DIEBOLD NIXDORF

USER GUIDE

ProBase POS 2

JavaPOS™, OPOS, POS for .NET and CPOS

September 2017

DIEBOLD NIXDORF

Imprint

Mathias Janke
Diebold Nixdorf
Product Line Retail
Wohlrabedamm 31
13629 Berlin
mathias.janke@dieboldnixdorf.com

DIEBOLD NIXDORF

Revision history

Version	Date	Author	Comment/Change
1.0	November 2016	Mathias Janke	Initial Version
2.0	November 2016	Mathias Janke	Changes for PBP 2.1
2.1	November 2016	Mathias Janke	Typos; WNLPOS 4 Support
3.0	March 2017	Mathias Janke	Changes for PBP 2.2
3.1	April 2017	Mathias Janke	Additional chapters/enhancements – Configuration, Logging, Tools, Samples
3.2	May 2017	Mathias Janke	Typos; path values corrected in chapter 7.1.1
3.3	June 2017	Mathias Janke	Additional information about UDM Client Logging, UDM Server start and CPOS examples
4.0	July 2017	Mathias Janke	Changes for PBP 2.3
4.1	August 2017	Mathias Janke	Corrections
5.0	September 2017	Mathias Janke	Changes for PBP 2.4

Copyright and Trademarks

DIEBOLD NIXDORF

Contents

DIEBOLD NIXDORF

1 Introduction

Starting with JavaPOS[1] and the requirements for a synchronization of the standard OPOS[2] with JavaPOS, the new standard UnifiedPOS[3] was created with the cooperation of Diebold Nixdorf in the respective committees[4]. UnifiedPOS (UPOS) stands as an acronym for the Unified Point of Service - as a standardization of OPOS and JavaPOS.

UPOS is not really a driver implementation but rather an interface definition, whereby the implementation of the definition in usable code as a driver for POS applications occurs in the one form as OPOS and in the other as JavaPOS.

With the introduction of the .NET framework and the market requirement for an extension of UPOS for .NET based POS applications, a POS for .NET (P4DN) implementation of the UPOS standard was implemented.

The provision of these three implementations (JavaPOS, OPOS and P4DN), which ultimately have all been implemented according to the same design principles, has led to ProBase POS.

As a further driver interface for Linux POS applications, which were developed in the programming language C, Diebold Nixdorf also offers the interface CPOS. This is also an implementation according to the UPOS standard.

The basis of ProBase POS is the JavaPOS from Diebold Nixdorf. The interfaces OPOS, CPOS and P4DN are supported when the Unified Device Manager (UDM) is used at the same time.

In addition to supporting existing POS peripherals, ProBase POS also supports devices from the self-checkout and automated checkout areas. Completely for the devices from the self-checkout area the provision of CIM[5] data is implemented in the driver interfaces, with which remote serviceability concepts can be implemented.

Initiated by working with the standard, Diebold Nixdorf developed a JavaPOS guide line that complements the UnifiedPOS standard with the main focus on standardized product installers, installation paths and folder structures as well as a simplified configuration of the delivered software stack to the requirements of the applications. The aim of these uniform installation routines, as defined by the guide line, is to simplify the integration of JavaPOS components from different manufacturers and to minimize the necessary manual adjustments.

With ProBase POS 2 Diebold Nixdorf follows this JavaPOS guide line. Many peripheral hardware manufacturers have already agreed to follow this directive or follow it already.

For this purpose, Diebold Nixdorf provides with the JavaPOS Configurator a tool in ProBase POS 2, which generates a common system configuration from the individual JavaPOS components of several manufacturers generated according to the directive. Using application-specific configuration files, this system configuration can be further customized by the JavaPOS Configurator.

---

[1] www.javapos.com
[2] OPOS - OLE for Retail POS
[3] www.nrf.com/resources/retail-technology-standards/unifiedpos
[4] www.nrf.com/resources/retail-technology-standards-0
[5] CIM - Common Information Model

DIEBOLD NIXDORF

2 General information

2.1 Background

The increasing use of Microsoft operating systems on retail market systems as well as the associated standard OPOS in 1994 had at that time led to the first module-based provision of the peripheral drivers according to the OPOS standard.

With the emerge of the Java technology to be usable for retail branch solutions and the introduction of this programming language into the POS application world in 1997, it became necessary for Diebold Nixdorf to provide the corresponding JavaPOS drivers.

Based on evaluation results on Java, a close cooperation between Diebold Nixdorf and its retail customers developed in the realization of projects based on Java technology. In the past, JOWE[6] (Java OPOS Wrapper) was the first available software / abstraction layer for controlling retail peripherals from Java applications under Microsoft Windows operating systems.

At the same time, the standard JavaPOS developed under the proactive cooperation of Diebold Nixdorf in the relevant committees for this technology approach. On the background of the progress in the international JavaPOS standardization, the retail market could now also be offered a pure JavaPOS adapted to the current requirements of the market.

About two years later, the standardization committee decided to meet the requirements for the synchronization of OPOS and JavaPOS and the standard UnifiedPOS was created. UnifiedPOS is an acronym for Unified Point of Service (UPOS), which is used to unify the OPOS and JavaPOS interfaces.

UPOS is not really a driver implementation but rather an abstract definition of interfaces with the realization of this definition into an applicable code as drivers for POS applications in the form of OPOS (as Active X or ATL objects in the world of Microsoft) and in the form of JavaPOS, respectively (as JAR module in the Java world and therefore for Windows and Linux operating systems).

Since 2002, with the introduction of the .NET Framework on Microsoft Windows operating systems, there has been another technology for application programming and starting 2003 there were requests for a further implementation of UPOS for .NET based POS applications. This POS for .NET (P4DN) implementation of the UPOS standard is based on the .NET technology of the Microsoft Windows operating systems.

The provision of these three driver stacks, which have all been implemented according to the same design principles, has led to ProBase POS. The peripheral interfaces JavaPOS, OPOS and P4DN will be delivered with ProBase POS in the future.

As a further interface, Diebold Nixdorf also offers the interface CPOS for POS applications, which were developed in the programming language C. This is also an implementation according to the UPOS standard; here the access takes place via the usual methods in the C programming world.

The basis of ProBase POS is the implementation of the UPOS standard for JavaPOS, which is always the interface to the peripherals devices. The OPOS, CPOS and P4DN interfaces are supported when

[6] no longer supported

the Unified Device Manager (UDM) is used at the same time. And can thus be used by applications, especially in the case of OPOS.

2.2 Environment at Diebold Nixdorf

ProBase POS can be used on all released POS BEETLE systems under the ProBase POS 2 approved operating systems such as Windows XP, POSReady 2009, Windows 7, POSReady 7, Windows 8.1 Pro, Windows 8.1 Industry Pro and Windows 10 IoT as well as WNLPOS 2, WNLPOS 3 and WNLPOS 4.

The approved operating systems for each BEETLE system can be found within the current POS configurator *BEETLE Systems and Peripherals configurator* in the DN intranet portal (go to Intranet > Portfolio > Our Portfolio > Services > Configurator/Price list > BEETLE Systems and Peripherals configurator & price list).

A list of Diebold Nixdorf peripherals and operating systems supported by ProBase POS 2 can be found in chapter *10 - Specifications*.

ProBase POS can also be used on some self-checkout/automated-checkout systems (BEETLE /Certo, BEETLE /iScan and Pay-Tower). Further peripherals and systems are supported by the ProBase Retail package specially designed for the ACO and SCO market.

2.3 Provision

The currently released ProBase POS versions are available for download in the DN intranet portal (go to Intranet > Portfolio > Our Portfolio > Software Solutions > Retail Software Solutions > System Software & Operating Systems > System oriented Software > ProBase POS) as well as for download on the DN website (go to Website > Support > Customer Reference Manuals > Wincor Customer Reference Manuals > Support > Downloads > POS-/Kiosk-Systems, Peripherals > Software > ProBase POS).

ProBase POS is also available as a software package on all BEETLE systems pre-installed with Linux or Windows.

Further documentation such as user manual, installation instruction and migration instruction can be found in the DN intranet portal and on the DN website under the above links.

In addition, ProBase POS has its own documentation in the form of ReadMe files and a technical documentation for integrators and developers as HTML files, which is located in the directory *\doc* below the ProBase POS installation folder (default path under Windows is *C:\Program Files\javapos\WN-ProBasePOS*, respectively */opt/wn/javapos* under Linux).

ProBase POS and the documentation for ProBase POS can also be provided via GCCC[7] with an email request to retailswsupport@dieboldnixdorf.com and the additional keywords "*probase pos*" within the email subject.

2.4 Maintenance and service

The maintenance service for ProBase POS is limited to the POS systems, POS peripherals and operating systems approved by Diebold Nixdorf.

[7] Global Customer Care Center

Diebold Nixdorf does not provide ProBase POS support for non-Diebold Nixdorf-approved system units, extensions and peripherals as well as for POS systems, peripherals and POS applications from third-party manufacturers.

An expansion of the support for new peripherals, improvements to the product as well as error corrections generally take place within the scope of the normal product cycle. Here, 3 to 4 ProBase POS releases are planned per year.

An error correction requires that an error has previously been reported by the customer or partner via a defined communication channel and that this error can be reproduced with the used and also the latest released ProBase POS version. By default this communication has to be done via GCCC by email to retailswsupport@dieboldnixdorf.com with the keywords "*probase pos*" in the subject.

Inquiries about customer-specific extensions for ProBase POS should also be made via the established communication channel to GCCC with an email to retailswsupport@dieboldnixdorf.com and the additional keywords "*probase pos*" in the subject.

Note: An automatic claim for short-term provision of a correction does not result from this basic maintenance and service. Diebold Nixdorf also reserves the right to refuse requests for expansion, or to implement it only within the limitation of project-related services.

2.5 License agreement / rights of use

The software ProBase POS 2 is subject to license and usage conditions. The exact wording of the End User License Agreement can be found in the appendix, Chapter *11.1 - End-user license agreement*.

DIEBOLD NIXDORF

3 Current service and features

ProBase POS 2 provides the well-known interfaces JavaPOS, OPOS and POS for .NET (P4DN) as well as CPOS, an interface for C-based applications, for Windows and Linux based operating systems. These interfaces are all implementations after Unified POS specifications with the following versions:

- JavaPOS 1.13
- OPOS UDM 1.13
- P4DN 1.12
- CPOS 1.13

Note: OPOS and P4DN are reserved for the Windows operating systems, since the basic technology is only available under Windows.

The base of ProBase POS is the JavaPOS of Diebold Nixdorf. The interfaces OPOS, CPOS and P4DN are supported if the UDM is used at the same time.

The interfaces OPOS 1.6 and 1.3, JavaPOS 1.7 and 1.5 as well as RDI and LRDI are not supported by ProBase POS and cannot be operated in parallel with the interfaces of ProBase POS.

3.1 Unified Device Manager

The UnifiedPOS standard describes the device interfaces independently to the implementation. Normally, this is done with a Unified Modeling Language (UML). However, different technologies and implementations are used in the retail sector, for which some separate attachments have also been added to the UnifiedPOS documentation. The most important implementations of the standard are:

- OPOS (since 1994)
- JavaPOS (since 1998)
- POS for .NET (since 2006)

Hardware manufacturers must offer all 3 implementations to support the entire UnifiedPOS standard. Differences in the implementation technologies and therefore in the methods, properties and events, as well as differences between Windows and Linux operating systems, can lead to different behavior of the addressed hardware although the implementations all follow the same standard. Further requirements of the application developers to support other technologies (for example in the Linux area further non-Java based API) have not been taken into account yet.

Normally, there is no way to allow an application written in a particular technology to access a UnifiedPOS device interface written in another technology. This usually requires an adapter or wrapper.

Diebold Nixdorf has developed the Unified Device Manager based on the different implementation technologies, the different operating system types (Windows and Linux) as well as the different implementations of the UnifiedPOS standard.

The Unified Device Manager provides a socket-based client-server construct, which is based on JavaPOS as the basis for communication with the hardware and provides all implementation technologies according to the UnifiedPOS standard via generic service objects. This approach is also important for JavaPOS as soon as the UDM client and the UDM server are to be used on different devices (e.g., mobile POS).

The following figure shows the UDM architecture as it is provided for Windows.

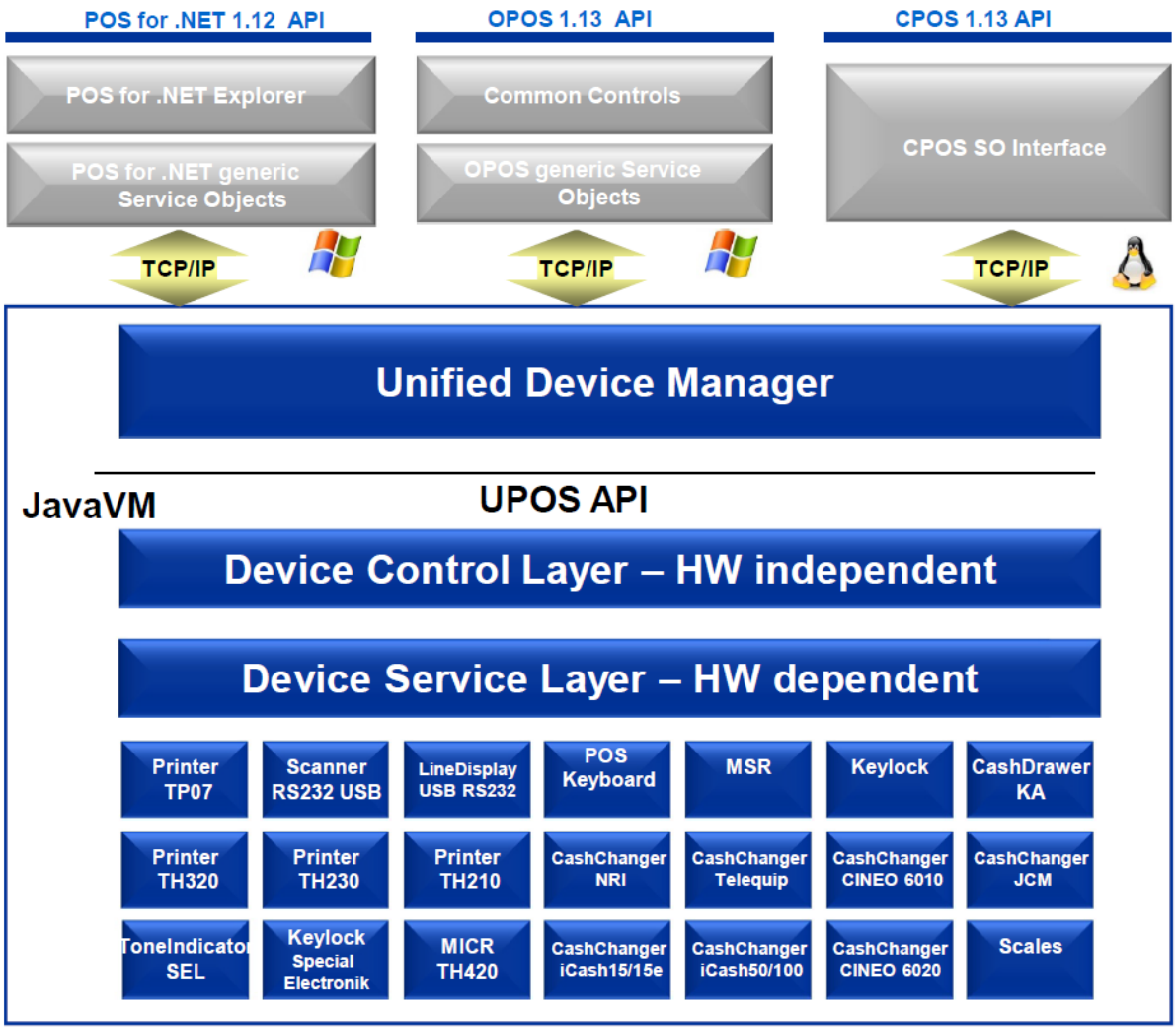


Figure 1: UDM architecture under Windows

The generic service objects convert the method calls of the application into corresponding JavaPOS calls, just as the answers from JavaPOS are converted back into the target implementation for the application. For the application itself, the UDM and the shared JavaPOS are presented transparently.

This results in the following advantages for the customer:

- An identical behavior of the device services under OPOS, JavaPOS, POS for .NET and CPOS
- In the OPOS area, the common controls can be used again
- Coexistence of Diebold Nixdorf device services or service objects with those of third-party manufacturers is possible
- UnifiedPOS remains the standard long-term constant
- A common software stack for POS peripherals, SCO and ACO systems
- Harmonized interfaces for applications such as TP.net, TPiScan and retail partner applications

3.2 JavaPOS guide line

By working with the UPOS standard, it turned out that not everything has been standardized or that there are no guidelines for the implementation. As a result, Diebold Nixdorf, together with selected partners and customers, developed a JavaPOS guideline[8][9] which includes the following topics:

- Packaging and delivery of JavaPOS components
- Configuration of the JavaPOS modules to the desired hardware
- Handling additional functions for firmware and device settings
- Handling of additional inventory and statistic data by JavaPOS
- Integration of JavaPOS components from different manufacturers

Main focus of this guideline is to simplify the integration of JavaPOS components from different manufacturers through uniform installation routines. The integration is not done by copying individual JAR files, but by referencing them. By embedding the JAR files into compliant installers, they are automatically updateable without manual adaptation.

With ProBase POS 2 Diebold Nixdorf follows this JavaPOS directive. Many peripheral hardware manufacturers have already agreed to follow this directive or follow it already.

The biggest differences between the ProBase POS 2 and the ProBase POS 1.1 versions and related to the JavaPOS guideline can be seen in the various installation paths, the folder structures therein, and in the storage locations of the user data such as JAR files, dynamic link libraries, shared objects, log files and configuration files, which are defined by the guideline. The requirements for the paths to be used correspond to the usual standards on the operating systems (for example, for Linux LSB) with the necessary extensions for the use of the JavaPOS architecture.

To minimize the integrational effort, Diebold Nixdorf provides with the JavaPOS Configurator in ProBase POS 2 a tool, which generates a common system configuration from the individual JavaPOS components of several manufacturers generated according to the directive. Using application-specific configuration files, this system configuration can be further customized by the JavaPOS Configurator.

3.3 Version numbering concept

ProBase POS 2 also converts to a semantic version numbering. The version numbers of ProBase POS follow the schema MAJOR.MINOR.BUILD.

Example: 2.0.45

The concept behind semantic version numbering is that the MAJOR number is changed when there are changes that lead to incompatibilities. The MINOR number is changed if the product changes or extensions do not lead to any incompatibilities and the product remains fully downwards compatible. The BUILD number represents only the development step of the product of the respective MAJOR.MINOR version and is generated during the build process within the development.

[8] JavaPOS General Requirements for Linux; Diebold Nixdorf; Peter Duellings; version 1.6
[9] JavaPOS General Requirements for Windows; Diebold Nixdorf; Peter Duellings; version 1.3

3.4 Product installer

3.4.1 Variants

The ProBase POS 2 product installer is available as 32-bit and 64-bit versions for Windows and Linux operating systems.

Note: The setup variants of 32 or 64-bit are based on the process architecture of the application or on the JavaVM architecture to be used. For example when using a 32-bit application/JavaVM on a 64-bit Windows, the 32-bit ProBase POS Installer must be used.

The provided product installers follow the naming scheme:

Windows

- WN-ProBasePOS-MAJOR.MINOR.BUILD-x64.exe
- WN-ProBasePOS-MAJOR.MINOR.BUILD-x86.exe

Linux

- wn-probase-pos-MAJOR.MINOR.BUILD-i386.rpm
- wn-probase-pos-MAJOR.MINOR.BUILD-x86_x64.rpm

Example: WN-ProBasePOS-2.4.7-x86.exe

For a detailed listing of supported operating systems, see Chapter *10.2 - Supported operating systems*.

3.4.2 Installation profiles

Windows

The current product installer for Windows operating systems is based on Inno-Setup, which provides the concept of installation profiles and can be called interactively as well as remote, silent and unattended.

Here a profile is connected to a certain implementation technology, which allows the programmatic access to peripheral devices. The concept allows the user to choose the appropriate technology for the application. The installer ensures that all components for the selected profile are installed correctly and completely, as well as configured. ProBase POS 2 itself is organized internally in components.

If necessary, the user can also make changes to the selection of the components specified by the preselected profile by selecting or deselecting components in the Custom Installation profile. However, the installer cannot perform a consistency check of the components in the modified profile, so the user must ensure that all required components are installed. Otherwise, the application may not or does not work properly.

The following table shows the currently defined profiles in the product installers for Windows:

Profile Name	Profile Meaning
JavaPOS Installation	Installation for JavaPOS based applications. (Installer default profile)
OPOS Installation	Installation for OPOS based applications. Forwards OPOS calls to WN's JavaPOS implementation using the UDM technology.
POS for .NET Installation	Installation for POS for .NET based applications. Forwards POS for .NET calls to WN's JavaPOS implementation using the UDM technology.
CPOS Installation	Installation for C based applications. Forwards C calls to WN's JavaPOS implementation using the UDM technology.
Custom Installation	For adjustments to the component selection, which were selected by the previously selected installation profile.

Note: Since OPOS generally exists only as a 32-bit API, the profile *OPOS Installation* is currently only available in the 32-bit installer!

It is recommended to use the predefined profiles and to carry out a customized installation only after consultation with the Diebold Nixdorf support team[10].

Linux

The current product installer for Linux operating systems is based on the Red Hat Package Manager (RPM) and does not provide any profiles. All dependencies to other packages are checked by the Package Manager and, if possible, automatically resolved.

The following table provides an overview of the implementations or rather profiles delivered with the respective product installers as there are significant differences between the 32- and 64-bit versions, respectively between the Windows and Linux installers.

[10] retailswsupport@dieboldnixdorf.com

API Name	Windows 32-bit JavaVM	Windows 64-bit JavaVM	Linux 32-bit JavaVM	Linux 64-bit JavaVM
JavaPOS	x	x	x	x
OPOS	x	-		
POS for .NET	x	x		
CPOS	x	x	x	x

3.4.3 Default paths

Windows

ProBase POS 2 will be installed in *C:\Program Files\javapos\WN-ProBasePOS* for architecture identical installations (e.g. 32-bit on 32-bit Windows) and in *C:\Program Files (x86)\javapos\WN-ProBasePOS* for architecture-non-identical installations (32-bit to 64-bit Windows). User data such as configuration files and log files are stored in subdirectories under *C:\ProgramData\javapos* according to the JavaPOS guide line.

Linux

On Linux, ProBase POS 2 will be installed in the directory */opt/wn/javapos*. Configuration files are stored in subdirectories under */etc/opt* and log files are stored in */var/log/wn* according to the JavaPOS guide line. Starting with ProBase POS 2.3, configuration files from ProBase POS are stored within the *config* directory under */etc/opt/wn/javapos*.

3.4.4 Special features

ProBase POS 2 does not only consist of UnifiedPOS implementations such as JavaPOS, OPOS and P4DN or the Unified Device Manager, but also provides a number of tools for configuration, testing and logging. Further information can be found in the chapter *6 - Configuration*, *7 - Logging* and *9.3 - Test tools*.

The product installer uses some of these tools in the last stages of the setup routine to configure the previously installed components, or creates tasks or services for these tools.

At the end of each installation under Windows or Linux, the JavaPOS Configurator will be called once in order to create an initial JavaPOS configuration. Further information about this can be found in chapter *6.1 - JavaPOS configuration*. The Linux installer also sets up an autostart task for the JavaPOS Configurator at system start.

Windows

If the OPOS profile or the OPOS UDM adapter in the *Custom Installation* profile has been selected, the product installer will call the OPOS UDM Configuration Updater once and set up a permanent task for the OPOS UDM Configuration Updater in Windows. This task is triggered by default by each system start and user login and then executed. Further information can be found in chapter *9.2 - OPOS UDM Configuration Updater*.

Note: It is recommended to restart the system after the installation, since the final configuration takes place in the start phase.

4 Installation of ProBase POS 2

4.1 Prerequisites

The installation of ProBase POS should only be carried out on the operating systems supported by ProBase POS (see chapter *10.2 - Supported operating systems*).

The use of ProBase POS and the installation of ProBase POS require additional software components and / or frameworks (see chapter 10.3 - Software requirements).

4.2 Installation under Windows

The product installer may run in interactive mode as well as in silent mode. It also provides the possibility of performing so-called unattended installations based on an initial master installation.

For more information on the installation, refer to the ProBase POS 2 Installation Guide for Windows.

4.2.1 Interactive installation

The product installer supports interactive installation, where the end-user can select interactively:

- the installation profile
- or a custom installation by selecting/deselecting particular components
- the Windows start menu shortcut group

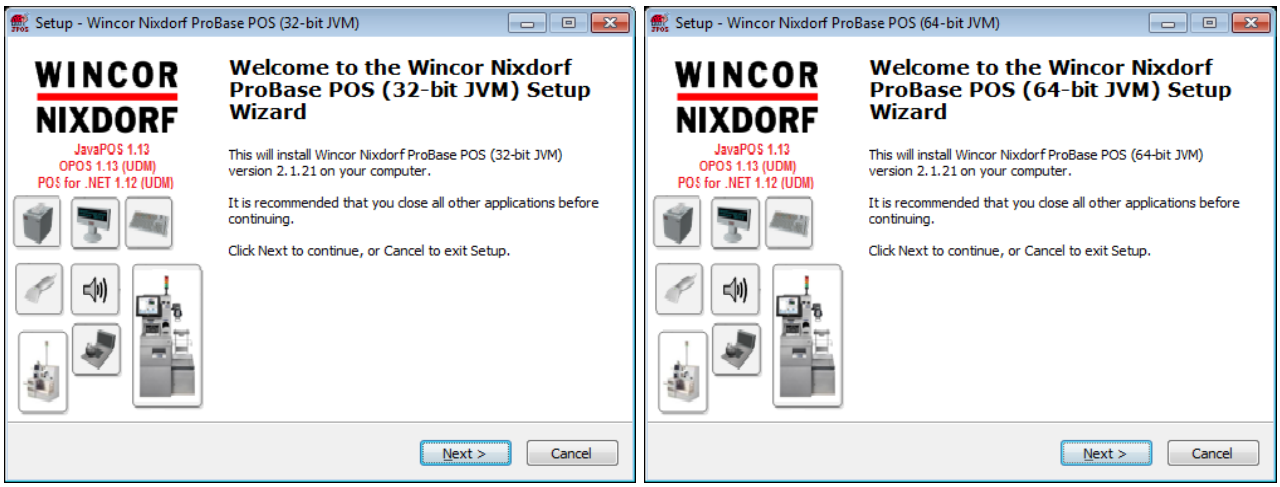
Example: Interactive installation

Note: Consecutively, the pictures of the 64-bit version are used except there is a difference between the installation for the 32-bit and 64-bit JavaVM.

Note: The installer setup can be cancelled on each step of the process by clicking the [Cancel] button. If the installation process is already in the step of copying files, then the installation abort will roll-back the changes done to the system. Before this step, nothing will have happened to your system. By clicking the [Back] button, you can go one step back in the installer setup if needed.

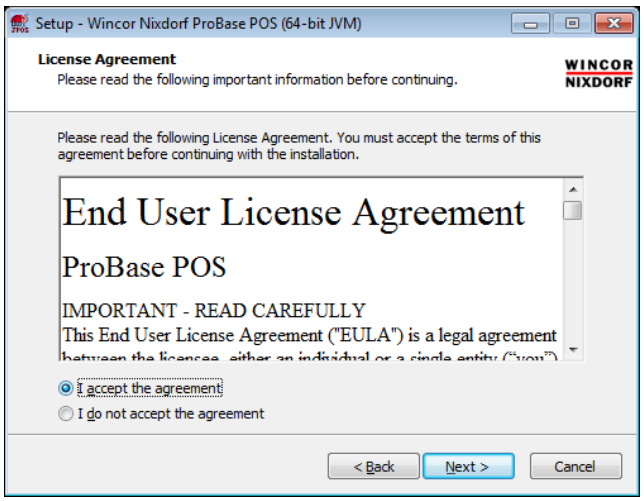
After launching the ProBase POS 2 installer, one of the following welcome-screens will appear.

Dialog: Welcome to Setup Wizard



- Click the [Next] button to go on with the setup.

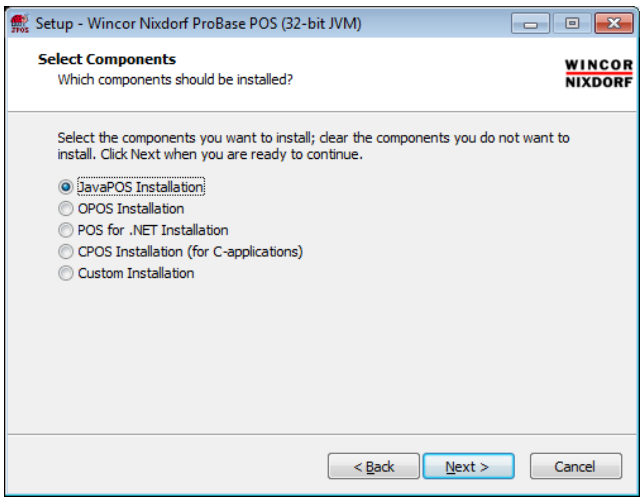
Dialog: License Agreement

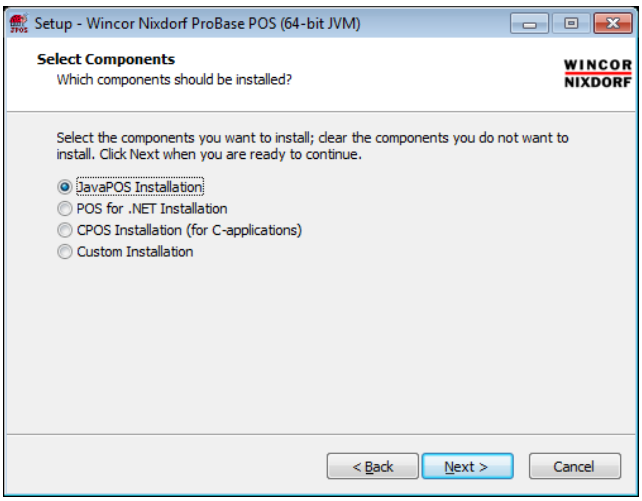


- Choose [I accept the agreement] after reading.
- Click the [Next] button.

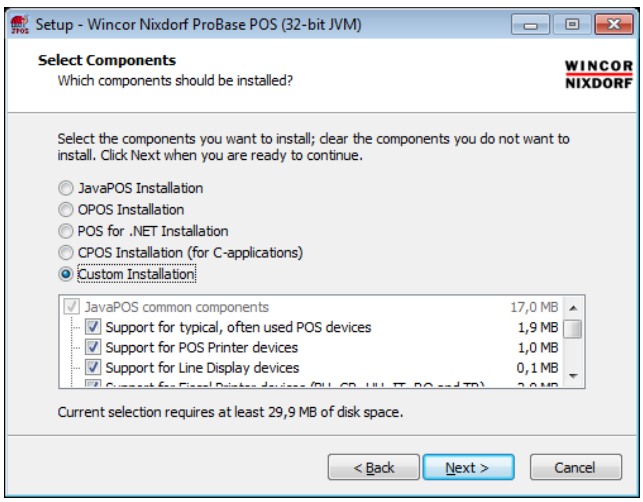
Dialog: Select Components

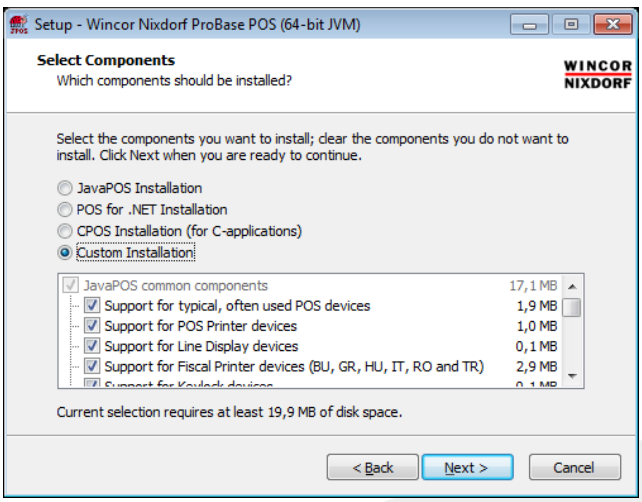
In this dialog, one of the predefined installation profiles can be selected. Alternatively, you can create your own installation using the [Custom Installation] profile and selecting the appropriate components.





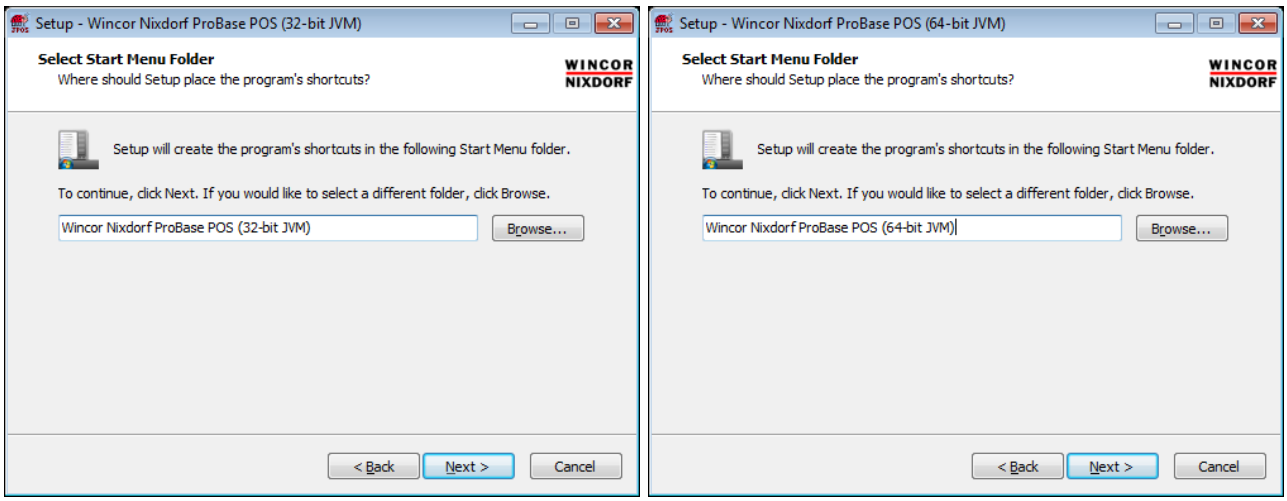
- Select the components to install by using one of the installation profiles.
- Click the [Next] button.





- Or choose [Custom Installation] and check the features you want to install from the list.
- Click the [Next] button.

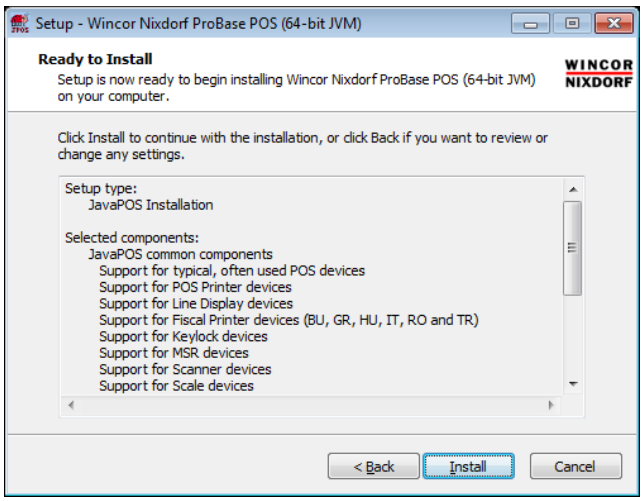
Dialog: Select Start Menu Folder



- Specify a start menu folder for this ProBase POS version.
- Click the [Next] button.

Dialog: Ready to Install

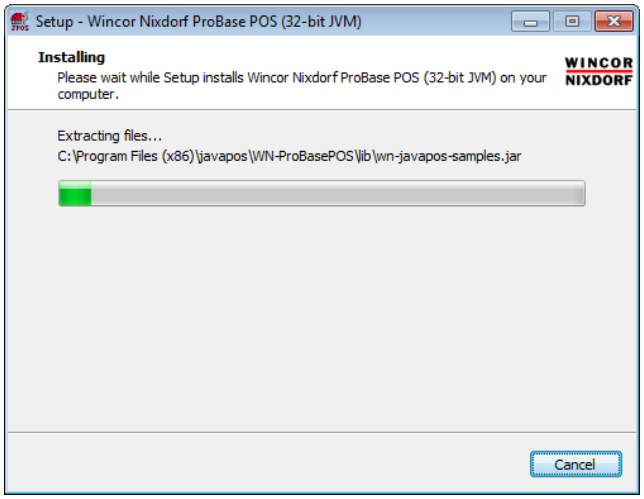
The installer is ready to proceed with the installation and provides an installation summary for letting you check all your previously made choices.

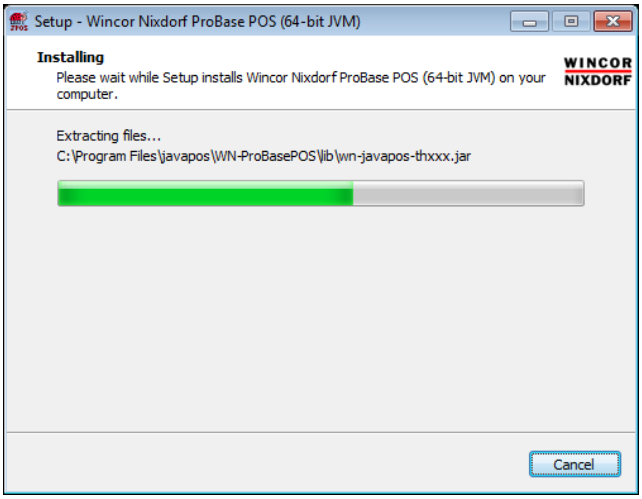


- Click the [Install] button.

Dialog: Installing

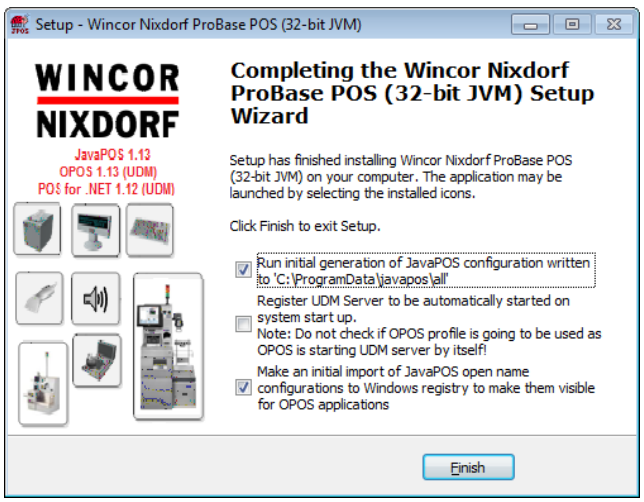
This dialog shows you the installation progress and what the installer does in this moment. Please wait until the installer finishes this step, except you want to cancel the installation (click the [Cancel] button if so).

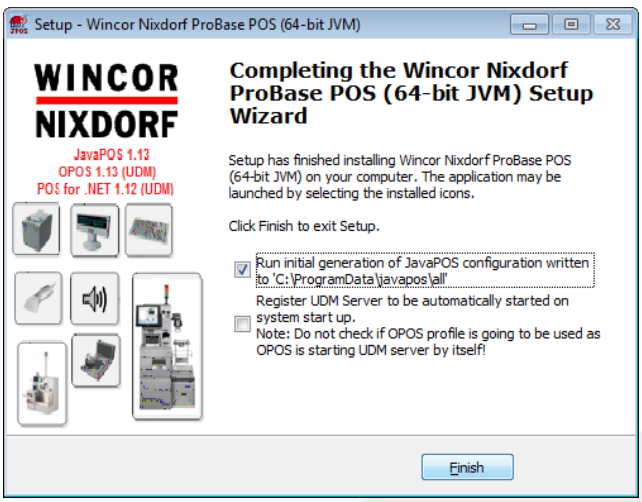




Dialog: Completing the Setup

If no error had been encountered and everything had been finished successfully, you are going to see a last dialog, informing you that the installation is complete and providing the possibility to perform prepared post installation actions.





- Click [Finish] to end the installation.

Note: The number of check boxes for post installation actions appearing in this dialog depends on the selected profile. Typically different JavaPOS configuration alternatives are going to be generated. By default all check boxes should be remain selected to ensure proper configuration for all parts.

4.2.2 Silent installation

A silent installation of ProBase POS 2 is possible. For this, the product installer must be called with the */SILENT* parameter via the command line.

During a silent installation, the wizard and other background windows are not displayed. However, a window showing the progress of the installation is displayed.

If this window is not to be displayed, you should use the completely silent installation mode with the */VERYSILENT* parameter. The JavaPOS installation profile is installed by default in silent installation modes.

If a different selection of components is required, an unattended installation is to be carried out (see chapter *4.2.3 - Unattended installation*).

4.2.3 Unattended installation

By using two command line parameters, an unattended installation is also possible.

To do this, a supervised installation on a system must be executed with the parameter */SAVEINF="pathToFile"* in the first step. The following installation must be carried out with all necessary specifications and settings. These settings and changes to the setup are stored in the specified file.

With this file, the product installer and the call parameters */LOADINF="pathToFile"* as well as */SILENT* or */VERYSILENT*, this recorded installation can now be run automatically, still and unattended on other systems.

4.3 Installation under Linux

RPM installation packages, as are common on Red Hat based operating systems, are delivered without interactive user mode. The installation can be started from the desktop with a double click on the installation package or from the terminal console by using of the following command:

```
rpm -ihv <ProBase POS installer-package.rpm>
```

Example:

```
rpm -ihv wn-probase-pos-2.0.46-i386.rpm
```

After all dependencies have been solved by the RPM Package Manager, the ProBase POS 2 components are installed. The installation progress and further information will be displayed on the console or the progress window.

Note: The product installer will install the complete content including UDM server and UDM C-client with the CPOS API. To use CPOS, the UDM server must be started manually before the application. It is possible to enter the UDM server as a service whereupon the UDM server will be started automatically at each system start. For more information please check chapter *6.6.1 - UDM server configuration*.

Further details on the installation can be found within the ProBase POS 2 Installation Guide for Linux.

DIEBOLD NIXDORF

5 Uninstallation of ProBase POS 2

5.1 Uninstallation under Windows

The product uninstaller will remove all components, previously installed by the product installer, even if the profile has been changed by subsequent installation attempts. See Inno-Setup documentation[11] for details.

5.1.1 Interactive uninstallation

There are three possibilities offered to run the uninstallation of the product interactively:

- direct call of the uninstaller *unins000.exe* located under the directory *<ProBase POS Installation-Directory>\Uninstall-WN-ProBasePOS*
- use the Windows start menu entry *Uninstall ProBase POS* at the location *Start Menu > all programs > Wincor Nixdorf ProBase (xx JVM)*, where xx is 32-bit or 64-bit
- use the Windows system tool with *Control Panel > Add or Remove Programs*

Note: It is recommended not to call the uninstaller directly from the Windows Explorer, as the uninstallation log won't be created (see chapter 7.2 - Logging during uninstallation for more details).

5.1.2 Silent uninstallation

Like for the installation, it is also possible to execute the uninstaller silently by applying the command line argument */SILENT* or */VERYSILENT*.

Example:

```
"C:\Program Files\javapos\WN-ProBasePOS\Uninstall-WN-ProBasePOS\
unins000.exe" /LOG="C:\ProgramData\javapos\wn\logs\uninstall_PBR.log"
/SILENT
```

Note: Unlike the installation, the uninstallation log is not activated by default, and therefore, it is recommended to call the uninstaller with the */LOG="PathToLogFile"* switch also.

5.2 Uninstallation under Linux

In order to uninstall ProBase POS 2, the RPM Package Manager is used as in the installation. The following call on the terminal console deletes all installed packages and services, even if they have been modified subsequently:

rpm –e <ProBase POS installer-paket name without extension>

Example:

```
rpm –e wn-probase-pos-2.0.46-i386
```

[11] http://www.jrsoftware.org/ishelp/

On WNLPOS 3 the system tool *Add/Remove Software* at *System* > *Administration* can also be used. Search for the packages containing *wn-pro*, uncheck the ProBase POS package and apply these changes to remove the ProBase POS package.

6 Configuration

6.1 JavaPOS configuration

ProBase POS 2 comes with an automated JavaPOS configuration generation to make integration with JavaPOS applications easier and more convenient, and to ensure that ProBase POS updates become visible to the application environments immediately after the installation and rebooting the applications. The automatic configuration is performed by the included JavaPOS Configurator.

The JavaPOS Configurator is a small Java program that lists all JavaPOS files and paths, analyzes all XML files (JavaPOS configuration files, peripheral configuration files), and generates the combined JavaPOS system configuration *jpos.xml* and the file *setenv.bat* on Windows or *setenv.sh* on Linux OS to set all necessary environment variables.

At the end of the installation process of ProBase POS 2, the JavaPOS Configurator is automatically started (if not deselected) to generate an initial configuration for JavaPOS.

Windows

With *config_javapos_startup.vbs* contained in the directory *<ProBase POS installation directory>\bin*, the JavaPOS configurator can be started manually. To successfully write the configuration, the JavaPOS configurator caller script needs administrator rights[12].

The JavaPOS Configurator can also be started from the Windows Start menu. By selecting *Start > All Programs > Wincor Nixdorf ProBase POS (xx-bit JVM) > Generate JavaPOS configuration* the script *config_javapos_startup.vbs* will be started to update the JavaPOS configuration.

Note: If additional packages have been installed or the installed packages have been updated, it is necessary to start the JavaPOS configurator manually.

Linux

With *wn_javapos_config.sh* contained in the directory *<ProBase POS installation directory>/bin*, the JavaPOS Configurator can be started manually, but must be executed with root privileges to successfully write the configuration in the defined directories. Restarting the POS system is also sufficient since the JavaPOS Configurator is configured to run at every system boot (via the link *71_wn_javapos_config.sh* in the directory *<ProBase POS installation directory>/startup.d*).

Note: If additional packages have been installed or the installed packages have been updated, it is necessary either to start the JavaPOS configurator manually or to restart the POS system.

It is possible to customize the output of the JavaPOS Configurator by means of an application-specific configuration file named javapos.config.properties. The configuration file contains only the application-specific adaptations that differ from the default configurations of the JavaPOS modules. These are, for example, adjustments such as the selection of the devices used, the necessary COM port settings for the RS232 devices used, or generally all adjustments to JavaPOS configuration parameters.

[12] Since ProBase POS 2.2, the caller script will call back to the user for administrator rights.

This application-specific configuration generation leads to a separate JavaPOS configuration file jpos.xml.

Due to the JavaPOS configurator and the possibility of application-specific customization, it is not necessary to change the original XML files in the <ProBase POS installation directory>\xml directory under Windows or <ProBase POS installation directory>/xml under Linux. In addition, this configuration is maintained through update installations of ProBase POS. Otherwise the changes to the original XML-files would be lost.

This customer-specific configuration is hereinafter referred to as the target configuration.

6.1.1 The JavaPOS Configurator output

At each run, the JavaPOS Configurator checks all available JavaPOS XML-files and the application-based configuration file *javapos.config.properties* and creates or replaces the existing files *jpos.xml* and *setenv.bat* of the target configuration in the directory *C:\ProgramData\javapos* under Windows or *jpos.xml* and *setenv.sh* of the target configuration in */etc/opt/javapos* under Linux.

Note: For reference and test purposes, the JavaPOS Configurator generates or updates an additional set of configuration files that are located in the directory *C:\ProgramData\javapos\all* under Windows, or */etc/opt/wn/javapos/all*[13] under Linux. This configuration includes all available devices and is hereinafter referred to as the *all device configuration*.

6.1.2 Customizing the JavaPOS Configurator output

Windows

To customize the output of the JavaPOS Configurator, the configuration file *javapos.config.properties* must be stored in a directory that is meaningfully named after the application. This directory with the configuration file must be located below the directory *C:\ProgramData\javapos* and this directory must still be made known to the JavaPOS configurator. To do so the application/directory name must be stored in the environment variable *%JAVAPOS_APPLICATION_NAME%*. This environment variable is only required at the runtime of the JavaPOS Configurator so that the subfolder and the associated configuration file can be found.

We recommend to use a small batch file named *setapplicationname.bat*, which should be located in the directory *C:\ProgramData\javapos* to set this environment variable because the JavaPOS Configurator will try to find and run this batch file first.

Example:

Content of C:\ProgramData\javapos\setapplicationname.bat

```
@REM Setting the Environment Variable JAVAPOS_APPLICATION_NAME
@SET JAVAPOS_APPLICATION_NAME=TestApplication
```

With the used application name *TestApplication*, the configuration file *javapos.config.properties* must be stored under *C:\ProgramData\javapos\TestApplication*.

[13] until ProBase POS 2.2 still /etc/opt/wn/jpos-all

Note: A template for *javapos.config.properties* can be found in the directory *<ProBase POS installation directory>\config*.

Linux

In order to customize the output of the JavaPOS configurator, the configuration file *javapos.config.properties* must be stored in a directory that is meaningfully named after the application. This directory with the configuration file must be located below the directory */etc/opt* and this directory must still be made known to the JavaPOS configurator. To do so, the application/directory name must be stored in the environment variable *$APPLICATION_NAME*. This environment variable is only required at the runtime of the JavaPOS Configurator so that the subfolder and the associated configuration file can be found.

We recommend to use a small shell script called *setapplicationname.sh*, which should be located in the directory */etc/opt/wn/javapos/config* to set this environment variable because the JavaPOS configurator will try to find and execute this shell script first.

Example:

Content of /etc/opt/wn/javapos/config/setapplicationname.sh

```
# Setting the Environment Variable APPLICATION_NAME
APPLICATION_NAME=TestApplication
```

With the application name *TestApplication*, the configuration file *javapos.config.properties* must be stored in the directory */etc/opt/TestApplication*.

Note: A template for javapos.config.properties can be found in the directory *<ProBase POS Installation directory>\config*.

6.1.2.1 Customization with javapos.config.properties

The so-called properties file should select all the devices used by the application and contain the necessary changes to the device configuration parameters

Note: Only the differences compared to the original configuration as supplied with ProBase POS or other manufacturer-specific products are necessary.

The file can contain comments (beginning with the comment symbol #) and should contain rows with <name> - <value> pairs.

Mainly, the file contains two types of information - a definition of used jposEntries for the intended configuration and a definitions of property values for these jposEntries.

Definition of JposEntries

JposEntries are specified by their names (JposEntry property "logicalDeviceName"; also known as OpenName). In addition, each JposEntry can be assigned a name which is more convenient for the application or is used by the application by default.

Syntax:

jpos.names=<Name1>,<Name2>,<Name3>,…

jpos.name.<Name1>=<Original-OpenName1>
jpos.name.<Name2>=<Originalname2>
jpos.name.<Name3>=<Originalname3>
…

Example:

```
# this is a comment
jpos.names=printer1,scanner1,cashdrawer1
jpos.name.printer1=WN_TH250_COM
jpos.name.scanner1=DLS-Gryphon-GD4135-USB-Scanner
jpos.name.cashdrawer1=WN_CD_PORT
…
```

In this example, the application uses OpenNames "printer1", "scanner1" and "cashdrawer1", where the names "WN_TH250_COM", "DLSGryphon-GD4135 USB Scanner" and "WN_CD_PORT" are the original names as they are defined in the original XML files supplied by the manufacturer.

Definition of property values

The definition of property values is intended for all JposEntry properties, which differ from the original values, as defined in the original XML files provided by the manufacturer.

This can be used, for example, if an RS232 device is connected to a different COM port, as defined in the original XML file, or the signaling time of the scanner must be adjusted.

Syntax:

jposentry.<Original-OpenName>.<Property-Name>=<New-Value>

Example:

```
# this is a comment
jposentry.DLS-Gryphon-GD4135-USB-Scanner.beepDuration=2
# the POS printer is connected to COM1 instead of COM2
jposentry.WN_TH250_COM.portName=COM1
# We are using a cash drawer reporting the status in iverted manner
jposentry.WN_CD_PORT.invertedStatusPolarity=true
…
```

6.2 JavaVM configuration

ProBase POS uses JavaPOS as the basis for all driver interfaces. A JavaVM is therefore required for the use of ProBase POS (see chapter *10.3.1 - Minimum prerequisites*).

Normally, JavaPOS verifies whether a JavaVM version is installed and automatically selects a suitable one from all found JavaVM versions.

Windows

Since ProBase POS 2.3, the user has the option to explicitly specify the JavaVM. To do so, a configuration file called *javahome.ini* has to be created under *<ProBase POS installation*

directory>*config*, which then contains the variable *JAVA_HOME* with the path to the desired JavaVM JRE.

Example:

Content of <ProBase POS Installation Directory>\config\javahome.ini

```
JAVA_HOME=C:\Program Files (x86)\Java\jre1.8.0_121
```

Linux

Since ProBase POS 2.3, the user has the option to explicitly specify the JavaVM. To do so, a configuration file called *javahome.env* has to be created under *<ProBase POS configuration directory>**config*, which then contains the variable *JAVA_HOME* with the path to the desired JavaVM JRE.

Example:

Content of <ProBase POS configuration directory>/config/javahome.env

```
JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk-1.7.0.45/jre
```

Note: The configuration file must be created in the directory *config* below *<ProBase POS Installation Directory>*. If it is not present, the Javapos Configurator behaves as before and tries to determine the JavaVM itself.

6.3 OPOS configuration

Typically, the device-specific configuration data in OPOS are stored in the Windows registry. The OPOS supplied with ProBase POS saves the configuration data in the subkey *\OLEforRetail\ServiceOPOS*, depending on the processor architecture, either under *HKLM\SOFTWARE* or under *HKLM\SOFTWARE\Wow6432Node*. Thereunder the devices are grouped into device categories after UnifiedPOS.

A modification of the configuration can be done directly in the Windows Registry. For this purpose, the parameters of the desired OPOS device must be edited below the open name for the device.

Note: Make sure that the changed parameters contain valid values as otherwise incorrect behavior may occur. A check of the values or a pre-selection of the values is not done when editing the configuration within the Windows Registry. In order for the changes to be permanently stored in the registry, the registry editor must be called with administrator rights.

Alternatively, the OPOS configuration can also be changed via the provided *OPOS Config Tool*. This can be done either with the application *HWD55ConfUDM.exe* from the directory *<ProBase POS installation directory>\opos\common\bin* or via the Windows Start menu with *OPOS Configuration Program* under *Start > All Programs > Wincor Nixdorf ProBase POS (xx-bit JVM) > OPOS (UDM) > OPOS Common*.

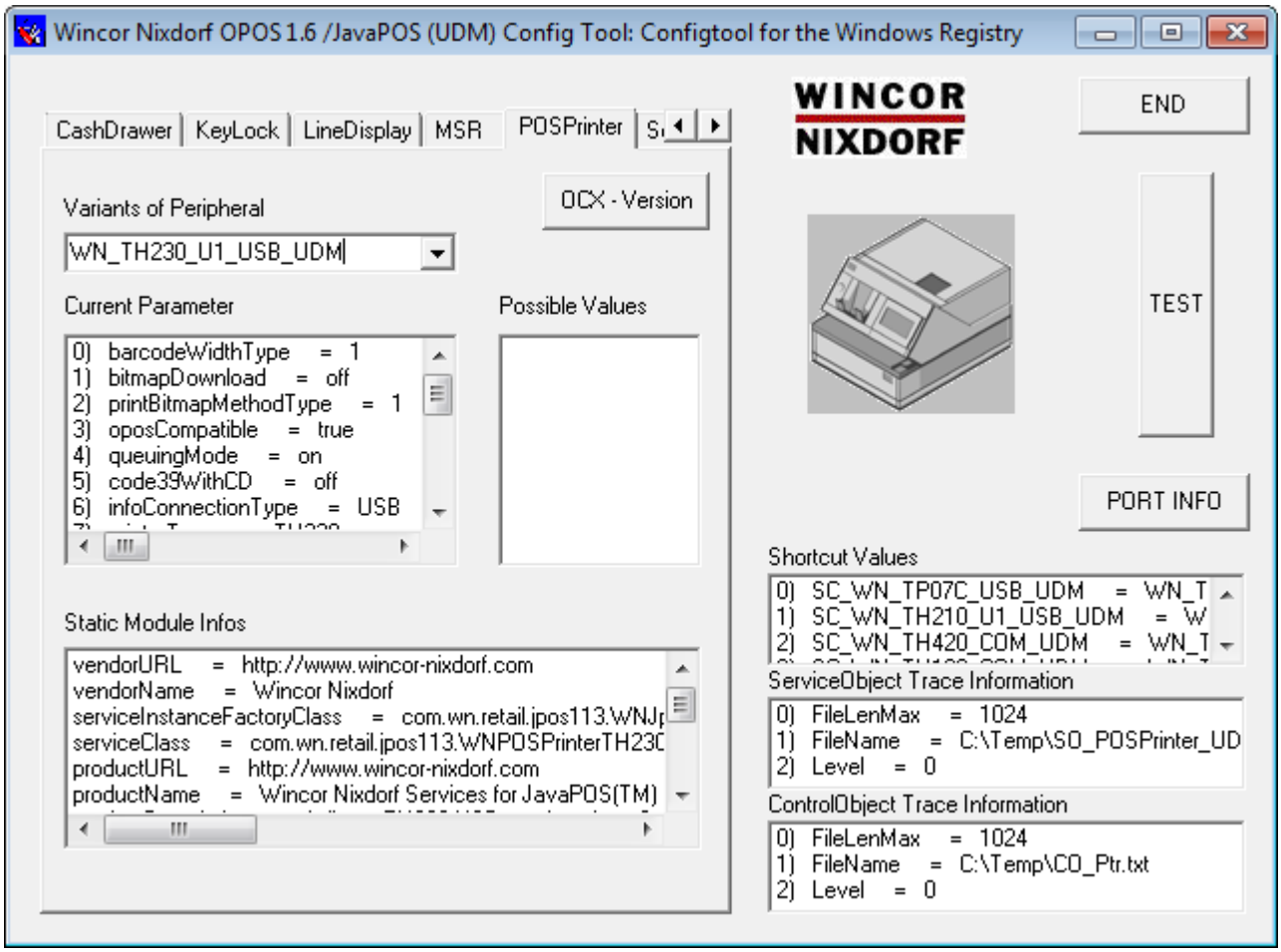


Figure 2: OPOS Config Tool

In this case too, the devices are grouped into device categories according to UnifiedPOS, which can be selected via the tabs in the tool. The device to be edited must then be selected via the *Variants of Peripheral* drop-down list. The configuration parameters shown in the list *Current Parameters* can then be changed via the selection in *Possible Values*.

Note: This variant of the configuration is the recommended way because the parameters can only contain valid values through the predefined selection.
The OPOS configuration tool must be started with administrator rights so that the changes can be permanently transferred to the registry.

6.4 P4DN configuration

The configuration of the POS for .NET interface above the UDM server/client can be set/changed using the tools and methods provided by the POS for .NET Framework from Microsoft.

6.5 CPOS configuration

The CPOS interface does not have a separate configuration above the UDM server/client, the JavaPOS configuration is used.

6.6 UDM configuration

ProBase POS offers the Unified Device Manager (UDM), a server-client architecture based on sockets, in order to be able to provide the OPOS, POS for .NET and CPOS interfaces on the basis of JavaPOS and, on the other hand, to provide a remote capability. The configuration of the UDM is done on the UDM server side via call parameters at the start of the UDM server. The UDM client (OPOS, P4DN and/or CPOS), however, is configured via environment variables.

6.6.1 UDM server configuration

The UDM server delivered with ProBase POS is already configured and ready for use with the default settings for POS systems.

Windows

The following parameters for the UDM server can be changed in the batch file *UDMServer.setup.bat* in the directory *<ProBase POS installation directory>\bin*.

Variable/Parameter	Meaning
JAVAPOS_DATA_HOME	Directory of the JavaPOS configuration; Default is C:\ProgramData\javapos\wn
UDM_DATA_HOME	Directory of the UDM server configuration; Default is C:\ProgramData\javapos\wn\udmserver
UDM_LOG_FILE	Directory and name of the UDM log file; Default is "%UDM_LOG_HOME%\udm-server.port%UDM_PORT%.%USERNAME%.log"
UDM_LOG_HOME	Directory of the UDM log file; Default is C:\ProgramData\javapos\wn\log\udm
UDM_NUM_CONNECTIONS	Maximum number of parallel connections; Default is 5
UDM_PORT	Defines a port for the TCP/IP socket connection; Default is 1131
WATCHFILE	Directory and name of the UDM server watchfile. Used to automatically restart the UDM server as long as the file exists; Default is %UDM_DATA_HOME%\delete_for_stopping_udm

Further parameters for the UDM server can be changed in the batch file *UDMServer.exe.bat* in the directory *<ProBase POS installation directory>\bin*.

Variable/Parameter	Meaning
UDM_SERVER_LOGGING	Enables or disables the logging for the UDM server; Default is „“ (empty)

Note: The UDM server can be started manually via *UDMServer.exe* and terminated via *StopUDMServer.exe*. The executable files start the similar-sounding batch files *UDMServer.exe.bat* and *StopUDMServer.exe.bat* and are found as well as the batch-files under *<ProBase POS installation directory>\bin*.

Note that the batch file StopUDMServer.exe.bat can only stop the UDM server if the same watchfile is referenced (to be set within the batch-file).

Note: In order to simplify the integration of the UDM server into the productive system, the UDM server was configured by default to an automatic startup behavior. The UDM server is automatically started on the first open() call on a device from the UDM client. This automatic startup behavior can currently only be used with OPOS and CPOS.

If this startup behavior is to be changed, the parameter *StartAsChildProcess* must be changed in the Windows Registry under *HKLM\SOFTWARE\Wincor Nixdorf\UDM\Server.*

For further information, please refer to chapter *5.1.4* of the *UDM User Guide* under *<ProBase POS installation directory>\doc*.

Variable/Parameter	Meaning
StartAsChildProcess	Enables automatic starting of the UDM server if the parameter is set to a value of 1 or greater. A value of 0 disables the automatic start and the UDM server has to be started independently before the application starts. Default is 1

Linux

The following parameters for the UDM server can be changed in the shell script *udmStartServer.sh* in the directory *<ProBase POS installation directory>\bin*.

Variable/Parameter	Meaning
UDM_HOME	Directory of the ProBase POS installation; Default is /opt/wn/javapos
UDM_VAR_DIR	Directory of the UDM log file; Default is /var/opt/wn/udm
UDM_ETC_DIR	Directory of the UDM server configuration; Default is /etc/opt/wn/jpos-all
UDM_PORT	Defines a port for the TCP/IP socket connection; Default is 1131

Variable/Parameter	Meaning
WATCHFILE	Directory and name of the UDM server watchfile. Used to automatically restart the UDM server as long as the file exists. Default is $WATCHFILE_DIR/delete_for_stopping_udm
WATCHFILE_DIR	Directory for the UDM Server watchfile. Default is $UDM_VAR_DIR

Note: The UDM server can be started manually via *udmStartServer.sh* to be found under *<ProBase POS installation directory>/bin*. The UDM server is then started as a process in the terminal console and can be terminated by closing the terminal console.

Since ProBase POS 2.2, a UDM server daemon is also included. The UDM server can be started, stopped or restarted via this service. Furthermore, the status of the service can also be queried. To do this, use the following command from the terminal console

service wn-udm <start | stop | restart | status>

Example: Start of the UDM servers

```
service wn-udm start
```

6.6.2 UDM client configuration - CPOS

The UDM client for CPOS delivered with ProBase POS is already configured and ready for use with the standard settings for POS systems.

If necessary, the default settings for the UDM client can be modified using the following environment variables.

Environment Variable	Meaning
WN_JAVAPOS_UDM_LOGFILE	Name und path of the log file of the UDM C-client. If “stdout” is used, then the log output will be directed to standard output (e.g. terminal console on display); Default is „“ (leer)
WN_JAVAPOS_UDM_PORT	The port number of the socket connection of the UDM server; Default is 1131
WN_JAVAPOS_UDM_HOST	The hostname on which the UDM server is running. Typically, it is the system on which the application is running and thus localhost. Default is 127.0.0.1
WN_JAVAPOS_UDM_MAXLINE	The maximum size (number of characters) that can be transmitted through the socket connection using a UnifiedPOS call; Default is 128.000

Environment Variable	Meaning
WN_JAVAPOS_UDM_MAXSTRINGPROP	The maximum size (number of characters) that a string variable can have during a transmission by means of a UnifiedPOS call; Default is 64.000
WN_JAVAPOS_UDM_CONNECTION_TIMEOUT	Time in milliseconds that the UDM client waits for a successful socket connection; Default is 60.000 ms (= 60 s)

Windows

Note: On Windows, environment variables can be created or changed temporarily using the command *set* at the command line or in the application start script. Environment variables can also be created or changed permanently via the system tools such as *Control Panel* > *System* > *Advanced system settings* > *Advanced* > *Environment variables* or the *Windows Registry*.

Linux

Note: On Linux, environment variables can be created or modified easily in the application start script using the command *export*. Environment variables can also be stored permanently under */etc/profile.d* in a separate sh- (bash shell) or csh-file (C shell) with an own filename and therein with the command *export*. In this case a system restart is necessary that the changes can take effect.

Example:

Content of /etc/profile.d/activate-udm-cclient-logging.sh

```
# Setting the Environment Variable WN_JAVAPOS_UDM_LOGFILE
export WN_JAVAPOS_UDM_LOGFILE=/var/opt/wn/log/udm-cclient.log
```

Note: Please make sure that the user does have the necessary execution rights for this script file. Also make sure that the logfile-path exists and that the user does have the write rights for this folder.

6.6.3 UDM client configuration - OPOS

OPOS uses the same UDM client as CPOS. See chapter *6.6.2 - UDM client configuration - CPOS* for more details.

6.6.4 UDM client configuration – P4DN

The UDM client configuration for POS for .NET is included in the *<P4DNUDMAdapter>* section in the configuration file *P4DNUDMAdapter.config* under *<ProBase POS installation directory>\p4dn\bin*.

Parameter	Meaning
port	The port number of the socket connection of the UDM server; Default is 1131
host	The hostname on which the UDM server is running. Typically, it is the system on which the application is running and thus localhost. Default is 127.0.0.1
reconnectTimeout	Time in milliseconds that the UDM client waits for a successful socket connection; Default is 3.000 ms (= 3 s)
latencyTime	Time in milliseconds, the UDM client waits, before responding to a connection error with an error message; Default is 1.000 ms (= 1 s)
ClientCreationStrategy	Defines the connection strategy (number of socket connections) between UDM client and UDM server.

The UDM client configuration for POS for .NET regarding logging is contained in the section *<log4net>* in the configuration file *P4DNUDMAdapter.config*.

Parameter	Meaning
file value	Name und path of the log file; Default is „C:\temp\P4DNUDMAdapter.log“
appendToFile	Determines whether the log entries are appended to the existing file or whether the file is created again; Default is „true“
level value	Defines the log level; Default is „INFO“

For more information, see the P4DN UDM adapter documentation *P4DNUDMAdapter.txt* under *<ProBase POS installation directory>\p4dn\doc*.

7 Logging

Based on the current implementation of ProBase POS, additional logs of the subjacent software layers can be required when using the OPOS, P4DN or CPOS API.

Example:

When using OPOS, logs/traces of the following layers can be required:

- OPOS (ServiceObject, ControlObject)
- UDM Client
- UDM Server
- JavaPOS (DeviceService, DeviceControls)

7.1 Logging during installation

Windows

Logging the installation process is always activated. If the installer is not called with the parameter */LOG="filename"*, the setup log will be stored in the with *%TMP%* defined directory with a unique filename based on the current date *Setup Log yyyy-mm-dd #<number>.txt*.

Example:

Setup Log 2016-10-01 #002.txt

Note: As post installation action (if not disabled), the JavaPOS configurator will be started to create an initial configuration. This JavaPOS configurator run will also be logged. The log files for the JavaPOS configurator are stored within *C:\ProgramData\javapos\all*.

Linux

The product installer is a RPM package, which does not provide any special options to log the installer activities separately. All activities of RPM packages will be registered within the */var/lib/rpm* database. This database can be queried at any time and the output can be formatted as needed and be saved to a dedicated file.

Additionally to the logging at the rpm database, the installation will also be logged to the *yum.log* file at */var/log*, with entries like *Installed: wn-probase-pos-<major>.<minor>-<build>.<architecture>*.

Example:

Installed: wn-probase-pos-2.4-7.i386

Note: As post installation action, the JavaPOS configurator will be started to create an initial configuration. This JavaPOS configurator run will also be logged. The log files for the JavaPOS configurator are stored within */var/log/wn/javapos*.

7.2 Logging during uninstallation

Windows

The logging feature for the uninstallation process cannot be generally activated like it is for the installation process. Therefore and in order to activate logging for the uninstallation process, the parameter */LOG="filename"* has to be appended to the uninstaller call.

The uninstallation logging has the following limitations:

- If the uninstaller is called directly (and without giving the */LOG* option), then no uninstallation log is created.
- The uninstaller cannot append a counter to the log file name as the installer can. Therefore only one uninstallation log per defined name is possible. If the product has been installed/uninstalled multiple times, the uninstallation log will always be re-written.

Note: We recommend to use the product version number as well as the date and/or the time within the log file name (e.g. /LOG=%TMP%/Uninstall_<ProBase POS> <Version>.<Build>.%DATE%.log).

Linux

The uninstallation has the same logging mechanism as the installation. Again, all activities of the RPM package are registered or de-registered within the */var/lib/rpm* database. After a successful uninstallation there is also a corresponding log entry in *yum.log* under */var/log*. This time with entries like *Erased: wn-probase-pos*.

7.3 JavaPOS logging

With ProBase POS 2.1, the JavaPOS logging was changed to the OpenSource logging concept log4j[14]. The new WN logger derived from it uses the log4j libraries in version 1.2.17.

The JavaPOS Logging can be activated with different degrees and different depths. It is possible to define logging globally, for individual device classes or for special devices. It is also possible to use different log levels, whereby the DEBUG log level should be sufficient.

For more information about the new JavaPOS Logging concept, including general and advanced configuration details, see the documentation *LoggingConcept.html* under *<ProBase POS installation directory>\doc\html* (or *<ProBase POS installation directory>/doc/html* under Linux).

Windows

JavaPOS Logging is generally active. The detail level and extent is determined by the WN logger configuration *wn-logger.properties* under *<ProBase POS installation directory>\config*.

To adjust the WN logger output, the following parameters can be modified.

[14] http://logging.apache.org/log4j/1.2/

Parameter	Meaning
log4j.appender.rollingfile.File	Path and name for the log file; Default is C:/ProgramData/javapos/wn/log/javapos.log **Note:** The path disclosure has to be done with „/“ instead the usual „\“ (Linux style).
log4j.appender.rollingfile.MaxFileSize	The maximum size for the log file; Default is 1MB
log4j.appender.rollingfile.MaxBackupIndex	Maximum number of log file backups for log file rotation; Default is 10

Linux

JavaPOS logging is generally active. The detail level and extent is defined by the WN logger configuration *wn-logger.properties* under *<ProBase POS configuration directory>/config*.

To adjust the WN logger output, the following parameters can be modified.

Parameter	Meaning
log4j.appender.rollingfile.File	Path and name for the log file; Default is var/log/wn/javapos/javapos.log
log4j.appender.rollingfile.MaxFileSize	The maximum size for the log file; Default is 1MB
log4j.appender.rollingfile.MaxBackupIndex	Maximum number of log file backups for log file rotation; Default is 10

With default settings the WN-logger records only a few diagnostic information. There are two methods to enable logging for JavaPOS devices.

1. General activation of the logging for all JavaPOS devices by simply commenting out the line with *log4j.rootLogger=DEBUG, rollingfile* and commenting the line with *log4j.rootLogger=INFO, diagnostics*.

Example:

Changing the logging from diagnose information level to debug information level.

Part of content of the file *<ProBase POS installation directory>\config\wn-logger.properties*

```
# default diagnostic logging configuration
#log4j.rootLogger=INFO, diagnostics
# for enabling extensive logging, comment the following line in and the
line above out
log4j.rootLogger=DEBUG, rollingfile
```

2. Activate the logging for individual device classes, devices (via the OpenNames) or even individual JavaPOS layers (DeviceControls, DeviceServices, DCAL) by commenting out the already specified lines or by adding additional entries according to the rules described within *LoggingConcept.html*.

Example:

Activation of logging for POSPrinter and POSKeyboard devices.

Part of content of the file *<ProBase POS installation directory>\config\wn-logger.properties*

```
# configuration patter for enabling the DEBUG logging for a particular
device:
# log4j.logger.<UPOS category name>.<open name>=DEBUG
# e.g., log4j.logger.POSPrinter.WN_TH230_COM=DEBUG, rollingfile


# alternatively: enabling logging on category level:
…
# log4j.logger.PointCardRW=DEBUG, rollingfile
log4j.logger.POSKeyboard=DEBUG, rollingfile
# log4j.logger.POSPower=DEBUG, rollingfile
log4j.logger.POSPrinter=DEBUG, rollingfile
```

Example:

Activation of logging especially for the POSPrinter TH230 COM and the line display BA63 COM

Part of content of the file *<ProBase POS installation directory>\config\wn-logger.properties*

```
# configuration patter for enabling the DEBUG logging for a particular
device:
# log4j.logger.<UPOS category name>.<open name>=DEBUG
# e.g., log4j.logger.POSPrinter.WN_TH230_COM=DEBUG, rollingfile

log4j.logger.POSPrinter.WN_TH230_COM=DEBUG, rollingfile
log4j.logger. LineDisplay.WN_BA63_COM=DEBUG, rollingfile
```

7.4 OPOS logging

The tracing at OPOS devices can, like the configuration, either be set directly in the Windows Registry or with the OPOS Config Tool. The tracing configuration for the individual OPOS ServiceObjects is stored within the Registry below the subkey *\OLEforRetail\ServiceOPOS* and for the OPOS ControlObjects below the subkey *\OLEforRetail\ControlOPOS*.

To activate the appropriate tracing, the parameter *Level* must be changed to a value greater than 0. The parameter *FileName* can be used to adjust the path and filename for the trace file.

Example:

Tracing activated at the OPOS ServiceObject for POSPrinter TH230

```
[HKLM\SOFTWARE\OLEforRetail\ServiceOPOS\POSPrinter\WN_TH230_USB_UDM\Trace]

"FileLenMax"="1024"
"FileName"="C:\Temp\SO_POSPrinter_UDM.txt"
"Level"="1"
```

Example:

Tracing activated at the OPOS ControlObject for POSPrinter

```
[HKLM\SOFTWARE\OLEforRetail\ControlOPOS\POSPrinter\Trace]
"FileLenMax"="1024"
"FileName"="C:\Temp\CO_Ptr.txt"
"Level"="1"
```

Note: In order for the changes to be stored permanently in the registry, the registry editor must be called with administrator rights.

Alternatively, the tracing for the OPOS devices can also be changed either with the application *HWD55ConfUDM.exe* from the directory *<ProBase POS installation directory>\opos\common\bin* or via the Windows Start menu with *OPOS Configuration Program* under *Start > All Programs > Wincor Nixdorf ProBase POS (xx-bit JVM) > OPOS (UDM) > OPOS Common*.

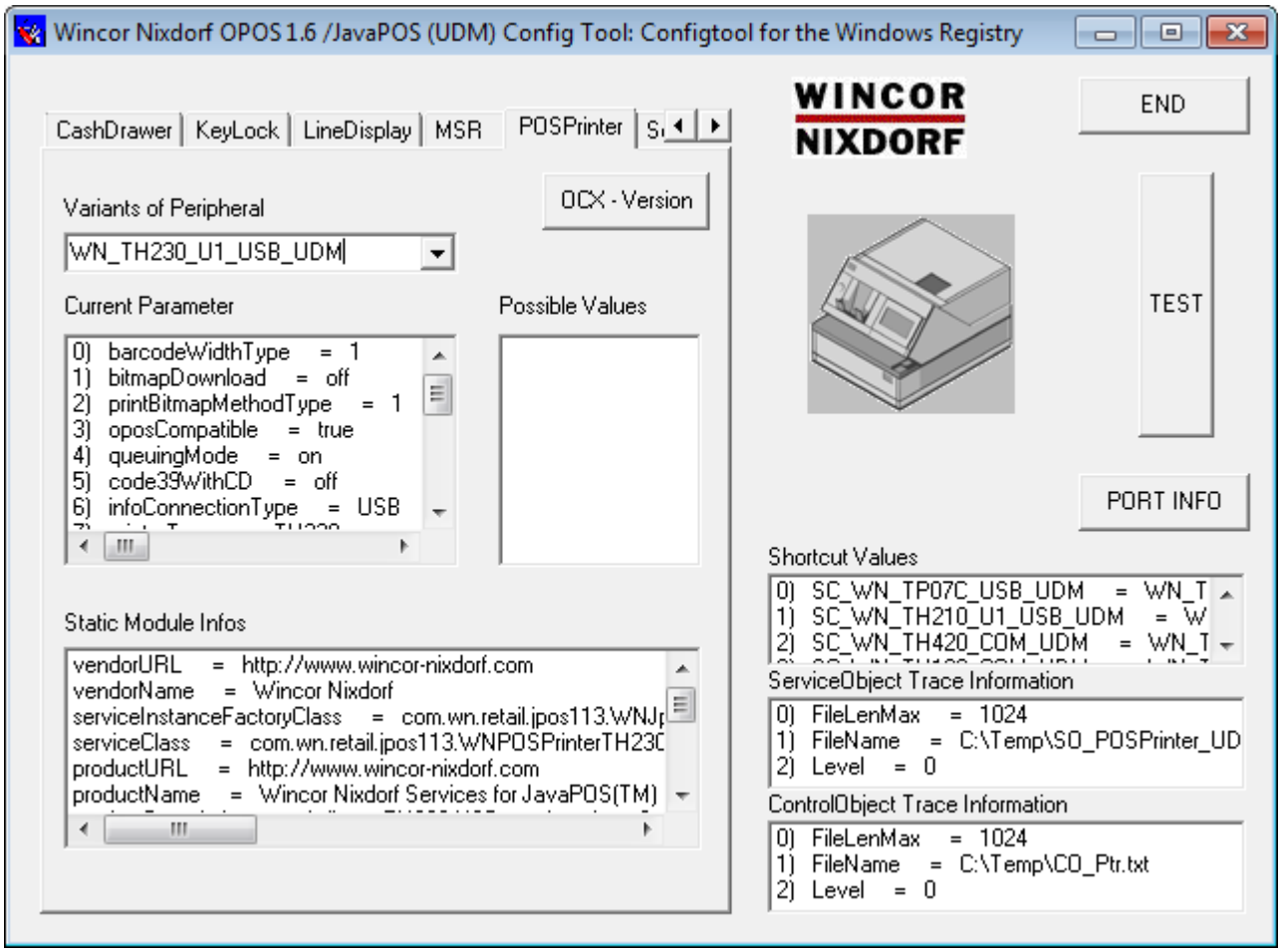


Figure 3: OPOS Config Tool

The *ServiceObject Trace Information* and *ControlObject Trace Information* sections can be used to enable the tracing for the current device or the used control object, and to modify the output.

To activate the appropriate tracing, the parameter *Level* must be changed to a value greater than 0. The parameter *FileName* can be used to adjust the path and filename for the trace file.

Note: The OPOS configuration tool must be started with administrator rights so that the changes are permanently transferred to the registry.

7.5 P4DN logging

Logging of the POS for .NET interface above the UDM server/client can be set/changed using the tools and methods provided by the POS for .NET Framework from Microsoft.

7.6 CPOS logging

The CPOS interface does not have separate logging above the UDM server/client because the application directly uses the UDM C-client layer.

7.7 UDM logging

7.7.1 UDM server logging

Windows

The logging for the UDM server can be activated in the batch file *UDMServer.exe.bat* in the directory *<ProBase POS installation directory>\bin*.

Variable/Parameter	Meaning
UDM_SERVER_LOGGING	Enables or disables the logging for the UDM server; Default is „“ (empty)

The name and directory for the log file of the UDM server can be changed in the batch file *UDMServer.setup.bat* in the directory *<ProBase POS installation directory>\bin*.

Variable/Parameter	Meaning
UDM_LOG_FILE	Directory and name of the UDM log file. Default is "%UDM_LOG_HOME%\udm-server.port%UDM_PORT%.%USERNAME%.log"
UDM_LOG_HOME	Directory of the UDM log file. Default is C:\ProgramData\javapos\wn\log\udm

Note: If the startup behavior of the UDM server has been changed in the registry, the logging of the UDM server can also be configured in the Windows registry. For further information, please refer to chapter *5.1.4* of the *UDM User Guide* under *<ProBase POS installation directory>\doc*.

Linux

The logging for the UDM server can be activated with the shell script *setlogging_udmserver.sh* from the directory *<ProBase POS installation directory>/bin*. Two log-levels are possible, which are passed to the script by a parameter with the value 1 or 2. The meaning of the values is as follows.

1 disables logging / activates normal logging

2 enables extended logging

Example:

Call of the shell script from the terminal

```
sh setlogging_udmserver.sh 1
```

Note: The log file *wn-udm.log* is written to */var/opt/wn* and automatically rotated by means of log file rotation.

7.7.2 UDM client logging - CPOS

The logging for the UDM client for CPOS is activated and configured with the environment variable *WN_JAVAPOS_UDM_LOGFILE*.

Environment Variable	Meaning
WN_JAVAPOS_UDM_LOGFILE	Name und path of the log file of the UDM C-client. If “stdout” is used, then the log output will be directed to standard output (e.g. terminal console on display); Default is „“ (leer)

Windows

Note: On Windows, environment variables can be created or changed temporarily using the command *set* at the command line or in the application start script. Environment variables can also be created or changed permanently via the system tools such as *Control Panel* > *System* > *Advanced system settings* > *Advanced* > *Environment variables* or the *Windows Registry*.

Linux

Note: On Linux, environment variables can be created or modified easily in the application start script using the command *export*. Environment variables can also be stored permanently under */etc/profile.d* in a separate sh- (bash shell) or csh-file (C shell) with an own filename and which contains the command *export*. In this case a system restart is necessary that the changes can take effect.

Example:

Content of /etc/profile.d/activate-udm-cclient-logging.sh

```
# Setting the Environment Variable WN_JAVAPOS_UDM_LOGFILE
export WN_JAVAPOS_UDM_LOGFILE=/var/opt/wn/log/udm-cclient.log
```

Note: Please make sure that this script file does have the necessary user rights for execution. Also make sure that the logfile-path exists and also has the necessary write rights for the user. User rights can be changed with the command chmod.

7.7.3 UDM client logging – OPOS

OPOS uses the same UDM client as CPOS. See chapter *7.7.1 - UDM server logging* for more details.

7.7.4 UDM client logging – P4DN

Logging for the UDM client for P4DN is contained in the section *<log4net>* in the configuration file *P4DNUDMAdapter.config* under *<ProBase POS installation directory>\p4dn\bin*.

Parameter	Meaning
file value	Name und path of the log file; Default is „C:\ProgramData\javapos\wn\log\P4DNUDMAdapter.log“
appendToFile	Determines whether the log entries are appended to the existing file or whether the file is created again; Default is „true“
level value	Defines the log level; Default is „INFO“

For more information, see the P4DN UDM adapter documentation *P4DNUDMAdapter.txt* under *<ProBase POS installation directory>\p4dn\doc*.

7.8 JavaPOS Configurator logging

The JavaPOS Configurator, a small Java program, generates information about the configuration generation at each call and forwards it to *stdout*. ProBase POS provides various scripts to access the JavaPOS Configurator and to write the output of the JavaPOS Configurator into defined log files.

Windows

The output of the JavaPOS Configurator, called by the batch script *config_javapos_startup.bat*, is still directed to stdout. Only the vbs script *config_javapos_startup.vbs* directs the output of the JavaPOS Configurator to the defined log-file.

The output for the generated JavaPOS configuration is then written into the log file *config_javapos_startup.bat.log* under *C:\ProgramData\javapos\wn\log*.

Linux

The output for the generated JavaPOS configuration is written into the log file *config_javapos_startup.sh.log* under */var/log/wn/javapos* by the bash shell script *config_javapos_startup.sh*.

At the end of each ProBase POS 2 installation, the JavaPOS Configurator can be executed as a post-install action to create an initial JavaPOS configuration. This option is activated by default. The log files generated during installation differ from the normal log files of the JavaPOS Configurator in the name. As a result, these log files are not overwritten by the normal calls of the JavaPOS Configurator.

Windows

The output from the JavaPOS Configurator during the installation is written into the log file *config_javapos_startup.bat.log* under *C:\ProgramData\javapos\wn\log*.

Linux

The output from the JavaPOS Configurator during the installation is written into the log file *config_javapos_startup.sh.log* under *var/log/wn/javapos*.

8 Programming examples

ProBase POS also provides programming examples for the interfaces JavaPOS, OPOS and CPOS. These are only intended as an aid to the application developer and do not claim to be complete and correct.

8.1 JPOS

Windows

The JavaPOS programming examples are located in the directory *<ProBase POS installation directory>\doc\examples*.

Linux

The JavaPOS programming examples are located in the directory *<ProBase POS installation directory>/doc/examples*.

8.2 OPOS

The OPOS programming examples are located in the directory *<ProBase POS installation directory>\opos\common\Samples*.

8.3 CPOS

Windows

The CPOS programming examples are located in the directory *<ProBase POS installation directory>\cpos-udm\samples*.

Linux

The CPOS programming examples are located in the directory *<ProBase POS installation directory>/cpos-udm/samples*.

8.4 P4DN

The POS for .NET programming examples are located in the directory *<ProBase POS installation directory>\p4dn\doc\examples*.

9 Tools

9.1 JavaPOS Tool Center

The JavaPOS Tool Center is a central hub for all the JavaPOS configuration and test tools provided with ProBase POS. Additional submenus or the corresponding tools can be started via the respective buttons. The JavaPOS configuration loaded with the JavaPOS Tool Center startup is basis for most of these tools.

Windows

The JavaPOS Tool Center is started either from the Windows Start menu at *Start Menu > All Programs > Wincor Nixdorf ProBase (xx JVM) > Wincor Nixdorf ProBase Tool Center* or directly with the batch file *start_probaseToolCenter.bat* from the directory *<ProBase POS installation directory>\bin*.

Linux

The JavaPOS Tool Center is started directly with the Shell-script *start_probaseToolCenter.sh* from the directory *<ProBase POS installation directory>/bin*.

The JavaPOS Tool Center is then presented as follows.

Figure 4: JavaPOS Tool Center

The most important functions/tools would be:

- Trace Configurator
- JCL Editor
- SwingSamples
- Toggle JavaPOS Configuration

9.1.1 Trace Configurator

The Trace Configurator makes it possible to modify the JavaPOS Logging configuration file using a graphical user interface.

Note: By changing the JavaPOS Logging concept, the Trace Configurator cannot be used in the current ProBase POS version. The graphical user interface provided by the Trace Configurator must be adapted to the new WN-logger configuration.

9.1.2 JCL Editor

The JCL editor allows to modify JavaPOS XML configuration files (original, as well as the result from the JavaPOS Configurator).

With the use of the JavaPOS Configurator and the application-specific configuration file *javapos.config.properties*, this editor is no longer necessary and should not be used any further. Changes to the original JavaPOS XML configuration files would be lost in a SW update and changes to the JavaPOS Configurator result would be lost after a restart from the JavaPOS Configurator.

Note: In Linux, the JavaPOS Configurator is called at every system start. Under Windows, the JavaPOS Configurator must be started manually after each system configuration change. For more information, see Chapter *6.1 - JavaPOS Configuration*.

9.1.3 Toggle JavaPOS configuration

The *Toggle JavaPOS configuration ()* button toggles the currently selected JavaPOS configuration between the *all device configuration* and the *target configuration*. The changeover also causes the configuration to be loaded, which affects the tools that are started by the JavaPOS ToolCenter. The JavaPOS configuration used by the POS application is not affected.

9.1.4 SwingSamples

The tool JavaPOS SwingSamples is a basic test tool to test and visualize the basic communication as well as class-specific basic methods with the peripheral devices via JavaPOS. The most important UnifiedPOS properties, methods and events are used here.

Note: This tool does not claim to be a complete implementation of the UnifiedPOS specification.

9.2 OPOS UDM Configuration Updater

With the installation profile *OPOS Installation*, a planned task for the OPOS UDM Configuration Updater is set up by the product installer. For this, the Windows tool Task Scheduler will be used. The name of the task is *OPOSUDMConfigurationUpdater* and the task is called at every system start and also every user login. The OPOS configuration, which is stored in the Windows Registry, is checked and, if necessary, updated.

With ProBase POS, this OPOS configuration in the Windows Registry is based on the JavaPOS XML files provided with ProBase POS. The OPOS Configuration Updater

- does not overwrite existing values,
- adds new values.
- checks for OPOS UDM OpenNames (all with suffix _UDM), whether corresponding JavaPOS OpenNames exists. If not, these entries will be deleted.

The OPOS Configuration Updater can be found in the directory *<ProBase POS installation directory>\oposudm\bin* as *OPOSUDMConfigUpdater.exe* and requires administrative rights to write the entries in the Windows Registry permanently.

Note: The OPOS Configuration Updater is executed automatically after the manual start of the JavaPOS Configurator. However, we recommend to restart the POS system every time the system configuration has changed so that the changed settings and affected libraries are reloaded.

9.3 Test tools

9.3.1 SwingSamples

The SwingSamples are a basic test tool to test the selected JavaPOS configuration for the respective peripheral devices. For this purpose, class-specific basic methods are offered for the peripheral devices. Please note that the scope of the SwingSamples methods is not sufficient for the scope of methods defined by UnifiedPOS.

After the call the SwingSamples present themselves as follows.

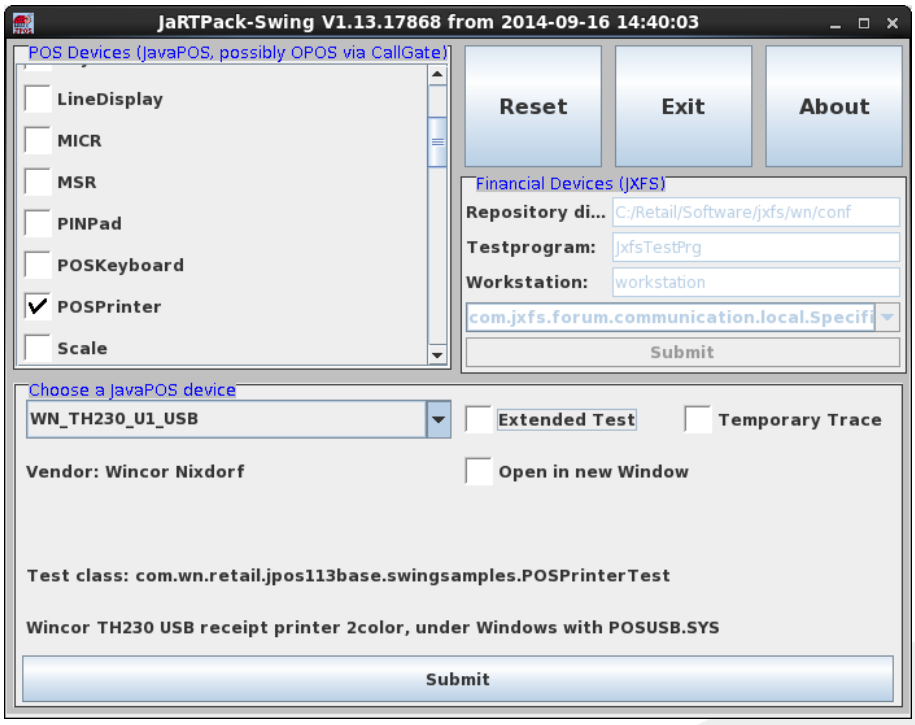


Figure 5: JavaPOS SwingSamples

The SwingSamples are called up either via the JavaPOS ToolCenter (see chapter *9.1 - JavaPOS Tool Center*) or directly via the respective call scripts.

Windows

<ProBase POS installation directory>\bin\start_jartpack.bat

Linux

<ProBase POS installation directory >/bin/start_jartpack.sh

After selecting a device category, a JavaPOS device (OpenName) and possibly necessary communication port parameters, either a simple test or an extended test can be called. In order to start the extended test, the checkbox for [Extended Test] has to be selected. The test is started by clicking [Submit].

The Simple Test mode

Figure 6: SwingSamples - Simple Test

By clicking on the button [TEST: "WN_TH230_U1_USB"] (or similar with other JavaPOS devices) the simple test can be executed. The result is displayed in the box below.

The Extended Test mode:

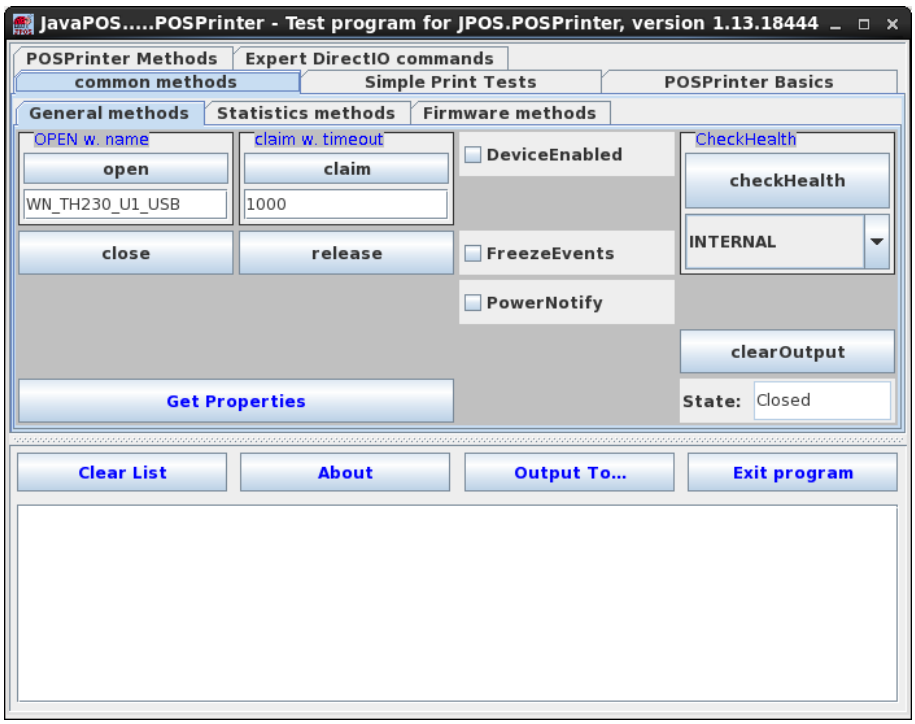


Figure 7: SwingSamples - Extended Test

The extended test mode is somewhat more complex and thus offers better test and analysis possibilities.

Each test usually begins by clicking [open], clicking [claim], and checking the [DeviceEnabled] checkbox. In a few cases, the device classes do not support the claim() command, so that must be omitted. Afterwards the other offered methods can be used for testing.

9.3.2 OPOS Config Tool

The OPOS Config Tool also provides a test to check the basic configuration in a simple communication test.

To start, the OPOS Config Tool is to be found in the Windows Start menu with *Start menu > All Programs > Wincor Nixdorf ProBase (xx JVM) > OPOS (UDM) > OPOS Common > OPOS Configuration program* or directly via the application *HWD55ConfUDM.exe* to be found within the directory *<ProBase POS installation directory>\opos\common\bin*.

After the start, the OPOS Config Tool presents itself as follows.

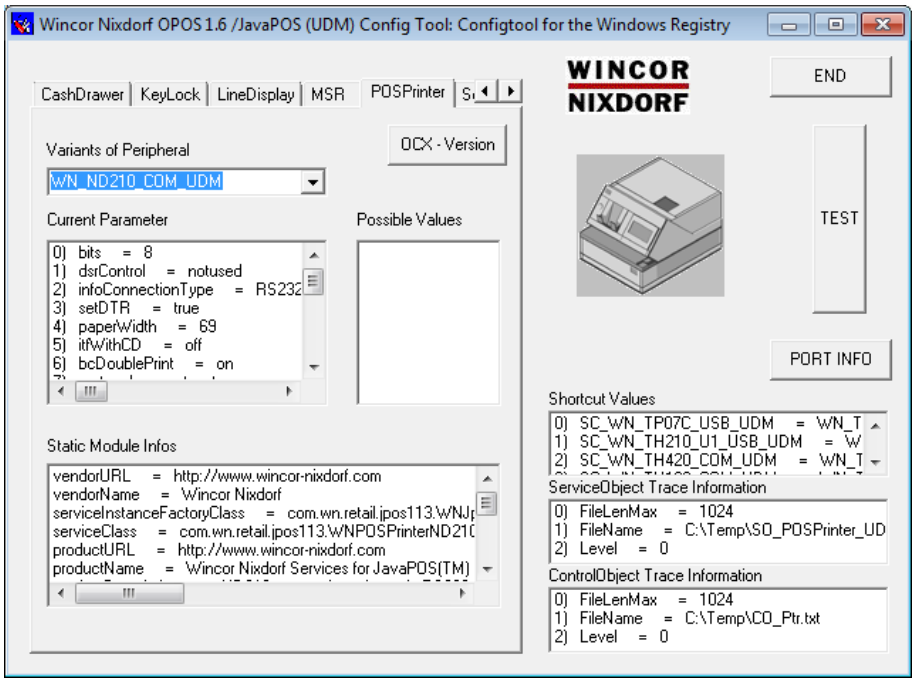


Figure 8: OPOS Config Tool

Clicking [TEST] starts a new dialog window.

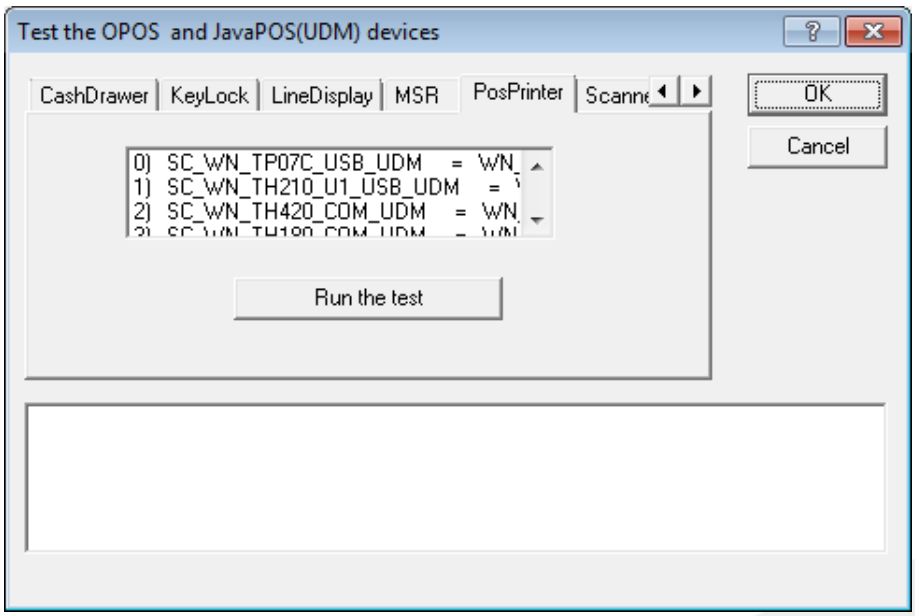


Figure 9: OPOS Config Tool - Simple Test

After selecting a device category and an OpenName from the list, a simple communication test for the selected device can be started using [Run the test]. The test result is displayed in the box below.

9.3.3 OPOS Sample Programs

The following class-specific OPOS Sample Programs are delivered and offered with ProBase POS.

- TestSample FiscalPrinter
- TestSample HardTotals
- TestSample Keyboard Keylock MSR
- TestSample LineDisplay
- TestSample Printer CashDrawer MICR LineDisplay
- TestSample Scale
- TestSample ScaleTransaction
- TestSample Scanner
- TestSample UPS

You can start the OPOS test samples (using the example of the test sample for fiscal printers) either via the Windows Start menu with *Start menu > All Programs > Wincor Nixdorf ProBase (xx JVM) > OPOS (UDM) > OPOS Common > TestSample FiscalPrinter* or directly via the application *FPrinter.exe* under *<ProBase POS installation directory>\opos\common\Samples\FiscalPrinter*.

The OPOS test sample is then presented as follows.

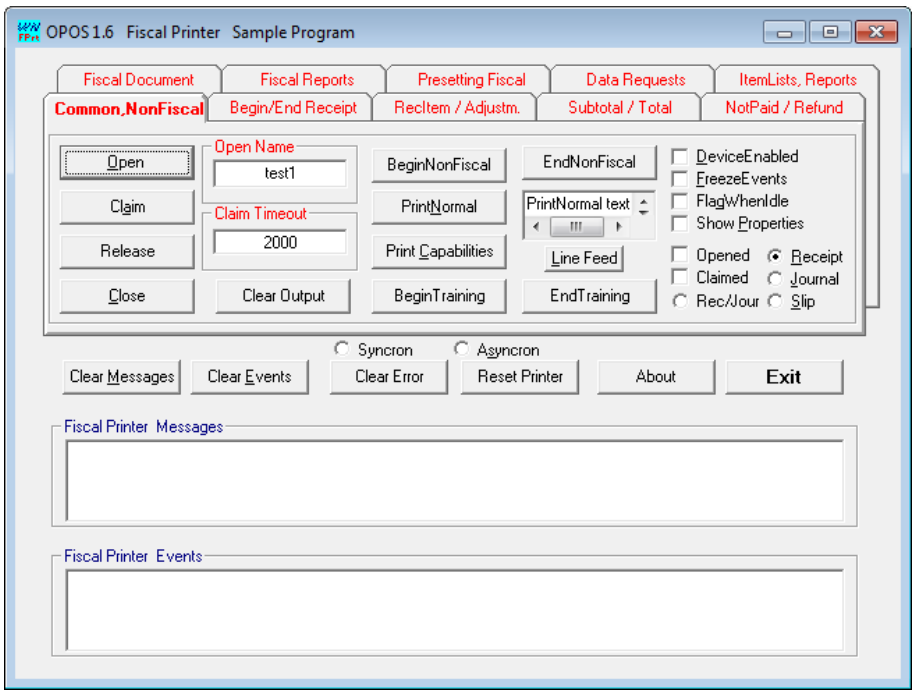


Figure 10: OPOS TestSample - Fiscal Printer

Each test usually begins by choosing an OpenName, and then clicking [Open], clicking [Claim], and checking the [DeviceEnabled] checkbox. In a few cases, the device classes do not support the claim() command, so that must be omitted. Afterwards the other offered methods can be used for testing.

For the other test tools, it is analogous.

9.3.4 P4DN test tool

ProBase POS does not provide a stand-alone test tool for the POS for .NET API. Here the test tool from the POS for .NET Framework from Microsoft is to use.

10 Specifications

10.1 Supported peripherals

CashChanger

- Wincor Nixdorf coin changer MCS
- Wincor Nixdorf coin changer MUX
- Wincor Nixdorf sub coin acceptor MCS
- Wincor Nixdorf sub coin dispenser TLQ
- Wincor Nixdorf note acceptor JCM iPro Notes Acceptor
- Wincor Nixdorf note acceptor JCM Notes Acceptor UBA10
- Wincor Nixdorf note dispenser JCM Bill Dispenser F53
- Wincor Nixdorf cash changer BCR 200

CashDrawer

- All Wincor Nixdorf cash drawer connected to the cash drawer port of BEETLE POS systems, connected via WN POS printer or via WN fiscal printer
- All Wincor Nixdorf cash drawer connected to Multi I/O Hub **NEW**

FiscalPrinter

- Wincor Nixdorf fiscal printer MF-EJ210 Greece
- Wincor Nixdorf fiscal printer MF-EJ210 Turkey
- Wincor Nixdorf fiscal printer MF-EJ320 Greece
- Wincor Nixdorf fiscal printer MF-EJ320 Turkey
- Wincor Nixdorf fiscal printer MF-ND77 Romania
- Wincor Nixdorf fiscal printer MF-TH210 Hungary with AEE (FL 2016 prepared)
- Wincor Nixdorf fiscal printer MF-TH230+ Hungary with AEE (FL 2016 prepared)
- Wincor Nixdorf fiscal printer MF-TH230+ Romania (incl. CommModule) (FL 2016 prepared)
- Wincor Nixdorf fiscal printer MF-TH230+ Italy with FFC
- Wincor Nixdorf fiscal printer MF-TH320 Hungary with AEE (FL 2016 prepared)
- Wincor Nixdorf fiscal printer MF-THF Romania
- Wincor Nixdorf fiscal printer TH230-MF Bulgaria (incl. TaxTerminal)
- Wincor Nixdorf fiscal printer TH230-MF Italy

HardTotal

- Non-volatile memory mapped on hard drive or flash memory

Keyboard

- All Wincor Nixdorf keyboards with related sub-devices keylock and MSR
- Wincor Nixdorf keyboard connected to WN display BA82/BA83

Keylock

- Wincor Nixdorf Waiter Keylock connected to WN POS keyboards
- Wincor Nixdorf Keylock connected to WN POS keyboards
- OLITRONIC Electronic key-lock-RS232LP connected to BEETLE /iPOS
- OLITRONIC Electronic keylock ICS-USB.B connected to BEETLE /Fusion
- Wincor Nixdorf Electronic Key Controller connected to WN BA8x, WN BA9x or WN Special Electronic
- Electronic key reader SKH301-001 from Sysking Technology Ltd. SKH300 series devices connected to BEETLE /iPOS+

LineDisplay

- Wincor Nixdorf two-line line display BA63
- Wincor Nixdorf four-line line display BA66
- Wincor Nixdorf line display BA63 / BA66 connected via ND77, ND210, TH230, TH230+ POS printer
- Wincor Nixdorf line display BA63 / BA66 connected via MF-ND77, MF-EJ210, MF-EJ320, MF-TH230+ (AEE and FFC), TH230-MF, MF-THF, MF-TH210, MF-TH320 fiscal printer
- Wincor Nixdorf line display connected to BEETLE /EXPRESS
- Wincor Nixdorf five-line line display VGA/4 over VGA/4 display server
- Wincor Nixdorf four-line line display on BA69 connected via Embedded Scale Controller
- Wincor Nixdorf line display connected to BEETLE /iPOS+
- Wincor Nixdorf virtual line display

MICR

- Wincor Nixdorf two-station thermal printer TH320
- Wincor Nixdorf two-station thermal printer TH420

MotionSensor

- Wincor Nixdorf Motion Sensor at COM port (RS232)

MSR

- Wincor Nixdorf MSR connected via WN POS keyboards
- MSR Nidec Sankyo Corporation ICM330
- MSR 213U connected to BEETLE /iPOS+
- MSR 7816 Swipe and Park
- Wincor Nixdorf MSR connected to WN BA7x, BA8x, BA9x, BEETLE /FUSION, BEETLE /iPOS, SNIkey
- MSR Hitachi-Omron V2X Series

POSPower

- External Uninterruptible Power Supply MPS1086

POSPrinter

- Wincor Nixdorf three-station matrix printer ND77
- Wincor Nixdorf single-station matrix printer ND210

- Wincor Nixdorf two-station inkjet Pharmacy-Printer
- Wincor Nixdorf single-station thermal printer TH180
- Wincor Nixdorf single-station thermal printer TH210
- Wincor Nixdorf single-station thermal printer TH230
- Wincor Nixdorf single-station thermal printer TH230+
- Wincor Nixdorf single-station thermal printer TH250
- Wincor Nixdorf two-station thermal printer TH320
- Wincor Nixdorf two-station thermal printer TH420
- Wincor Nixdorf single-station thermal printer TP07
- Wincor Nixdorf single-station thermal printer TP07c
- Wincor Nixdorf single-station stamp printer PP01
- Wincor Nixdorf single-station label printer PP02
- Zebra single station label printer GX430t
- Wincor Nixdorf single-station kiosk printer VKP80III

Scale

- All Wincor Nixdorf scales named WExx using CHECKOUT DIALOG 06
- Datalogic scale Magellan 8202
- Datalogic scale Magellan 8502
- All Wincor Nixdorf scale named connected to Embedded Scale Controller using Scale Transaction Module
- Shekel security scale
- Mettler security scales
- Mettler Toledo scale BC15
- Mettler Ariva scale using CHECKOUT DIALOG 06 via VCO-disp software
- Mettler Viva scale using CHECKOUT DIALOG 06 via VCO-disp software
- Mettler Spider Software scale

Scanner

- All Wincor Nixdorf scanner named ELxx with WN communication protocol NIXDORF RS232C Mode A or B
- All Wincor Nixdorf scanner named ELxx with default settings mode according to IBM´s 'USB OEM POS Device Interface Specification' labelled as 'IBM Hand-held USB' or 'IBM Table Top USB'
- Motorola Symbol scanner (formerly Symbol Technologies, Inc.) supporting the WN communication protocol NIXDORF RS232C Mode A
- Motorola Symbol scanner (formerly Symbol Technology, Inc.) that comply with IBM´s 'USB OEM POS Device Interface Specification", version 1.29
- Motorola Symbol SE3223 scan engine / barcode reader
- Datalogic scanner supporting the WN communication protocol NIXDORF RS232C Mode A
- Datalogic scanner Magellan 8400 (RS232 Mode)
- Datalogic scanner Gryphon D120 (RS232 Mode)
- Datalogic scanner Gryphon D130 (RS232 Mode)
- Datalogic scanner Gryphon M100 (RS232 Mode)
- Wincor Nixdorf Scanner EL 71 (Zebra scanner DS4308)

- Intermec scanner ED40

ToneIndicator

- Wincor Nixdorf internal BEETLE POS system loudspeaker
- Wincor Nixdorf OPT built in Special Electronic ACO USB
- Wincor Nixdorf C1030 built in Special Electronic ACO USB
- Wincor Nixdorf ACO Kiosk Box SEL
- Wincor Nixdorf Special Electronic CDL for Modular Postal Systems
- Wincor Nixdorf Special Electronic ACO USB
- Wincor Nixdorf SCO compact built in Special Electronic ACO USB
- Wincor Nixdorf LED-Status-Box Pole light R/G/B at COM (RS232)
- Wincor Nixdorf Mini LED Pole light R/G at COM (RS232)

10.2 Supported operating systems

ProBase POS 2 can be used on and is released for the following operating systems.

Windows:

- Windows XP
- POSReady 2009
- Windows 7
- POSReady 7
- Windows 8.1 Pro
- Windows 8.1 Industry Pro
- Windows 10 Pro (since PBP 2.2)
- Windows 10 IoT (since PBP 2.2)

Linux:

- WNLPOS 2
- WNLPOS 3
- WNLPOS 4 (since PBP 2.1)

10.3 Software requirements

10.3.1 Minimum prerequisites

The following software must already be installed on the target machine in order to successfully run the ProBase POS installer:

- Java Virtual Machine (JavaVM) (at least version 6)

Windows

Note: A JavaVM is also required for the installation profiles *OPOS Installation*, *POS for .NET Installation* and *CPOS Installation* of the product installer. Especially for this the UDM Default Runtime, a Diebold Nixdorf distribution package of a Java Runtime Engine (JRE) based on OpenJDK 6 is proposed.

The UDM Default Runtime installation package does not install a publicly available JRE and takes some other security measures (see separate documentation[15]). The user profile used for the application must be added to the user group *wndev*, so that the UDM DefaultRuntime can be executed.

The installation package of the UDM Default Runtime Engine[16][17] (currently available only as 32-bit version) is provided with the pre-installations or as a separate download.

Linux

Note: All Diebold Nixdorf Linux pre-installations provide sufficient JavaVM packages for ProBase POS. Up to and including WNLPOS 3 these are only 32-bit (i386) JavaVM versions. As of WNLPOS 4, the 64-bit (x64_86) JavaVM versions are also provided with the pre-installation.

10.3.2 Additional prerequisites

Windows

In case the *POS for .NET Installation* profile or corresponding POS for .NET components have been selected (see chapter*4.2.1* - Interactive installation), ensure that the following software has already been installed on the target machine:

- Microsoft .Net Framework (at least version 2.0 up to 4.0)
- Microsoft POS for .Net (version 1.12)

Additional software packages (e.g., drivers such as WNPOSUSB) may be necessary to support corresponding devices. This information can be found in the respective user manuals of these devices.

10.4 Components included

ProBase POS 2.4 consists of the following.

CIM adapter components

- wn-cim 1.7.6

Common components

- dn-common 0.3.0
- wn-common 1.11.0
- wn-common-doc(ument) 1.2.51
- wn-common-jnaio 1.2.1
- wn-common-rs232 1.2.3
- wn-common-rs232-native 1.5.12

[15] UDM Security Considerations; Diebold Nixdorf; Denis Kuniss; version 1.0

[16] > Intranet > Portfolio > Our Portfolio > Software Solutions > Retail Software Solutions > System Software & Operating Systems > System oriented Software > UDM

[17] ProBase POS 2.1 (32-bit) benötigt die UDM Default Runtime Engine Version 1.1-3 oder neuer

- wn-common-usb 1.4.0
- wn-common-usb-native 1.11.1

JavaPOS components

- wn-javapos-beeper-native 1.5.9
- wn-javapos-cashchangers 1.24.5
- wn-javapos-cashdrawer 0.1.5
- wn-javapos-common 1.9.0
- wn-javapos-config 2.4.0
- wn-javapos-controls 2.0.1
- wn-javapos-diagnostics 1.5.1
- wn-javapos-f53 1.19.1
- wn-javapos-fiscalprinter 1.34.1
- wn-javapos-iscan 1.29.0
- wn-javapos-jcl 1.4.0
- wn-javapos-jcl-editor 1.4.0
- wn-javapos-kbdclaimer-native 1.5.8
- wn-javapos-keylock 1.7.1
- wn-javapos-kkmusb 1.19.0
- wn-javapos-linedisplay 1.9.3
- wn-javapos-mps1086 1.10.0
- wn-javapos-msr 1.7.1
- wn-javapos-portalscanner 1.21.1
- wn-javapos-portio-native 1.6.6
- wn-javapos-posprinter 1.14.5
- wn-javapos-ps7000 1.4.1
- wn-javapos-retail 1.28.3
- wn-javapos-samples 1.47.1
- wn-javapos-scale 1.4.1
- wn-javapos-scanner 1.9.1
- wn-javapos-selaco 1.18.0
- wn-javapos-th250 2.0.1
- wn-javapos-thxxx 1.41.0
- wn-javapos-tp07 1.16.0
- wn-javapos-trace 1.5.1
- wn-javapos-tsop 1.6.2

OPOS components

- wn-opos-1.6C00-common package 1.6.9.1

UDM adapter components

- wn-udm-cpos 1.6.0
- wn-udm-javapos 3.5.6
- wn-udm-opos 2.5.0
- wn-udm-p4dn (pos for .net) 3.3.1

Tools and other components

- wn-fwu-api 5.6.6
- wn-logger 3.9.1
- wn-tools 1.1.0

10.5 Currently available add-ons

Windows

Peripherals	Installer version	Support of 32-bit JavaVM	Support of 64-bit JavaVM
Scale Transaction Module	1.4.8	x	x
Asia add-on package (supports following devices: TH200, ND220, ND210(Asia), TH230(Asia), BA64, BA63GU, CashDrawer SMBUS for iPOS+ Adv. and iPOS+ Braswell, BA9x NFC/RFID and BA9x BCR	1.13.206-1	x	x

Linux

Peripherals	Installer version	Support of 32-bit JavaVM	Support of 64-bit JavaVM
Line display BA64	1.13.1-1.17.8	x	x
Scale Transaction Module	1.4.1	x	x
POS Printer TH200	1.13.1-1.11	x	x
BA9x NFC/RFID module	1.13.1-2.3	x	x
POS Printer ND220	1.13.1-1.9	x	x
Cash drawer port at AiO (iPOS+ (Adv.))	1.13.1-5	x	x

10.6 Restrictions in 2.4

The following restrictions on ProBase POS 2.4 are already known and will be fixed with the next ProBase POS version:

- The support for the POS for .NET Framework 1.14 from Microsoft is not fully tested
- Support for the Scale Transaction Module is limited to OPOS, CPOS and JavaPOS
- On Windows, OPOS UDM is not installed correctly during a change installation (other profile) (workaround: uninstalling PBP 2 and installing with the new profile).
- The tool for updating the OPOS UDM configuration within the registry does not add any missing sub-entries (workaround: remove the entire registry key of the device and restart the system)
- Currently, no GUI is available for the new trace/log mechanism or to customize the JavaPOS configurator
- Some expert tests from the JavaPOS ToolCenter can terminate after installing additional JavaPOS modules
- Temporary tracing in the SwingSamples is still not functional after the trace/log mechanism has been changed
- The trace configurator within the JavaPOS ToolCenter has only a minimal function
- 32-bit Java Runtime Engines on 64-bit Linux systems are not detected correctly
- The UDM server is not automatically restarted under CPOS (Linux) as under OPOS, P4DN and CPOS (Windows)
- The JavaVM is not determined correctly, if there are residues of an installation of the Oracle JRE
- The cash drawer port at the Multi I/O Hub is not supported properly on Linux

11 Appendix

11.1 End-user license agreement

End-User License Agreement

ProBase POS

IMPORTANT - READ CAREFULLY
This End User License Agreement ("EULA") is a legal agreement between the licensee, either an individual or a single entity (“you”) and Wincor Nixdorf International GmbH (“Diebold Nixdorf”) for the Diebold Nixdorf Software that accompanies this EULA as well as possibly associated media, related documentation and Internet-based services ("Software"). An amendment or addendum to this EULA may accompany the Software. YOU AGREE TO BE BOUND BY THE TERMS OF THIS EULA BY DOWNLOADING, INSTALLING, COPYING, OR USING THE SOFTWARE. DO NOT INSTALL, COPY, OR USE THE SOFTWARE, IF YOU DO NOT AGREE.

1. GRANT OF LICENSE.
Intellectual property rights in the Software are owned by Diebold Nixdorf and made available to you under a restricted license as set out in this EULA. Any rights not expressly granted are reserved by Diebold Nixdorf.

Diebold Nixdorf grants you the following rights provided that you comply with all terms and conditions of this EULA:

- Diebold Nixdorf grants you a non-exclusive, non-transferable right to permanent use of the Software only in conjunction with hardware products from Diebold Nixdorf. Use of the Software in conjunction with non-Diebold Nixdorf hardware products is not permitted hereunder.
- You may copy the Software for data protection, archiving and backup purposes. However, only the strictly necessary amount of backup copies may ever be stored.
- You may enhance the Software with third party software via the defined interfaces.
- In case you received the Software together with hardware of Diebold Nixdorf, you may only transfer the right of use granted to you to a third party only in full and together with ownership of the hardware supplied with it and/or ownership of the original data carrier supplied by Diebold Nixdorf and only in full acknowledge of this EULA by the third party.

2. OTHER RIGHTS AND LIMITATIONS.
You hereby expressly guarantee that you will not copy, modify, rent, sale, distribute or transfer any part of the Software except within the scope of the rights of use granted within this EULA.

3. RESERVATION OF RIGHTS AND OWNERSHIP.
Diebold Nixdorf or its suppliers own the title, copyright, and other intellectual property rights in the Software. The Software is protected by copyright and other intellectual property laws and treaties. Diebold Nixdorf reserves all rights not expressly granted to you in this EULA. This EULA does not grant you any rights to trademarks of Diebold Nixdorf.

4. USER RESTRICTIONS.
You may not decompile, disassemble or reverse engineer the Software, except and only to the extent that such activity is expressly permitted by applicable law notwithstanding this limitation.
You may not rent, lease, lend or provide commercial hosting services with the Software.

5. DISCLAIMER.
You may reach third-party sites through the usage of the Software or associated media or services. Diebold Nixdorf does not control third-party sites and Diebold Nixdorf is not responsible for the contents of any third-party sites, any links contained in third-party sites, or any changes or updates to third-party sites. Diebold Nixdorf is providing these links and access to third-party sites and services to you only as a convenience, and the inclusion of any link or access does not imply an endorsement by Diebold Nixdorf of the third-party site or service.

6. ADDITIONAL SOFTWARE/SERVICES.
This EULA applies to updates, enhancements, add-on components, or Internet-based services components, of the Software that Diebold Nixdorf may provide to you or make available to you after the date you obtain your initial copy of the Software, unless they are accompanied by separate terms. For the avoidance of this EULA does not provide any binding obligation for Diebold Nixdorf for the delivery of any future update, upgrade or new releases. Diebold Nixdorf reserves the right to discontinue any Internet-based services provided to you or made available to you through the use of the Software.

7. TERMINATION.
This EULA is effective from the date on which the Software is downloaded by you until terminated. Diebold Nixdorf may terminate the license granted to you under this EULA by written notice at any time without stating a reason. Diebold Nixdorf shall have the right to immediately terminate the license of use under this EULA for cause in case you fail to comply with any provision of this Agreement and you do not cure the relevant breach within a reasonable time frame after written notification of Diebold Nixdorf. Upon termination, you must immediately destroy all copies of the Software or return all copies of the Software to Diebold Nixdorf.

8. EXCLUSION OF WARRANTIES.
To the maximum extent permitted by applicable law, Diebold Nixdorf and its suppliers or its resellers provide the Software and support services (if any) AS IS AND WITH ALL FAULTS, and hereby disclaim all other warranties and conditions, whether express, implied or statutory, including, but not limited to, any (if any) implied warranties, duties or conditions of merchantability, of fitness for a particular purpose, of reliability or availability, of accuracy or completeness of responses, of results, of workmanlike effort, of lack of viruses, and of lack of negligence, all with regard to the Software, and the provision of or failure to provide support or other services, information, software, and related content through the Software or otherwise arising out of the use of the Software. ALSO, THERE IS NO WARRANTY OR CONDITION OF TITLE, QUIET ENJOYMENT, QUIET POSSESSION, CONCORDANCE TO DESCRIPTION OR NON-INFRINGEMENT WITH REGARD TO THE SOFTWARE. Modifications and amendments to the Software may occur without notice.

9. EXCLUSION OF INCIDENTAL, CONSEQUENTIAL AND CERTAIN OTHER DAMAGES.
IN NO EVENT SHALL DIEBOLD NIXDORF OR ITS SUPPLIERS OR RESELLER BE LIABLE FOR ANY INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, BUT NOT LIMITED TO, DAMAGES FOR LOSS OF PROFITS OR CONFIDENTIAL OR OTHER INFORMATION, FOR BUSINESS INTERRUPTION, FOR PERSONAL INJURY, FOR LOSS OF PRIVACY, FOR FAILURE TO MEET ANY DUTY INCLUDING OF GOOD FAITH OR OF REASONABLE CARE, FOR NEGLIGENCE, AND FOR ANY OTHER PECUNIARY OR

OTHER LOSS WHATSOEVER) ARISING OUT OF OR IN ANY WAY RELATED TO THE USE OF OR INABILITY TO USE THE SOFTWARE, THE PROVISION OF OR FAILURE TO PROVIDE SUPPORT OR OTHER SERVICES, INFORMATON, SOFTWARE, AND RELATED CONTENT THROUGH THE SOFTWARE OR OTHERWISE ARISING OUT OF THE USE OF THE SOFTWARE, OR OTHERWISE UNDER OR IN CONNECTION WITH ANY PROVISION OF THIS EULA, EVEN IN THE EVENT OF THE FAULT, TORT (INCLUDING NEGLIGENCE), MISREPRESENTATION, STRICT LIABILITY, BREACH OF CONTRACT OR BREACH OF WARRANTY OF DIEBOLD NIXDORF OR ANY SUPPLIER, AND EVEN IF DIEBOLD NIXDORF OR ANY SUPPLIER HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

10. LIMITATION OF LIABILITY AND REMEDIES.
Notwithstanding any damages that you might incur for any reason whatsoever (including, without limitation, all damages referenced herein and all direct or general damages in contract or anything else), the entire liability of Diebold Nixdorf and any of its suppliers or resellers under any provision of this EULA and your exclusive remedy hereunder shall be limited to the greater of the actual damages you incur in reasonable reliance on the Software but in maximum up to the amount actually paid by you for the Software. The foregoing limitations and exclusions shall apply to the maximum extent permitted by applicable law, even if any remedy fails its essential purpose.

11. APPLICABLE LAW.
This EULA is governed by the laws of the Federal Republic of Germany. The application of the United Nations Convention of Contracts for the International Sale of Goods is expressly excluded. Place of Jurisdiction is Düsseldorf, Germany.

12. ENTIRE AGREEMENT; SEVERABILITY.
This EULA (including any addendum or amendment to this EULA which accompanies the Software) is relating to the Software and the support services (if any) the entire agreement between you and Diebold Nixdorf. It supersedes all prior or contemporaneous oral or written communications, proposals and representations with respect to the Software or any other subject matter covered by this EULA. To the extent the terms of any Diebold Nixdorf policies or programs for support services conflict with the terms of this EULA, the terms of this EULA shall precede. If any provision of this EULA is held to be void, invalid, unenforceable or illegal, the other provisions shall continue in full force and effect.

11.2 Changes to version 2.4

Apart from the listed changes, further development-related changes may be included in ProBase POS 2.4.

11.2.1 General

PG4 Java VM 1.6 required

Issue: Minimum requirement for Java VM increased from version 1.5 to version 1.6.

11.2.2 Add-ons

PA54 Multi IO Hub cash drawer port

Issue: Added support for cash drawer port of Multi-IO-Hub with JposEntry DN_CD_HUB.

Affects: wn-javapos-cashdrawer

PA53 Linedisplay BA63/66 at Greek fiscal printer

Issue: Added support for BA63/66 line display connected to a Greek fiscal printer including proper code page mapping from Unicode to ELOT928 as required by the fiscal printer FW.

Affects: wn-javapos-linedisplay

PA52 Linedisplay configuration property clearDisplayOnDisable

Issue: Added configuration property "clearDisplayOnDisable" to enable automatic clearing of the display if the device is set to disable.

Affects: wn-javapos-linedisplay

PA51 Second pipe for Bizerba USB scales

Issue: Added second pipe support for Bizerba USB scale solutions (PTC 2185460)

Affects: wn-javapos-retail

PA50 Codepage 1252 for older THxxx POS printer

Issue: Added support for codepage 1252 older THxxx POS printer (MKS-2207159).

Affects: wn-javapos-thxxx

PA49 More error codes

Issue: Added new error codes for connection reject situations and PDH specific errors.

Affects: wn-udm-cpos; wn-udm-opos

11.2.3 Fixes

PF45 POS printer using usbprint.sys caused NullPointerException under Linux

Issue: POS printer command open caused NullPointerException under WNLPOS / Linux for all printer using usbprint.sys (e.g. VKP80III).

Affects: wn-common-jnaio

PF44 Update installation on WNLPOS (Linux) failed sometimes

Issue: The update installation on WNLPOS had some minor issues related to FIFO, UDM server, and config files (PTC 2027478).

Affects: wn-javapos-config; wn-javapos-samples; wn-javapos-retail

PF43 Unicode encoding at line displays connected to Greek fiscal printer was incorrect

Issue: The Unicode character encoding for line displays connected to a Greek MF-EJ210 or MF-EJ320 fiscal printer was not correct (PTC#2185611).

Affects: wn-javapos-fiscalprinter

PF42 Linedisplay command displayText caused IndexOutOfBound exception

Issue: The command displayText used with trailing NL or CR caused an IndexOutOfBound exception.

Affects: wn-javapos-linedisplay

PF41 Scale command readWeight failed

Issue: The scale command readWeight() failed with errorMessage "readWeight - handleCheckAndKorr returns. Try again... "

Affects: wn-javapos-retail

PF40 Desktop icon for ProBase POS Tool Center under Linux was missing

Issue: The Desktop link to ProBasePOS Tool Center under WNLPOS / Linux environments was missing (PTC 1938222).

Affects: wn-javapos-samples

11.2.4 Changes

PC76 BCR 200 now recognizes ‘EU’as CurrencyCode

Issue: The BCR 200 CoinRecycler now recognizes 'EU' as CurrencyCode for EUR also.

Affects: wn-javapos-cashchangers

PC75 Additional error state for the JavaPOS configurator

Issue: The JavaPOS configurator has an additional return error if the configuration was not written because of "Zugriff verweigert"/”Access denied”.

Affects: wn-javapos-config

PC74 Error handling improved for JavaPOS configurator

Issue: The error handling for the JavaPOS configurator is improved for cases where the manual configuration of the JavaVM within javahome.ini is wrong. An error will be raised (PTC 2196841).

Affects: wn-javapos-config; wn-udm-javapos

PC73 Several loggerBaseNames changed.

Issue: Several loggerBaseNames were changed for better readability.

Affects: wn-javapos-iscan; wn-javapos-mps1086; wn-javapos-portalscanner; wn-javapos-ps7000; wn-javapos-retail; wn-javapos-thxxx

PC72 Logging improved for several Device Services

Issue: Logging improved.

Affects: wn-javapos-keylock; wn-javapos-msr; wn-javapos-scale; wn-javapos-scanner

PC71 Configuration parameter names for linedisplays changed

Issue: The configuration parameter "FColor" and "BColor" changed to "foregroundColor" and "backgroundColor". The old parameter names are still applicable for backward compatibility.

Affects: wn-javapos-linedisplay

PC70 Zebra label printer support was removed

Issue: The support for the Zebra label printer (open names WN_ZEBRA_COM and WN_ZEBRA_USB) is removed (PTC#2186591).

Affects: wn-javapos-posprinter

PC69 The POS printer TH250 RS232 configuration changed

Issue: The TH250 RS232 POS printer require "RS232" as configuration value for the "infoConnectionType" configuration property.

Affects: wn-javapos-posprinter

PC68 Old JavaPOS example files were removed.

Issue: To regain coherence between PBP for Windows and Linux, the doc/examples/*SimpleTest.java files were removed (PTC 2027717).

Affects: wn-javapos-samples

PC67 Improved logging

Issue: The logger names were shortened for better readability.

Affects: wn-javapos-thxxx

PC66 TSOP events logging improved.

Issue: Avoiding SUE corresponding TSOP events are logged to the diagnostics file.

Affects: wn-javapos-tsop

PC65 Logger behavior and config file improved

Issue: The Rolling File Appender was re-added and within the logging config file, the categories were sorted and double entries removed (PTC 2175979).

Affects: wn-logger

PC64 UDM server logging improved

Issue: Serious JRE errors at UDM server level are now logged as fatal errors to get a trace of them.

Affects: wn-udm-javapos

PC63 Error code 1008 changed to 1012 for moPOS

Issue: In case of PDH is not claimed, now the error code -1012 will be returned instead of error code -1008 as error code -1008 is already used in other connection reject situations.

Affects: wn-udm-javapos

PC62 Installer check box text at the post run dialog enhanced

Issue: The check box note text at the installer post run dialog is now well-defined.

Affects: wn-udm-javapos

PC61 UDM Server automatic start is enabled for P4DN devices

Issue: The automatic UMD server start is now also enabled for P4DN devices at the open call (PTC#1029820).

Affects: wn-udm-p4dn

PC60 P4DN UDM client log file output moved

Issue: The P4DN UDM client logging moved to %ProgramData%/javapos/wn/log.

Affects: wn-udm-p4dn

PC59 Performance of the UDM socket communication was optimized.

Issue: The performance on the UDM socket communication was optimized (PTC 2157377).

Affects: wn-udm-p4dn

11.3 Changes to version 2.3

Apart from the listed changes, further development-related changes may be included in ProBase POS 2.3.

11.3.1 General

PG3 JavaPOS Controls 1.14

Issue: Updated JavaPOS controls to version 1.14

Note: JavaPOS support now JavaPOS device services after UnifiedPOS 1.14.

DIEBOLD NIXDORF

11.3.2 Add-ons

PA48 POS printer WN TH250

Issue: Added support for POS printer WN TH250 by integrating previous JavaPOS add-on for TH250

Affects: wn-common-jnaio; wn-javapos-th250

PA47 Kiosk printer WN VKP80III

Issue: Added support for kiosk printer WN VKP80III. Current open name is WNVKP80III_USB.

Affects: wn-javapos-posprinter

PA46 USB handheld scanner ED40

Issue: Added support of USB HID handheld scanner ED40.

Affects: wn-common-usb-native

PA45 Configuration option to define JavaVM version to be used

Issue: Added configuration option to define a JavaVM version to be used by JavaPOS by using dedicated config files in Linux and Windows. (PTC 2146133)

Affects: wn-javapos-config

PA44 JavaPOS Configurator option to reference newly defined open names

Issue: Added configuration option for javapos configuration properties files allowing to reference newly defined open names at "jpos.entry.<open-name>" definitions. (PTC 2185484)

Affects: wn-javapos-config

PA43 JavaPOS Configurator option to allow JAR files ordering at Java classpath

Issue: Added configuration option for javapos configuration properties files allowing ordering of JAR files in the generated Java class path by adding an entry "jpos.order.jar=<list-of-jars>" (PTC 1994435)

Affects: wn-javapos-config

PA42 Configuration option “printingAEELogUnresponsivenessTime” for MF HU

Issue: AEE Hungary: added configuration option "printingAEELogUnresponsivenessTime" to allow configuration of the real time command timeout in case of AEE log data printing which may under circumstances be longer than the global one as reported by application programmers

Affects: wn-javapos-fiscalprinter

PA41 Configuration option “resuncOnPrintZReportTimeout” for MF HU

Issue: AEE Hungary: added configuration option "resyncOnPrintZReportTimeout" to avoid wrongly thrown timeout exceptions on printZReport calls caused under circumstances by the power off/on cycle initiated by the fiscal FW during day end procedure.

Affects: wn-javapos-fiscalprinter

PA40 DirectIO 1203 - MAP_IMAGE_KEY for MF IT

Issue: TH230, TH230-FFC Italy: added directIO 1203 - MAP_IMAGE_KEY, particular for OPOS legacy applications migrating to OPOS/UDM (PTC 2186414)

Affects: wn-javapos-fiscalprinter

PA39 Open Name configuration for RS232 based fiscal printer for MF IT

Issue: TH230-FFC Italy: added open name configuration for RS232 based fiscal printer type

Affects: wn-javapos-fiscalprinter

PA38 Training mode capabilities for MF RO and MF BG

Issue: MF Romania, MF Bulgaria: enabled training mode capabilities

Affects: wn-javapos-fiscalprinter

PA37 Configuration parameter “initialZeroValue” for command readWeight()

Issue: Added configuration parameter "initialZeroValidValue=true" to the configuration files. Caution the default value is set to TRUE instead of FALSE as defined by UPOS spec. (PTC 2100814)

Affects: wn-javapos-iscan

PA36 JavaPOS entries for SEL of ACO KioskBox and Electronic Keylock behind SEL of ACO KioskBox

Issue: Added the JposEntries "WN_SELACO_KioskBox" and "WN_ELECTRONIC_KEYLOCK_ACOKioskBox" for SEL of ACO Kiosk Box

Affects: wn-javapos-selaco

PA35 Simple logging

Issue: Added script setlogging.vbs for enabling and disabling of simple JavaPOS logging (PTC 2113602)

Affects: wn-logger

PA34 UnifiedPOS category ImageScanner

Issue: Added support for UnifiedPOS category ImageScanner for CPOS.

Affects: wn-udm-cpos

PA33 Buffering of transaction print calls at UDM client side

Issue: Added configurable buffering of transaction print calls on UDM client side to speed up the print processing. This feature is configurable via the property "transactionPrintBuffering" at JavaPOSUDMAdapter.config.xml and/or P4DNUDMAdapter.config (PTC 2152068)

Affects: wn-udm-javapos; wn-udm-p4dn (pos for .net)

PA32 Optional parameter “session ID” for createInstance call to UDM Server

Issue: Added optional parameter “session Id” for 'creationInstance' command to allow multiple connection to a PDH claimed UDM server from different applications (PTC 1702774)

Affects: wn-udm-javapos

PA31 Configuration parameter “applianceID” for moPOS PDH

Issue: Added configuration parameter “applianceID” at P4DNPOSUDMAdapter.config to allow multiple applications running on one remote client to access the devices on a POS Device Hub in parallel without being rejected even if the particular application does not have claimed the POS Device Hub itself but it was claimed by a "master" application on the same tablet (PTC 1702774)

Affects: wn-udm-p4dn (pos for .net)

PA30 Configuration parameter “connectionTimeout” for UDM Server connections

Issue: Added configuration parameter “connectionTimeout” for controlling the UDM connection time out. This can improve the success condition in receiving the UDM server greeting message. By default set to 30 seconds (PTC 2088111)

Affects: wn-udm-p4dn (pos for .net)

11.3.3 Fixes

PF39 BA64 USB showed error messages on the terminal console

Issue: Using the BA64 USB showed some irritating error message on the terminal console under Linux.

Affects: wn-common-usb-native

PF38 DirectIO 11114 – EJ_SET_LIMITS_UDM throws ILLEGAL exceptions at MF-HU

Issue: The directIO 11114 - EJ_SET_LIMITS_UDM – used at MF HU (AEE) throws an ILLEGAL exception in case wrong data parameter are passed.

Affects: wn-javapos-fiscalprinter

PF37 Fiscal printer could make state transitions in case of LOCKED state

Issue: The fiscal printer could falsely state transitions in case of LOCKED state (aka fiscal printer blocked)

Affects: wn-javapos-fiscalprinter

PF36 Totalizer type FPRT_TT_RECEIPT was not adapted to AEE specifications at MF-HU

Issue: The command getTotalizer() for totalizer type FPTR_TT_RECEIPT used general Hungary implementation instead of AEE specific one.

Affects: wn-javapos-fiscalprinter

PF35 DirectIOs GET_ARTICLE_LIST, GET`_DEPARTMENT_LIST and GET_HEADER return values are not

correct decoded when CP866 was used at MF-BG

Issue: Cyrillic names returned by DirectIOs GET_ARTICLE_LIST, GET_DEPARTMENT_LIST, and GET_HEADER were not properly decoded to Unicode, when CP866 was used (PTC 2097520)

Affects: wn-javapos-fiscalprinter

PF34 Payment change receipt printing did not allowed positive rounding at MF-HU

Issue: The payment change receipt printing failed when using positive rounding. Especially when foreign currencies were involved. (PTC 2186388)

Affects: wn-javapos-fiscalprinter

PF33 DirectIO 1203 – MAP_IMAGE_KEY was faulty at MF-RO

Issue: The directIO 1203 MAP_IMAGE_KEY was faulty.

Affects: wn-javapos-fiscalprinter

PF32 Payment change receipt definition was not up to date to current fiscal law at MF-HU

Issue: The definition and example for Payment Change Receipt was not up to date, as CASH_OUT was still allowed to be used. Therefore, the total value was wrong.

Affects: wn-javapos-fiscalprinter

PF31 TP07 MBean throwed NullPointerException

Issue: A NullPointerException occurred in TP07 MBean (PTC 2155619)

Affects: wn-javapos-tsop

PF30 Update issues on WNLPOS installations

Issue: On update installations on WNLPOS pre-installations, there were several minor issues (PTC 2027478)

Affects: wn-udm-javapos

11.3.4 Changes

PC58 Moved JavaPOS Configurator output for “all devices” configuration

Issue: The all devices JavaPOS configuration will now be created under /etc/opt/wn/javapos/all instead of /etc/opt/wn/javapos/jpos-all under Linux.

Affects: wn-javapos-config; wn-udm-javapos; wn-javapos-samples

PC57 JavaPOS Device Controls updated to 1.14

Issue: The JavaPOS device control implementation was updated to controls compliant with UnifiedPOS 1.14 (before controls were compliant with UnifiedPOS 1.13)

Affects: wn-javapos-controls

PC56 Fiscal printer DS update due to new FW version 00-14-34 for MF-RO

Issue: The fiscal printer DS was updated due to latest FW version 00-14-34 for MF-RO. This included:

Removed getting/setting TAX terminal mode, now mandatory by law

Added article category processing at description parameter to printRec* methods

Avoiding paper waste by creating small cut paper pieces collected in the cutter during receipt begin

Affects: wn-javapos-fiscalprinter

PC55 Negative receipts layout changed accordingly fiscal homologation at MF-HU

Issue: The quantity line in negative receipts will now be printed if item header lines has been printed, even if quantity is one. This was required at the homologation for MF-HU (AEE)

Affects: wn-javapos-fiscalprinter

PC54 Power-on detection will issue reinitialization of TH180 and WNPP0x printer

Issue: The power-on detection at TH180 and WNPP0x kiosk printers will issue the re-initialization commands now.

Affects: wn-javapos-posprinter

PC53 CashDrawer DeviceService behavior is improved and more configurable

Issue: The CashDrawer Device Service has a new parameter and the behaviour regarding status feedback shanged slightly for better handling of situations where the drawer is still/again closed after openDrawer returns (PTC 2152059)

Affects: wn-javapos-retail

PC52 USB enable/disable behavior for USB scanner is now configurable

Issue: The USB enable/disable behavior for USB HID scanner is now configurable (PTC Issue 2105635)

Affects: wn-javapos-retail

PC51 Smaller barcodes can be printed

Issue: The size check of the call BarcodeToImage() changed to allow smaller barcodes to be printed (PTC 1727140)

Affects: wn-javapos-retail

PC50 JavaPOS tool center will restart with “all-devices” configuration” if no “target” configuration is available.

Issue: The JavaPOS tool center will not end if no target configuration has been generated and will be restarted with the “all-devices” configuration instead.

Affects: wn-javapos-samples

PC49 Changed ToolCenter function “Trace Configurator” to “Logging Configurator” to enable/disable

simple logging

Issue: The JavaPOS ToolCenter functionality "Trace Configurator" was changed to "Logging Configuration" for enabling/disabling simple logging for JavaPOS.

Affects: wn-javapos-samples

PC48 Removed internal calls <GS I 67> at the THxxx DS due to being obsolete

Issue: The call <GS I 67> to check the printer type for TH210 or TH230 is removed from the THxxx JavaPOS DS because the TH230 has its own device service.

Affects: wn-javapos-thxxx

PC47 TP07 under OPOS UDM checks now the configuration parameter "setRecLineSpacing"

Issue: The TP07 running under OPOS UDM will now check the optional configuration parameter setRecLineSpacing (PTC 1727219).

Affects: wn-javapos-tp07

PC46 WN-Logger will now check the result code from the UAC dialog

Issue: The WN-Logger will now check the result code from the UAC dialog under Windows to avoid hang-ups at the logger start-up when the UAC dialog is canceled by the user. (PTC 2113602)

Affects: wn-logger

PC45 UDM server logging will now create the log directy

Issue: The UDM server logging under Linux will now create the log directory if not available to avoid that the UDM server is not starting.

Affects: wn-udm-javapos

PC44 UDM Adapter avoids failed answer analysis.

Issue: The UDM Adapter avoids now UDM answer analysis failures if dollar '$' (0x24) or backslash '\' (0x5c) characters are contained in the answer from the UDM server.

Affects: wn-udm-javapos

PC43 OPOS UDM Config Updater maps now "sHydraProfileName"s also

Issue: The OPOS UDM Config Updater can now map "sHydraProfileName" value names to UDM OPOS open names as the OUCU does for "uses" values already. Necessary for devices connected to POSPrinter TH250.

Affects: wn-udm-opos

11.4 Changes to version 2.2

The following is a list of all changes and bug fixes between ProBase POS version 2.1 and ProBase POS 2.2.

11.4.1 General

PG2 Windows 10

Issue: Added support for Microsoft Windows 10 operating systems.

Note: USB peripherals (using the WNPOSUSB driver up to now) have to use the DNPOSUSB driver version 3.00.0.0-1 or later on Microsoft Windows 10 operating systems.

11.4.2 Add-ons

PA29 Configuration property "paperNearEndReceiptCounter" for MF-IT

Issue: Added configuration property "paperNearEndReceiptCounter" to support new day begin run time FW property at TH230-FFC

Affects: wn-javapos-fiscalprinter

PA28 Configuration property "partialCutPercentage" and directIO 811 for MF-IT

Issue: Added configuration property "partialCutPercentage" to control partial cut static as well as directIO 811 to control partial cut dynamically for TH230-FFC and TH230-MF

Affects: wn-javapos-fiscalprinter

PA27 DirectIO 602 - FORCE_DOC_TOTAL_POPULATION for MF-HU

Issue: AEE Hungary: added directIO 602/FORCE_DOC_TOTAL_POPULATION for giving application control whether internally computed document total value is populated to fiscal device on last printRecTotal call

Affects: wn-javapos-fiscalprinter

PA26 Pairing check for Till Check receipt for MF-HU

Issue: Added pairing check for Till Check receipt as requested by fiscal law for AEE

Affects: wn-javapos-fiscalprinter

PA25 Specify payment Id for currency payments for MF-HU

Issue: Added possibility to specify payment Id for currency payments, overriding automatically maintained payment Ids given by configuration property currencyMap for AEE.

Affects: wn-javapos-fiscalprinter

PA24 Command printRecTaxId for MF-RO

Issue: Added printRecTaxId command to allow tax Id registration for being printed with endFiscalReceipt at TH230-FFC

Affects: wn-javapos-fiscalprinter

PA23 Default unit name for printRecItem for MF-RO

Issue: Added default unit name for printRecItem as unit name is mandatory by fiscal law for TH230-FFC

Affects: wn-javapos-fiscalprinter

PA22 Command case getData(FPTR_GD_FISCAL_REC) for MF-RO

Issue: Added command case getData(FPTR_GD_FISCAL_REC) for retrieving number of fiscal receipts with printed tax id for TH230-FFC

Affects: wn-javapos-fiscalprinter

PA21 DirectIO 820 - PRINT_SELFTEST for MF-RO

Issue: Added directIO 820 - PRINT_SELFTEST for issuing a fiscal self-test printout (required by homologation in Romania) for TH230-FFC

Affects: wn-javapos-fiscalprinter

PA20 DirectIO 301 - DAY_OPEN_RECEIPT_REQUIRED for MF-HU

Issue: Added direction 301 - DAY_OPEN_RECEIPT_REQUIRED as predicate whether an open day receipts still needs to be issued for AEE

Affects: wn-javapos-fiscalprinter

PA19 Support for BA9x MSR PID 0x0405

Issue: Added BA9x M4 MSR product ID 0x0405 to supported usb product id's (change applies to Linux only) at JposEntry "WN_MSR_USB"

Affects: wn-javapos-kkmusb

PA18 Method QueryTabletID and direction 70 – QUERY_TABLET_ID for PDH

Issue: Added method QueryTabletId as well as directIO 70 - QUERY_TABLET_ID for allowing POSDeviceHub to retrieve Id of docked tablet (PTC#2081676)

Affects: wn-udm-p4dn; wn-udm-javapos

PA17 DirectIO – GET_DOCKED for PDH

Issue: Added directIO GET_DOCKED for POS Device Hub controller (PTC#2060837)

Affects: wn-udm-javapos

PA16 UDM Server Logging configuration

Issue: Added UDM server specific logging configuration to create separate log file udm-server.log.

Affects: wn-logger

11.4.3 Fixes

PF29 WN-Logger throws exception when used with Trace log level

Issue: When using the WN-Logger with log level “Trace”, exceptions were thrown. Some other small issues with JMX, wrong re-initialization at device open calls occurred also. (PTC#2027756)

Affects: wn-javapos-cashchangers; wn-javapos-f53; wn-javapos-iscan; wn-javapos-retail; wn-javapos-selaco; wn-javapos-tp07; wn-javapos-thxxx; wn-javapos-tsop

PF28 PrintBarcode EAN128 got false result and was limited

Issue: The PrintBarcode command for EAN128 result was wrongly Code128 and was in general below the maximum of possible barcode data (PTC 1772682)

Affects: wn-javapos-retail

PF27 Starting scripts of JavaPOS ToolCenter and SwingSamples runs endless or crashes

Issue: In certain configurations it was possible to have the caller scripts of JavaPOS ToolCenter or SwingSample run endless or to crash with exceptions. (PTC#2026949; #1938198; #1938198)

Affects: wn-javapos-samples

PF26 Very long Arabic receipts caused StackOverflowError

Issue: A StackOverflowError occurred during processions of very long receipts with Arabic text (MKS-2000376)

Affects: wn-javapos-thxxx

PF25 Capability CapSlpNearEndSensor handling was wrong

Issue: The result of the method getSlpNearEndSensor was not managed by the capability CapSlpNearEndSensor of the TH230 USB

Affects: wn-javapos-thxxx

PF24 Conversion of byte array properties failed

Issue: The conversion for properties including byte arrays failed at the P4DN UDM adapter (PTC#1830492)

Affects: wn-udm-p4dn

PF23 UDM Server logging configuration was broken

Issue: The logrotate configuration for the UDM server logging was broken and the log file was overwritten at system start, when UDM server was startet automatically causing lost log file. (PTC#2065690)

Affects: wn-udm-javapos

PF22 Payment line alignment not correct

Issue: The payment line alignment was not correct at TH230-FFC (MF-IT)

Affects: wn-javapos-fiscalprinter

PF21 Payment logic for special cases not correct

Issue: The payment logic as well as the method printRecTotal for a special cases where the total is zero achieved by rounding only (e.g. 1Ft amount total - 1Ft rounding == 0Ft real total) was not correct at AEE (MF-HU)

Affects: wn-javapos-fiscalprinter

PF20 New FW led to wrong cash-out handling

Issue: New FW 00-14-26 affected cash-out handling at TH230-FFC (MF-RO)

Affects: wn-javapos-fiscalprinter

PF19 IndexOutOfBound exception detection was not correct

Issue: The auto detection for IndexOutOfBound exception in case of wrong answers from MFC was not correct at AEE (MF-HU)

Affects: wn-javapos-fiscalprinter

PF18 NPE happening in case of weight items was not correct

Issue: NPE happening for printRecItemRefund and printRecItemRefundVoid calls in case of weight items was not correct at AEE (MF-HU)

Affects: wn-javapos-fiscalprinter

PF17 Encoding of several printRec methods led to problems printing special Hungarian chars

Issue: The encoding for printRecItemAdjustment, printRecItemAdjustmentVoid, printRecRefund, printRecRefundVoid, printRecSubtotalAdjustment, and printRecSubtotalAdjustVoid methods was not correct and led to problems printing Hungarian special characters at MF-HU

Affects: wn-javapos-fiscalprinter

PF16 Documentation of negative receipts was wrong

Issue: The documentation of negative receipts was not correct at AEE (MF-HU)

Affects: wn-javapos-fiscalprinter

PF15 DirectIO 1203 and 808 did not avoid other directions called subsequently

Issue: The directIOs 1203 - MAP_IMAGE_KEY and 808 - SET_CASH_DRAWER_BALANCE_TYPE did not avoid other directIO commands to be called subsequently at TH230-FFC (MF-RO)

Affects: wn-javapos-fiscalprinter

PF14 New FW command leads to wrong printRecItemVoid command

Issue: Corrected FW command for weight affected printRecItemVoid() calls for all MF-Printer at MF-RO

Affects: wn-javapos-fiscalprinter

11.4.4 Changes

PC42 Substitute all Windows API calls with secure API calls

Issue: All unsecure Windows API calls are substituted by corresponding secure calls (PTC#1553222)

Affects: wn-common-usb-native; wn-common-rs232-native; wn-javapos-beeper-native; wn-javapos-kbdclaimer-native; wn-javapos-portio-native;

PC41 Modified PID at JposEntry “WN_BA9x_MSR_USB”

Issue: The JposEntry "WN_BA9x_MSR_USB" was modified with the corrected usb product id 0x0405 (change applies to Linux only)

Affects: wn-javapos-kkmusb

PC40 DirectIO 1053 accepts now parameter

Issue: The direction 1053 - payInPayOutAbort accepts now a parameter for rejectMoney = true|false

Affects: wn-javapos-cashchangers

PC39 Method DepositAutoChange renamed to payInPayOut

Issue: The method depositAutoChange operations is renamed to payInPayOut*

Affects: wn-javapos-cashchangers

PC38 DirectIO 1052 changes to asynchronous call

Issue: The direction 1052 - PayIn-PayOut / Deposit Auto Change is now turned into an asynchronous call

Affects: wn-javapos-cashchangers

PC37 Create log file for JavaPOS Configurator run at Linux system start

Issue: The log file 'config_javapos_startup.sh.log' will now be created on every call of 'wn_javapos_config.sh' (PTC#2021695)

Affects: wn-javapos-config

PC36 Modified timer usage for scale

Issue: The timer instance is now reused instead of creating new instances (MKS-1985117)

Affects: wn-javapos-iscan

PC35 Reduce load from high frequency calls to scales

Issue: ReadWeight() calls in readFast Mode (directIO 8) are now spaced out in case they are called high frequently (MKS-1985117)

Affects: wn-javapos-iscan

PC34 Documentation update for C1030

Issue: Updated SELACO / C1030 documentation (added info how to know if CITBox is closed or full)

Affects: wn-javapos-selaco

PC33 Correct log file output to follow JavaPOS guidelines

Issue: Moved logging file creation from <data base dir>\logs\jddConfigFiles to <data base dir>\log\jddConfigFiles to follow specification requirements more strictly (PTC#1952421)

Affects: wn-javapos-selaco

PC32 Added Launcher and test configuration

Issue: Added launcher and test config file (PTC#1772682)

Affects: wn-javapos-tp07

PC31 Enabled print barcode on slip station for Pharmacy printer

Issue: Enabled the print barcode command on slip enabled for Pharmacy printer (Jump)

Affects: wn-javapos-thxxx

PC30 Changed log file behavior to log rotate

Issue: Changed the logging to log4net as log file rotation to be configured with 10 files per 1 MB

Affects: wn-udm-p4dn

PC29 Split P4DNUDMAdapter.dll for multi access configurations

Issue: The POSDeviceHub.dll is now separated from the P4DNUDMAdapter.dll to get several application running on the same machine which utilize POSDeviceHub access.

Affects: wn-udm-p4dn

PC28 Modified exception messages and replace %20 characters with spaces

Issue: The exception messages coming from the UDM server are now modified by properly replacing '%20' characters with spaces

Affects: wn-udm-p4dn

PC27 Avoid logging errors about trailing spaces to reduce misunderstandings

Issue: Logging errors about trailing spaces coming from UDM server reduced to avoid misunderstandings at application side

Affects: wn-udm-p4dn

PC26 Modified trace message to avoid misunderstandings

Issue: The trace message on the xOpenService method is modified to avoid a info note being misunderstood as error

Affects: wn-udm-opos

PC25 UDM Server logging changes also to WN-Logger

Issue: The proprietary UDM server logging to stdout is replaced by common WN logging; the UDM server logging is also controlled by the wn-logger.properties file; however, it can still be enabled programmatically through the '-debug' command line option within the caller script.

Affects: wn-udm-javapos

PC24 UDM server will now send the tablet undock command directly instead of calling a shell command

Issue: The UDM server will now send the tablet undock command directly from inside and not by executing a configured shell command (precondition for realization of Windows based POS Device Hubs).

Affects: wn-udm-javapos

PC23 PDH call made available under Windows

Issue: The POS Device Hub call checker configuration file is now available under Windows as "config\pdh.properties"

Affects: wn-udm-javapos

PC22 Extended size of log files

Issue: The logrotate configuration for wn-udm.log is extended to size of 5MB

Affects: wn-udm-javapos

PC21 Library is compiled statically against C runtime to avoid missing runtime library

Issue: The library is now statically compiled against Microsoft C runtime library to avoid missing msvcrt*.dll at runtime.

Affects: wn-udm-cpos

PC20 Programming API improved

Issue: The programming API was improved by introducing com.wn.log.WNLoggerFactory as the main facade and marking com.wn.log.liblogger.WNLibLoggerFactory as deprecated.

Affects: wn-logger

PC19 Avoid interferences with log4j configurations used by applications

Issue: The log4j configuration will now avoid interferences with application's log4j configuration provided by the application as log4j.xml or log4j.properties resources file on the class path.

Affects: wn-logger

11.5 Changes up to version 2.1

The following is a list of all changes and bug fixes between ProBase POS version 2.0 and ProBase POS 2.1.

11.5.1 General

PG1 WN Logger

Issue: Switched from proprietary WN JavaPOS Tracing/Logging to WN Logger based on log4j.

Note: The Trace Configurator from the JavaPOS ToolCenter cannot be used to change the logging settings for JavaPOS devices. This needs to be done by modifying the related WN Logger properties file.

11.5.2 Add-ons

PA15 DisableASBonClose

Issue: Added DisableASBonClose to help after Stall PID / Abort Pipe / Missing IN Token / USBD_STATUS_XACT_ERROR issues (.

Affects: wn-javapos-thxxx (DS WNPOSPrinterTH230)

PA14 Support several USB peripherals under Linux

Issue: Linux USB accessing library now supports:
BA63 GU2 (0x0AA7:0x0206)
BA63 GU2 M4 (0x0AA7:0x0207)
BA9x MSR M4 (0x0AA7:0x0408)
BA9x Modular Keylock (0x0AA7:0x0409)
BA9x MSR (0x0AA7:0x0405)

Affects: wn-common-usb-native

PA13 JavaPOS entries for ITL CashChanger

Issue: Added JavaPOS entries (OpenNames) for ITL CashChanger

Affects: wn-javapos-cashchangers

PA12 JavaPOS entries for BCR-200 CashChanger

Issue: Added JavaPOS entries (OpenNames) for BCR-200 CashChanger

Affects: wn-javapos-cashchangers

PA11 DirectIO for dispenseAutoChange

Issue: Added dispenseAutoChange directIO

Affects: wn-javapos-cashchangers

PA10 OEM code page mapping at MF-HU

Issue: Added OEM code page mapping for payment descriptions ensuring Hungarian special characters are printed well (PTC #1958558) for AEE

Affects: wn-javapos-fiscalprinter

PA9 Device Service for IOBox on the Fastlane V5

Issue: Added new Device Service WNIOBoxNCR (including configuration and documentation) for handling the IOBox on the Fastlane V5.

Affects: wn-javapos-iscan

PA8 Support and JavaPOS entries for EL 71 - Zebra scanner DS4308

Issue: Added support and JavaPOS entries for Zebra scanner DS4308 as EL 71 (PTC 1910729)

Affects: wn-javapos-retail

PA7 DirectIO commands 114, 115, and 116 for WNScannerUSB.class

Issue: Added directIO commands 114, 115, and 116 (similar implementation as already done for WNScanner.class) (PTC 1951607)

Affects: wn-javapos-retail (com.wn.retail.jpos113.WNScannerUSB.class)

PA6 Configuration options for WNScannerUSB.class

Issue: Added configuration options 'commandGoodBeep', 'commandBadBeep', and 'commandNotOnFileBeep' (similar implementation as already done for WNScanner.class) (PTC 1951607)

Affects: wn-javapos-retail (com.wn.retail.jpos113.WNScannerUSB.class)

PA5 JavaPOS entry for WN_Scanner_Intermec_ED40

Issue: Added JavaPOS configuration "WN_Scanner_Intermec_ED40"

Affects: wn-javapos-scanner

PA4 Symbology Identifier for Intermec ED 40 scanner

Issue: Added new WNScanner global Symbology Identifier to WNScanner.class being supported by Intermec ED 40 Scanner

Affects: wn-javapos-scanner

PA3 DeviceAdapter for Intermec ED40 Scanner

Issue: Added DeviceAdapter for Intermec ED40 Scanner (as requested by MPS project)

Affects: wn-javapos-scanner

PA2 Label type id definitions

Issue: Added label type id definitions not defined in UPOS but supported by Intermec scanners.

Affects: wn-javapos-scanner (WNScanner)

PA1 Configuration option enableNixdorfModeCompatibility

Issue: Added configuration option 'enableNixdorfModeCompatibility' that will, if enabled, replace the ScanData as received from DeviceAdapter by NixdorfPrefix + LabelDate + \n.

Affects: wn-javapos-scanner

11.5.3 Fixes

PF13 Administrator rights were not checked and requested from JavaPOS Configurator

Issue: The JavaPOS Configurator does not check if executed with or requests for administrator rights for execution if not (PTC 1937968).

Affects: wn-javapos-config

PF12 Installer has not checked the JRE availability

Issue: The product installer does not check whether a JRE is installed before installing the components.

Affects: wn-javapos-config

PF11 Wrong data received when using getData(PRINTER_ID) at MF-IT

Issue: corrected getData(PRINTER_ID) for Italy, devices TH230-MF, MF-EJ-THF and MF-THF (PTC 1963625)

Affects: wn-javapos-fiscalprinter

PF10 Using the virtual Linedisplay could freeze the application

Issue: The WN_VIRTUAL_LINEDISPLAY could freeze the Java program if text in the virtual line display was selected (PTC 1937961).

Affects: wn-javapos-linedisplay

PF9 KBDClaimerTestMain class was not available at Low Level Tests anymore

Issue: The KBDClaimerTestMain class was not available anymore at the "POSKeyboard Low Level Test" in Probase ToolCenter (PTC #1938238)

Affects: wn-javapos-retail

PF8 Log-level TRACE led to error on Open() for different classes

Issue: The TRACE level logging caused errors on device open for CashDrawers, MSR7816, USB LineDisplays, MotionSensors, and POSKeyboards.

Affects: wn-javapos-retail

PF7 Path of sample files was missing

Issue: Sample .bmp and .txt files had missing path (PTC 1938006)

Affects: wn-javapos-samples

PF6 JavaPOS examples were not executable

Issue: JavaPOS examples compilation scripts (compile.bat and runtest.bat) were not executable. Now prepared to be run as Administrator from the explorer (PTC 1950122)

Affects: wn-javapos-samples

PF5 Reset bitmap with empty string showed wrong error code

Issue: Resetting of stored bitmaps by calling empty string (for path and filename) on printer throw an error 'Error at UDM server: setBitmap: file name not found' (PTC 1856175)

Affects: wn-javapos-thxxx (DS WNPOSPrinterTH230)

PF4 Missing timeout for receipt prints

Issue: Receipt prints (Jump) missed a timeout (PTC 1965206).

Affects: wn-javapos-thxxx (DS WNPOSPrinterApoPtr)

PF3 Alignment of rotated barcode was not possible

Issue: The alignment of a rotated barcode was not possible (PTC 1767999).

Affects: wn-javapos-thxxx (DSWNPOSPrinterTH230)

PF2 CPOS examples could not be run out-of-box

Issue: Provided examples compilation could not be used out of the box (PTC 1947176, 1952624)

Affects: wn-udm-cpos

PF1 UDM server connection could dead lock application

Issue: The UDM server connection could dead lock the application under circumstances, where the connections was not fully established before forcing createInstance (PTC 1636333)

Affects: wn-udm-cpos

11.5.4 Changes

PC18 CIM log files

Issue: CIM log files will now be stored within <data base dir>\log\cim on Windows (previously <data base dir>\logs\cimAdapter) accordingly the JavaPOS guide line requirements (PTC 1952421)

Affects: wn-cim

PC17 JavaPOS log files

Issue: JavaPOS log files will now be stored within /var/log/wn/javapos on Linux (previously var/log/wn_javapos) accordingly the JavaPOS guide line requirements

Affects: wn-common-usb-native

PC16 Diagnostics log files

Issue: Diagnostics log files will now be stored within <data base dir>\log\diagnostics on Windows (previously <data base dir>\logs\diagnostics) accordingly the JavaPOS guide line requirements (PTC 1952421)

Affects: wn-javapos-diagnostics

PC15 Error will be thrown for VLD when used on Linux without XWindow

Issue: A JposException ILLEGAL will be thrown in case the virtual line display is opened on a Linux without a running XWindows system.

Affects: wn-javapos-linedisplay

PC14 Behaviour of paper cut for PP0x printer is changed

Issue: The paper cut of full cut is now with paper management (the cut will be between the labels), the partial cut cuts the paper at the actual position for all WN_PP0x printer.

Affects: wn-javapos-posprinter

PC13 Changed MSR parameter clearTime to msrClearTime

Issue: The MSR config param clearTime was changed to msrClearTime and will be initialized different for Windows and Linux (default here is 1250).

Behaviour of MSR thread start & retry to open pipes is now configurable (default is set to 3) because the MSR thread still purges old/illegal card data for 200 ms after thread start (PTC 1565155)

Affects: wn-javapos-retail, wn-javapos-kkmusb

PC12 Change general behaviour of direction commands for USB scanner

Issue: Behavior of directIO commands sending commands to scanner changed and will now also validate the responses. If command is rejected by the device a JposException is thrown (PTC 1951607)

Affects: wn-javapos-retail

PC11 Changed default Line display device name and parameter at SwingSamples

Issue: The default device name and the parameter for row and column at linedisplay test as well as the xPositionVD default value (set to 100) for the virtual line display was changed (PTC 1950099)

Affects: wn-javapos-samples

PC10 JDD log/config files

Issue: JDD log files will now be stored within <data base dir>\log\jddConfigFiles on Windows (previously <data base dir>\logs\jddConfigFiles) accordingly the JavaPOS guide line requirements (PTC 1952421)

Affects: wn-javapos-selaco

PC9 JavaPOS log files

Issue: JavaPOS log files will now be stored within <data base dir>\log on Windows (previously <data base dir>\logs) accordingly the JavaPOS guide line requirements (PTC 1952421)

Affects: wn-javapos-trace, wn-logger

PC8 UDM Server log files

Issue: UDM Server log files will now be stored within <data base dir>\log\udm on Windows (previously <data base dir>\udmserver\logs) respectively within /var/log/wn/log/udm on Linux (previously /var/opt/wn/javapos/logs/udm) accordingly the JavaPOS guide line requirements (PTC 1952421)

Affects: wn-udm-cpos, wn-udm-javapos

PC7 Avoid trace pop-up window

Issue: Avoid trace pop-up window, which is shown when X-server was started and trace is enabled on Linux (PTC 1982499)

Affects: wn-udm-javapos

PC6 DirectIO(999) modified

Issue: The directIO(999) was modified.

Affects: wn-javapos-scanner, wn-javapos-scale, wn-javapos-posprinter, wn-javapos-msr, wn-javapos-retail

PC5 Behaviour of option hardwareAutoDisable changed

Issue: The behaviour of scanner devices when the configuration option 'hardwareAutoDisable' is set to true is changed. Now explicit calls to setDataEventEnabled(true|false) will enable or disable laser in order to enable or disable scanning. Default value of this configuration option, also valid if key is not set at all, is changed to true. (PTC 1930509)

Affects: wn-javapos-scanner

PC4 WN Tracing removed

Issue: The com.wn.retail.jpos113.TraceLogger was removed as it will not be used anymore (replaced by new WN Logger)

Affects: wn-javapos-common

PC3 MF-HU implementation changed based on law change in 2016

Issue: The implementation of AEE based Hungarian fiscal printer devices is now adapted as required by law change in 2016

Affects: wn-javapos-fiscalprinter

PC2 Exceptions corrected

Issue: The exceptions in case of an error at Open() and Close() are changed from JPOS_E_CLOSED to JPOS_E_FAILURE.

Affects: wn-javapos-kkmusb

PC1 Enable() and Disable() scanner will wait for response

Issue: The Enable() and Disable() requests for scanner peripherals will now wait for its related responses.

Affects: wn-javapos-retail

55 / 77 / 46 / 5