Application Note

8421-15950-18

EAN-18

WEIGH-TRONIX EAN-18

This document describes the Weigh-Tronix OPOS Scale-Class Service Object (SO) and its Installation/Administrator program.

WTScaleSO
OPOS Scale Control
&
WTScaleAdmin
Administrator Program

TITLE:				
APPLICATION NOTE: OPOS SCALE CONTROL AND WTOPOSAdmin Install / Test Program				
PART: 8421-15950-18	REV: B	ECO:	APPV'D:	22-FEB-02

Table of Contents

1 INTRODUCTION

1.1 Purpose

This document provides a general specification and description of WTScaleSO.DLL and WTScaleAdmin.EXE. **WTScaleSO.DLL** is an OPOS Scale-Class ActiveX in-process server (code component) while the **WTScaleAdmin.EXE** is the installation, test and administration program for the DLL server. The intended audience for this document are the technical personnel who will install, configure and test the OPOS Scale Control for use in an OPOS compliant Point-of-Sale system.

1.2 Scope

OLE for Retail POS (**OPOS**) is an evolving, standard specification that defines an open driver architecture for Point-of-Sale (**POS**) hardware devices on the Windows 95/98 and Windows NT platforms. The OPOS specification defines two intermediate layers that sit between the POS application and the hardware devices. This document elaborates on those aspects of the specification relevant to the Weigh-Tronix Scale device, and provides a detailed description of the architecture implemented in the Weigh-Tronix OPOS Scale Control. For a good background description of OLE for Retail POS we recommend you visit the following website:

`www.monroecs.com/opos.htm`

2 OPOS CONTROL DESCRIPTION

2.1 Concept of Operation

The OPOS specification defines a two-layered architecture between a POS application and each hardware device for optimum flexibility and device independence. The upper layer, known as the Control Object (**CO**), is a device class-specific OLE ActiveX Control used by a POS application to interact with a POS device. The CO communicates with the lower layer of the OPOS architecture, which is known as the Service Object (**SO**). The SO is implemented as an in-process OLE Automation Server (a code DLL). Each SO will interact directly with and drive a specific POS device, thus making it a vendor and device-specific component.

In the past, a POS device vendor would be responsible for providing both the service object and the control object to support their particular device. Unfortunately, experience has shown that the CO developed by one manufacturer generally does not work well with, and is not guaranteed to work with, the SO from other manufacturer. This is usually because the manufacturer develops the CO and SO concurrently and can sometimes make assumptions or take liberties in the implementation because they assume they have control over both pieces. Basically, any POS system that is "hard-coded" to use a specific brand of CO is going to run into problems interfacing with the SOs of a different brand.

This CO shortcoming was the impetus behind the independent development of a set of Common Control Objects (**CCO**s) by Curtiss Monroe Consulting Services (**MCS**) Dayton, Ohio. The goal of MCS was to develop a vendor-neutral set of COs that would be distributed free-of-charge to the POS industry. These CCOs would save the device manufacturer the effort required to develop the CO and ensure compatibility within the POS system for alternative manufacturers devices in the same class.

Unfortunately, implementation of this CCO approach comes well after there were many vendor-specific SO/CO combinations already in use in the industry. This means that some back-compatibility issues may present some initial minor difficulties when switching to the CCO approach.

The developer of the "Common Control Object" describes "Deviations from OPOS Specification" that were necessary to work under COM. Basically, these were name changes to some methods. "Claim" (for example) was changed to "ClaimDevice" and "Release" was changed to "ReleaseDevice". The methods "Claim" and "Release" can still be accessed by an app but only through the IDispatch interface (for back-compatibility). Similarly, methods “Open” and “Close” were renamed to “OpenService” and “CloseService” respectively.

He goes on further to state that "this will require changing a few lines of code in applications that use early-binding as they are updated to use these (common) control objects". Further, "...in other words, with early-binding, swapping a control object is an application change, such that these method renames can be done at that time; if the application is using late-binding, it won't need to be changed since Claim and Release are still supported by the IDispatch interface."

2.2 The Weigh-Tronix OPOS Scale Control and WTOPOSAdmin Program

The Weigh-Tronix OPOS Scale Control is an implementation of the current release of the OPOS specification.

Installation of the WTOPOSAdmin program will automatically install and register the combination of a Common Control Object (of the scale-device class) from OPOS/MCS, and a Service Object specifically designed by Weigh-Tronix to interface with a Weigh-Tronix/NCI POS scale. The scale must be configured for serial communication as per the Weigh-Tronix SCP-02 protocol document. These two components are collectively referred to as an OPOS (scale) Control.

In addition to installing the OPOS Control, the Weigh-Tronix WTOPOSAdmin program will provide configuration and test capabilities for the control. This program is described later in this document along with control installation instructions.

3 SYSTEM ARCHITECTURE

3.1 Architecture Overview

The system overview is depicted in Figure 3.1. The Control Object is an OLE ActiveX CCO (provided by OPOS/MCS) that implements the OPOS-defined interface for the scale device class. This control is mostly an empty wrapper that exposes methods, properties, and events, which in turn interact with the Service Object. The control object implements almost no logic of its own, except for error checking and reporting related to the *closed state*, or the state of the control before it binds to the Service Object.

The Service Object is one level closer to the hardware and implements a device driver for the Weigh-Tronix POS scale. This component uses the Win32 Serial API and other ActiveX controls to communicate and interact directly with the scale.

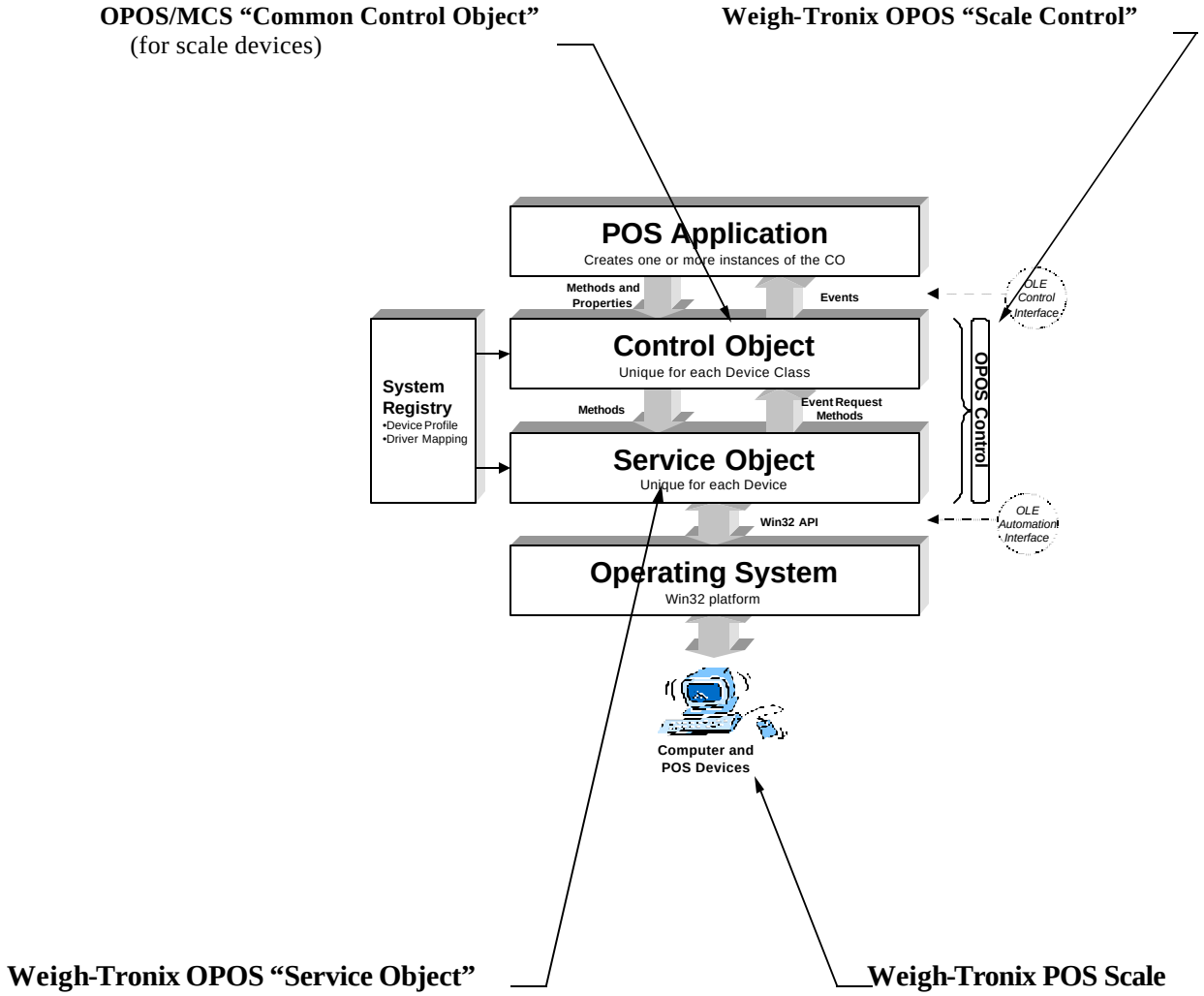


Figure 3-1. System Architecture

4 SOFTWARE/HARDWARE SPECIFICATION

4.1 Software Specifications

The OPOS Service Object (SO) software has been developed to work under Windows® 95 (OSR2), Windows 98®, Windows® NT 4.0 (SP6), and Windows 2000 operating systems.

The SO is opened and bound by the CCO. The SO is responsible for executing its exposed methods when called upon by the CCO. These methods are used to set and retrieve property values, as well as perform various operations such as reading the weight from the scale. Upon successfully opening, the SO will load a named Device Profile (i.e. Scale Profile) specified by the calling CCO from the system registry. The Scale Profile describes the behavior and connection settings of the scale hardware that the SO will drive.

When the SO is enabled (refer to the *DeviceEnabled* property of the OPOS A.P.G.), it will open and initialize a communication channel to the serial port specified by the loaded Device Profile.

Whenever the device I/O handler requests and receives data from the scale, it packages it up and enqueues the data for future processing. This data is later dequeued by the event dispatch handler. In addition to data-related events, the event dispatch handler may dequeue error-related events, status update events, and others supported by the OPOS scale specification. The event dispatch handler will then process the data event and fire up a data event to the Common Control Object, via the CCO's corresponding Event Request Method.

When the event dispatch handler fires a data event, depending on the current state of the device properties, it may have to disable the device (refer to the *AutoDisable* property of the OPOS A.P.G.).

4.2 Hardware Specifications

The OPOS Service Object itself requires no special hardware. It is a software-only component. However, a weigh-scale is (of course) a necessary peripheral to perform the weighing requirements of the POS application. The SO is designed to be used with a Weigh-Tronix/NCI Model 67XX (or similar model) scale that communicates using the Weigh-Tronix SCP-02 Serial Communications Protocol. While newer model scales support special OPOS extensions to the scales serial protocol, legacy scales can still be used and special items configured on the 'Advanced' tab in the Administration program.

5 IMPLEMENTATION

This section documents the details of the Weigh-Tronix Service Object implementation.

5.1 Device Profile Registry Settings

The Device Profile is the set of configuration settings that describe the operational characteristics of a specific instance of a Weigh-Tronix POS scale. The Device Profile is stored in the Windows® Registry under a well known key as specified by the OPOS specification. All device instance information is rooted at the following registry key:

HKEY_LOCAL_MACHINE\SOFTWARE\OLEforRetail\ServiceOPOS

Under this key will exist sub-keys for each device class. For the scale device class, this key is **Scale**. Under the Scale key, a Device Profile is stored within a sub-key of its own. The name of the Device Profile key is the identifier passed from the Control Object to the Service Object for the *DeviceName* parameter in the *OpenService* method.

As per the OPOS specification, all OPOS registry properties are implemented as string values. However, each property can take on a logical type of String or Number. In addition, the Weigh-Tronix OPOS Control supports embedded comments in the value of the property to facilitate manual device configuration. The format of a registry property value is as follows:

<value> ||| <comment string>

The value of the property is delimited by the comment indicator, which is three vertical bars (|||) with no spaces in between. The comment indicator, the white space preceding it, and all text following it will be safely ignored by the Weigh-Tronix Scale OPOS Service Object, as well as all tools distributed with the Weigh-Tronix OPOS Control that interface with the registry.

Some Scale Profile settings are defined as OPOS standard while others are Weigh-Tronix OPOS-specific settings. The following table summarizes these.

Value	Logical Type	Description
(default)	String	The default value of a device instance key is the OLE Programmatic Class ID of the Service Object that will drive this device instance. Prog ID: **WTScaleSO.clsWTSCL**
ServiceObject	String	This is set to the filename of the actual Service Object Dynamic Link Library (DLL). Service Object: **WTScaleSO.DLL**
ServiceObjectVersion	Number	The is the version of the Service Object. This setting indicates what level of OPOS compliance the Service Object meets. This can be used to resolve compatibility issues with the Control Object and host application.
ServiceObjectDescription	String	A brief description of the Weigh-Tronix Service Object **WTScaleSO.DLL**
Description	String	A description of the device instance. The WTOPOSAdmin configuration tool of the OPOS Control distribution places a datestamp here when this Device Profile is created.
AdminVersion	String	The version of the Administration program **WTScaleAdmin.EXE**
ModelNumber	String	Hard-coded to "SCP-02" which the protocol used by the scale to communicate with the Service Object.
SerialPort	String	Configured by user in the 'Scale Profile' section of the administration program. Values range from COM1 to COM9.
BaudRate	Number	Configured by user in the 'Scale Profile' section of the administration program. Values included 1200, 2400, 4800, 9600, 19200. Default = 9600 **Must match the baud configuration setting in the scale.**
Parity	String	Configured by user in the 'Scale Profile' section of the administration program. Values include Even, Odd and None. **Must match the parity configuration setting in the scale.**
DataBits	Number	Configured by user in the 'Scale Profile' section of the administration program. Values include 7 or 8. **Must match the data bits configuration setting in the scale (if separate setting).**
Initial_MaximumWeight	Number	MaximumWeight property. Configured by user in the 'Advanced' section of the administration program. Typically used when the scale is not able to return this 'capability' property.
Initial_WeightUnits	String	WeightUnits property. Configured by user in the 'Advanced' section of the administration program. Typically used when the scale is not able to return this information.
Initial_CapDisplay	Boolean	CapDisplay property. Configured by user in the 'Advanced' section of the administration program. Typically used when the scale is not able to return this 'capability' property.
Initial_CapDisplayText	Boolean	CapDisplayText property. Configured by user in the 'Advanced' section of the administration program. Typically used when the scale is not able to return this 'capability' property.
Initial_CapPriceCalculating	Boolean	CapPriceCalculating property. Configured by user in the 'Advanced' section of the administration program. Typically used when the scale is not able to return this 'capability' property.
Initial_CapTareWeight	Boolean	CapTareWeight property. Configured by user in the 'Advanced' section of the administration program. Typically used when the scale is not able to return this 'capability' property.
Initial_CapZeroScale	Boolean	CapZeroScale property. Configured by user in the 'Advanced' section of the administration program. Typically used when the scale is not able to return this 'capability' property.
Initial_MaxDisplayTextChars	Number	MaxDisplayTextChars property. Configured by user in the 'Advanced' section of the administration program. Typically used when the scale is not able to return this information.

5.2 The Weigh-Tronix Scale-Device Profile

Device Profiles provide many conveniences for the POS system administrator. First, it allows a single Service Object driver to be used to control multiple scales. For example, a POS terminal may have two serial ports and a different Weigh-Tronix/NCI POS scale connected to each. We can create a **Scale Profile** for each scale, yet we only need a single Service Object to drive both scales. Second, it allows replacement scales to be used (in many cases) by simply adding a new profile in the Administration program with known serial communications settings. Scales are shipped from the factory with default communications settings. This saves having to open some scales in order to change those configuration settings to match an existing profiles settings. Finally, it allows a system to use any combination of new or legacy Weigh-Tronix/NCI scales in the same system. New Weigh-Tronix/NCI POS scales provide communications extensions which provide certain scale-capability attributes to the SO. Legacy scales (that do not have this capability), nevertheless may continue to be used because the Administration program allows these attributes to be set (on the 'Advanced' tab) and saved to the scales associated profile.

The following figure shows a screen snapshot within REGEDIT of a typical Weigh-Tronix/NCI scale profile.

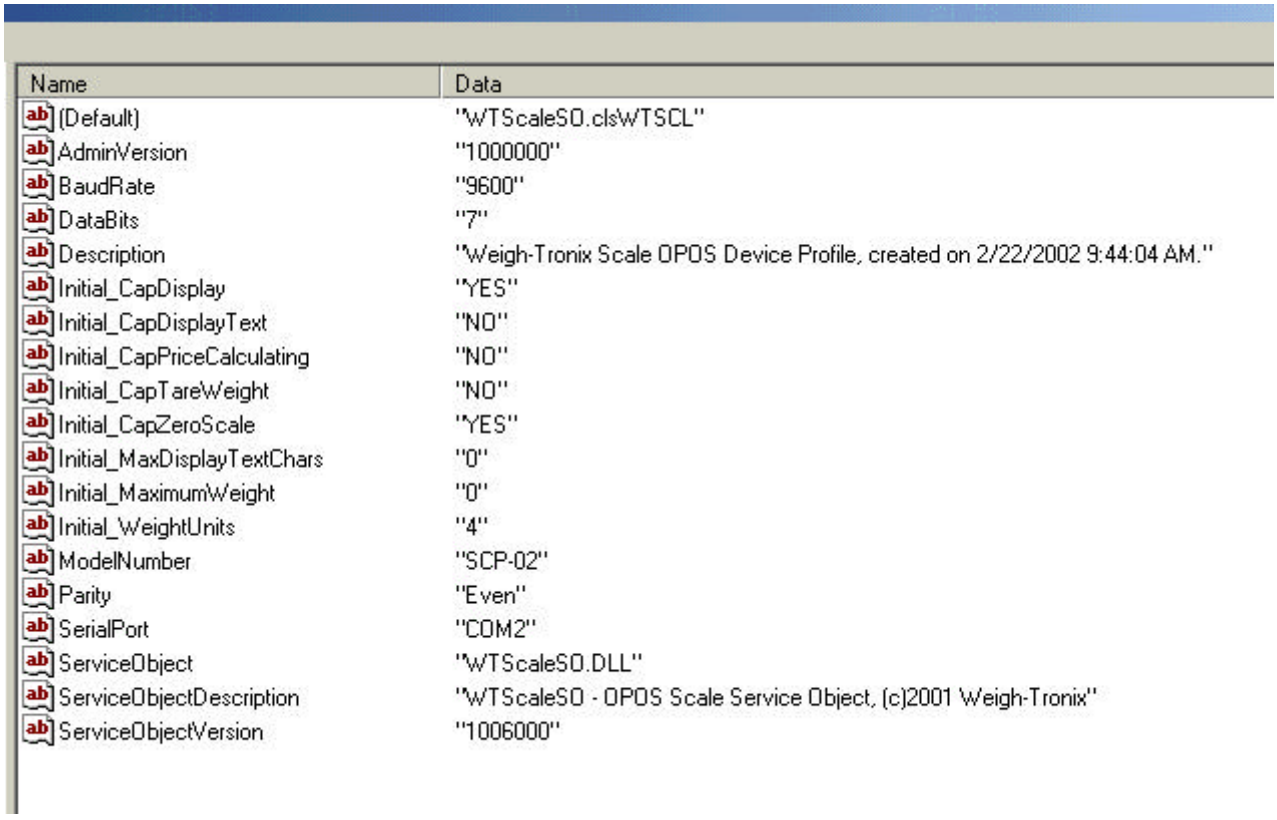


Figure 5-1. Sample Registry Scale Profile

This example shows a typical Scale Profile. The default value of the Device (i.e. Scale) Name key is the Service Object's programmatic ID **WTScaleSO.clsWTSCL** while the filename of the Service Object is **WTScaleSO.DLL**.
Notice that the description field shows when the scale profile was created.

In addition to serial communications setings, the profile also contains settiings for various scale capabilities that might be desired to be associated with a particular scale. These settings are prefixed by "**Initial_**" and then use the actual property name as defined in the OPOS specification. The values shown in the sample happen to be the default values that are automatically assigned. The user may override them by changing the values in the '**Advanced**' section of the Administration program.

5.3 Service Provider Registry Values

In addition to the registry settings associated with each Device Profile, the OPOS specification makes provisions for registry settings that affect all device instances for a common service provider (vendor) globally. For the Weigh-Tronix OPOS Control, this key is located under the following path:

`HKEY_LOCAL_MACHINE\SOFTWARE\OLEforRetail\ServiceInfo\Weigh-Tronix`

The current release of the Weigh-Tronix Scale Service Object does not use any vendor-specific settings.

5.4 The Weigh-Tronix Scale Service Object (SCP-02)

The Weigh-Tronix Scale OPOS Service Object is an OLE in-process server which drives a Weigh-Tronix/NCI POS scale that uses the SCP-02 serial communications protocol.

Device Class:	Scale
Device Sharing Rule:	Exclusive
OLE Prog. Class ID:	WTScaleSO.clsWTSCL
Device Driver File:	WTScaleSO.DLL

The following tables summarize the implemented interface to the Service Object as specified by the OPOS Application Programmer's Guide.

Common Properties

Common	*Type*	*Access*	*Scale Cmd*	*May Use After*	*Comments and Notes*
AutoDisable	Boolean	R/W		Open	ok
BinaryConversion	Long	R/W		Open	ok, (OPOS_BC_NONE only!)
CapPowerReporting	Long	R		Open	ok, (OPOS_PR_NONE only!)
CheckHealthText	String	R		Open	ok
Claimed	Boolean	R		Open	ok
DataCount	Long	R		Open	ok
DataEventEnabled	Boolean	R/W		Open	ok
DeviceEnabled	Boolean	R/W		Open & Claim	ok
FreezeEvents	Boolean	R/W		Open	ok
OpenResult	Long	R		--	ok
OutputID	Long	R		*Not Supported*	*Not Supported*
PowerNotify	Long	R/W		Open (& Not Enabled)	ok, (OPOS_PR_NONE only!)
PowerState	Long	R		Open	ok, (OPOS_PS_UNKNOWN only!)
ResultCode	Long	R		--	ok
ResultCodeExtended	Long	R		Open	ok
State	Long	R		--	ok
ControlObjectDescription	String	R		--	ok, (Response handled by CO)
ControlObjectVersion	Long	R		--	ok, (Response handled by CO)
ServiceObjectDescription	String	R		Open	ok
ServiceObjectVersion	Long	R		Open	ok
DeviceDescription	String	R		Open	ok
DeviceName	String	R		Open	ok

Common & Scale-Specific Properties

Common	*Scale Cmd*	*May Use After*	*Comments and Notes*
Open		--	ok, (OpenService in S.O.)
Close		Open	ok, (CloseService in S.O.)
ClaimDevice		Open	ok
ReleaseDevice		Open & Claim	ok
CheckHealth		Open, Claim, & Enable	ok, (no OPOS_CH_EXTERNAL)
ClearInput		Open & Claim	ok, (note 6)
ClearOutput		*Not Supported*	*Not Supported*
DirectIO		Open	*Not Implemented (at this time)*
Specific (i.e. Scale)			
DisplayText	**D**	Open, Claim, & Enable	*Not Implemented (at this time)* **[W], [BC]**
ReadWeight	**W**	Open, Claim, & Enable	ok **[R]**
ZeroScale	**Z**	Open, Claim, & Enable	ok **[W]**

Scale-Specific Properties

Specific (i.e. Scale)	*Type*	*Access*	*Scale Cmd*	*May Use After*	*Comments and Notes*
CapDisplay	Boolean	R	**A**	Open, *(Claim & Enable)*	ok, (notes 1,2) **[R], [*]**
CapDisplayText	Boolean	R	**A**	Open, *(Claim & Enable)*	ok, (notes 1,2) **[R], [*]**
CapPriceCalculating	Boolean	R	**A**	Open, *(Claim & Enable)*	ok, (notes 1,2) **[R], [*]**
CapTareWeight	Boolean	R	**A**	Open, *(Claim & Enable)*	ok, (notes 1,2) **[R], [*]**
CapZeroScale	Boolean	R	**A**	Open, *(Claim & Enable)*	ok, (notes 1,2) **[R], [*]**
AsyncMode	Boolean	R/W		Open	ok
MaxDisplayTextChars	Long	R	**C**	Open, *(Claim & Enable)*	ok, (note 2) **[R], [*]**
MaximumWeight	Long	R	**m**	Open, *(Claim & Enable)*	ok, (note 2) **[R], [*]**
SalesPrice	Currency	R	**s**	Open, Claim, & Enable	ok **[R][W]**
TareWeight	Long	R/W	**T**	Open, Claim, & Enable	ok, (note 3) **[W]**
UnitPrice	Currency	R/W	**P**	Open, Claim, & Enable	ok **[W]**
WeightUnit	Long	R	**u**	Open, *(Claim & Enable)*	ok, (note 2) **[R]**

Events

Name		*May Occur After*	*Comments and Notes*
DataEvent		Open, Claim, & Enable	ok
DirectIOEvent		Open, Claim	*Not Implemented (at this time)* **[BC]**
ErrorEvent		Open, Claim, & Enable	ok, (limited, note 4)
OutputCompleteEvent		*Not Supported*	*Not Supported*
StatusUpdateEvent		Open, Claim, & Enable	*Not Implemented (at this time)*

Internal Control/Service Object Methods

Name		*May Use After*	*Comments and Notes*
COFreezeEvents		--	ok (note 5)
GetOpenResult		--	ok, (optional method is supported)

Notes:

1) These "scale capability" attributes are all returned together as a single response to an 'A' scale command.

2) These are properties that might be available with some scales but not with others. The 'Advanced' tab in the WTO-POSAdmin program allows setting of these properties for those cases where the scale is unable to furnish the attributes. Additionally, it can be used to preset the values so that they are available to be read by the application after the Service Object has been 'Opened' but before actual values can be read from the scale (i.e. after 'Enabled'). Attributes read from the scale (if supported), will always override those that were set in the 'Advanced' tab.

3) These properties require special options in the scale and are not currently available.

4) The current release of the Service Object will raise only an OPOS_EL_INPUT event and only accept accept an OPOS_ER_CLEAR in response from the application.

5) The OPOS specification is unclear as to what value should be returned when this method is called. This Service Object will return a 4-byte TRUE (i.e. a Variant Bool) which is set to a Long value of 1.

6) Although not explicitly defined in the OPOS specification, this Service Object will return an OPOS_E_CLOSED value if the application attempts to read the value after 'Open' but before 'Claim'.

[BC] refers to a method, event or property that is affected by the BinaryConversion of text strings.

[*] refers to an attribute that is determined by interrogating the scale.

[W] refers to a value which which the S.O. must be able to 'write' to the scale.

[R] refers to a value which the S.O. must be able to 'read' from the scale.

Vendor-Specific (Weigh-Tronix) Methods

These special methods allow an application to communicate directly with the Service Object for functionality that is not provided by the Common Control Object. These are special functions that are provided by Weigh-Tronix for use with their scales that may be of use to developers. Using vendor-specific methods (through the DirectIO function) means that your POS application will not be portable.

The interface mechanism used is the **DirectIO** function call. The parameter(s) passed and returned may vary from one funtion to another. In order to access any of the Weigh-Tronix (vendor-specific) methods you must include the WTCommands.BAS file (or equivalent) in your project. Alternatively, you can code equivalent constant definitions for the method names and return values as described in the following pseudo-code:

OPOS "Vendor-Specific" (Weigh-Tronix) Function Constants

Long Const **WT_MfgCmd_SIOProtocol** = 1	Read serial protocol being used
Long Const **WT_MfgCmd_0Wt_Valid** = 2	0.00 weight is a valid weight
Long Const **WT_MfgCmd_0Wt_NotValid** = 3	0.00 weight is not a valid weight
Long Const **WT_MfgCmd_EnableTransactionRead** = 4	Enable reading of price computing transaction
Long Const **WT_MfgCmd_DisableTransactionRead** = 5	Disable reading of price computing transaction
Long Const **WT_MfgCmd_ReadTransaction** = 6	Read a price computing transaction from scale

Vendor-Specific (Weigh-Tronix) Error Codes

OPOS "ResultCodeExtended" Constants

Long Const **WT_MFG_NULL** = 0
Long Const **WT_MFG_ECRPROTO** = 1
Long Const **WT_MFG_OPOSPROTO** = 2

Long Const **WT_MFG_VALIDLATCHEDVALUE** = 80
Long Const **WT_MFG_INVALIDLATCHEDVALUE** = 81
Long Const **WT_MFG_TRANSACTIONLATCHDISABLED** = 82
Long Const **WT_MFG_TRANSACTIONLATCHENABLED** = 83

Long Const **WT_MFG_OTHERERROR** = 99

Calling Vendor-Specific Methods:

The **DirectIO** command accepts three parameters; a *Command* (command number), a *pData* (pointer to additional optional data) and a *pString* (pointer to additional optional string data). The function returns a Long value to indicate the OPOS_SUCCESS or OPOS_E_FAILURE of the operation. The syntax of the DirectIO function is:

LONG DirectIO (LONG *Command*, LONG * *pData*, BSTR* *pString*)

for example (in pseudo-code);

```
long lret
long result

lret = OPOSScale1.DirectIO(WT_MfgCmd_SIOProtocol, 0, "")
IF (lret = OPOS_SUCCESS)
THEN result = OPOSScale1.ResultCodeExtended
```

will request that the S.O. interrogate the scale and determine which serial protocol is being used. For the Model 67XX scale the protocol will be either the Standard ECR or the ECR with OPOS extensions.

If the value returned from the call is OPOS_SUCCESS, then the application may then read the **ResultCodeExtended** (i.e. **RCE**) property for the actual result of the special inquiry. In this particular example, the value returned will be either **WT_MFG_ECRPROTO** (for the *Standard* protocol) or **WT_MFG_OPOSPROTO** (for the *ECR with OPOS extensions*) protocol.

Summary of Vendor-Specific Methods:

Vendor Command *(Command)*	*(pData)*	*(pString)*	*May Use After*	*Return Values (in RCE)*
WT_MfgCmd_SIOProtocol (determine scale protocol type)	**0**	**“”** (null)-	Open, Claim, Enable	**WT_MFG_ECRPROTO** **WT_MFG_OPOSPROTO**
WT_MfgCmd_0Wt_Valid (allow `0.0` weight from scale)	**0**	**“”** (null)	Open [2]	**WT_MFG_NULL**
WT_MfgCmd_0Wt_NotValid (don’t allow `0.0` weight from scale)	**0**	**“”** (null)	Open [2]	**WT_MFG_NULL**
WT_MfgCmd_EnableTransactionLatch (enable price computing transaction reads from scale)	**0**	**“”** (null)	Open, Claim, Enable	[1]
WT_MfgCmd_DisableTransactionLatch (disable price computing transaction reads from scale)	**0**	**“”** (null)	anytime [3]	[1]
WT_MfgCmd_ReadTransaction (read price computing transaction from scale)	***Timeout*** [5]	**“”** (null)	Open, Claim, Enable	[4]

Notes:

[1] **WT_MFG_TRANSACTIONLATCHENABLED** or **MFG_TRANSACTIONLATCHDISABLED**
[2] Open is required due to the use of the DirectIO command.
[3] Even though DirectIO command requires that the device be Opened, we will still disable the latch even if device is closed.
[4] **WT_MFG_VALIDLATCHEDVALUE** or **WT_MFG_INVALIDLATCHEDVALUE**
[5] The LONG ***Timeout*** value is the number of milliseconds to wait for settled weight. Please refer to ***Read Weight*** method in the OPOS Application Programmers Guide for details.

Price Computing Transaction Function Return Values:

The group consisting of *weight*, *unit pr*ice and *sales price* are considered to be the price computing transaction values. Normally, these values are aquired from the scale using separate OPOS Service Object function; ***ReadWeight***, ***UnitPrice*** and ***SalesPrice*** respectively. Accessing them in this way however means that there may be some (indeterminate) time between each function call. Therefore, the possibility exists that all three values of the transaction (after reading them individually), may no longer correspond to the instantaneous transaction conditions that were present when acquisition began. This could result in a total sales price calculation that does not agree with the weight and unit price values.

To prevent these errors, Weigh-Tronix has implemented a set of three vendor-specific functions that allow the transaction data to all be aquired simultaneously and '*latched*'. In order to return the correct data types to the application that is making the calls, the standard OPOS functions *ReadWeight, UnitPrice* and *SalesPrice* will still be used. However, the values returned for the function(s) as well as the **ResultCode** and **ResultCodeExtended** will be based on the 'latched' reading.

In order to use the standard functions to access the latched transaction values, the application must first have called the **WT_MfgCmd_EnableTransactionLatch** (i.e the *enable*) function and then the **WT_MfgCmd_ReadTransactionLatch** (i.e. the *read*) function successfully**.** Subsequently, all uses of the standard *ReadWeight, UnitPrice* and *SalesPrice* will return only the *latched* transaction values not the current scale values! This will continue until such time as the application releases the latched transaction mode by calling the **WT_MfgCmd_DisableTransactionLatch** (i.e. the *disable*) function.

While accessing the latched transaction data, the application should monitor the function return value and/or the **ResultCode** after calling *ReadWeight, UnitPrice* and *SalesPrice*. If the value of ResultCode is OPOS_E_FAILURE, then the application should check the **ResultCodeExtended** value for a further description of the error. A special vendor-specific error code will be placed there that describes the error relating to transaction data access. These error codes are described in the section entitled Vendor-Specific (Weigh-Tronix) Error Codes on page 15.

Description of Vendor-Specific Methods:

Method: **WT_MfgCmd_SIOProtocol**

Remarks: Request to S.O. to determine what serial protocol is being used by the scale.

Syntax: **`OPOSScale1.DirectIO(WT_MfgCmd_SIOProtocol, 0, "")`**

Usage: May use after Open, Claim and Enable

Return: Function returns OPOS_SUCCESS or OPOS_E_FAILURE

RCE: If function returns OPOS_SUCCESS then the **OPOSScale1.ResultCodeExtended** property should be read immediately for one of the return values:

WT_MFG_ECRPROTO
Means that the scale is communicating using the standard Weigh-Tronix ECR protocol. This is the "legacy" protocol.

WT_MFG_OPOSPROTO
Means that the scale is communicating using the Weigh-Tronix ECR protocol with OPOS extensions. This is the latest protocol for use with the Weigh-Tronix Service Object.

Method: **WT_MfgCmd_0Wt_Valid**

Remarks: Instructs the S.O. to return the scale weight (in response to the ***ReadWeight*** method) even if the weight is zero (i.e. **`0.00`**). The OPOS specification states that a weight value is *not* to be returned when it is zero. This command overrides the standard OPOS operation.
See also: WT_MfgCmd_0Wt_NotValid.

Syntax: **`OPOSScale1.DirectIO(WT_MfgCmd_0Wt_Valid, 0, "")`**

Usage: May use after Open

Return: Function returns OPOS_SUCCESS or OPOS_E_FAILURE

RCE: This is a direct command. That is, it is not necessary to read a returned value. However, if you choose to do so, and if the function returns OPOS_SUCCESS, then the **OPOSScale1.ResultCodeExtended** property should be read immediately for the return value:

WT_MFG_NULL

Description of Vendor-Specific Methods (continued):

Method: **WT_MfgCmd_0Wt_NotValid**

Remarks: This is the corollary to WT_MfgCmd_0Wt_Valid. It instructs the S.O. to restore the standard OPOS weight response criteria so such that the scale weight (in response to the ***ReadWeight*** method) will be returned only if the weight value is non-zero.
See also: WT_MfgCmd_0Wt_Valid.

Syntax: **`OPOSScale1.DirectIO(WT_MfgCmd_0Wt_NotValid, 0, "")`**

Usage: May use after Open

Return: Function returns OPOS_SUCCESS or OPOS_E_FAILURE

RCE: This is a direct command. That is, it is not necessary to read a returned value. However, if you choose to do so, and if the function returns OPOS_SUCCESS, then the **OPOSScale1.ResultCodeExtended** property should be read immediately for the return value:

WT_MFG_NULL

Method: **WT_MfgCmd_EnableTransactionLatch**

Remarks: This function instructs the S.O. to return *'latched'* price computing transaction values when subsequently requested using the standard ***ReadWeight*** method, ***UnitPrice*** read-property and ***SalesPrice*** read-property.
This method must be followed by a successful **WT_MfgCmd_ReadTransaction** function call in order to aquire the latched price computing transaction values from the scale.

See Also: WT_MfgCmd_DisableTransactionLatch

Syntax: **`OPOSScale1.DirectIO(WT_MfgCmd_EnableTransactionLatch, 0, "")`**

Usage: May use after Open, Claim and Enable

Return: Function returns OPOS_SUCCESS or OPOS_E_FAILURE

RCE: This is a direct command. That is, it is not necessary to read a returned value. However, if you choose to do so, and if the function returns OPOS_SUCCESS, then the **OPOSScale1.ResultCodeExtended** property should be read immediately for the return value:

WT_MFG_TRANSACTIONLATCHENABLED
or
WT_MFG_TRANSACTIONLATCHDISBLED

Note: **See the discussion regarding price computing transaction function return values on page 18.**

Description of Vendor-Specific Methods (continued):

Method: **WT_MfgCmd_DisableTransactionLatch**

Remarks: This function instructs the S.O. to return to 'normal' operation whereby it returns separately obtained price computing transaction values (from the scale) when requested by the standard ***ReadWeight*** method, ***UnitPrice*** read-property and ***SalesPrice*** read-property.

See Also: WT_MfgCmd_EnableTransactionLatch

Syntax: `OPOSScale1.DirectIO(WT_MfgCmd_DisableTransactionLatch, 0, "")`

Usage: May use after Open

Return: Function returns OPOS_SUCCESS or OPOS_E_FAILURE

RCE: This is a direct command. That is, it is not necessary to read a returned value. However, if you choose to do so, and if the function returns OPOS_SUCCESS, then the **OPOSScale1.ResultCodeExtended** property should be read immediately for the return value:

WT_MFG_TRANSACTIONLATCHENABLED
or
WT_MFG_TRANSACTIONLATCHDISBLED

Note: **See the discussion regarding price computing transaction function return values on page 18.**

Description of Vendor-Specific Methods (continued):

Method: **WT_MfgCmd_ReadTransaction**

Remarks: This function instructs the S.O. to aquire *' latched'* price computing transaction values from the scale. This means that a single command to the scale wil return weight, unit-price and total sales-price simultaneously thereby guaranteeing that there is no time difference between the values.

Before this function can be successfully used, the **WT_MfgCmd_EnableTransactionLatch** vendor-specific function must have been previously called. After the 'transaction latch' has been enabled, this vendor-specific **...ReadTransaction** method may be called. Each time it is called it will aquire the transaction values from the scale and latch them in it's internal registers. The application may then ac ces the individual values by using the standard OPOS ***ReadWeight*** method, ***UnitPrice*** read-property, and ***SalesPrice*** read-property. These standard functions will return the most recently 'latched' transac tion values rather than request them from the scale.

NOTE: It is important to remind the application programmer that the same values will remain latched and returned unless the **...ReadTransaction** method is called again to update the values. Also, latched values will continue to be returned (by the standard OPOS commands) until the WT_MfgCmd_DisableTransactionLatch function is called.

See Also: WT_MfgCmd_EnableTransactionLatch, WT_MfgCmd_DisableTransactionLatch

Syntax: `OPOSScale1.DirectIO(WT_MfgCmd_ReadTransaction, 0, "")`

Usage: May use after Open, Claim and Enable

Return: Function returns OPOS_SUCCESS or OPOS_E_FAILURE

RCE: This is a direct command. That is, it is not necessary to read a returned value. However, if you choose to do so, and if the function returns OPOS_SUCCESS, then the **OPOSScale1.ResultCodeExtended** property should be read immediately for the return value:

WT_MFG_VALIDLATCHEDVALUE
or
WT_MFG_INVALIDLATCHEDVALUE

If the function returns OPOS_E_FAILURE, a check of the ResultCodeExtended value will likely indicate the error condition:

WT_MFG_TRANSACTIONLATCHDISABLED

Note: **See the discussion regarding price computing transaction function return values on page 18.**

6 INSTALLATION

This section describes how to install the Scale Object, Common Control Object and the configuration/test program.

6.1 OPOS Scale Control & WTOPOSAdmin Installation

A single self-extracting installation file contains all necessary software components to be installed. This file is called **WTOPOS.EXE** and includes:

- The Weigh-Tronix OPOS Scale Service Object (**WTScaleSO.DLL**)
- The Scale-Class Device Common Control Object (**OPOSScale.OCX**) (courtesy of OPOS administration & RCS)
- The Weigh-Tronix Administrator Configuration/Test Program (**WTOPOSAdmin.EXE**).

The installation process will begin by simply double-clicking on the WTOPOS.EXE file. All necessary files will be automatically extracted and installed. In addition, the WTScaleSO.DLL and OPOSScale.OCX components will be registered in the system registry.

After the installation process is complete, you can launch the WTOPOSAdmin configuration/test program from the "start" button (assuming you have installed to the default names and directories) by selecting:

`Start, Programs, Weigh-Tronix WTOPOSAdmin, Weigh-Tronix WTOPOSAdmin`

6.2 OPOS Scale Control & WTOPOSAdmin Uninstall

If you need to uninstall the application, you should do so through the Windows ® Control Panel.

Click on: **`Start, Settings, Control Panel`**, then double-click on **`Add/Remove Programs`**, click on the **`Install/Uninstall`** tab and then locate and click on the **`Weigh-Tronix WTOPOSAdmin`** entry.

7 USER INTERFACE

7.1 User Interface (WTOPOSAdmin) Overview

A user-friendly administration program allows users to install, configure, and test the OPOS scale control for subsequent use by the main POS application program. The application is called **WTOPOSAdmin.EXE** (aka the administrator). The administrator user interface is presented as a set of tabbed folders. Each folder provides a group of related items.

7.1.1 Scale Profile Tab (see Appendix-C)

This is where the serial communications parameters (and other parameters) are set to interface with the scale.

Once set, they are saved in the Windows® Registry by a *Profile Name* that you choose. Any number (up to a maximum of 100) profiles may be saved. New profiles may be added and old ones removed. Any changes to profile parameters that are made will automatically be updated in the Registry without the need to click a "save" button. At least one properly configured profile must be established for proper operation of the scale control in the POS application. Other detailed information regarding the Service Object is also displayed but cannot be changed.

7.1.1.1 Creating New Profiles

To create a new scale device profile click on the **Add** button on the *Scale Profile* tab panel.

A dialog box will prompt you to enter your new *Device Name*. Type in the name that you desire to be associated with this profile, and then click the **Ok** button. If the device name is allowed it will automatically be added to the system Registry and will have the following serial communications defaults saved: COM1, 9600 baud, 7 data bits, Even parity.

IMPORTANT: The serial communications settings that you choose for *Serial Port*, *Baud R*ate, *Data Bits* and *Parity* must match those as set in the scales Configuration menu. Unless otherwise specified, the *Data Bits* value must be set to **7** to work with most Weigh-Tronix scales. Also, the protocol setting in the scale must be **`Ecr`**.

The name in the *Profile Name* selection box will revert back to the first profile selection in the list. To make changes to the profile you just created, you must click on the *Profile Name* drop-down list box and select your profile by clicking on it's name. Then, you may change any communications or advanced settings. Each change will automatically update the profile in the system Registry.

7.1.1.2 Deleting Profiles

To delete an existing scale device profile, first select the desired profile so that it is displayed in the *Profile Name* drop-down list box, then click the **Remove** button. The profile will be removed from the system Registry and the The name in the *Profile Name* selection box will revert back to the first profile selection in the list.

7.1.2 Scale Test Tab (see Appendic-C)

This provides a simple means of testing and verifying proper operation of the scale, it's configuration settings, interconnect cable and serial communications port on the host PC. It is a good means of testing because it completely bypasses the CCO and SO control software.

First, you must determine the current communications settings of the scale. This is done by entering the scales *'Configuration'* mode. Please refer to the scales User's Manual for instructions. Note the scales current baud rate and parity settings.

IMPORTANT: The scale protocol must be set to the **`Ecr`** selection for proper operation. Please verify the protocol setting while in the configuration mode.

In the Admin program, select the *Scale Test* tab and set the *Baud Rate*, *Data Bits* and *Parity* settings the same as they are set in the scale. Then, select a known available COM port in the *Serial Port* list box. From the *Scale Function* list box, select the desired command to be sent to the scale and press the **Send** button. The scales response will be indicated in the *Scale Response* display.

A detailed description of the commands and expected responses are described in the Weigh-Tronix document entitled;

SERIAL COMMUNICATIONS PROTOCOL:
SCP-02 (ECR STANDARD and OPOS EXTENSIONS)
P/N: 8408-14788-02

Note: Not all scales will support the OPOS extension commands.

Responses from the scale should be immediate (i.e. less than 100mS). If you do not receive a responses, verify all settings and cable connections.

7.1.3 Common Tab (see Appendix-C)

This section was originally created to aid in the development, testing and validation of the Service Object control software at Weigh-Tronix. It's purpose is to exercise all common properties, methods and events for the scale control. There are **no required settings** that must be made on this tab in order to make the CCO/SO function properly. However, it can be a very convenient debugging tool for POS application developers.

All property, method and event names are identical to those defined in the OPOS specification. Read-only properties are displayed in yellow text boxes. Properties that may be written to are presented in white selection list boxes or as option check boxes. Methods are accessed as buttons with method names in bold text. Some methods (e.g. **ClaimDevice**) require parameters to be passed when the method is called. Those methods will have an associated selection list box that allows an appropriate parameter to be selected.

The **ResultCode** and **ResultCodeExtended** properties are usually updated after each method call or setting of properties. However, the values displayed will be for the most recent method call (or property update). If the test panel internally calls multiple methods (or properties) in succession to perform an operation, then only the final result code will be displayed. For the users convenience, the ResultCode and ResultCodeExtended are also displayed in the status bar at the bottom of the panel in **RC** and **RCE** respectively. The status bar also indicates the current *Opened, Claimed* and *Enabled* state of the device.

Finally, the Common tab panel also indicates the version and description of the scale-class Common Control Object that is being used.

7.1.4 Specific Tab (see Appendix-C)

This section was originally created to aid in the development, testing and validation of the Service Object control software at Weigh-Tronix. It's purpose is to exercise all specific scale-device properties, methods and events for the scale control. There are **no required settings** that must be made on this tab in order to make the CCO/SO function properly. This tab panel can also be a very convenient debugging tool for POS application developers.

All property, method and event names are identical to those defined in the OPOS specification. Read-only properties are displayed in yellow text boxes. Properties that may be written to are presented in white text-entry boxes or as option check boxes. Methods are accessed as buttons with method names in bold text. Some methods (e.g. **ReadWeight**) require parameters to be passed when the method is called. Those methods will have an associated selection list box that allows an appropriate parameter to be selected.

WARNING: If you click on the **ReadWeight** method button and it's parameter is set to **OPOS_FOREVER**, the button will dim and the Admin progrogram will wait forever (if necessary) for there to be a valid, stable, non-zero weight to be returned from the scale. If any conditions prevent that from occuring you may click on the **ZeroScale** button to terminate the wait.

7.1.5 Advanced Tab (see Appendix-C)

This section allows POS system administrator personnel to set some scale attributes that might not be available directly from the scale itself. Current Weigh-Tronix/NCI scales (for POS applications) have the ability to return information to the Service Object that define the scales capabilities. These attributes are then available to the POS application by making calls to the service object. Older legacy scales, however, did not have the ability to return those scale attributes therefore this *Advanced* tab was added to provide a means of setting attributes and associating them with a particular scale profile.

For example, if a legacy scale does not have the inherent ability to return the value of the maximum weighing capacity, then the system administrator could manually enter a value in the **MaximumWeight** entry box on the *Advanced* tab panel. A `30` lb capacity scale would require a value of `30000` to be entered (in order to comply with the OPOS Specification which states that MaximumWeight property has an implied decimal point of three places). Now when a POS application accesses one of these properties it will receive a legitimate value.

NOTE: If the scale is capable of returning these attributes, then the actual values read from the scale will override those that are set on the *Advanced* tab panel. The scale is not interrogated for these values until the device is Enabled. Therefore, any of these attributes that are read after *Open* and *Claim* (but before *Enable*) will return the values that are displayed on the *Advanced* tab panel.

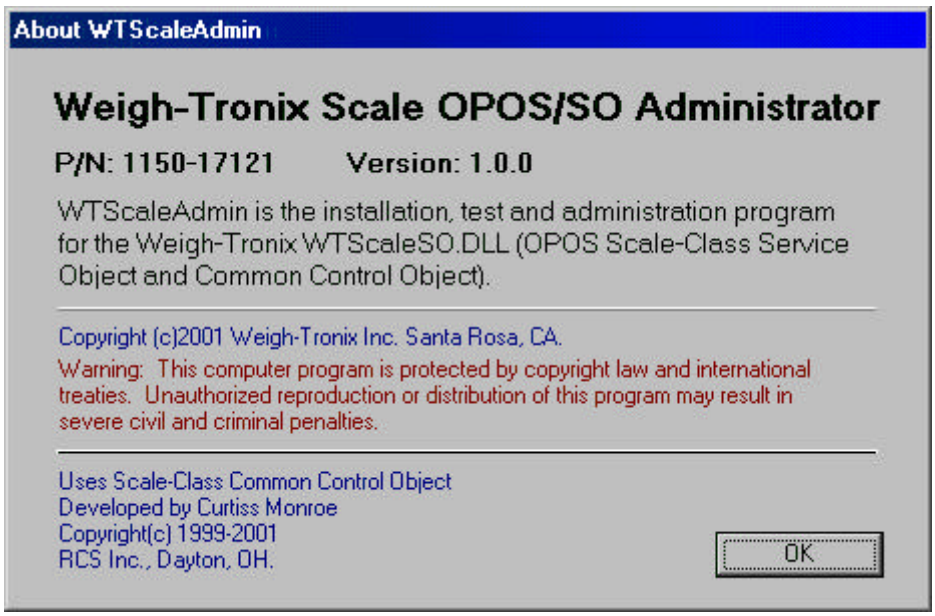
Vendor-Specific Commands:

The Advanced tab also has a section that allows the testing and simulation of a limited set of commands that are unique to the Weigh-Tronix Service Object. These commands are described in detail beginning on p.15. Please note that using any of these commands will make your POS application program non-portable.

7.1.6 About WTScaleAdmin

The version number describes the software version of the Administrator program only. It is not related to the Common Control Object or Service Object software version. Those version numbers may be viewed on the *Common* tab panel.

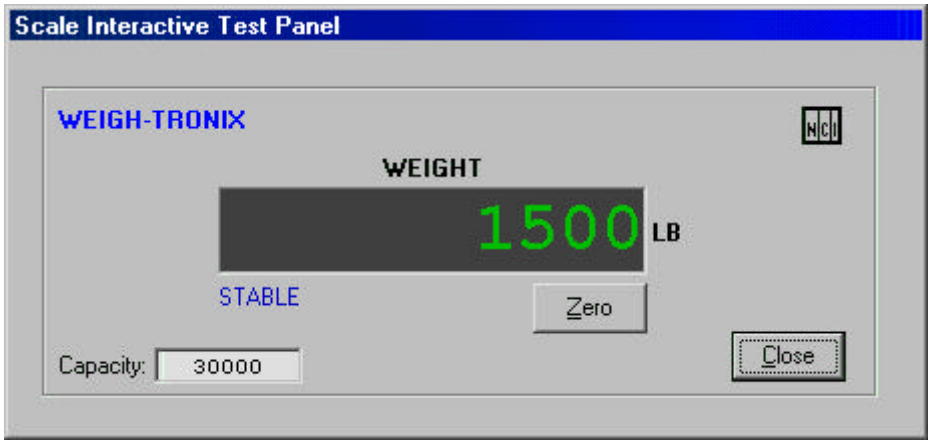
The Common Control Object is provided courtesy of the OPOS Committee and RCS.
The Service Object was developed by Weigh-Tronix.
Installing the WTOPOSAdmin program also automatically installs and registers the CCO and SO.



7.1.7 Scale Interactive Test Panel

The OPOS Specification requires that the Service Object provide the ability to perform an interactive test of the (scale) device. The SO typically should display a modal dialog box to present test options an results.

The following scale test dialog box is displayed in response to a call the **CheckHealth** method (with **OPOS_CH_INTERACTIVE** passed as the Level parameter). The Administrator program provides convenient access to this test program by clicking on the **CheckHe alth** button in the Comon Methods section of the *Common* tab panel.



In operation, the panel will dynamically display the current weight on the scale with its units-of-measure. In addition, it will indicate whether the weight is STABLE (or not) and error conditions OVER CAPACITY or UNDER CAPACITY (if/when applicable).

The Capacity label indicates the maximum rated capacity of the scale and is the value read from the scale (if supported) or the value entered on the *Advanced* tab panel. Clicking on the **ZERO** button will cause the weight on the scale to be cleared to zero if all scale-zeroing criteria are met. The example above shows a weight reading of `1.5` pounds on a `30` pound capacity scale.

NOTE: As required by the OPOS Secification, this is a modal dialog box and must be closed before access to the POS application will again be allowed.

7.2 WTOPOSAdmin Initial Setup and Operating Instructions Overview

With just a few simple configuration settings in the WTOPOSAdmin program (and the scale) we will be ready to verify that the control installed successfully and that it works with the scale.

- First, it is important to verify the current scale settings for Protocol, Baud Rate and Parity. To do this, please refer to the operators guide that was provided with the scale. It is necessary to become familiar with the procedure for entering the scales setup menu and for scrolling and making selections.

  a) Enter the setup menu mode by setting the appropriate rocker switch.

  b) Scroll and select "**`Conf`**" to enter the Configuration sub-menu.

  c) Scroll and select "**`Prot`**" to enter the serial protocol sub menu.

  IMPORTANT: the "**`Ecr`**" protocol must be selected. If it is not, then scroll and select the **`Ecr`** protocol at this time. The OPOS control will not work with any other protocol setting.

  d) Scroll and select "**`BAUd`**" to enter the baud rate sub-menu. Make note of the current baud rate setting, or change it to a preferred rate at this time. (Factory default is 9600 baud).

  Note: Some scales combine the parity selection with the baud rate. Other scales have a separate "**`Parity`**" menu where the choice is made. In either case, simply make note of the current parity setting or scroll to change it to a preferred parity at this time. (Factory default is Even parity).

  e) Exit the menu mode by returning the rocker switch to its original position. This should place the scale back into the normal weighing mode again.

- Connect the scale to the PC using a standard 9-pin RS-232 cable. Make sure the COM port on the PC is available and has no other device attached to it (e.g. an internal modem).

- Launch the WTOPOSAdmin program. Click on the **Add** button to enter a name for this scale instance in the **Scale Instance** box. (Refer to table 7-1 for a description of the various user interface controls available in the administration configuration/test program).

- Set the communications **Configuration** for **Serial Port**, **Baud Rate** and **Parity** to match the settings that the scale is configured to. Note: unless otherwise speified, the scale communicates using **7** (seven) data bits per character.

- Click on the **Enable Scale** button. This will enable the **Read Weight** and **Zero Scale** buttons.

- Click the **Read Weight** button and the weight currently indicated on the scale should be displayed. If not, check cable connections, COM port, and baud rate/parity settings in the scale and test program.

- With a small weight on the scale, click the **Zero Scale** button and the weight displayed should go to zero. You must click the **Read Weight** button again to see this zero weight on the test program as it does not automatically perform the read weight function after doing a zero scale operation.

  Note: The maximum amount of weight that may be zeroed on the scale is often limited to **2%** of full scale capacity. A 30 lb capacity scale, for example, will not allow zeroing if the displayed weight is greater than 0.6 lb. Therefore, if the **Zero Scale** button appears not to have zeroed the scale, please verify that the weight displayed on the scale does not exceed this maximum. Also, the scale will not zero if the weight is in-motion, undercapacity or overcapacity.

7.3 POS Application Test

The final (and most important) step in testing the Weigh-Tronix OPOS Scale Control is to verify that it works properly with your existing POS application.

Since there are many different POS applications in use it is impossible to give a step-by-step test sequence. In general, the POS application will have a configuration mode in which the operator may select particular POS devices. Simply select the Weigh-Tronix POS Scale (SCP-02) as your scale device. Then run your POS application to verify that weight is properly accessed from the scale. If your application does not function properly, please review the list of possible in stallation issues below.

7.4 Installation Issues

- Make sure that the scale is configured for "**ECR**" protocol.
- Make sure that the scale and WTOPOSAdmin program have the same baud rate and parity settings.
- Make sure that the COM port setting in the WTOPOSAdmin program is really available for use and that the scale is connected to that serial port.
- Problems frequently occur when using the CO from one vendor with the SO from another vendor. In one particular case, a POS application (CASPOS) was compiled (and early-bound) to an IBM CO. The CO would not work with the Weigh-Tronix OPOS Scale Control, so the application was recompiled. Once recompiled, CASPOS properly bound the WT SO/CCO combination with the application thereby, effectively, eliminating the IBM CO.
  This recompile was actually beneficial in that it brought the POS application up to date as far as using the OPOS recommended "Common Control Object" (**CCO**) technique.

  Note:

  The author of the "common control object" CCO (Curtiss Monroe) has a web-site (www.monroecs.com) where he describes "Deviations from OPOS Specification" that were necessary (to the CCO) to work under COM. Basically, these were name changes to two methods. "Claim" was changed to "ClaimDevice" and "Release" was changed to "ReleaseDevice". The methods "Claim" and "Release" can still be accessed by a (POS) app but only through the IDispatch interface (for back-compatibility).

  He goes on further to state that "this will require changing a few lines of code in applications that use early-binding as they are updated to use these (common) control objects". Further, "...in other words, with early-binding, swapping a control object is an application change, such that these method renames can be done at that time; if the application is using late-binding, it won't need to be changed since Claim and Release are still supported by the IDispatch interface."

  Another option for this particular case would have been for the POS application to be modified to late-bind to the CO, instead of being "hard-coded" to a specific vendors CO. Then, the linkage to the new CCO could have been resolved at run time.

- In one particular case, the Weigh-Tronix OPOS Scale Control did not immediately function properly when used with a POS application that was written in Visual FoxPro v5.0.

  The incompatibility was with Visual FoxPro v5.0 and the CCO and is related to COM's built-in ActiveX Dual Interface Support and binding. These issues were worked-around by code changes to the application details of which are unknown. However, it is known that for Visual FoxPro v5.0, the default for dual interface support is **enabled**, therefore, one possible solution is to review the FoxPro v5.0 documentation and make manual code changes to override the dual interface support.

  Another possible solution is to upgrade to Visual FoxPro v6.0 who's default support for dual interface support is **disabled** which complies with the CCO and standard COM usage.

Appendix-A

Glossary of Terms:

SO - Service Object;
Provided by scale vendor and performs scale to CCO (or CO) interface.

CO - Control Object;
Provided by scale vendor or other third party and performs SO to application interface. These vendor-specific control objects will be eventually be replaced by their equivalent common control object.

CCO - Common Control Object;
Provided courtesy of OPOS/RCS and performs SO to application interface. The CCO is intended to be a common replacement for all vendor supplied CO's. There is one CCO for each device class.

WT OPOS Control -
The Weigh-Tronix (combined SO/CCO) OPOS-scale control as distributed. The Weigh-Tronix OPOS control will install an SO specific for the WT/NCI scale (using the SCP-02 serial protocol) and the CCO provided by courtesy of the OPOS committee.

Object References -
When writing application programs, object references are *early-bound* if they use object variables declared as variables of a *specific* class. Object references are *late-bound* if they use object variables declared as variables of the *generic* Object class. Even though it is slower, there are times when *late-binding* is necessary.

Late Binding -
When an object variable is declared as a generic Object, the compiler cannot determine at compile-time what sort of object reference the variable will contain. Therefore the language will use late-binding, requiring that the language determine at run time whether or not that object will actually support the properties and methods used in the application code. Late binding forces the compiler to add extra code to do this run time checking which slows execution of the properties, methods and events for these objects.

Early Binding -
When an object variable is declared using the specific class that defines the object, the compiler can resolve the references to the object at compile time. This results in a minimal amount of code in the compiled executable to actually invoke the object's properties, methods and events.

Appendix-B

References:

Title: **Serial Communications Protocol SCP-02 (ECR Standard)**
Doc. No.: 8408-14788-02 (SCP-02)
Source: Weigh-Tronix Inc.
3990 Brickway Blvd.
Santa Rosa, CA. 95403
Web Site: www.wt-nci.com
Desc.: Describes the proprietary serial communications protocol used to connect Weigh-Tronix POS scales to personal computers. Also describes "OPOS Extensions" to ECR protocol which

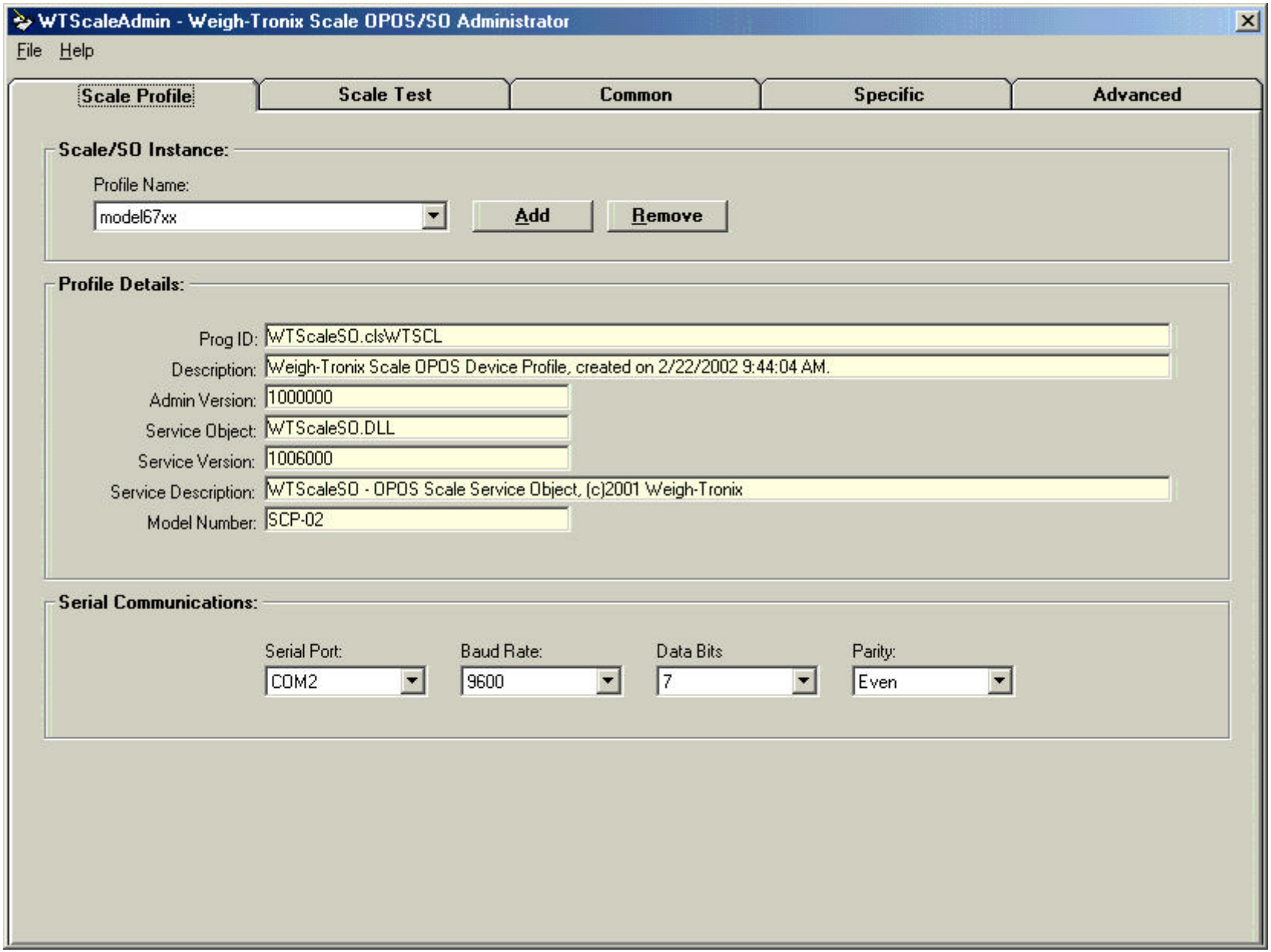
Title: **WTScaleAdmin Administrator Program**
Doc. No.: 8421-15950-18 (EAN-18)
Source: Weigh-Tronix Inc.
3990 Brickway Blvd.
Santa Rosa, CA. 95403
Web Site: www.wt-nci.com
Desc.: This document. Describes the Weigh-Tronix OPOS Scale-class Service Object (SO) and its Installation, Test and Administration program.

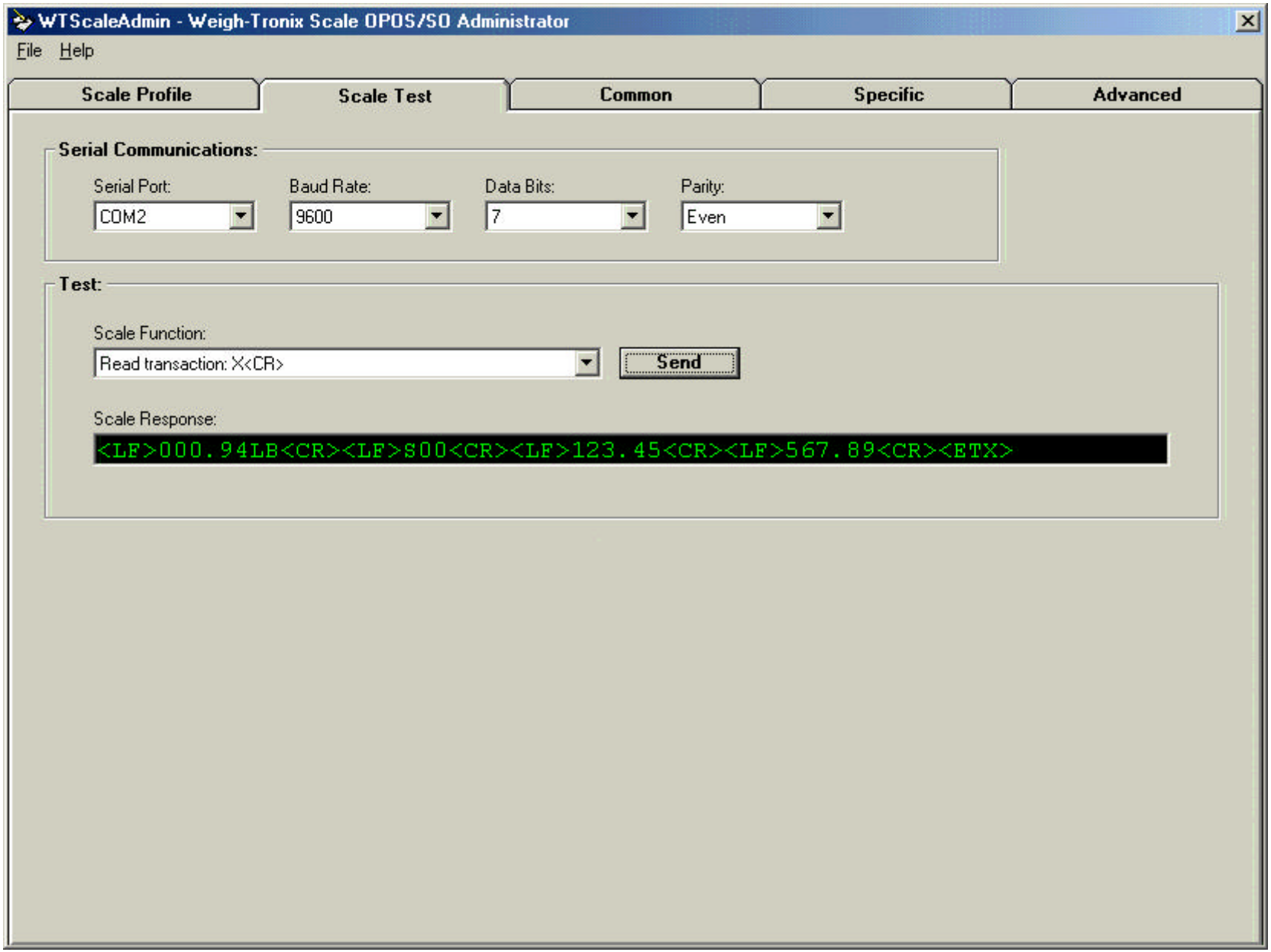
Title: **OLE for Retail POS**
Application Programmer's Guide
Doc. No.:
Source: OPOS/Monroe Consulting Services
Web Site: www.monroecs.com/oposccos_current.htm
Desc.: International Standard
for Windows 95/98, Windows NT or other OLE/ActiveX compliant 32-bit operating system.

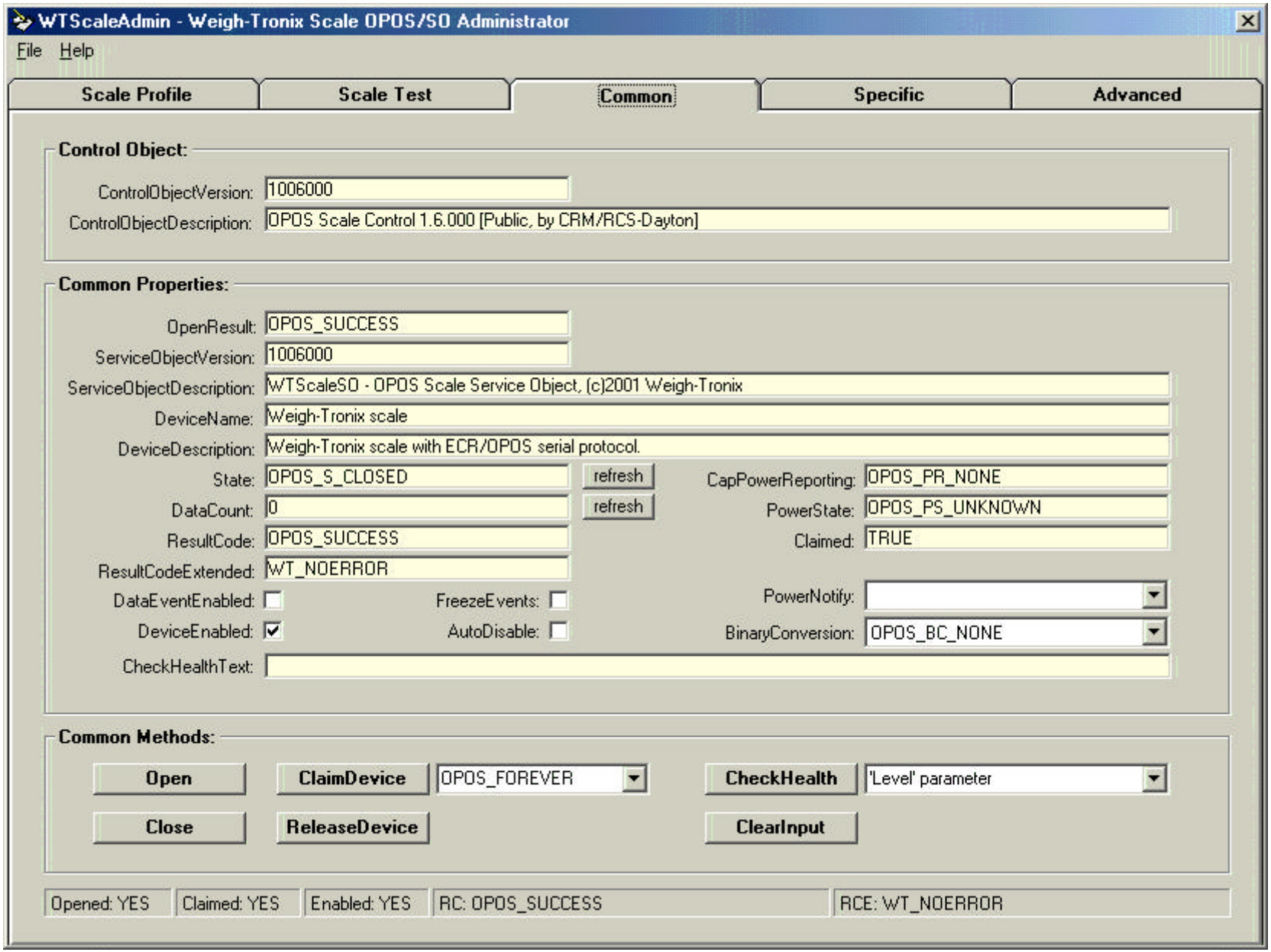
Title: **OLE for Retail POS**
Control Programmer's Guide
Doc. No.:
Source: OPOS/Monroe Consulting Services
Web Site: www.monroecs.com/oposccos_current.htm
Desc.: International Standard
for Windows 95/98, Windows NT or other OLE/ActiveX compliant 32-bit operating system.

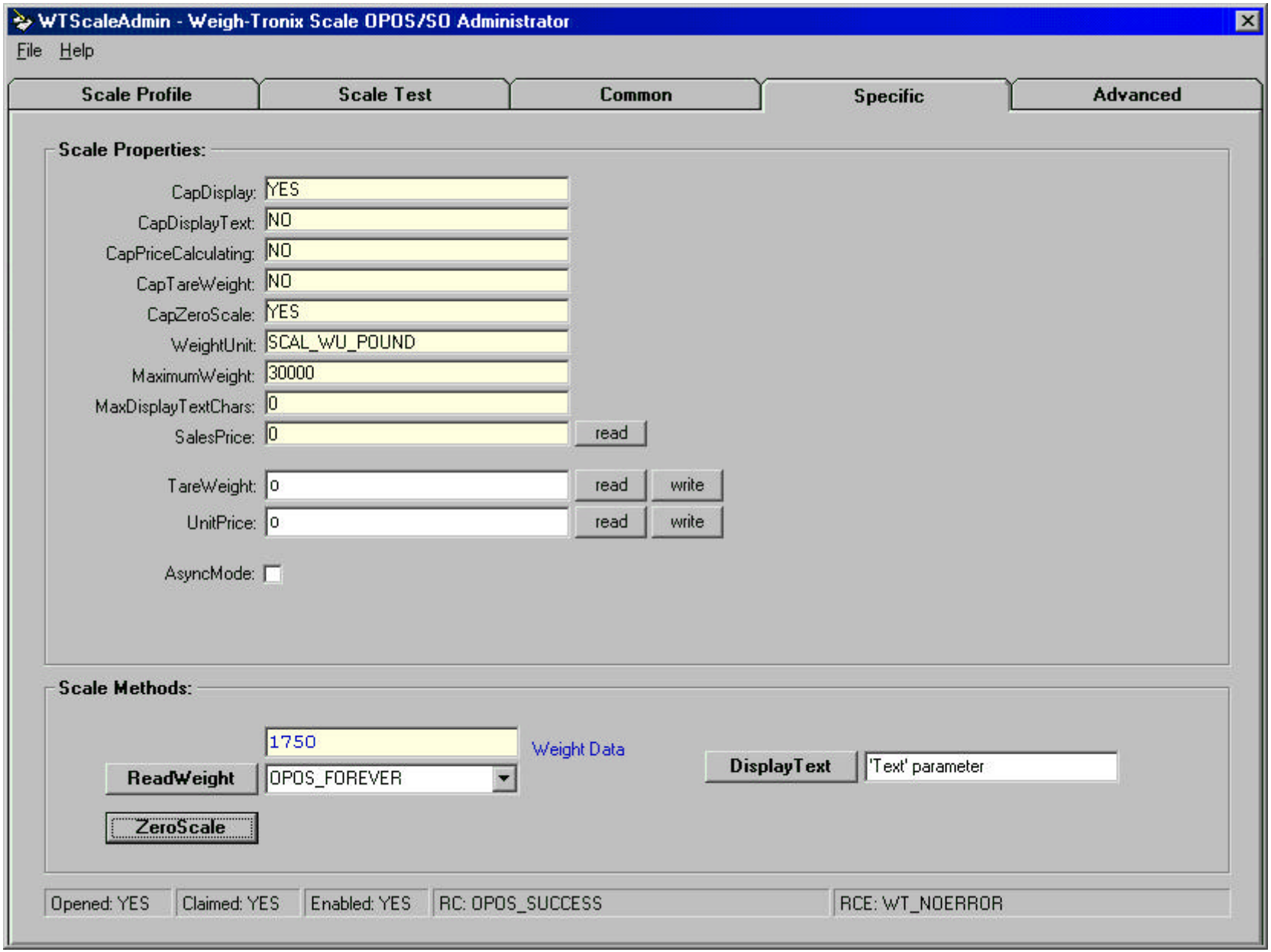
Appendix-C

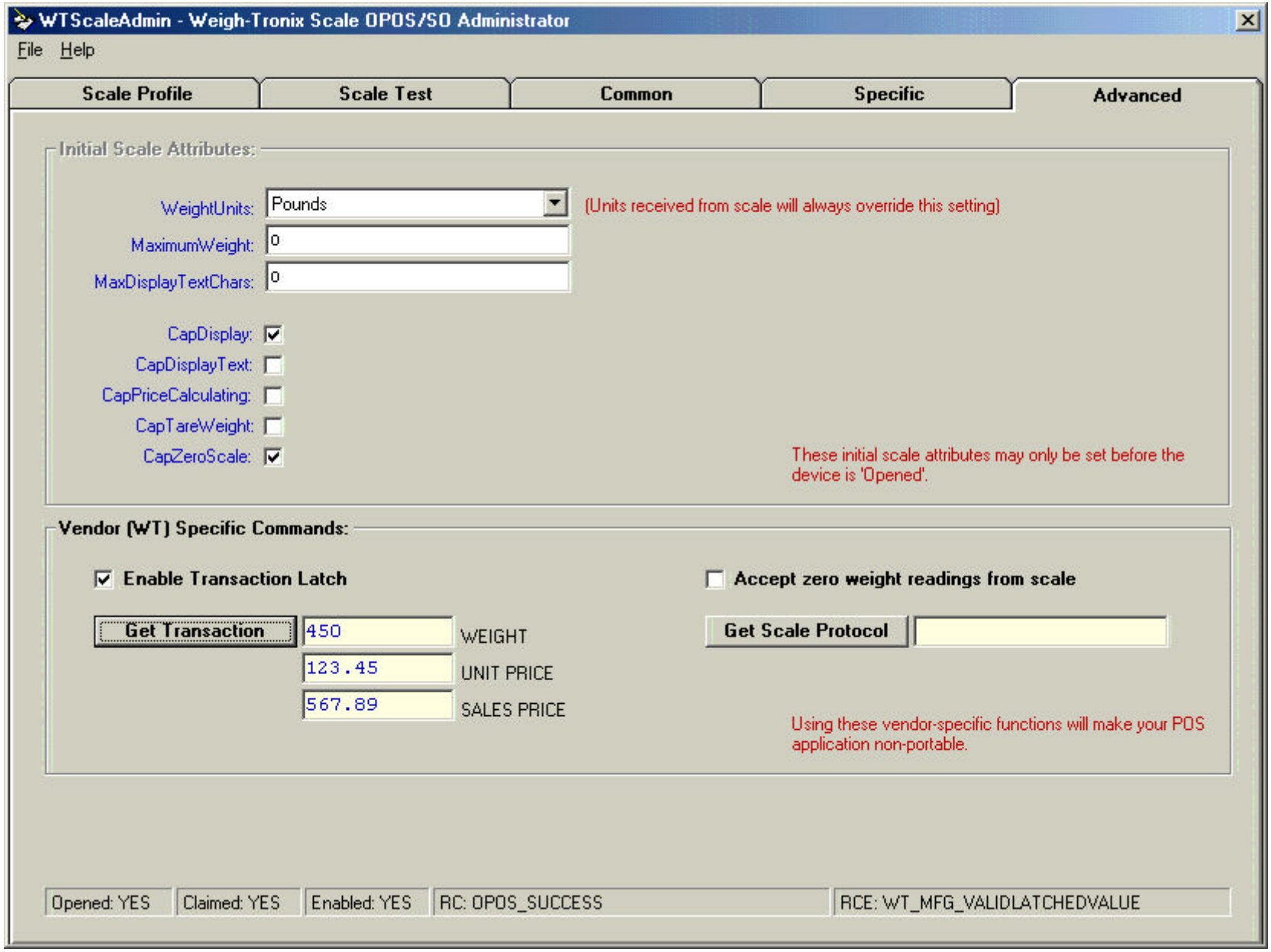
Sample screen shots of the Weigh-Tronix **WTScaleAdmin** (Administration) program.


WTScaleAdmin - Weigh-Tronix Scale OPOS/SO Administrator
File Help
Scale Profile
Scale Test
Common
Specific
Advanced
Scale/SO Instance:
Profile Name:
model67xx
Add
Remove
Profile Details:
Prog ID: WTScaleSO.clsWTSCL
Description: Weigh-Tronix Scale OPOS Device Profile, created on 2/22/2002 9:44:04 AM.
Admin Version: 1000000
Service Object: WTScaleSO.DLL
Service Version: 1006000
Service Description: WTScaleSO - OPOS Scale Service Object, (c)2001 Weigh-Tronix
Model Number: SCP-02
Serial Communications:
Serial Port: COM2
Baud Rate: 9600
Data Bits 7
Parity: Even


WTScaleAdmin - Weigh-Tronix Scale OPOS/SO Administrator
File Help
Scale Profile
Scale Test
Common
Specific
Advanced
Serial Communications:
Serial Port:
COM2
Baud Rate:
9600
Data Bits:
7
Parity:
Even
Test:
Scale Function:
Read transaction: X<CR>
Send
Scale Response:
<LF>000.94LB<CR><LF>S00<CR><LF>123.45<CR><LF>567.89<CR><ETX>


WTScaleAdmin - Weigh-Tronix Scale OPOS/SO Administrator
File Help
Scale Profile
Scale Test
Common
Specific
Advanced
Control Object:
ControlObjectVersion: 1006000
ControlObjectDescription: OPOS Scale Control 1.6.000 [Public, by CRM/RCS-Dayton]
Common Properties:
OpenResult: OPOS_SUCCESS
ServiceObjectVersion: 1006000
ServiceObjectDescription: WTScaleSO - OPOS Scale Service Object, (c)2001 Weigh-Tronix
DeviceName: Weigh-Tronix scale
DeviceDescription: Weigh-Tronix scale with ECR/OPOS serial protocol.
State: OPOS_S_CLOSED
refresh
DataCount: 0
refresh
ResultCode: OPOS_SUCCESS
ResultCodeExtended: WT_NOERROR
DataEventEnabled:
FreezeEvents:
DeviceEnabled:
AutoDisable:
CheckHealthText:
CapPowerReporting: OPOS_PR_NONE
PowerState: OPOS_PS_UNKNOWN
Claimed: TRUE
PowerNotify:
BinaryConversion: OPOS_BC_NONE
Common Methods:
Open
ClaimDevice
OPOS_FOREVER
CheckHealth
'Level' parameter
Close
ReleaseDevice
ClearInput
Opened: YES
Claimed: YES
Enabled: YES
RC: OPOS_SUCCESS
RCE: WT_NOERROR


WTScaleAdmin - Weigh-Tronix Scale OPOS/SO Administrator
File Help
Scale Profile
Scale Test
Common
Specific
Advanced
Scale Properties:
CapDisplay: YES
CapDisplayText: NO
CapPriceCalculating: NO
CapTareWeight: NO
CapZeroScale: YES
WeightUnit: SCAL_WU_POUND
MaximumWeight: 30000
MaxDisplayTextChars: 0
SalesPrice: 0 read
TareWeight: 0 read write
UnitPrice: 0 read write
AsyncMode:
Scale Methods:
1750
Weight Data
ReadWeight
OPOS_FOREVER
DisplayText
'Text' parameter
ZeroScale
Opened: YES
Claimed: YES
Enabled: YES
RC: OPOS_SUCCESS
RCE: WT_NOERROR


WTScaleAdmin - Weigh-Tronix Scale OPOS/SO Administrator
File Help
Scale Profile
Scale Test
Common
Specific
Advanced
Initial Scale Attributes:
WeightUnits: Pounds
(Units received from scale will always override this setting)
MaximumWeight: 0
MaxDisplayTextChars: 0
CapDisplay:
CapDisplayText:
CapPriceCalculating:
CapTareWeight:
CapZeroScale:
These initial scale attributes may only be set before the device is 'Opened'.
Vendor (WT) Specific Commands:
Enable Transaction Latch
Accept zero weight readings from scale
Get Transaction
450
WEIGHT
123.45
UNIT PRICE
567.89
SALES PRICE
Get Scale Protocol
Using these vendor-specific functions will make your POS application non-portable.
Opened: YES
Claimed: YES
Enabled: YES
RC: OPOS_SUCCESS
RCE: WT_MFG_VALIDLATCHEDVALUE

Appendix-D

This section provides a complete revision history of the Weigh-Tronix **WTScaleAdmin.EXE** (Administration) program and the **WTScaleSO.DLL** Service Object.

Date	Release	Significant Changes
September 2001	**Beta**	**First beta test release of the Weigh-Tronix OPOS Service Object (WTScaleSO.DLL) and installation/administration program (WTScaleAdmin.EXE).**
March 2002	**Release 1.0 [1]**	**WTScaleAdmin.EXE (v2.0)** Added *SalesPrice* and *UnitPrice* property test functions. Added tests for all new vendor-specific commands. Changed *Version* to *AdminVersion* in Registry Profile. Added *ServiceObjectVersion* and *ServiceObjectDescription* to Registry Profile. **Note: Because of changes to Registry Profiles, you should first delete all existing profiles and uninstall existing WTScaleAdmin before installing this new release.** **WTScaleSO.DLL (v1.6)** Added *SalesPrice* and *UnitPrice* property functionality. Currency value strings no longer accept currency (e.g. '**$**' prefix). Added vendor-specific commands: 1)Determine scale protocol in use. 2)Accept scale zero weight values. 3)Do not accept scale zero weight values. 4)Enable price transaction value latching. 5)Disable price transaction value latching. 6)Read and latch price transaction values from scale.

NOTES:

[1] This first release makes changes in the information stored in the system registry. Therefore, before installing this new release you should do the following:

a)Launch existing WTScaleAdmin program

b)On the **ScaleProfile** tab, **Remove** all profile names after making note of serial communications settings for later reentry. This deletes scale profile(s) from the system registry.

c)Exit the WTScaleAdmin program and use the Windows Uninstall feature to completely remove the cur rent WTScaleAdmin program.

d)Install the new release of WTScaleAdmin.

Weighing Products & Systems

Postal Scales

POS Scales

Dot Matrix Impact Printers

Thermal Graphic Label Printers

OPOS Scale Controls

www.wt-nci.com

Weigh-Tronix Inc.
3990 Brickway Blvd.
Santa Rosa, CA. 95403-1098

Tel: (707) 527-5555
Fax: (707) 527-5517