JavaPOS Linux Driver for Mettler Toledo Checkout Scales

User Manual

Mettler-Toledo Retail Group

Last Revision: 4/2/2019

1.0 Introduction

This JavaPOS Linux driver implements the Device Service component according to the JavaPOS architecture and standard specifications.

This Device Service implementation is intended to work with Mettler Toledo Ariva Checkout scales (Ariva-S, Ariva-B and Ariva-H).

Two RS-232 protocols are supported: Dialog-06 and 8217.

- Dialog-06 and Dialog-06-PIPE protocols are primarily for the European market.
    - The MettlerScaleDialog-06-PIPE protocol uses named pipes to display weights directly on a POS screen, using the Mettler Toledo Virtual Checkout Displays (VCODisp) technology.
- The 8217 protocol is used for both the N. America and European markets.

The POS may use USB Virtual COM Ports to transport the RS-232 protocol. Ariva scales also provide a Virtual COM Port driver, upon request. In this case the Ariva scale must be configured for Virtual COM Ports (menu 3.1 = 1).

This release of JavaPOS Device Service implementation is **JavaPOS version 1.8** compliant.

2.0 System Requirements

The Mettler Toledo JPOS Installer detects whether your Linux OS is 32 or 64 bits and installs the corresponding JPOS driver. Note that mismatching your Java and OS (32 bit Java on a 64 bit Linux, for example) will not likely work with this JPOS driver.

1. On 32-bit Linux OS's install a 32 bit Java.

2. On 64-bit Linux OS's install a 64 bit Java.

3. To determine if your Linux OS is 32 or 64 bits:
    a. In the shell window execute the "arch" command.
        i. If "x86_64" is returned, then you have a 64-bit machine.
        ii. If "x86" is returned, then you have a 32-bit machine.
    b. "uname –a" is also a useful command.

4. To determine if your Java is 32 or 64 bits:
    a. Execute the "*java -version*" command.

To determine if your Windows OS is 32 or 64 bits:

- Go to Start->Control Panel\System and Security\System

To determine if your Java is 32 or 64 bits:

- Go to Start->Control Panel->Programs->Java
- Select the Java tab.
- Select View
- If the Architecture tab says "x86" then it is a 32 bit Java installation.

Verify that your Java is enabled.

- Go to Start->Control Panel->Programs->Java (32-bit).
    - In the Java Control Panel, select "Java"
    - Select "View"
    - Verify that Java is "Enabled" (checked).

NOTE: *nrjavaserial.jar*: For Serial Communications, including USB Virtual COM Ports, the MT JavaPOS driver uses *nrjavaserial.jar* (and not *javax.comm.SerialPort, gnu.io.SerialPort, librxtxSerial.so,* etc.)

3.0 JavaPOS Scale Device Configuration

The "jpos.xml" configuration file contains *JposEntries* configuration information for each device managed through JavaPOS drivers.

Note that the property names used inside JposEntry are *case sensitive*.

jpos.xml contains three entries: MAKE SURE YOU EDIT THE CORRECT ENTRY!

- The "***logicalName***" attribute is the identifier of the JposEntry and therefore must be unique at level of jpos.xml'. Its value must be used by the high-level application when opening the JavaPOS scale device.

- The "***factoryClass***" attribute indicates the fully-qualified name of the class responsible for the creation of the instance of the JavaPOS Device Service for Mettler Scale supported models.

- The "***serviceClass***" attribute indicates the fully-qualified name of the class used by the JavaPOS Device Service implementation for the target scale model. For Mettler Scale L2-SCx models the Device Service is implemented by "***com.mt.jpos.MettlerL2SCScaleService***" class.

- The "***Vendor name***" and "***url***" attributes can be set to appropriate Mettler values.

- The "***category***" MUST have "Scale" value and the "***version***" attribute should reflect the JavaPOS version used ("1.8").

- The "***product description***" property can be set to a desired string that will be returned as JavaPOS Device Description when an application requests it.

- The "***physicalDeviceDescription***" and "***physicalDeviceName***" properties can be set to desired string values that will be returned as JavaPOS PhysicalDeviceDescription and PhysicalDeviceName when an application requests them.

- The ***"deviceServiceChecksum"*** property is mandatory and indicates the checksum calculated over the '.jar' archive that contains the drivers' implementation (*mtscale.jar*).
    - This checksum value represents the CRC32 checksum calculated over the 'MessageDigest' hash-content in the '.jar' archive.
    - When the '**open**' method is invoked for a scale device, the checksum over the custom implementation part of the driver (that is, the Device Service component in the '.jar' archive) is computed at runtime. This obtained value is compared to the jpos.xml property value. If they are different then an error is raised for the 'open' operation and the Device Service component is not opened.
    - If this property is missing or has no value then the 'open' operation fails and an error is raised.

  The proper checksum value to use for this property is delivered together to each new release of the '.jar' archive (mtscale.jar).

- The "***port***" property value indicates the serial port the scale is connected on:
    - on Windows "COM1", "COM2" and so on;
    - on Linux "/dev/ttyS0", "/dev/ttyS1" and so on.

  This property is mandatory only if "***communicationType***" is set to "COM"

Note: for the following values (baud rate, parity, stop bits and data bits), the following values are typically standard for Mettler Toledo scales:

1.) *For the US and Canada scales are typically configured for 9600 baud, 7 data bits, Even parity and 1 stop bit, but this depends upon the POS system's configuration.*
2.) *For Europe the Ariva scales are typically configured for 9600 baud, 7 data bits, Odd parity and 1 stop bit, but this depends upon the POS system's configuration.*

The scale RS-232 settings must match these settings. The Ariva RS-232 settings can be configured via the Ariva display menu.

- The "***baudrate***" property's value indicates the communication speed appropriate for the scale. If this property is missing or invalid then the default value 9600 is used.

- The "***parity***" property's value indicates the parity for serial communication with the scale connected on the indicated serial port. If this property is missing or invalid then the default value 1 (meaning 'ODD') is used.
    - Parity: 1 is for Odd Parity, 2 is for Even Parity, 0 is for No Parity.

- The "***stopbits***" property's value indicates the number of stop bits for serial communication with the scale connected on the indicated serial port. If this property is missing or invalid then the default value 1 is used.

- The "***databits***" property's value indicates the number of bits for data content during serial communication with the scale connected on the indicated serial port. If this property is missing or invalid then the default value 7 is used.

- The "***metricUnit***" property indicates through its "true"/"false" values whether the scale reports weight values in Kilogram (metric unit) or in Pound (English unit).
    - If set to "false" the MeasuredWeight field will display in Lbs. or Oz.
        - This is typically true for the US.
    - If set to "true" the MeasuredWeight field will display the weight in grams.
        - This is typically true for the EU.

- The "***maximumWeight***" property indicates the maximum weight value for that scale. The value is expressed as integer number, which has an assumed decimal place located after the "thousands" digit position. Here are the values:
    - 0-15 lbs.: 15000
    - 0-6/6-15 lbs.: 15000
    - 240 Oz. 240000
    - 0-3/3-6 kg: 6000
    - 0-15 kg: 15000
    - 0-6/6-15kg: 15000
    - 0-30 lbs. 30000
    - 0-15/15-30 lbs. 30000

- The "***tracing***" property indicates through its "true" / "false" values whether the tracing mechanism is turned On or Off.

- The "***tracingOutputFile***" property indicates the file where to write the trace-messages. The file can be indicated using absolute or relative paths. If the path does not exist or the file cannot be created at that location (because insufficient access rights) then the tracing mechanism is turned Off.

- The "***tracingLevel***" property indicates the tracing level. This property can take two possible values "INFO" and "DEBUG". INFO is default tracing level. DEBUG is for tracing extra information. This property is optional.

- The "***refreshSerialConnection***" property indicates the time interval for forced refresh of serial communication, measured in minutes. This property accepts numeric values. This property is optional. If not present, serial communication will not be refreshed.

The following values **are** only for Dialog-06 pipe communications:

- The "***communicationType***" property indicates the communication type desired. This property can take two possible values "COM" for serial communications and "PIPE", for pipe communication. This property is mandatory for Dialog-06.

- The "***pipeInputFile***" and '***pipeOutputFile***" properties indicate the names (including path if needed) of the named pipes to use. If these names contain a path then these values are system-dependent:
    - on Windows "..\\pipe\\VCODispIn", "VCODispIn" and so on;
    - on Linux "../pipe/VCODispIn", "VCODispIn" and so on.

  These properties are mandatory for pipe communication ("***communicationType***" property is set to "PIPE").

4.0 How to Send non-JavaPOS Commands to a Scale

The Mettler Ariva Scale Dialog-06 protocol offers commands that are not included in the JavaPOS standard. These commands can be sent to the scale logical device through the "**directIO**" call:

void directIO(int commandCode, int[] data, Object auxiliarData)

In the Mettler Scale JavaPOS driver implementation, the directIO call can be used for the following special commands:

- Request an item weight by providing the item description:
    - commandCode = MettlerScaleDirectIOCommands. GET_WEIGHT
    - data = int[1] and acts as input-output parameter:
        - On input data[0] provides the timeout value in milliseconds to wait for a weight result from the scale.
        - In synchronous mode (AsyncMode = false) the weighing result is returned in data[0];
    - auxiliarData – the String representing the item description to be displayed (maximum 13 characters.)
    - The unit price and the tare weight, if available, must be set before calling this directIO method.

Code sample:

```
// myScale instance is previously open, claimed and enabled

int[] data = new int[1];
String itemDescription= “Apples”;

data[0] = 2000; // 2 seconds

try {
      myScale.setAsyncMode(false);
      myScale.setUnitPrice(14500);// meaning 1.4500
      myScale.setTareWeight(300); //meaning 0.300

      myScale.directIO(MettlerScaleDirectIOCommands.GET_WEIGHT, data,
                        itemDescription);

      int weight = data[0];
      System.out.println(“Measured weight = “ + weight);
      System.out.println(“Sales Price = “ + myScale.getSalesPrice());

} catch (JposException jposEx) {
      //Exception handling
}
```

If the operation is asynchronous (AsyncMode = true) then the weighing result is returned through the DataEvent event and the calculated sales price is returned via the JavaPOS scale instance’s SalesPrice attribute.

- Display the protocol’s version number (version number On):

- commandCode=MettlerScaleDirectIOCommands.DISPLAY_PROTOCOL_VERSION
- data = null (not used)
- auxiliarData = null (not used)

According to the Dialogue06 protocol, after receiving the command to display the protocol version number the scale will ignore further commands until it receives the command to stop displaying the protocol version number (version number Off). Therefore the application should also send the following command to put the scale back into its operational state:

- Stop displaying the protocol’s version number (version number Off):

- commandCode = MettlerScaleDirectIOCommands. CLEAR_PROTOCOL_VERSION
- data = null (not used)
- auxiliarData = null (not used).

5.0 Generate Checksum over the Scale Driver Jar File

Users: Note that the proper checksum value to use for this property is delivered with each new release of the mtscale.jar archive, so this operation is typically only used by the software developers who created this .jar file.

When a new JavaPOS Jar archive is released, a new checksum value must be generated to use as the value for the "***deviceServiceChecksum***" property in the entry for Mettler scale device in jpos.xml configuration file.

The **mtchecksumutil.jar** tool is used to generate a checksum over a Jar file.

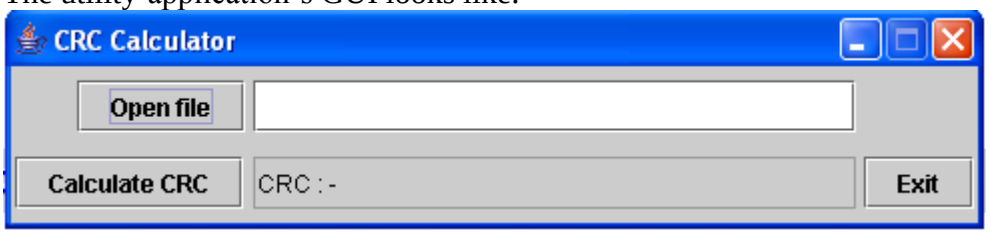
Run this **mtchecksumutil.jar** from a command-line from the directory where the jar is located, using either:

java –cp $CLASSPATH:./mtchecksumutil.jar com.mt.intern.util.CRC32CalcApp

or:

java –jar mtchecksumutil.jar

The utility-application's GUI looks like:



Use "Open file" button to browse and locate the target ***mtscale.jar*** file. It assumes by default the directory where ***mtchecksumutil.jar*** is located.

Once the target jar file is located and selected, press the "Calculate CRC" button and the checksum calculated over the selected jar file will be displayed on the GUI and written into a text file named 'mtscale_crc.txt' in the same directory where ***mtchecksumutil.jar*** is located.

This checksum value must be used in the *jpos.xml* file for the ***"deviceServiceChecksum"*** property for the jpos-entry corresponding to Mettler scale device being managed by JavaPOS.

6.0 Test Application

This Mettler Toledo JavaPOS driver implementation includes a Java test application GUI contained in '**mtscaletestapp.jar**'.

To run on a Linux platform, this application must be launched from an X-Window session.

Assuming you are in the directory where **mtscaletestapp.jar** is located, the application can be started from the command prompt: ***startMTScaleTestApptest.bat***

The application GUI for 8217, Dialog-06 and Dialog-06-PIPE *are at the end of this section.*

6.1 Test Application Guide

Errors are reported in the 'Reported errors' text area.

1 Open scale

The first operation is to open the JavaPOS Scale logical device. If the open operation succeeds the scale's principal characteristics are displayed into the 'Scale properties' panel.

After the 'Open' operation, the user can either 'Claim' or 'Close' the Scale logical device.

2 Claim scale

This operation opens the serial port connection and enables the scale's logical device. After this operation it is possible to start weighing operations, to 'Release' the scale or to 'Close' the scale.

3 Weighing operations

3.1 Get Weight

After the scale is claimed one can send the "Get Weight" command to the scale.

- When the 'Get weight' button is pressed the command is sent to the physical scale and the weighing result is displayed in the 'Measured weight' field.

3.2 Tare

This application support both Platter Tare (Tare Compensation) and Preset Tare, as well as the ability to Clear Tare.

- A Preset Tare weight can be entered into the Preset Tare Weight Field.

- A Platter Tare weight can also be placed onto the scale and registered by pressing the Tare Compensation button.

3.3 Zero Scale

This command zeros the scale.

3.4 Dialog-06

- The ‘Unit price’ is mandatory when using Dialog-06 and, when provided, the calculated ‘Sales price’ is displayed.
- The Item Description may also be provided.

3.5 Dialog-06-Pipe

- The Dialog-06-Pipe Zero Scale command is provided to work with the Virtual Checkout Display (VCODisp).

3.6 Other

Note that the Timeout, Asynchronous Operations and Confidence Test commands may not be particularly useful with the Ariva scale. The use of these commands is left as an exercise for the reader.

4 Release scale

During this operation the connection to the serial port is closed and the scale instance is signaled as released.

After this operation one may ‘Claim’ again the scale or ‘Close’ it.

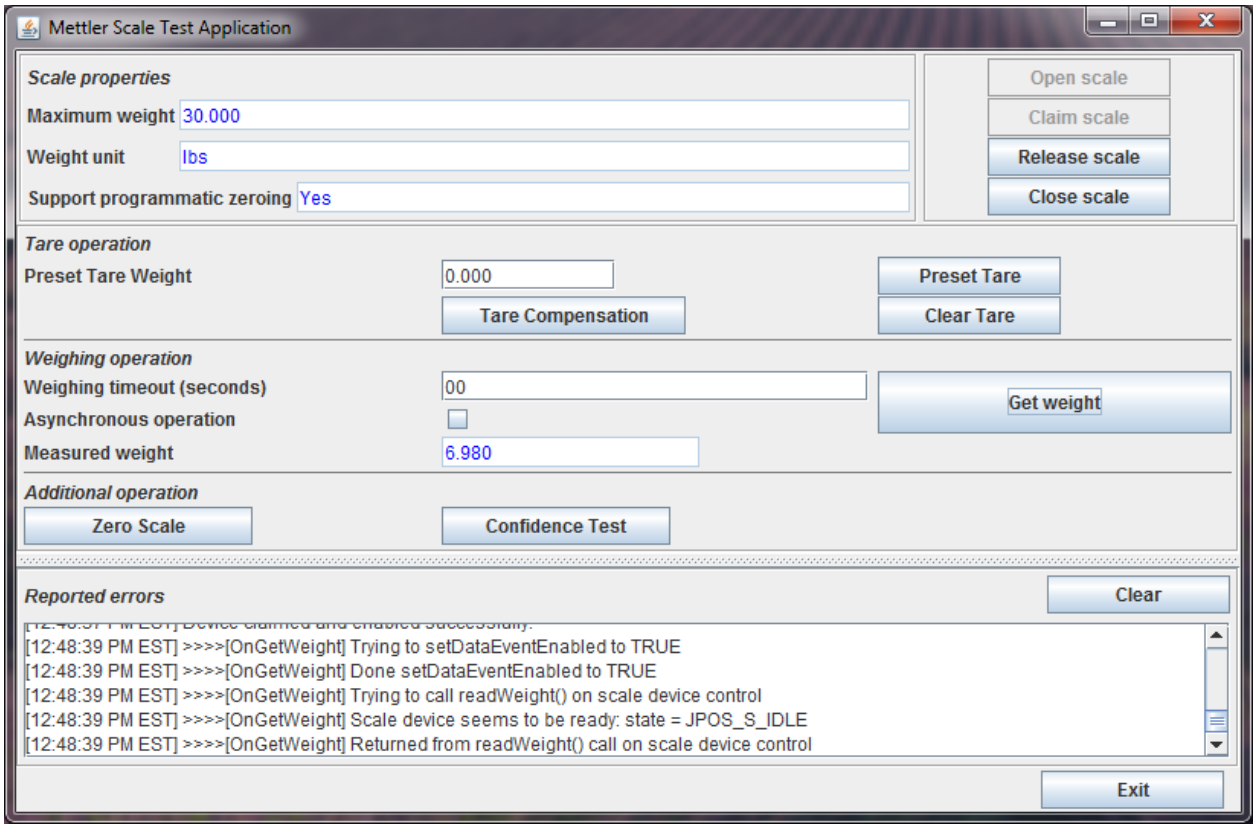
5 Close scale

During this operation the scale instance is closed.

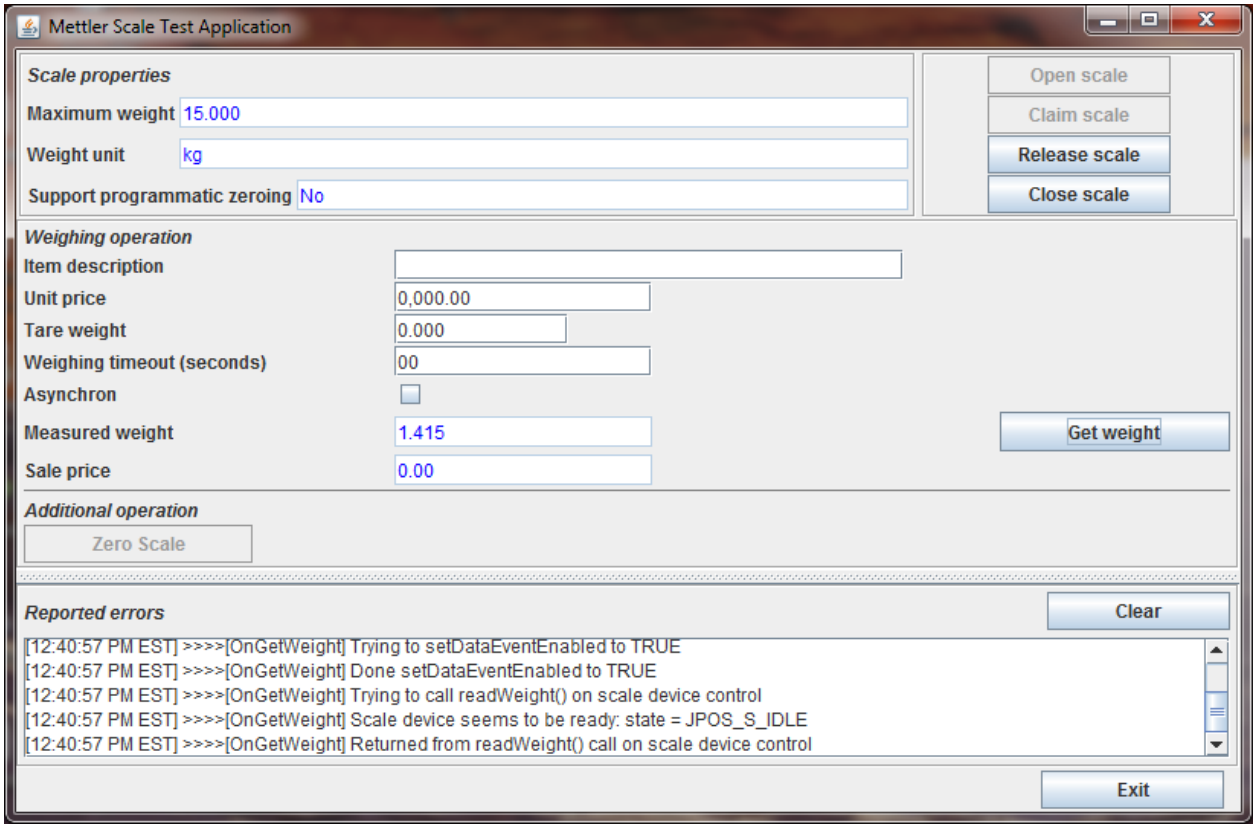
6 Exit

Exit the application.

6.2 8217 Image



6.3 Dialog-06 Image



6.4 Dialog-06-PIPE Image