned data received from the scanner through the Zebra Scanner OPOS Driver. The Zebra Scanner SDK includes source code for this VC++ test utility.

OPOS Test Utility Window Functionality

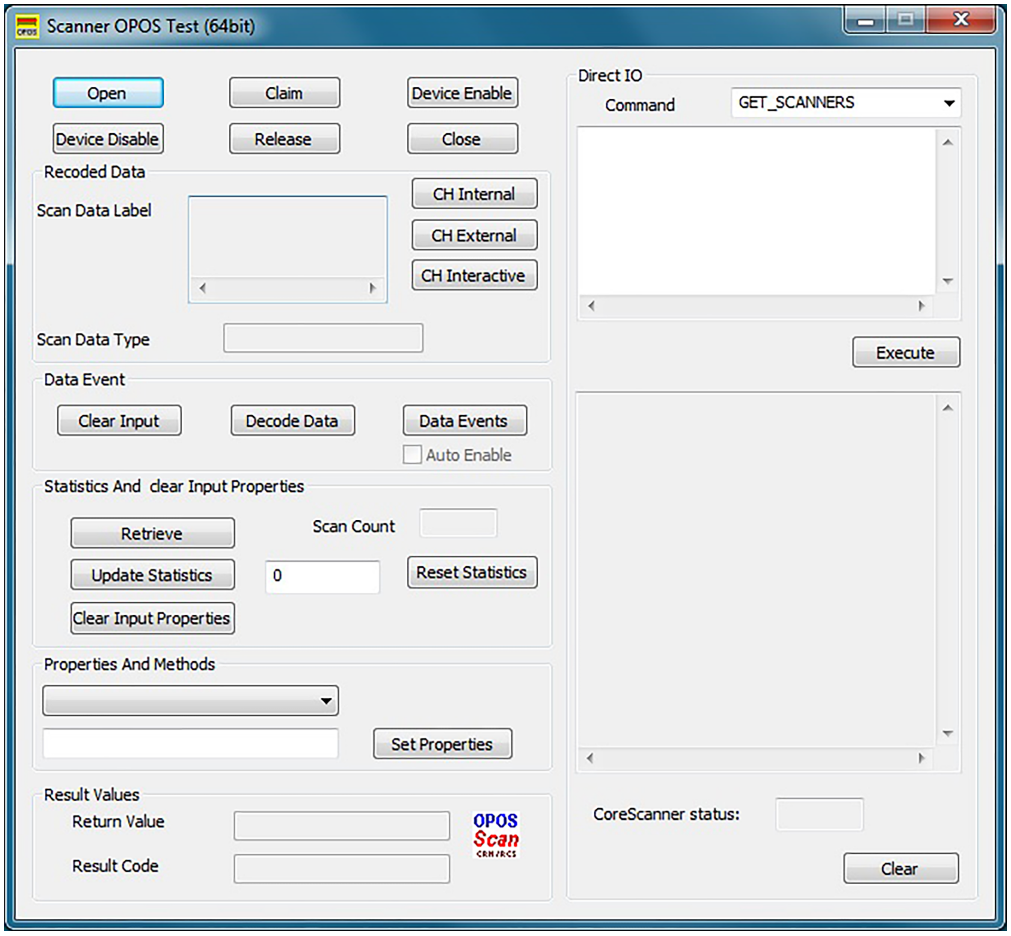


Figure 4-1 *Scanner OPOS Test Window*

Table 4-1 *Scanner OPOS Test Utility Button/Field Functionality*

Button/ Field/ Check Box	Description	Values	Code Sample
Open	Open Method.	ZEBRA_SCANNER	`m_ctScanner.Open("ZEBRA_SCANNER")`
Claim	Claim the device with time out value.	-1, Any integer starting from zero	`m_ctScanner.Claim(1000)`
Device Enable	Enable the scanner. Must enable before using scanners.	n/a	`m_ctrScanner.SetDeviceEnabled(TRUE)`
Device Disable	Disable the scanner.	n/a	`m_ctrScanner.SetDeviceEnabled(FALSE)`
Release	Release the scanner.	n/a	`m_ctrScanner.ReleaseDevice()`

Table 4-1 *Scanner OPOS Test Utility Button/Field Functionality (Continued)*

Button/ Field/ Check Box	Description	Values	Code Sample
Close	Close the scanner.	n/a	`m_ctrScanner.CloseDevice()`
Recorded Data			
Scan Data Label	Label of the scan data.	n/a	`m_ctrScanner.GetScanDataLabel()`
Scan Data Type	Type of the scanned data. This is only a readable property.	n/a	`m_ctrScanner.GetScanDataType()`
CH Internal	Perform a Health check that does not physically change the device. The device is tested by internal tests to the extent possible.	n/a	`m_ctrlScanner.CheckHealth(OPOS_CH_INTERNAL)`
CH External	Performs a more extensive test that may change the device. When executed, the scanner may beep.	n/a	`m_ctrlScanner.CheckHealth(OPOS_CH_INTERNAL)`
CH Interactive	Not supported.	n/a	
Data Event			
Clear Input	Clear method. Clears the input data.	n/a	`m_ctrScanner.ClearInput()`
Decode Data	Set decode data enable.	n/a	`m_ctrScanner.SetDecodeData(1)`
Data Events	Set data event enabled. Must enable data event to get data.	n/a	`m_ctrlScanner.SetDataEventEnabled(1)`

Table 4-1 *Scanner OPOS Test Utility Button/Field Functionality (Continued)*

Button/ Field/ Check Box	Description	Values	Code Sample
Auto Enable	Check box to automatically enable the scanner after a decode scan.	n/a	Refer to OPOS Scanner Sample Application source code provided with the SDK installation.
Statistics And Clear Input Properties			
Retrieve	Retrieve statistic.	GoodScanCount	`m_ctrlScanner.RetrieveStatistics(&test)`
Reset Statistics	Reset statistics.	GoodScanCount	`m_ctrlScanner.ResetStatistics("GoodScanCount")`

Update Statistics	Update statistics.	GoodScanCount	`CString strTemp;`
			`m_nGoodScanCount=100;`
			`strTemp.Format("GoodScanCount=%d",`
			`m_nGoodScanCount );`
			`m_ctrlScanner.UpdateStatistics(strTemp)`

Clear Input Properties	Clear input properties.	n/a	`m_ctrlScanner.ClearInputProperties()`
Properties and Methods			

Set Properties	Set the value of property to the given value.	n/a	`m_ctrScanner.SetFreezeEvents(0)`
			`m_ctrScanner.SetAutoDisable(1)`

Result Values			
Return Value	Return value of the last function call.	Read only.	
Result Code	Return value of result code.	Read only.	`m_ctrScanner.GetResultCode()`
Direct IO			
Command	Select the Direct IO command to be executed.	GET_SCANNERS RSM_ATTR_GETALL RSM_ATTR_GET RSM_ATTR_GETNEXT RSM_ATTR_SET RSM_ATTR_STORE	This is the current set of op-codes supported by Zebra OPOS Direct I/O implementation. For additional information refer to the OPOS Scanner Sample Application source code provided with the SDK installation.
inXML field	Input XML for selected Direct IO command	Modifiable input XML code	This is the man input parameter for the Direct I/O function. The format of the XML in very similar to inXML for CoreScanner API. For additional information refer to Scanner SDK Developer Guide and OPOS Scanner Sample Application source code provided with the SDK installation.

Table 4-1 *Scanner OPOS Test Utility Button/Field Functionality (Continued)*

Button/ Field/ Check Box	Description	Values	Code Sample
Execute	Execute the selected command.	n/a	
outXML field	XML output resulting from selected command	Read-only output XML	This is the man output parameter for the Direct I/O function. The format of the XML in very similar to inXML for CoreScanner API. For additional information refer to Scanner SDK Developer Guide and OPOS Scanner Sample Application source code provided with the SDK installation.
CoreScanner status	Result code from the underlying CoreScanner layer	Read-only	Direct I/O function helps to communicate an application with underlying device/device drivers. This field represent CoreScanner state after each Direct I/O call. Refer to OPOS Scanner Sample Application source code provided with the SDK installation.
Clear	Clear the contents of the outXML field	n/a	This clears the Direct I/O related text fields in the OPOS sample application. Please refer to OPOS Scanner Sample Application source code provided with the SDK installation.

Viewing Bar Code Data

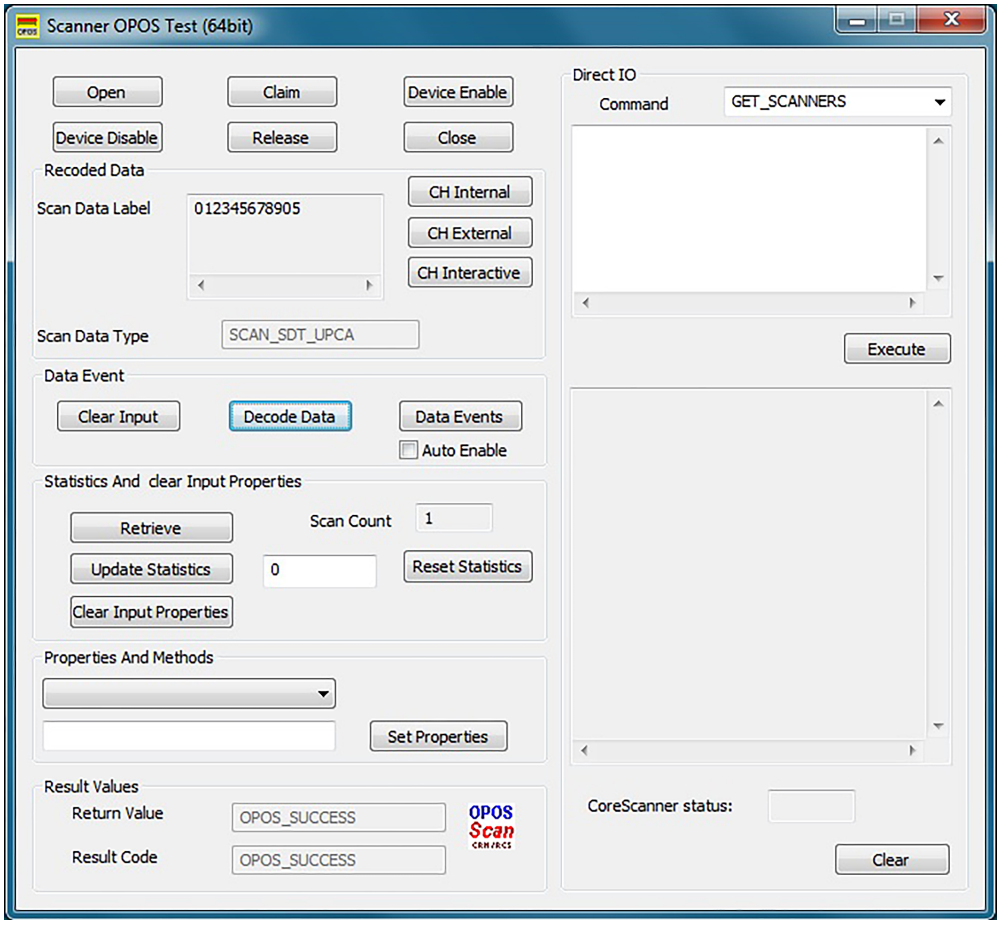
To view bar code data using the Scanner OPOS Test Utility:

1. Scan the USB OPOS (Hand-held) bar code, SNAPI bar code or Wincor-Nixdorf RS-232 Mode B bar code *on page 2-2* to configure the scanner for the correct communication protocol.
2. Under the folder \Program Files\Zebra Technologies\Barcode Scanners\Scanner SDK\OPOS\Scanner OPOS\Sample Applications\bin, enter the sub-folder appropriate to the host architecture (x86 or x64) and run Scanner-OPOS-Test.exe to launch the Scanner OPOS Test Utility.
3. Select **Open**. The *Open Service Object* window appears.
4. Select **Ok** to use the Zebra Scanner Service Object that the Scanner SDK Installshield setup program loaded on the PC.
5. Select **Claim**.
6. Select **Device Enable**.
7. Select **Data Events**.
8. Select **Decode Data**.
9. Scan the following sample bar code:

0 12345 67890 5

UPC-A

10. Select **Data Events** to view the scanned UPC-A bar code data. The bar code data that the driver processed appears in the *Scan Data Label* and *Scan Data Type* boxes:


Scanner OPOS Test (64bit)

Recoded Data
Scan Data Label: 012345678905
Scan Data Type: SCAN_SDT_UPCA

Direct IO
Command: GET_SCANNERS

Statistics And clear Input Properties
Scan Count: 1

Result Values
Return Value: OPOS_SUCCESS
Result Code: OPOS_SUCCESS

CoreScanner status:

Figure 4-2 *Scanner OPOS Test Window - Scan Data*

11. Select **Clear Input** to clear the data from the *Scan Data Type* and *Scan Data Label* boxes.

12. To perform a second test, scan another bar code.

Getting and Setting OPOS Properties

This utility allows getting and setting the OPOS properties of the Zebra Scanner OPOS Driver via the *Properties And Methods* drop-down list.

To get and set the OPOS properties of the Zebra Scanner OPOS Driver:

1. Scan the USB OPOS (Hand-held) bar code, SNAPI bar code or Wincor-Nixdorf RS-232 Mode B bar code *on page 2-2* to configure the scanner for the correct communication protocol.
2. Under folder \Program Files\Zebra Technologies\Barcode Scanners\Scanner SDK\OPOS\Scanner OPOS\Sample Applications\bin, enter the sub-folder appropriate to the host architecture (x86 or x64) and run Scanner-OPOS-Test.exe to launch the Scanner OPOS Test Utility.
3. Select **Open**. The *Open Service Object* window appears.
4. Select **Ok** to use the Zebra Scanner Service Object that the SDK Installshield loaded on the PC.
5. Select **Claim**.
6. Select **Device Enable**.
7. Select a configurable (setable) property in the *Properties And Methods* drop-down list. You can configure some properties (e.g., AutoDisable, FreezeEvents); other properties are read only (e.g., ServiceObjectVersion, DataCount).
8. The current value of the OPOS driver appears in the edit box below the property selected in the list box. The values 1 and 0 represent true and false, respectively.
9. To change the configurable property, change the value in the edit box and select **Set Properties**. This updates the property with the new value.

Creating a Custom OPOS Sample Application

You may use any programming language to create a custom OPOS sample application. However, Microsoft supported languages are recommended (e.g., Visual Basic, Visual C++ or C#).

To create a custom OPOS sample application:

1. Create a project in the desired Microsoft Visual Studio development environment.
2. Select *Choose Toolbox Items…* from the *Tools* menu.
3. Select (check) OPOS Scanner Control from the *COM Components* tab.
4. Add OPOSScanner.ocx to the project.
5. Drag and drop to the form/Dialog window.
6. Add a variable (handle) for the scanner control added to the form/Dialog window.
7. Call Open(), Claim() methods [e.g., Open ("SYMBOL_SCANNER"); Claim (2000);].
8. SetDeviceEnabled to TRUE.
9. Set Freeze Events property to FALSE [e.g., SetFreezeEvents (FALSE)].
10. SetDataEventsEnable to TRUE to get scan data events.
11. When done, SetDeviceEnabled to FALSE, Release() and Close() the service.
12. Call Device Disable property, release and close methods to close the connection.

Return Value and Result Code

When calling any method, check whether the return value is 0 (=OPOS_SUCCESS) to ensure the method is successful. Otherwise it returns an error code, which indicates the reason for the error. After setting property values, check that the result code returns 0 (=OPOS_SUCCESS), indicating success. If unsuccessful, it returns an error code.

Direct I/O

See *Zebra OPOS Driver Direct IO API on page 3-6* for a description of Direct IO functionality.

Statistics Methods

The Zebra OPOS Driver supports the retrieveStatistics, resetStatistics, and updateStatistics methods. GoodScanCount is the only defined statistic in the Zebra OPOS Driver and can be used as a parameter for these methods.

Modified Claim Functionality

Model number, serial number and the Type (Scanner Host Mode) parameters are available in the system registry as configurable entries so that user can configure them according to the business requirement. Claiming a scanner compares the scanner details provided in system registry with the attached scanner properties. The claim is successful when they match.

Enter * to include anything for the particular entry. For example, enter * for the serial number to claim scanners with any serial number. Otherwise, the claim is successful only if the provided serial number and model number matches the present scanner.

For the model number and serial number, provide the exact value, or part of the string and a star (*). Do not enter a star (*) in the middle of a string; in this case, all data after star (*) is ignored. Provide a value and star as a model number (e.g., LS4208*) to accept all scanners starting with that model number (LS4208). Provide a star (*) for the serial number to accept all scanners regardless of serial number.

It is required to add the full name of each scanner type as a comma or space delimited list to enable the scanners to be Claimed. As an example, to include SNAPI scanners and IBM hand-held mode scanners the "Type" entry should be (USBIBMHID SNAPI). To include all scanner modes (Types) the value should be (ALL)

Since non-RSM scanners, including serial scanners in Wincor-Nixdorf RS-232 Mode B, do not provide the model number and serial number, to claim these scanners set a " * " to both ModelNumber and SerialNumber registry values.

CHAPTER 5 SCALE OPOS SAMPLE APPLICATION

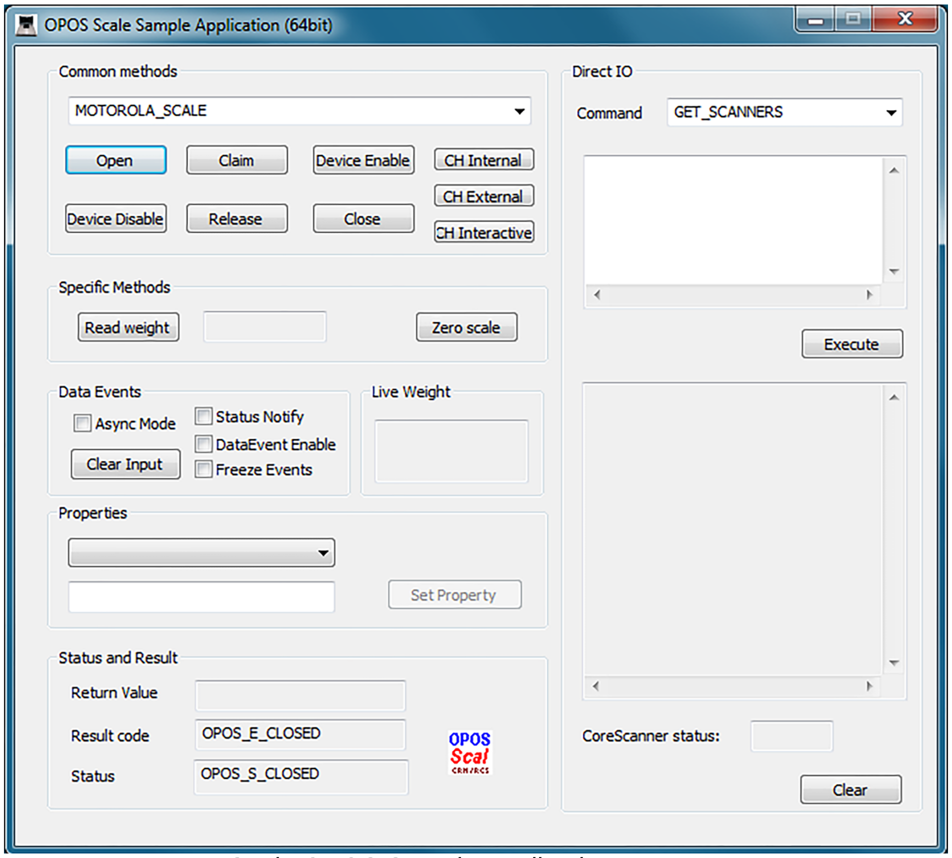
Overview

The Zebra Scale OPOS Driver suite ships with a sample application that demonstrates all the OPOS operations on a connected Zebra scale.

Scale OPOS Sample Application

The Scale OPOS Sample Application allows you to simulate an application communicating with the Zebra Scale OPOS Driver. This utility displays received weight data from the scale through the Zebra Scale OPOS Driver. The Zebra Scanner SDK includes source code for this VC++ test utility.

Scale OPOS Sample Application Window Functionality



Common methods: MOTOROLA_SCALE

Command: GET_SCANNERS

Result code: OPOS_E_CLOSED

Status: OPOS_S_CLOSED

Figure 5-1 *Scale OPOS Sample Application*

Table 5-1 *Scale OPOS Sample Application Button/Field Functionality*

Button/ Field/ Check Box	Description	Values	Code Sample
Open	Open Method	ZEBRA_SCALE	`OposScale.Open("ZEBRA_SCALE"`
Claim	Claim the device with time out value.	-1, Any integer starting from zero	`OposScale.ClaimDevice(1000)`
Device Enable	Enable the scale. Must enable before using scale.	n/a	`OposScale.put_DeviceEnabled(TRUE);`
Device Disable	Disable the scale.	n/a	`OposScale.put_DeviceEnabled(FALSE);`
Release	Release the scale.	n/a	`OposScale.ReleaseDevice()`
Close	Close the scale.	n/a	`OposScale.Close()`
CH Internal	Perform a health check that does not physically change the device. The device is tested by internal tests to the extent possible.	n/a	`m_ctrlScanner.CheckHealth(OPOS_CH_INTERNAL)`

Table 5-1 *Scale OPOS Sample Application Button/Field Functionality (Continued)*

Button/ Field/ Check Box	Description	Values	Code Sample
CH External	Performs a more extensive test that may change the device. When executed, the scanner may beep.	n/a	`m_ctrlScanner.CheckHealth(OPOS_CH_INTERNAL)`
CH Interactive	Not supported.	n/a	
ReadWeight	Read the weight on the scale.	`&lweightData` is the weight on the scale (a pointer to the variable holding the returned weight value). `5000` is the time in milliseconds to wait before returning an error (if no valid weight measurement read).	`OposScale.ReadWeight(&lweightData, 5000)`
Zero Scale	Set the scale weight value to zero	n/a	`OposScale.ZeroScale();`
Async Mode	Set the AsyncMode Property	True, false	`OposScale.put_AsyncMode(true);`
Clear Input	Sets all properties populated as result of firing DataEvent or ErrorEvent back to their default value.	True, false	`OposScale.ClearInput();`
Status Notify	Enables Live Weight values (must be set before Device Enable is sent to the scale).	True, false	`OposScale.put_StatusNotify(true);`
Live Weight	Reports the Live Weight value.		Refer to OPOS Scanner Sample Application source code provided with the SDK installation.
Freeze Events	Freeze events.	True, false	`OposScale.put_FreezeEvents(true);`
DataEvent Enable	Enable Data Events.	True, false	`OposScale.put_DataEventEnabled(true);`
Set Property	Set the value of property to the given value.	n/a	Refer to OPOS Scanner Sample Application source code provided with the SDK installation.

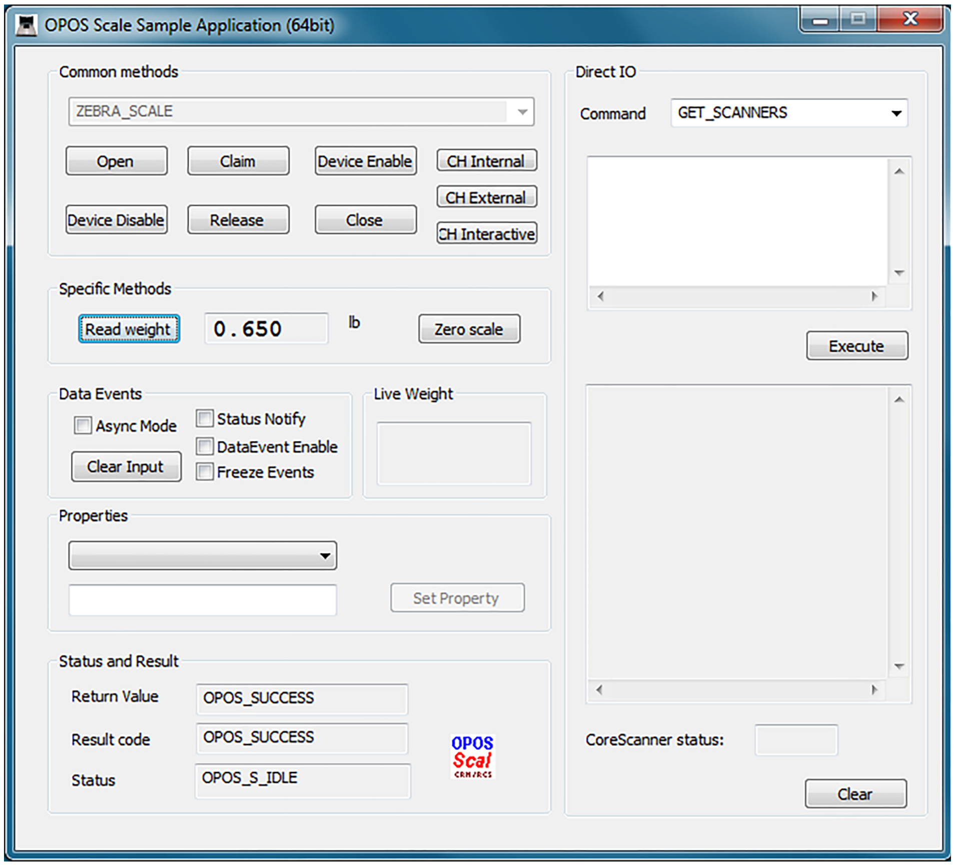
Table 5-1 *Scale OPOS Sample Application Button/Field Functionality (Continued)*

Button/ Field/ Check Box	Description	Values	Code Sample
Status and Result			
Return Value	Return value of the last function call.	This is only a readable property.	n/a
Result code	Return value of result code.	This is only a readable property.	n/a
Status	Status of the scale service object.	This is a read only property.	n/a

Retrieving Weight Data

To retrieve weight data using the OPOS Scale sample application:

1. Connect the MP6200 scanner/scale unit to a host PC with a USB, or serial cable.
2. Switch the scanner to a management-capable host mode by scanning one of the bar codes *on page 2-2* to configure the scanner for the correct communication protocol.
   a. IBM Hand-held
   b. IBM Table-top
   or
   c. SSI.
3. Under folder \Program Files\Zebra Technologies\Barcode Scanners\Scanner SDK\OPOS\Scale OPOS\Sample Applications\bin, enter the sub-folder appropriate to the host architecture (x86 or x64) and run OPOSScaleSampleApp.exe to launch the Scale OPOS sample application.



ZEBRA_SCALE

0.650 lb

Command GET_SCANNERS

Return Value OPOS_SUCCESS

Result code OPOS_SUCCESS

Status OPOS_S_IDLE

Figure 5-2 *Zebra Scale OPOS Sample Application*

4. Place the item to weigh on the platter of the scale.
5. On the sample application screen, select **Open** to open the logical device named in the text box (which is by default installed by the Scanner SDK Installshield).
6. Select **Claim**.
7. Select **Device Enable**.
8. Select **Read Weight**. The weight of the item, and the measuring unit appear in the text box.

Retrieving Weight Data Asynchronously

To retrieve weight data asynchronously using the OPOS Scale sample application:

1. Connect the MP6000 scanner/scale unit to a host PC with a USB or serial cable.
2. Switch the scanner to a management-capable host mode by scanning one of the bar codes *on page 2-2* to configure the scanner for the correct communication protocol.
   a. IBM Hand-held
   b. IBM Table-top

      or
   c. SSI.
3. Under folder \Program Files\Zebra Technologies\Barcode Scanners\Scanner SDK\OPOS\Scale OPOS\Sample Applications\bin, enter the sub-folder appropriate to the host architecture (x86 or x64) and run OPOSScaleSampleApp.exe to launch the Scale OPOS sample application. See *Figure 5-2 on page 5-5*.
4. Place the item to weigh on the platter of the scale.
5. On the sample application screen, select **Open** to open the logical device named in the text box (which is by default installed by the Scanner SDK Installshield).
6. Select **Claim**.
7. Check the *AsyncMode* box in DataEvent section.
8. Check *DataEventEnable*.
9. Select **Device Enable**.
10. Select **Read Weight**. The weight of the item, and the measuring unit appear in the text box.

Getting and Setting OPOS Properties

This utility allows getting and setting the OPOS properties of the Zebra OPOS Driver via the Properties drop-down list.

To get and set the OPOS properties of the Zebra OPOS Driver:

1. Scan the USB OPOS (Hand-held) bar code, SNAPI bar code or SSI bar code *on page 2-2* to configure the scanner for the correct communication protocol.
2. Under folder \Program Files\Zebra Technologies\Barcode Scanners\Scanner SDK\OPOS\Scale OPOS\Sample Applications\bin, enter the sub-folder appropriate to the host architecture (x86 or x64) and run OPOSScaleSampleApp.exe to launch the Scale OPOS sample application.
3. Select **Open** to use the Zebra OPOSScale Service Object that the SDK Installshield loaded on the PC.
4. Select **Claim**.
5. Select **Device Enable**.
6. Select a configurable (setable) property in the *Properties* drop-down list. You can configure some properties (e.g., *AutoDisable*, *FreezeEvents*); other properties are read only (e.g., *ServiceObjectVersion*, *DataCount*).
7. The current value of the OPOS property appears in the edit box below the property selected in the list box. The values 1 and 0 represent true and false, respectively.
8. To change the configurable property, change the value in the edit box and select **Set Property**. This updates the property with the new value.

Creating a Custom OPOS Sample Application

You may use any programming language to create a custom OPOS sample application. However, Microsoft supported languages are recommended (e.g., Visual Basic, Visual C++, or C#).

To create a custom OPOS sample application:

1. Create a project in the desired Microsoft Visual Studio development environment.
2. Select *Choose Toolbox Items…* from *Tools* menu.
3. Select (check) *OPOS Scale Control* from the *COM Components* tab.
4. Add OPOSScale.ocx to the project.
5. Drag and drop to the form/*Dialog* window.
6. Add a variable (handle) for the scanner control added to the form/*Dialog* window.
7. Call Open(), Claim() methods [e.g., Open ("ZEBRA_SCALE"); Claim (2000);].
8. *SetDeviceEnabled* to TRUE.
9. Add button to call *ReadWeight* method, and add edit control to display the weight data.
10. When done, *SetDeviceEnabled* to FALSE, Release() and Close() the service.

Return Value and Result Code

When calling any method, check whether the return value is 0 (=OPOS_SUCCESS) to ensure the method is successful. Otherwise it returns an error code, which indicates the reason for the error. After setting property values, check that the result code returns 0 (=OPOS_SUCCESS), indicating success. If unsuccessful, it returns an error code.

Direct I/O

See *Zebra OPOS Driver Direct IO API on page 3-6* for a description of Direct IO functionality.

Statistics Methods

The Zebra OPOS Scale Driver does not support the *Statistics* method. Statistics related to a scale could be obtained through the RSM parameters of the MP6200. Refer to the *Zebra Scanner SDK for Windows Developer’s Guide* (p/n 72E-149784-xx).

Modified Claim Functionality

Model number, serial number and the Type (Scanner Host Mode) parameters are available in the system registry as configurable entries so that user can configure them according to the business requirement. Claiming a scanner compares the scanner details provided in system registry with the attached scanner properties. The claim is successful when they match.

Enter * to include anything for the particular entry. For example, enter * for the serial number to claim scanners with any serial number. Otherwise, the claim is successful only if the provided serial number and model number matches the present scanner.

For the model number and serial number, provide the exact value, or part of the string and a star (*). Do not enter a star (*) in the middle of a string; in this case, all data after star (*) is ignored. Provide a value and star as a model number (e.g., LS4208*) to accept all scanners starting with that model number (LS4208). Provide a star (*) for the serial number to accept all scanners regardless of serial number.

It is required add the full name of the each scanner type as comma or space delimited list to get the scanners Claimed. As an example to include SNAPI scanners and IBM Hand Held mode scanners the "Type" entry should be (USBIBMHID SNAPI). To include all scanner modes (Types) the value should be (ALL)

Since non-RSM scanners, including serial scanners in Wincor-Nixdorf RS232 Mode B, do not provide the model number and serial number, to claim these scanners set a " * " to both ModelNumber and SerialNumber registry values.

CHAPTER 6 SUPPORTED SYMBOLOGY TYPES VS. SCANNER MODE

Overview

This chapter provides a matrix of scanner modes and supported symbology types in each mode.

Supported Symbology Types vs. Scanner Mode

Table 6-1 *Supported Symbology Types vs. Scanner Modes*

Symbology		Scanner Mode		
Type	**Value**	**IBM HID**	**SNAPI**	**Nixdorf Mode B**
UPC-A	SCAN_SDT_UPCA	X	X	X
UPC-A with supplemental bar code	SCAN_SDT_UPCA_S	X	X	X
UPC-E	SCAN_SDT_UPCE	X	X	X
UPC-E with supplemental bar code	SCAN_SDT_UPCE_S	X	X	X
UPC-D1	SCAN_SDT_UPCD1	X	X	X
UPC-D2	SCAN_SDT_UPCD2	X	X	X
UPC-D3	SCAN_SDT_UPCD3	X	X	X
UPC-D4	SCAN_SDT_UPCD4	X	X	X
UPC-D5	SCAN_SDT_UPCD5	X	X	X
EAN 8 (=JAN 8)	SCAN_SDT_EAN8	X	X	X
JAN 8 (= EAN 8)	SCAN_SDT_JAN8	X	X	X
EAN 8 with supplemental barcode	SCAN_SDT_EAN8_S	X	X	X

Table 6-1 *Supported Symbology Types vs. Scanner Modes (Continued)*

Symbology		Scanner Mode		
Type	**Value**	**IBM HID**	**SNAPI**	**Nixdorf Mode B**
EAN 13 (= JAN 13)	`SCAN_SDT_EAN13`	X	X	X
JAN 13 (= EAN 13)	`SCAN_SDT_JAN13`	X	X	X
EAN 13 with supplemental barcode	`SCAN_SDT_EAN13_S`	X	X	X
EAN-128	`SCAN_SDT_EAN128`	X	X	X
Standard (or Discrete) 2 of 5	`SCAN_SDT_TF`	X	X	X
Interleaved 2 of 5	`SCAN_SDT_ITF`	X	X	X
Codabar	`SCAN_SDT_Codabar`	X	X	X
Code 39	`SCAN_SDT_Code39`	X	X	X
Code 128	`SCAN_SDT_Code128`	X	X	X
OCR "A"	`SCAN_SDT_OCRA`	X	X	-
OCR "B"	`SCAN_SDT_OCRB`	X	X	-
GS1 DataBar Omnidirectional (normal or stacked)	`SCAN_SDT_GS1_DATABAR`	X	X	-
GS1 DataBar Expanded (normal or stacked)	`SCAN_SDT_GS1_DATABAR_E`	X	X	-
Composite Component A	`SCAN_SDT_CCA`	-	X	-
Composite Component B	`SCAN_SDT_CCB`	-	X	-
Composite Component C	`SCAN_SDT_CCC`	-	X	-
PDF 417	`SCAN_SDT_PDF417`	X	X	-
MAXICODE	`SCAN_SDT_MAXICODE`	X	X	-
Data Matrix	`SCAN_SDT_DATAMATRIX`	-	X	-
QR Code	`SCAN_SDT_QRCODE`	-	X	-
Micro QR Code	`SCAN_SDT_UQRCODE`	-	X	-
Aztec	`SCAN_SDT_AZTEC`	-	X	-
Micro PDF 417	`SCAN_SDT_UPDF417`	-	X	-

When the scanner is in Wincor-Nixdorf RS-232 Mode B, the Zebra OPOS return value for the ScanDataType property differs from the expected value for the bar code types listed in *Table 6-2*.

Table 6-2 *Bar Code Types Not Accurately Identified in Wincor-Nixdorf RS-232 Mode B*

Symbology Type	Expected Value	Zebra RSM OPOS Return Value	Comments
UPC-A with supplemental bar code	SCAN_SDT_UPCA_S	SCAN_SDT_UPCA	Nixdorf Mode B cannot distinguish UPCA since it identifies bar code types UPCA, UPCA_S, EAN13, EAN13_S, and BOOKLAND as one type.
UPC-E with supplemental bar code	SCAN_SDT_UPCE_S	SCAN_SDT_UPCE	Nixdorf Mode B identifies both bar code types UPCE and UPCE_S as UPCE.
EAN 8 with supplemental bar code	SCAN_SDT_EAN8_S	SCAN_SDT_EAN8	Nixdorf Mode B identifies both EAN8 and EAN8_S bar code types as EAN8.
EAN 13	SCAN_SDT_EAN13	SCAN_SDT_UPCA	Nixdorf Mode B cannot distinguish EAN 13 since it identifies bar code types UPCA, UPCA_S, EAN13, EAN13_S, and BOOKLAND as one type.
EAN 13 with supplemental bar code	SCAN_SDT_EAN13_S	SCAN_SDT_UPCA	Nixdorf Mode B cannot distinguish EAN 13_S since it identifies bar code types UPCA, UPCA_S, EAN13, EAN13_S, and BOOKLAND as one type.

APPENDIX A WINDOWS REGISTRY KEYS FOR OPOS DRIVER

Overview

This appendix describes the Windows registry keys for the OPOS Scanner and Scale logical names and drivers.

Table A-1 *Registry Paths to Default Scanner OPOS Logical Names*

Key Path (x64)	HKEY_LOCAL_MACHINE\SOFTWARE\OLEforRetail\ServiceOPOS\Scanner\ZEBRA_SCANNER HKEY_LOCAL_MACHINE\SOFTWARE\OLEforRetail\ServiceOPOS\Scanner\MOTOROLA_SCANNER HKEY_LOCAL_MACHINE\SOFTWARE\OLEforRetail\ServiceOPOS\Scanner\STI_USBSCANNER HKEY_LOCAL_MACHINE\SOFTWARE\OLEforRetail\ServiceOPOS\Scanner\SYMBOL_SCANNER

✓ ***NOTE*** On 64-bit Windows, there are registry keys for 32-bit applications under the corresponding Wow6432Node.

Table A-2 *Registry Keys for OPOS Scanner Logical Names*

Value Name	Sample Data	Description
(Default)	ZebraScannerSO.ScannerSO	Service object name of the scanner OPOS.
Baud rate	9600 115200	Baud rate for serial port.
CheckDigitAutoCalculate	0 1	Automatically calculate Check Digit and append to the decode data label. 0 = Disable 1 = Enable

[1] Added to maintain the backward compatibility and are optional if the CompatibilityMode is set to disable backward compatibility.

Table A-2 *Registry Keys for OPOS Scanner Logical Names (Continued)*

Value Name	Sample Data	Description
CompatibilityMode	0 1	Specifies how the decode data is stored in the ScanData and ScanDataLabel properties. 0 = Decode data is stored according to the UPOS specification. 1 = ScanData property contains the same data as the ScanDataLabel property.
Connection[1]	ConnUSB	Added for backward compatibility.
Description	Zebra scanner logical device	Logical device description.
ExclusiveClaimLevel	0 1	Specifies whether an OPOS claim is required to be exclusive to the scanners in the logical device. For example, if a scanner is exclusively claimed by an application, other applications cannot claim it or send management commands to that scanner. 0 = OPOS claim exclusivity is not required for all scanners in the logical device. 1 = OPOS claim is exclusive to all scanners in the logical device. Any other value = OPOS claim is able to claim at least one scanner in the logical device.
ModelNumber	1. DS6707-SR20001ZZR,DS9808* (* Represents any model of the DS9808.)	Comma separated list of scanner model numbers for use with OPOS driver.
PID[1]	* (Represents all PIDs)	Product IDs of Zebra bar code scanners.
Port	COM1 COM1,COM2 * (Represent any COM port)	Comma or space delimited list of serial ports needed for use with OPOS driver.
SerialNumber	1. 7116000500337,7087000501981 2. * (Represents all serial numbers)	Comma separated list of scanner serial numbers for use with OPOS driver.
Type	SNAPI NIXMODB USBIBMHID USBIBMTT SSI ALL	Comma or space delimited list of scanner modes from the TypePool.
TypePool	SNAPI NIXMODB USBIBMHID USBIBMTT SSI ALL	All the supported types only for reference. OPOS driver does not read this entry.
VID	0x05E0	Vendor ID of Zebra bar code scanners.

[1] Added to maintain the backward compatibility and are optional if the CompatibilityMode is set to disable backward compatibility.

Table A-3 *Registry Paths to Default Scale OPOS Logical Names*

Key Path (x64)	HKEY_LOCAL_MACHINE\SOFTWARE\OLEforRetail\ServiceOPOS\Scale\ZEBRA_SCALE HKEY_LOCAL_MACHINE\SOFTWARE\OLEforRetail\ServiceOPOS\Scale\MOTOROLA_SCALE

✓ ***NOTE*** On 64-bit Windows, there are registry keys for 32-bit applications under the corresponding Wow6432Node.

Table A-4 *Registry Keys for OPOS Scale Logical Names*

Value Name	Sample Data	Description
(Default)	ScaleOPOS.ScalSO	Service object name of the Scale OPOS.
DeviceDescription	Zebra Technologies Scale logical device	Logical device description.
ExclusiveClaimLevel	0 1	0 =OPOS Scale service object does not get exclusive access of the device on claiming the device. This may be required if an application needs to access both scanner and scale of the MP6200 scanner at same time. 1 =Scale service object does claim the device exclusively so that no other POS application can access the device.
LiveWeightFrequency	2	Number of live weight events per second (values from 1 to 100).
NoTimeOutOnReadWeight	0 1	0 = ReadWeight method returns OPOS_E_TIMEOUT if no valid weight data is retrieved before the time out elapse. 1 = ReadWeight method does not return OPOS_E_TIMEOUT if no valid weight data is retrieved before the time out elapse. It always reports OPOS_SUCCESS as result code.
ZeroValidOveride	0 1	0 = ReadWeight does not accept zero weight as a valid reading. 1 = ReadWeight does accept zero weight as a valid reading.

Table A-4 *Registry Keys for OPOS Scale Logical Names (Continued)*

Value Name	Sample Data	Description
WeightChangeThreshold	Unit of the value is based on scales weight unit with assumed decimal point. For example, 10 = 0.01Kg or 0.01lb	Defines how large the weight reported from the scanner should differ from the previously reported value in order for the last reported weight value to be a legitimate weight.
WeightChangeMonitorInterval	Polling interval in milliseconds	Defines how fast the SO should poll the scale to detect a change of weight on the platter.
UnstableWeightIsADifferentWeight	0 = unstable no weight change 1 = unstable reported as weight change	Defines whether even an unstable weight reported after a stable weight is considered as a change of weight on the platter.

Table A-5 *Registry Path for Scanner OPOS Driver*

Key Path (x64)	HKEY_LOCAL_MACHINE\SOFTWARE\Zebra\Zebra Scanners\OPOS\Scanner

✓ ***NOTE*** On 64-bit Windows, there are registry keys for 32-bit applications under the corresponding Wow6432Node.

Table A-6 *Registry Keys for Scanner OPOS Driver - Applies to all Scanner OPOS Logical Names*

Value Name	Values	Description
AutoDisableControl	0 = Disable 1 = Enable	0 = Normal AutoDisable property operation. 1 = Automatically disables the scanner after each data event regardless of the value of OPOS property AutoDisable.
AutoDisableDelay	0	Delay in milliseconds to disable the scanner automatically after each data event. Default value is 0.
DataEventAutoDisableControl	0 = Disable 1 = Enable	0 = Normal DataEventEnable operation. 1 = Scanner SO overrides the value of OPOS property DataEventEnable and always delivers the data events to application layer.
DataEventDelay	0	Minimum time gap in milliseconds between two decode data events. Default value is 0 ms.
EnableTrace	0 = Disable 1 = Enable	Enable or disable debug/engineering message entries to the log file specified by LogFilePath.

Table A-6 *Registry Keys for Scanner OPOS Driver - Applies to all Scanner OPOS Logical Names (Continued)*

Value Name	Values	Description
InterCommandDelay	0	Minimum time gap in milliseconds between OPOS method calls. Default value is 0 ms.
LogFilePath	\Program Files\Zebra Technologies\Barcode Scanners\Scanner SDK\OPOS\Scanner OPOS\bin\Logs\stiopos.txt	Path and filename for log file if EnableTrace is enabled.
PowerNotifyControl	0 1	This is to control reporting the value for the OPOS PowerState property. When the value is set to 1, the PowerState property always has the value True regardless of the value of the PowerNotify property. This defaults to 0.
SetDeviceEnableStateOnFailure	0 = Disable 1 = Enable	0 = Normal Result Code operation 1 = Regardless of the result of DeviceEnable call, Result Code is set to OPOS_SUCCESS.

Table A-6 *Registry Keys for Scanner OPOS Driver - Applies to all Scanner OPOS Logical Names (Continued)*

Value Name	Values	Description
SetEnableDisableOnEvent	0 = Disable 1 = Enable	0 = Normal Enable/Disable operation 1 = SO keeps the device Enable/Disable state in same state as SO DeviceEnable state. If an external application changes the device state, the SO immediately corrects it.
SetHardwareAutoDisableState	0 = Disable 2 = Enable	Controls the hardware 'Scan Disable Mode' setting on the scanner itself. 0 = Normal operation 2 = The SO sets the 'Scan Disable Mode' on the scanner to 'Auto Disable'. This disables scanning after transmission of a bar code, and remains disabled until the host sends a Scan Enable. This feature is not supported by all scanner models.
SyncDeviceEnableStateOnDiscovery	0 = Disable 1 = Enable	Sync a newly connected device's enable / disable state with the current enable / disable state of the OPOS driver. This is helpful if a new scanner is connected to a POS system where it already has a connected scanner. At the time the new scanner is connected, if the state of the OPOS SO is 'device disabled' the newly connected scanner should also has to change its state to 'device disable'. This synchronization will be performed if "SyncDeviceEnableStateOnDiscovery" is enabled.

Table A-7 *Registry Path for Scale OPOS Driver*

Key Path (x64)	HKEY_LOCAL_MACHINE\SOFTWARE\Zebra\Zebra Scanners\OPOS\Scale

✓ ***NOTE*** On 64-bit Windows, there are registry keys for 32-bit applications under the corresponding Wow6432Node.

Table A-8 *Registry Keys for Scale OPOS Driver - Applies to all Scale OPOS Logical Names*

Value Name	Values	Description
DebugPrint	0 = Disable 1 = Enable	Enable debug log messages viewable through a debug log viewer such as Microsoft DebugView.
Enable_OPOS_ESCAL_SAME_WEIGHT_Error	0 = Disable 1 = Enable	Enable / disable reporting the OPOS_ESCAL_SAME_WEIGHT error. This is disabled by default.
EngDbgStr	0 = Disable 1 = Enable	Enable detailed engineering-level debug messages through a debug log viewer.
FileLog	0 = Disable 1 = Enable	Enable or disable debug/engineering message entries to the log file specified in the Location key.
Level	1 - 5	Level of log messages, from minimal (1) to verbose (5).
Location	\Program Files\Zebra Technologies\Barcode Scanners\Scanner SDK\OPOS\Scale OPOS\bin\Logs	Path name to log file if FileLog is enabled.

For more details on how to use the registry, see *Modified Claim Functionality on page 5-9*.

INDEX

ZEBRA

Zebra Technologies Corporation
Lincolnshire, IL U.S.A.
http://www.zebra.com

72E-149783-06 Revision B - January 2020