Oracle® Retail EFTLink

Core Configuration Guide

Release 16.0.0.1

E85721-07

March 2019

Oracle® Retail EFTLink Core Configuration Guide, Release 16.0.0.1

E85721-07

Primary Authors: Ian Williams, Matt Preston

Contributors: Tracy Gunston

Value-Added Reseller (VAR) Language

Oracle Retail VAR Applications

The following restrictions and provisions only apply to the programs referred to in this section and licensed to you. You acknowledge that the programs may contain third party software (VAR applications) licensed to Oracle. Depending upon your product and its version number, the VAR applications may include:

(i) the **MicroStrategy** Components developed and licensed by MicroStrategy Services Corporation (MicroStrategy) of McLean, Virginia to Oracle and imbedded in the MicroStrategy for Oracle Retail Data Warehouse and MicroStrategy for Oracle Retail Planning & Optimization applications.

(ii) the **Wavelink** component developed and licensed by Wavelink Corporation (Wavelink) of Kirkland, Washington, to Oracle and imbedded in Oracle Retail Mobile Store Inventory Management.

(iii) the software component known as **Access Via™** licensed by Access Via of Seattle, Washington, and imbedded in Oracle Retail Signs and Oracle Retail Labels and Tags.

(iv) the software component known as **Adobe Flex™** licensed by Adobe Systems Incorporated of San Jose, California, and imbedded in Oracle Retail Promotion Planning & Optimization application.

You acknowledge and confirm that Oracle grants you use of only the object code of the VAR Applications. Oracle will not deliver source code to the VAR Applications to you. Notwithstanding any other term or condition of the agreement and this ordering document, you shall not cause or permit alteration of any VAR Applications. For purposes of this section, "alteration" refers to all alterations, translations, upgrades, enhancements, customizations or modifications of all or any portion of the VAR Applications including all reconfigurations, reassembly or reverse assembly, re-engineering or reverse engineering and recompilations or reverse compilations of the VAR Applications or any derivatives of the VAR Applications. You acknowledge that it shall be a breach of the agreement to utilize the relationship, and/or confidential information of the VAR Applications for purposes of competitive discovery.

The VAR Applications contain trade secrets of Oracle and Oracle's licensors and Customer shall not attempt, cause, or permit the alteration, decompilation, reverse engineering, disassembly or other reduction of the VAR Applications to a human perceivable form. Oracle reserves the right to replace, with functional equivalent software, any of the VAR Applications in future releases of the applicable program.

Contents

Send Us Your Comments

Oracle Retail EFTLink, Core Configuration Guide, Release 16.0.0.1

Oracle welcomes customers' comments and suggestions on the quality and usefulness of this document.

Your feedback is important, and helps us to best meet your needs as a user of our products. For example:

- Are the implementation steps correct and complete?
- Did you understand the context of the procedures?
- Did you find any errors in the information?
- Does the structure of the information help you with your tasks?
- Do you need different information or graphics? If so, where, and in what format?
- Are the examples correct? Do you need more examples?

If you find any errors or have any other suggestions for improvement, then please tell us your name, the name of the company who has licensed our products, the title and part number of the documentation and the chapter, section, and page number (if available).

> **Note:** Before sending us your comments, you might like to check that you have the latest version of the document and if any concerns are already addressed. To do this, access the Online Documentation available on the Oracle Technology Network Web site. It contains the most current Documentation Library plus all documents revised or released recently.

Send your comments to us using the electronic mail address: retail-doc_us@oracle.com

Please give your name, address, electronic mail address, and telephone number (optional).

If you need assistance with Oracle software, then please contact your support representative or Oracle Support Services.

If you require training or instruction in using Oracle software, then please contact your Oracle local office and inquire about our Oracle University offerings. A list of Oracle offices is available on our Web site at www.oracle.com.

Preface

This *Oracle Retail EFTLink Core Configuration Guide* describes the requirements and procedures to set up EFTLink to interface between the specific POS and the selected EFT payment system

Audience

This *Oracle Retail EFTLink Core Configuration Guide* is for the following audiences:

- System administrators and operations personnel
- Database administrators
- System analysts and programmers
- Integrators and implementation staff personnel

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit
http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit
http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Documents

For more information, see the following documents in the Oracle Retail EFTLink Release 16.0 documentation set:

- *Oracle Retail EFTLink Release Notes*
- *Oracle Retail EFTLink Framework Installation Guide*
- *Oracle Retail EFTLink Security Guide*
- *Oracle Retail EFTLink Configuration Utility User Guide*

Customer Support

To contact Oracle Customer Support, access My Oracle Support at the following URL:

https://support.oracle.com

When contacting Customer Support, please provide the following:

- Product version and program/module name
- Functional and technical description of the problem (include business impact)
- Detailed step-by-step instructions to re-create
- Exact error message received
- Screen shots of each step you take

Review Patch Documentation

When you install the application for the first time, you install either a base release (for example, 16.0) or a later patch release (for example, 16.0.1). If you are installing the base release or additional patch releases, read the documentation for all releases that have occurred since the base release before you begin installation. Documentation for patch releases can contain critical information related to the base release, as well as information about code changes since the base release.

Improved Process for Oracle Retail Documentation Corrections

To more quickly address critical corrections to Oracle Retail documentation content, Oracle Retail documentation may be republished whenever a critical correction is needed. For critical corrections, the republication of an Oracle Retail document may at times **not** be attached to a numbered software release; instead, the Oracle Retail document will simply be replaced on the Oracle Technology Network Web site, or, in the case of Data Models, to the applicable My Oracle Support Documentation container where they reside.

This process will prevent delays in making critical corrections available to customers. For the customer, it means that before you begin installation, you must verify that you have the most recent version of the Oracle Retail documentation set. Oracle Retail documentation is available on the Oracle Technology Network at the following URL:

http://www.oracle.com/technetwork/documentation/oracle-retail-100266.html

An updated version of the applicable Oracle Retail document is indicated by Oracle part number, as well as print date (month and year). An updated version uses the same part number, with a higher-numbered suffix. For example, part number E123456-**02** is an updated version of a document with part number E123456-**01**.

If a more recent version of a document is available, that version supersedes all previous versions.

Oracle Retail Documentation on the Oracle Technology Network

Oracle Retail product documentation is available on the following web site:

http://www.oracle.com/technetwork/documentation/oracle-retail-100266.html

(Data Model documents are not available through Oracle Technology Network. You can obtain them through My Oracle Support.)

Conventions

Navigate: This is a navigate statement. It tells you how to get to the start of the procedure and ends with a screen shot of the starting point and the statement “the Window Name window opens.”

```
This is a code sample
    It is used to display examples of code
```

1 Introduction

After installing EFTLink from the *Oracle Retail EFTLink Framework Installation Guide*, and as part of that selected a core (step 2 in that document), the implementer will need to configure the specific core with the required settings to allow the POS to communicate with the selected EFT System. This guide consists of separate sections for each available core; go to the pertinent section for each core to be installed.

Note: Also refer to the *Oracle Retail EFTLink Security Guide* for core specific actions to ensure secure configuration.

2

Adyen Core

Disambiguation

This core implementation is for use with Adyen JNI wrapper with communication based on a socket or serial protocol, implemented internally within the JNI, to the terminal.

EFTLink General

See also the *EFTLink Framework Installation Guide.*

This document assumes static EFTLink configuration. When deploying with a POS that supports dynamic configuration, all property settings referred to below should be set on the POS, and not directly into local property files.

Minimum Version

The Adyen interface requires a minimum EFTLink version of v15.0

System Architecture

EFTLink connects to Adyen's PED, via JNI wrapper.

> **Note:** This document does not cover installation of Adyen software

Fileset

In addition to standard EFTLink files, Adyen uses:

- `cores/Adyen/AdyenCore.jar` – executable code for the Adyen EFTLink core.
- `adyen.properties` – configuration settings to specify which features are enabled and to define communication parameters for the interface with the EFT payment system.
- `data/adyen.keystore` – keystore file to encrypt a password in adyen.properties, this file need to be generated at installation. Please see the next section for details.

Keystore

The encryption key must be generated and stored in a keystore. To achieve this, the following steps must be followed:

Windows Operating Systems

- Open a command prompt, and change directory to the eftlink location.
- Type `encrypt.bat -k <keystore name> <properties file>`

  For example, `encrypt.bat -k adyen.keystore adyen.properties.`

Keystore file will be generated and stored in the data directory

Password Encryption

The password within the `adyen.properties` file needs to be encrypted. To achieve this, the following steps must be followed:

Windows Operating Systems

To encrypt a password; open a command prompt and change directory to eftlink's location.

- Type `encrypt.bat –e <keystore name> <properties> <password>`.

  For example `encrypt.bat –e adyen.keystore adyen.properties [`followed by the required password as a final parameter].
- Password and initialization vector will be outputted to the console.

  Copy and paste it to `adyen.password` and `adyen.password.iv` in `adyen.properties`.

To re-encrypt a password with new encryption settings; open a command prompt and change directory to eftlink's location.

- Type `encrypt.bat –r <keystore name> <properties> <encrypted password> <previous initialization vector> <keygen type> <cipher type> <key size> <iterations>.`

  For example, `encrypt.bat –r adyen.keystore adyen.properties [Encrypted password] [Encrypted password iv] AES AES/CBC/PKCS5Padding 128 10000.`
- Re-encryption uses existing crypto settings in the properties file to decrypt the password. Once the password is decrypted, a new keystore file is generated using the new crypto parameters specified at the command line and the new encrypted password / initialization vector is generated. /
- When using AES algorithm with a keysize that is greater than 128, you may get java.security.InvalidKeyException: Illegal key size or default parameters. If so, Additional Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files will need to be downloaded and extracted to %JAVA_HOME%/jre/lib/security/

Third Party

Note: Critically important

The following file is also needed, not supplied by Oracle:

`POS_JNI32.jar/POS_JNI64.jar` is a JNI wrapper supplied by Adyen to allow communication to Adyen’s PED.

This should be placed in `cores\Adyen` alongside `AdyenCore.jar`. Use appropriate version according to VM environment, `POS_JNI32.jar` for 32-bit and `POS_JNI64.jar` for 64-bit.

Language

There are translation files in AydenCore.jar. These should not need to be modified, but if a translation needs to be changed, they can be extracted to the base eftlink folder.

`LangEN_Ayden.properties`

The file in use follows the language setting for EFTLink itself, defined in `EftlinkConfig.properties` e.g. `DisplayLanguage = EN`

Additional files could be added for other supported languages for the small set of translations required by the Adyen core.

Core Classname

The following should have been set in the EftlinkConfig.properties file by installcore.bat or installcore.sh

```
EPSCore0 = manito.eft.adyen.AdyenCore
```

Configuration Settings

The full set of configuration properties is defined and commented in ayden.properties.

Key Settings

Settings that may be different for all POSs.

Setting	Description	Example
adyen.environment	Live or Test environment. Default is Test.	adyen.environment = Live
adyen.merchant.account	Merchant account code provided by Adyen.	adyen.merchant.account = OracleTest
adyen.username	Username provided by Adyen.	adyen.username = ws_371398
adyen.password	Encrypted password, see password encryption section for details.	Adyen.password = [encrypted password string]
Adyen.password.iv	Encrypted password initialization vector, see password encryption section for details.	Adyen.password.iv = [encrypted password iv string]
ped.address	IP address of the PED. If serial ped then com port number.	ped.address = 10.0.0.99

Secondary Settings

These settings are normally correct at their default values, but can be overridden if necessary.

Setting	Description	Default	Example
ped.name	Any symbolic name of the PED.		ped.name = VX680_01
merchant.reference	Unique merchant reference.		merchant.reference = Merchant_A
tender.options	Specify tender options to be used. Currently supported options are: AskGratuity,AttendantActionHandler,BypassPin,DontPrintReceipt,EnableMagstripeFallback,Error,ForcedDecline,ForcedOnline,GetAdditionalData,KeyedCardDetailsHandler,KeyedEntry,NoCTLS,NoProcess,ReceiptHandler,UNKNOWN		tender.options = GetAdditionalData,ReceiptHandler,AttendantActionHandler

Setting	Description	Default	Example
tokenized.refund	Enables refund by token if set to true or auto. If set to false, standard refund will always be performed. If set to auto, tokenized refund will be performed if a token is supplied in the request otherwise standard refund will be used.	auto	`tokenized.refund = auto`
combine.receipt	When `combine.receipt` is true, sets which line number to suppress		`combine.receipt = true`
combine.receipt.suppress.lines	When `combine.receipt` is true, sets which line number to suppress		`combine.receipt.suppress.lines = 1,2,3,4,5`
combine.receipt.suppress.strings	When `combine.receipt` is true, sets which line to suppress when strings are matched.		`combine.receipt.suppress.lines = Date,Time`
pos.id.override	Overrides POS ID from the POS with a specify ID. Should be used in PED POOL mode.		`pos.id.override = 10`
cashback	Enables cashback. The following options are supported: off – no cashback pos – POS will prompt for cashback options via DeviceRequest ped – PED will prompt for cashback options via its display		`cashback = ped`
print.all.receiptSets	When set to true, enables all receipts sent from Adyen to be printed. When set to false, prints only the latest receipt set.		`print.all.receiptSets = false`
crypto.keygenType	Sets keygen algorithm type.		`crypto.keygenType = AES`
crypto.cipherType	Sets cipher algorithm type.		`crypto.cipherType = AES`
crypto.keySize	Sets size of the keystore. Note that when keysize is greater than 128, you may get java.security.InvalidKeyException: Illegal key size or default parameters. If this happens you will need to download additional Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files and extract those files to `${java.home}/jre/lib/security/`		`crypto.keySize = 128`
crypto.iterations	Sets number of iterations.		`crypto.iterations = 10000`

Supported Functions

Below is a list of supported functionalities of the interface to Adyen. Some functions provided by Adyen, such as Loyalty, Giftcard, Cashback etc. are not implemented in this release because of the business requirement.

Function	Description
Payment	EFTLink sends payment requests to Adyen. Adyen will return a response message with unformatted receipt strings for customer and/or merchant receipts. If successful, appropriate receipts will be printed at the end of transaction.
Reversal	EFTLink sends reversal requests to Adyen. This will reverse a transaction specified by the transaction number, found on the receipt, which must be captured by the POS and pass on to EFTLink.
Refund	EFTLink sends refund requests to Adyen. This will refund a transaction with specified amount.
Tokenized Refund	EFTLink sends refund requests to Adyen. This will refund a transaction with specified token id.

3

AJB FIPay

This FIPay implementation is for use with AJB FIPay software with communication via TCP/IP based on a proprietary socket protocol. It should be read in conjunction with the *Oracle Retail EFTLink Framework Installation Guide*.

Disambiguation

This FIPay implementation is for use with any compatible terminal that has AJB firmware installed, with communication based on a socket protocol.

EFTLink General

This document assumes static EFTLink configuration. When deploying with a POS that supports dynamic configuration, all property settings referred to below should be set on the POS, and not directly into local property files.

Minimum Version

The FIPay interface requires a minimum EFTLink version of 1.1.118

System Architecture

EFTLink connects directly to the terminal using a proprietary socket protocol.

> **Note:** This document does not cover installation of AJB software

Fileset

In addition to standard EFTLink files, FIPay uses:

- `FIPayCore.jar` – executable code for the FIPay EFTLink core.
- `fipay.properties` – configuration settings to specify which features are enabled and to define communication parameters for the interface with the EFT payment system.
- `Giftcard.properties` – Gftcard specific properties.
- `LangEN_FIPay.properties` – English language translation file, alternative translation maybe used by adding a new file and change the 2-character language code and translation text as appropriate.
- `AJBComm.jar` – API supplied by AJB to allow communication to the terminal.

> **Note:** If the POS supports dynamic configuration, properties can be set there instead of in `fipay.properties`.

Third Party

Note: Critically important

The following file is also needed, not supplied by Oracle: `AJBComm.jar`. This is an API supplied by AJB to allow communication to FIPay software. It should be placed in cores\FIPay alongside `FIPayCore.jar`.

Language

There are translation files in FIPayCore.jar. These should not need to be modified, but if a translation needs to be changed, they can be extracted to the base eftlink folder.

```
LangEN_FIPay.properties
LangFR_FIPay.properties
```

The language used will follow the language set in the EFTLink framework; see the *EFTLink Framework Installation Guide*, EFTLink General Information, Translation section.

```
EftlinkConfig.properties
DisplayLanguage = EN
```

Possible values are: EN, FR

Additional files could be added for other EFTLink supported languages for the small set of translations required by the FIPay core.

Core Classname

```
manito.eft.ajb.FIPayCore
```

This should be set as `EPSCore<x>` in `eftlinkconfig.properties`.

Configuration Settings

The full set of configuration properties is defined and commented in `fipay.properties`.

Key Settings

Settings that may be different for each POS/PED.

Setting	Description	Example
ip.address	Terminal address IP address	`ip.address = localhost`
store.number	The unique store number allocated by AJB.	`store.number = 100`

Secondary Settings

These settings are normally correct at their default values, but can be overridden if necessary.

Setting	Description	Default	Example
ip.port	IP port number	24900	`ip.port = 24900`
creditdebit.prompt	Credit/Debit prompt, controls whether to prompt operator for the card type (debit or credit), a specific terminal may have this built-in so this property maybe turned off (set to false).	true	`creditdebit.prompt = true`
response.timeout	FIPay response timeout, specify the number of seconds to wait for response from FIPay.	120	`response.timeout = 120`
pos.validate.swipe	Card validation prompt, controls whether to continue with the payment for this card. The prompt will display the card type.	false	`pos.validate.swipe = false`
electronic.signature	Enable electronic signature capture, if false signature prompt will appear after receipts are printed.	true	`electronic.signature = true`
enable.signature.logging	Enable logging of signature data (for debugging purposes ONLY). **Note**: This should be enabled for debugging purposes only. As soon as the debugging is complete, set back to false.	false	`electronic.signature.logging = false`
enable.emv.initialization	Enable emv transaction processing, when enabled, it will send an 'initDebit' command to FiPay at pos logon, an admin option is also available to allow adhoc initialization.	false	`enable.emv.initialization = false`
emvkeys.provided	EMV initialisation keys 'EMV_KEYS.DAT' which resides in FIPAYEPS, set to true if this file exists. Ignored if 'enable.emv.initialization' is false.	true	`emvkeys.provided = true`
enable.tokenization	Enable tokenization for refund.	false	`enable.tokenization = false`
currency.symbol	Currency symbol for customer display. If set to 'default', symbol base on operating system regional setting will be used.	$	`currency.symbol = $`
combine.receipt	Turn on/off POS combine receipt, default true	true	`combine.receipt = true`
combine.receipt.suppress.lines	When combine receipt is true, set which line to suppress.		`combine.receipt.suppress.lines = 1,2,3,4`

Setting	Description	Default	Example
tokenexpirydate	Specifies whether an expiry date will be included when performing tokenized refunds.	false	`tokenexpirydate = false`
combine.receipt.suppress.strings	When combine receipt is true, set what line to suppress when strings are matched		`combine.receipt.suppress.strings = DATE,DCC Not Available`
giftcard.handler	Giftcard handler, fully qualified class name		`giftcard.handler = manito.eft.ajb.giftcard.StandardFiPayGiftCardHandler`
giftcard.provider	Giftcard provider, fully qualified class name. Possible values are: manito.eft.ajb.giftcard.FiPayBlackhawk, manito.eft.ajb.giftcard.FiPaySVS, manito.eft.ajb.giftcard.FiPayGiveX, manito.eft.ajb.giftcard.FiPayInComm, manito.eft.ajb.giftcard.FiPayValueLink		`giftcard.provider =`

Supported Functions

The following operations are supported by this implementation of the AJB FIPay interface.

Function	Description
Payment	EFTLink sends payment requests to AJB FIPay. AJB will return a response message with formatted receipt strings for customer and/or merchant receipts. In an event of referral where authorization cannot be obtained online then a prompt for authorization code will appear; the authorization code must be obtained via telephone and entered here. If successful, appropriate receipts will be printed at the end of transaction.
Reversal	EFTLink sends reversal requests to AJB FIPay. This will reverse a transaction specified by the transaction number, found on the receipt, which must be captured by the POS and passed on to EFTLink.
Refund	EFTLink sends refund requests to AJB FIPay. This will refund a transaction with specified amount.
Reconciliation / Settlement	This is not supported directly by AJB FIPay via TCP/IP request; instead a batch script supplied by AJB must be used. This can be set up to run automatically at a specific time or on-demand at user's discretion.
Sale State Notifications	Sends line items through to the device so the customer display can be updated in line with the POS.

SVC Payment	Sends a Gift or Merchandise credit card payment request to the terminal. If there are not enough funds available, only the funds available will be deducted. The POS client will have to settle the transaction with another tender in this scenario.
SVC Activate	Sends a Gift or Merchandise credit card activation request to the terminal.
SVC Deactivate	Sends a Gift or Merchandise credit card deactivation request to the terminal. The account is disabled after this as the request is intended to be used for lost or stolen cards. It is not possible to use the card or account once this request has been issued and accepted.
SVC Add Value	Sends a Gift or Merchandise credit card add value request to the terminal. This will only add value to an account that has been activated.
SVC Balance Enquiry	Sends a Gift or Merchandise credit card balance enquiry request to the terminal.
SVC Unload (Cashout)	Sends a Gift or Merchandise credit card cash out request to the terminal. All funds are deducted from the account and the cash back amount is returned to the POS. The account is not deactivated as part of this process.

4

Banksys Core

General Information

This document covers EFTLink Integration with **Banksys VIC** Payment Systems. It should be read in conjunction with the *Oracle Retail EFTLink Framework Installation Guide*.

EFTLink General

This document assumes static EFTLink configuration. When deploying with a POS that supports dynamic configuration, all property settings referred to below should be set on the POS, and not directly into local property files.

Minimum Version

The **Banksys** interface requires:

- Aa minimum EFTLink version of 15.0.
- Java 1.6 or later.

System Architecture

EFTLink connects directly to the payment terminal using serial communication using the VIC protocol.

Fileset

In addition to standard EFTLink files:

- `cores/Banksys/banksysvic.jar` – executable code for the core
- `lib/RXTXcomm.jar` – Serial comms library.
- `lib/rxtxParallel.dll` – Windows DLL library for RXTXcomm.jar.
- `lib/rxtxSerial.dll` - Windows DLL library for RXTXcomm.jar.

Language

There are translation files in banksysvic.jar

```
EftlinkConfig.properties
DisplayLanguage = EN
```

Instead the translations selected will follow the value of a Banksys specific setting LANGUAGE, see secondary setting below. Possible languages are English, French and Dutch.

Core Classname

```
com.torexretail.eftlink.core.vic.VicCore
```

This should be set as `EPSCore0 = com.torexretail.eftlink.core.vic.VicCore` in `EftlinkConfig.properties` by installcore.bat or installcore.sh

Configuration Settings

The banksys core does not have a dedicated core property file, instead the properties are defined by entries in the framework property file, `EftlinkConfig.properties`. These entries are not present by default, so a readme.txt in the cores/Banksys folder contains commented examples of these entries, which can be pasted into `EftlinkConfig.properties`

Key Settings

Settings that may be different for each POS/PED.

Setting	Description	Default	Example
VIC_PORTNAME	Serial port name.		`VIC_PORTNAME = COM1`
LANGUAGE	Language code, options: en = English, fr = French, nl = Dutch	en	`LANGUAGE = en`
COUNTRY	Country code.	UK	`COUNTRY = UK`

Secondary Settings

Settings that are normally correct at their default values, but can be overridden if necessary.

Setting	Description	Default	Example
VIC_AUTOLOGON	Automatically perform a log-on at initialization.	false	`VIC_AUTOLOGON = false`
VIC_MANDATORY VOUCHER	If receipt data is not available, receipt voucher will be created using the data from the response.	false	`VIC_MANDATORYVOUCHER = false`
VIC_PRINTWIDTH	Receipt print width.	38 (characters)	`VIC_PRINTWIDTH = 38`

Supported Functions

The following operations are supported by this implementation of the Banksys interface:

Payment

Sends payment request to the Banksys terminal. The terminal will return a response message with formatted receipt strings for customer and/or merchant receipts.

If successful, appropriate receipts will be printed at the end of transaction.

5
Cayan Core

This Cayan implementation is for use with Genius terminals in the US, with communication based on a webservice protocol.

EFTLink General

See also the EFTLink general deployment guide if not already familiar with EFTLink.

This document assumes static EFTLink configuration. When deploying with a POS that supports dynamic configuration, all property settings referred to below should be set on the POS, and not directly into local property files.

Minimum Version

The Cayan interface requires a minimum EFTLink version 16

System Architecture

Cayan Genius is deployed as an intelligent terminal. EFTLink connects directly to the terminal using a proprietary web services protocol.

Genius 5.0 and later versions supports a HTTPS interface in addition to its traditional HTTP interface. Only the protocol scheme (https vs. http) and port (8443 vs 8000) differ. The cayancore can communicate with the Genius device using TLS to secure the connection. The terminal will generate appropriate certificates as required in order to serve the TLS connection, and all certificates generated by the terminal will be signed by the Cayan CA.

The cayan certificate is automatically stored upon startup in the file `cayan.public.jks`

To enable TLS in `cayan.properties`, change all the `http.action` entries containing http://cedIp:cedPort *into* https://cedIp:cedPort and set `ced.port=8443`

Fileset

In addition to standard EFTLink files:

- `cayancore.jar` – executable code for the Cayan EFTLink core.
- `cayanTA.crt` – cayan root certificate
- `cayan.properties` – configuration settings to specify which features are enabled and to define communication parameters for the interface with the terminal.
- `langEN_cayan.properties` – English translation file for the Cayan core
- `cayanruntime.properties` – core logging settings that are automatically reloaded at runtime (checked every 10 seconds)
- `cayandynamic.properties` – merchant specific details that can be accessed through the administration functions
- `cayan_receipt.properties` – links a receipt template file to a ReceiptType XML element
- `cayan_giftadd_receipt, cayan_giftbalance_receipt, cayan_payment_receipt, cayan_refund_receipt, cayan_reversal_receipt` – customer configurable receipt template files

Runtime files

- `cayan.public.jks` – keystore file containing the cayan root certificate to allow TLS communication
- `cayan.secure` – storage file for the random encryption key that is used to protect merchant information

Account Information Entry

At initial software startup, a keystore is created for encryption information and the Cayan certificate is placed into a second keystore. Account information is added to the EFTLink system via the EFTLink admin menus. Five parameters are required to be entered via the admin function:

- Account Name
- Account Software Key
- Site Identifier
- Account DBA
- Terminal Identifier

Both the Account Name and Account Software Key are automatically encrypted. All 5 parameters are held in the `cayandynamic.properties` file.

See the Administration Functions section below for entry of the parameters.

Account Information Re-Encryption

The password within the `cayandynamic.properties` file needs to be encrypted. To achieve this, the following steps must be followed:

Windows Operating Systems

To re-encrypt a password with new encryption settings; open a command prompt and change directory to eftlink's location.

- Type `encrypt.bat -g <keystore name> <properties> <certificate> <dyanamicProperties> {<Colon-Separated List of Properties>} <keygenType> <cipherType> <keySize> <iterations>`

  For example, `encrypt.bat -g cayan.secure cayan.properties cayan.public.jks cayandynamic.properties {merchant.name:merchant.key} AES AES 128 10000`
- Re-encryption uses existing crypto settings in the properties file to decrypt the password. Once the password is decrypted, a new keystore file is generated using the new crypto parameters specified at the command line and the new encrypted password / initialization vector is generated.
- When using AES algorithm with keysize is greater than 128, you may get **java.security.InvalidKeyException: Illegal key size or default parameters**. If so Additional Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files will need to be downloaded and extracted to ${java.home}/jre/lib/security/

Core Classname

`manito.eft.cayan.CayanCore`

This should be set as `EPSCore<x>` in `eftlinkconfig.properties`.

Configuration Settings

The full set of configuration properties is defined and commented in `cayan.properties`.

Key Settings

Settings that may be different for each POS/PED.

Setting	Description	Example
Terminal address	IP of Genius terminal.	`ced.ip =`
Simulator	Simulation mode.	`ced.simulator = false`
Receipt handling	Separate EFT receipts or EFT receipt as part of the regular POS receipt.	`EmbeddedReceipt = false`
Signature Verification	Enable/Disable signature verification dialog	`SignatureVerification = false`
Reversal Failure	Enable/Disable reversal failure dialog	`ReversalDialog = false`

Secondary Settings

These settings are normally correct at their default values, but can be overridden if necessary.

Setting	Description	Default	Example
Terminal address	Port number	8080 for http and 8443 for https.	`ced.port =`
Timeout	Overall response timeout in seconds	600	`ced.get.timeout =`
Status Timeout	Timeout period for checking status of device.	1	`ced.status.timeout`
LineItem Timeout	Timeout period for outputting a line item to the device.	1	`ced.item.timeout`
Signature display scaling	Signature display scaling	3	`SignatureScaling =`
Status Checks	Perform periodic status checks during a transaction.	false	`ced.statuschecks = false`
Status Check Ondemand	Perform status check at the end of transaction.	false	`ced.statuschecks.ondemand = false`
Auto Reversal	Not used	false	`ced.abortautoreversal = false`
statusMngr	Interval of periodic status checks when not in a transaction.	2	`ced.status.interval.inactive = 2`
Device Masking	Specifies whether sensitive fields are masked in the logfile.	true	`DeviceInputLogMasking = true`
Admin menu	Specifies the admin menu configuration.		`AdminMenu =`

Setting	Description	Default	Example
Maintenance Timeout	Timeout for maintenance menu.	60	`MaintenanceTimeout = 60`
Operator Response Timeout	Operator prompt timeout on POS.	60	`OperatorTimeout = 60`
Signature Scaling	Used to scale the signature from the CED for displaying on the POS	3	`SignatureScaling = 3`
sSgnature MaxY	Specifies the maximum size of the signature to be scaled.	100	`SignatureMaxY = 100`
Signature Verification	Determines whether the signature will be verified on the POS if returned from the device.	true	`SignatureVerification = true`
Receipt Handling	Embed the receipt in the card service response.	false	`EmbeddedReceipt = false`
Sale Receipt	Send sale receipt to POS for printing	true	`EmbeddedReceiptSale = true`
Gift Receipt	Send gift receipt to POS for printing	true	`EmbeddedReceiptGift = true`
Reversal Msg	Not USED – prompt for reversal on test system.	false	`ReversalDialog = false`
Status Interval	Interval of periodic status checks when in a transaction.	2	`ced.status.interval = 2`
	Not USED	`false`	`auto.report = false`

Administration Functions

The terminal has some administration/maintenance functions. These are normally invoked from a dedicated “EFT Maintenance” button, but if this is not available, they could be accessed by an engineer using the EFTLink built-in test harness.

EFTLink uses DeviceProxy messages to display input prompts on the POS to manage these functions.

Cayan will provide the merchant credentials that are required to setup the connection with the Cayan host. The information consists of five elements: Name, Key, SiteID, DBA and TerminalID

These credentials must be entered through the administration functions. The information is stored in the file `cayandynamic.properties`. The fields Name and Key are stored in an encrypted form. For each POS system, the cayancore will create a random encryption key to protect sensitive information. The encryption key itself is stored in the file `cayan.secure` using an EFTLink specific encryption algorithm.

Cayan has created an Oracle account for testing purposes. To connect to the Cayan host from non-US IP addresses, a ‘WhitelistRequest’ document containing the static IP of the Genius terminal must be sent to Cayan first. It typically takes 2-3 business days for Cayan security to review and then IT to process.

Operation	Description
Merchant Name	This operation allows the technician/cashier to enter the merchant name and store it encrypted in `cayandynamic.properties`.
Merchant Key	This operation allows the technician/cashier to enter the merchant key and store it encrypted in `cayandynamic.properties`.
Merchant Site ID	This operation allows the technician/cashier to enter the merchant site identifier and store it in `cayandynamic.properties`.
Merchant DBA	This operation allows the technician/cashier to enter the merchant dba and store it in `cayandynamic.properties`.
Merchant Terminal ID	This operation allows the technician/cashier to enter the merchant terminal identifier and store it in `cayandynamic.properties`.

Supported Functions

Below is a list of supported functionalities of the interface to Cayan.

Function	Description
Payment	Sends payment request to the terminal. Terminal will return a response message with receipt strings.
Reversal	Sends reversal request to the terminal. This will reverse a transaction specified by the transaction number, found on the receipt, which must be captured by the POS and pass on to EFTLink.
Refund	Sends refund request to the terminal. This will refund a transaction with specified amount.
Sale State Notifications	Sends line items through to the device so the customer display can be updated in line with the POS.
SVC Payment	Sends a Gift or Merchandise credit card payment request to the terminal. If there are not enough funds available, only the funds available will be deducted. The POS client will have to settle the transaction with another tender in this scenario.
SVC Activate	Sends a Gift or Merchandise credit card activation request to the terminal.
SVC Deactivate	Sends a Gift or Merchandise credit card deactivation request to the terminal. The account is disabled after this as the request is intended to be used for lost or stolen cards. It is not possible to use the card or account once this request has been issued and accepted.
SVC Add Value	Sends a Gift or Merchandise credit card add value request to the terminal. This will only add value to an account that has been activated.
SVC Balance Enquiry	Sends a Gift or Merchandise credit card balance enquiry request to the terminal.

SVC Unload (Cashout)	Sends a Gift or Merchandise credit card cash out request to the terminal. All funds are deducted from the account and the cash back amount is returned to the POS. The account is not deactivated as part of this process.

6

CCVPOS (CCV ITS)

This CCVPOS implementation is for use with payment terminals that support the CCV ITS interface, with communication based on a socket/XML protocol.

EFTLink General

This document assumes static EFTLink configuration. When deploying with a POS that supports dynamic configuration, all property settings referred to below should be set on the POS, and not directly into local property files. It should be read in conjunction with the *Oracle Retail EFTLink Framework Installation Guide*.

Minimum Version

The CCVPOS interface requires a minimum EFTLink version of v15.0.

System Architecture

CCV ITS is deployed as an intelligent terminal. EFTLink connects directly to the terminal via TCP/IP using a socket/XML protocol derived from the OPI/IFSF standard.

Fileset

In addition to standard EFTLink files, CCVPOS uses:

- `cores/ccvpos/ccvposcore.jar` – executable code for the CCVPOS EFTLink core.
- `CcvposConfig.properties` – configuration settings to specify which features are enabled and to define communication parameters for the interface with the EFT terminal.

Language

There are translation files in ccvposcore.jar, that should not need to be modified, but if a translation needs to be changed, they can be extracted to the base eftlink folder.

```
LangEN_CCV.properties
LangFR_CCV.properties
LangNL_CCV.properties
```

The language used will follow the language set in the EFTLink framework; see the *EFTLink Framework Installation Guide*, EFTLink General Information, Translation section

```
EftlinkConfig.properties
DisplayLanguage = EN
```

Possible values are: EN, FR, NL

Core Classname

The following should have been set in the `EftlinkConfig.properties` file by installcore.bat or installcore.sh

```
 EPSCore0 = manito.eft.ccvpos.CcvPosCore
```

Configuration Settings

Settings are defined in `pointus.properties`.

Key Settings

Settings that may be different for each POS/PED.

Setting	Description	Default	Example
CCVServerIP	IP address of the CCV terminal.		`CCVServerIP = 10.0.0.99`
CCVWorkstationId	Name by which the local POS is identified by the terminal and/or host.	POS01	`CCVWorkstationId = POS01`

Secondary Settings

These settings are normally correct at their default values, but can be overridden if necessary.

Setting	Description	Default	Example
CCVChannel1	TCP/IP port used by EFTLink to allow the CCV terminal to connect back to the POS..	4102	`CCVChannel1 = 4102`
CCVResponseTimeout	Time allowed in seconds for the transaction to complete at the terminal. This must be set longer that maximum terminal activity timeout, which is generally 5 minutes. This needs to be long enough to cover all customer interaction and host authorization.	330	`CCVResponseTimeout = 330`
EmbedCustomerText	Option to embed the customer EFT voucher within the POS receipt to save paper. **Note**: This goes against the CCV general rule of guaranteeing customer printout, so this feature should not be enabled without prior approval by CCV.	false	`EmbedCustomerText = false`
EmbedJournalText	Option to return the merchant EFT voucher to the POS as part of the payment response, rather than as a direct print request, so that it can be store in an Electronic Journal.	false	`EmbedJournalText = false`

Setting	Description	Default	Example

ManagementMenuX	Specifies which administration function to show at position 'n' of the menu and sets a label for it. See section on Administration Menu.		`ManagementMenu1 = Reprint Last Transaction, REPRINT_LAST_TICKET`
			`ManagementMenu2 = Reconciliation - shift totals, SHIFT_TOTALS`
			`ManagementMenu3 = Reconciliation - close shift, CLOSE_SHIFT`
			`ManagementMenu4 = List card types, GET_CARD_CIRCUITS`
			`ManagementMenu5 = Version, VERSION`
			`ManagementMenu6 = Cancel, CANCEL`

Supported EFT Operations

The following operations are supported by this implementation of the CCV ITS interface.

Payment / Refund

Payment and refund by credit/debit card.

Reversal / Void

Payment/refund transactions can be cancelled by request from the POS.

Reconciliation / Settlement

If required, EFT batch management and reporting can be managed via the Administration Menu. (See below).

Administration Menu

The CCV ITS interface requires some administration/maintenance operations. These are normally invoked from a dedicated "EFT Maintenance" button at the POS, though the content of the display screen then presented to the operator is controlled by EFTLink.

Reprint Last Transaction

Print a copy-receipt of the most recent transaction. This can be used to recover payment details in the event of a system failure.

List Card Types

Print a report of the card type supported by the terminal.

Reconciliation - Shift Totals

Print a non-closing shift/batch report.

Reconciliation - Close Shift

Close the current shift/batch.

7

Ingenico

This document section covers EFTLink Integration with Ingenico Payment Systems. It should be read in conjunction with the *Oracle Retail EFTLink Framework Installation Guide*.

EFTLink General

This document assumes static EFTLink configuration. When deploying with a POS that supports dynamic configuration, all property settings referred to below should be set on the POS, and not directly into local property files.

Minimum Version

The Ingenico interface requires a minimum EFTLink version of 15.0.

System Architecture

EFTLink connects to the payment system using a proprietary socket protocol. The Ingenico EPS runs as a software package called C3 installed on the POS PC.

Note: This document does not cover the installation of C3.

Fileset

The following files are used:

- `cores/Ingenico/ingenicoCore.jar`
- `ingenico.properties`

Language

There are translation files in injenicoCore.jar, that should not need to be modified, but if a translation needs to be changed, they can be extracted to the base eftlink folder.

```
LangEN_Ingenico.properties
LangES_Ingenico.properties
LangFR_Ingenico.properties
```

The language used will follow the language set in the EFTLink framework; see the *EFTLink Framework Installation Guide*, EFTLink General Information, Translation section

```
EftlinkConfig.properties
```

```
DisplayLanguage = EN
```

Possible values are: EN, ES, FR

EFTLink Configuration

The following line should have been set in the `EftlinkConfig.properties` file by running `installcore.bat` or `installcore.sh`.:

```
EPSCore0 = manito.eft.ingenico.IngenicoCore
```

Configuration Settings

The core is configured via properties contained in `the ingenico.properties` file, which should have been copied from the cores\Ingenico folder to the base eftlink folder by installcore.bat or installcore.sh.

The available settings are listed below.

Setting	Description	Default	Example
c3path	Path to where C3 is located, this enables C3 to be initiated automatically by EFTLink. If left blank then the user must ensure that C3 is already running before launching EFLink.		`c3path = C:/Program Files/Ingenico/C3Generic/bin/c3inet.exe`
ip.address	The IP address of C3, default 127.0.0.1.		`ip.address = 10.0.0.5`
ip.port	The IP port of C3.	9518	`ip.port = 9518`
confirmation_list	A list of strings separated by a comma [,] which determines whether a user acknowledgement is required. **Note**: This list may require modification before it can be used in a Spanish configured Ingenico system.		`confirmation_list = APPEL PHONIE,<>,VALIDEZ,Montant,PAN,FIN VAL,APPEL TELECOL`
comms_timeout	The timeout in seconds between EFTLink and C3.		`comms_timeout = 120`
totalisation_type	The report type when doing reconciliation – not currently used.		`totalisation_type = C`
default_currency	The currency being used, defined by using the ISO 4217 numerical code, for example Euro 978, US Dollar 840, Sterling 826.		`default_currency = 978`

Supported Functions

The following operations are supported by this implementation of the Ingenico C3 interface.

- Logon and logoff (at the beginning and end of a shift or trading period)
- Sale (2-stage)
- Refund
- Maintenance
- Gift card/private card
- Cheque
- Customer receipt re-print (via the maintenance menu)

8 Merchant Link

Disambiguation

This Merchant Link implementation is for use with any compatible terminal that has Merchant Link firmware installed, with communication based on a socket protocol.

EFTLink General

This document assumes static EFTLink configuration. When deploying with a POS that supports dynamic configuration, all property settings referred to below should be set on the POS, and not directly into local property files.

Minimum Version

The Merchant Link interface requires a minimum EFTLink version of 1.1.124.

System Architecture

Merchant Link is deployed as an intelligent terminal. EFTLink connects directly to the PoslynxMINI device which in turn connects to the Verifone MX925 device using a proprietary socket/XML protocol. The PoslynxMINI device acts as a message broker to the MX925 device. The MX925 device will need to know the PoslynxMINI device address. When setting the system up for the first time it is best to contact the Merchant Link technician who will be able to talk through the process.

Fileset

In addition to standard EFTLink files:

- `poslynxcore.jar` – executable code for the Merchant Link EFTLink core.
- `poslynx.properties` – configuration settings to specify which features are enabled and to define communication parameters for the interface with the terminal.
- `LangEN_Poslynx.properties` – English language translation file, alternative translation maybe used by adding a new file and change the 2-character language code and translation text as appropriate.

Note: If the POS supports dynamic configuration, properties can be set there instead of in `poslynx.properties`.

Core Classname

`manito.eft.poslynx.PoslynxCore`

This should be set as `EPSCore<x>` in `eftlinkconfig.properties`.

Configuration Settings

The full set of configuration properties is defined and commented in `poslynx.properties`.

Key Settings

Settings that may be different for each POS/PED.

The Core will connect to the PoslynxMINI device, this device is connected to the internet and the LAN and is accessible by Merchant Link. The device is DNS enabled and will have an address of the following format:

Setting	Description	Example
Terminal address	IP of Mx925 terminal	`TerminalIP =` **`xxxxxx`**`.poslynx.org` Where **xxxxxx** is the last 6 digits of the MAC address on the TL device.

Secondary Settings

Setting	Description	Default	Example
Terminal address	Port number	20100	`TerminalPort =`
Timeout	Overall response timeout in seconds	120	`ResponseTimeout =`
EmbeddedPrinting	Option to buffer customer printout generated during the transaction and then include it in the POS authorisation response so that it can be merged with the POS receipt line to form a single receipt/voucher.	false	`EmbeddedPrinting =`

Supported Functions

Below is a list of supported functionalities of the interface to Merchant Link. Many functionalities are provided by Merchant Link, such as Loyalty, Cashback etc. (please refer to interface specification for details) but are not implemented because of the business requirement.

Administration Functions

The terminal has some administration/maintenance functions. These are normally invoked from a dedicated “EFT Maintenance” button, but if this is not available, they could be accessed by an engineer using the EFTLink built-in test harness.

EFTLink uses DeviceProxy messages to display input prompts on the POS to manage these functions.

Function	Description
Terminal Connect Test	Sends a request to test the connection to the device. This Core builds an XML request with the “CONNECTIONTEST” command and should receive a response with an “APPROVED” result if successful.

Reset Pin Pad	Sends a request to reset the pin pad. This command will force the terminal to restart after acquiring Certificate Authority Public Key (CAPK) file from the host processor.
Signature Capture	Sends request to test the signature capture functionality on the device.
Batch Summary	Sends a batch summary request to the terminal. This provides a summary report of the current information for that batch of transactions prior to settlement. It should be considered as flash report typically referred to as an X read report.
Batch Report	Sends a batch card totals request to the terminal. This will return a report which will summarise the card type totals for the current batch prior to settlement.
Batch Close	Sends a batch close request to the terminal. This will initiate a batch close process on the host processor which triggers transactions to be submitted for financial settlement. This returns a day end report.
Payment	Sends payment request to the terminal. Terminal will return a response message with unformatted receipt strings for customer and/or merchant receipts. In an event of referral where authorization cannot be obtained online then a prompt for authorization code will appear; authorization code must be obtained via telephone and entered here. If successful, appropriate receipts will be printed at the end of transaction.
Reversal	Sends reversal request to the terminal. This will reverse a transaction specified by the transaction number, found on the receipt, which must be captured by the POS and pass on to EFTLink.
Refund	Sends refund request to the terminal. This will refund a transaction with specified amount.
Sale State Notifications	Sends line items through to the device so the customer display can be updated in line with the POS.
SVC Payment	Sends a Gift or Merchandise credit card payment request to the terminal. If there are not enough funds available, only the funds available will be deducted. The POS client will have to settle the transaction with another tender in this scenario.
SVC Activate	Sends a Gift or Merchandise credit card activation request to the terminal.
SVC Deactivate	Sends a Gift or Merchandise credit card deactivation request to the terminal. The account is disabled after this as the request is intended to be used for lost or stolen cards. It is not possible to use the card or account once this request has been issued and accepted.
SVC Add Value	Sends a Gift or Merchandise credit card add value request to the terminal. This will only add value to an account that has been activated.

SVC Balance Enquiry	Sends a Gift or Merchandise credit card balance enquiry request to the terminal.
SVC UnlOad (Cashout)	Sends a Gift or Merchandise credit card cash out request to the terminal. All funds are deducted from the account and the cash back amount is returned to the POS. The account is not deactivated as part of this process.

9

SixPay

General Information

This section of the document covers EFTLink Integration with SixPay Payment Systems. It should be read in conjunction with the *Oracle Retail EFTLink Framework Installation Guide*.

EFTLink General

This document assumes static EFTLink configuration. When deploying with a POS that supports dynamic configuration, all property settings referred to below should be set on the POS, and not directly into local property files.

System Architecture

Six Payment Services MPD is deployed as a store server application to manage the connection to the authorization host and to handle all the local PEDs. PEDs use IP, so must be connected to the LAN. EFTLink connects to the store server, not directly to any PED. EFTLink communicates with MPD using an implementation of the IFSF/OPI protocol.

Note: This document does not cover the installation of MPD.

Fileset

In addition to standard EFTLink files the following are used:

- `Cores/SixPay/sixpaycore.jar` – executable code for the MPD OPI interface
- `sixpay.properties` – configuration settings to specify which features are enabled and to define communication parameters for the interface with the store server.

Note: If the POS supports dynamic configuration, properties can be set there instead of in `sixpay.properties`.

Language

There are translation files in sixpaycore.jar, that should not need to be modified, but if a translation needs to be changed, they can be extracted to the base eftlink folder.

```
LangDE_Sixpay.properties
LangEN_Sixpay.properties
LangFR_Sixpay.properties
LangIT_Sixpay.properties
LangNL_Sixpay.properties
```

The language used will follow the language set in the EFTLink framework; see the *EFTLink Framework Installation Guide*, EFTLink General Information, Translation section

```
EftlinkConfig.properties
```

```
DisplayLanguage = EN
```

Possible values for SixPay are: DE, EN, FR, IT, NL

Core Classname

This should have been set as `EPSCore0= manito.eft.sixpay.SixpayMPDOPIClient` in eftlinkconfig.properties by `installcore.bat` or `installcore.sh`

Configuration Settings

Configuration settings are made in `sixpay.properties`, which would have been copied from cores/SixPay to the base eftlink folder by `installcore.bat` or `installcore.sh`

Key Configuration Settings

Setting	Description	Default	Example
SixpayServerIP	IP address of the store server running MPD	127.0.0.1	`SixpayServerIP = 10.0.0.50`
SixpayWorkstationID	Optional Setting for specific WorkstationID, and to set the WorkstationID format. **Note:** This becomes the base number when SixpayWorkstationIDPosBased is enabled. The default is for this not to be set (property is commented) – the workstation number will be taken directly from the OPI message from the POS.		`SixpayWorkstationID = POS1`
SixpayWorkstationIDPosBased	Option to automatically set the MPD workstation ID from the numeric suffix of a mixed numeric/ non-numeric POS workstation ID. Boolean. If this feature is enabled, the SixpayWorkstationID setting is taken as the value for POS #1 and the numerical component is incremented for all other POSs.	false	`WorkstationIDPosBased = true` This would mean that for POS2 with the `SixpayWorkstationID = POS1` set above, messages to MPD would be from POS2. Careful use of WorkstationID settings and overrides in both the POS and EFTLink should make it possible to deploy a standard `sixpay.properties` file across all POSs.

Optional Configuration Settings

These settings are normally left on defaults.

Setting	Description	Default	Example
SixpayChannel0	TCP/IP port used for primary channel to MPD	20002	`SixpayChannel0 = 20002`
SixpayChannel1	TCP/IP port for device requests from MPD		`SixpayChannel1 = 20007`

Setting	Description	Default	Example
SixpayResponseTimeout	Timeout in seconds for EFTLink to wait for the response from MDP.	300	`SixpayResponseTimeout = 300`
IncludeSaleItems	If enabled, sale item details are included in the payment request.	false	`IncludeSaleItems = true`
EmbeddedPrinting	Whether customer printout is to be buffered and included in the POS authorization response such that it can be embedded in the POS receipt.	false	`EmbeddedPrinting = false`
ElectronicJournal	Whether merchant printout (other than signature slips) is buffered and included in the POS authorization response such that it can be stored in an electronic journal.	false	`ElectronicJournal = false`
SignatureCheckTag	Trigger tag/text to detect that a signature has been asked for and should be checked, default "Signature:"		`SignatureCheckTag=sign`
SignatureCheckTimeout	Timeout for Signature OK? Question.	30 seconds	`SignatureCheckTimeout = 30`

Fixed Configuration Settings

The property file `sixpay.properties` has a section of settings headed as Fixed Configuration settings, which should not be changed.

Other Information

PED Identification/Selection

The PED is identified to MPD by the `WorkstationID` in the IFSF/OPI message. By default, this is copied through from the `WorkstationID` in the POS-EFTLink message. Thus, the POS numbering needs to be kept in sync with the PED configuration in MPD. If this is not possible, or if the POS uses non-numeric `WorkstationID`, override settings must be used in the `sixpay.properties` files as described above.

10

SolveConnect Core

General Information

This document covers EFTLink Integration with TLG (The Logic Group) Payment Systems. It should be read in conjunction with the *Oracle Retail EFTLink Framework Installation Guide*.

EFTLink General

This document assumes static EFTLink configuration. When deploying with a POS that supports dynamic configuration, all property settings referred to below should be set on the POS, and not directly into local property files.

System Architecture

EFTLink connects directly to the SolveConnect software usually installed on the same PC as the POS, using a proprietary socket protocol.

> **Note**: This document does not cover the installation of SolveConnect software.

Fileset

In addition to standard EFTLink files the following are used:

- `Core/SolveConnect/SolveConnect.jar` – Core interface to TLG's SolveConnect software.
- `SolveConnect.POS.properties`
- `SolveConnect.properties`

Language

There are translation files in SolveConnect.jar, that should not need to be modified, but if a translation needs to be changed, they can be extracted to the base eftlink folder.

```
LangEN_SolveConnect.properties
LangES_SolveConnect.properties
```

The language used will follow the language set in the EFTLink framework; see the *EFTLink Framework Installation Guide*, EFTLink General Information, Translation section

```
EftlinkConfig.properties
DisplayLanguage = EN
```

Possible values are: EN, ES

Core Classname

This should have been set as `EPSCore0= manito.eft.solveconnect.SolveConnectCore` in `EftlinkConfig.properties`. by `installcore.bat` or `installcore.sh`

Configuration Settings

There are two configuration files – `SolveConnectPOS.properties` and `SolveConnect.properties`. These are copied from `cores/SolveConnect` to the base eftlink folder by installcore.bat or installcore.sh.

`SolveConnectPOS.properties` carries only the POS specific identifiers, `SolveConnect.properties` carries everything else, and can usually be deployed on a retailer's estate without other changes.

Key Settings

`SolveConnectPOS.properties`

Setting	Description	Default	Example
SourceID	The POS specific identifier, allocated by retailer, to be unique across the retailer's estate.		`SourceID = DPOS0001`
Store.ID	A 4-digit store identifier which forms part of the reference number assigned to each transaction.	9999	`Store.ID = 1234`
POS.ID	2 digit POS identifier which forms part of the reference number assigned to each transaction.	99	`POS.ID = 25`

Note: Together, the Store.ID and POS.ID settings can be used to create a transaction reference that will be unique across all sites in a group.

Secondary Configuration Settings

`SolveConnect.properties`

Setting	Description	Default	Example
ServiceHost	Hostname or IP address of SolveConnect service.		`ServiceHost = 127.0.0.1`
TransactionTimeoutPeriod	Number of seconds to allow a transaction to complete.	180	`TransactionTimeoutPeriod = 180`
CancellationTimeout period	Maximum number of seconds the core will wait for a transaction response following a cancellation.	30	`CancellationTimeoutPeriod = 30`
MaintenanceMenuTimeout	The number of seconds to wait for an option to be selected before dismissing the Maintenance menu.	30	`MaintenanceMenuTimeout = 45`
AuditLoggingEnabled	Enable/Disable logging of transaction results to an audit log.	false	`AuditLoggingEnabled = false`

Setting	Description	Default	Example
TransactionReferenceScheme	The format and source of Store and Till-ID values. Recognised values are Properties and PowerPOS. If set to PowerPOS, the POS.ID value will be automatically extracted from the POS system name at run time, so the setting in `SolveConnect.POS.properties` can be left at zero.	Properties	`TransactionReferenceScheme = PowerPOS`
TransactionNumberFromPOS	Whether to use the transaction number from the POS (with suffixes to ensure uniqueness) rather than the default auto-incrementing number.	true	`TransactionNumberFromPOS = true`
ForcePurchaseWithCashback	Force all POS Purchase requests to be converted to Solve Purchase with Cashback requests.	true	`ForcePurchaseWithCashback = true`
PromptForCashbackCharge	Prompt for a cashback charge.	true	`PromptForCashbackCharge = true`
TransactionReferenceFormat	Format for the transaction reference to be passed to SolveConnect. Built from the store id (S), POS ID (P) and POS Transaction number (T).	SSSSPPTTTTTT	`TransactionReferenceFormat = SSSSPPTTTTTT`
EmbeddedPrinting	Whether customer printout is to be buffered and included in the POS authorization response such that it can be embedded in the POS receipt.	false	`EmbeddedPrinting = false`

DCC Keywords	DCC keywords for extracting DCC from status message. There are no defaults.		`DCCAmountKeyword = DCC Amount`
			`DCCExchangeRateKeyword = Exchange Rate`
			`DCCMarginKeyword = Margin`

AuthTokenOrigin	Whether to automatic token recognition to establish local/central origin.	false	`AuthTokenOrigin = false`

Token Formats	Token formats to identify local/central token. There are no defaults.		`LocalTokenFormat = 1234567890123456789`
			`CentralTokenFormat = 123456ABCDEFGHI1234`

CardSwipeTimeoutPeriod	Number of seconds to allow for a standalone card read/swipe to complete. This will need to be extended e.g. to 9999 if an open/background card read operation is required.	30	`CardSwipeTimeoutPeriod = 30`

Setting	Description	Default	Example
PEDLogoffDelayTime	Delay time between POS logoff and PED logoff, to allow for operator changeover without PED disconnection. Applies to networked PEDs only. Time in seconds. Set to 0 to disable PED logoff.	300	`PEDLogoffDelayTime = 300`
SelectiveMerchantPrint	Determine whether merchant print is selective. I.e. enable for some conditions disabled for others.	false	`SelectiveMerchantPrint = false`
MercantPrint.not_present.not_performed	In selective mode, all merchant print is disabled by default, but can be selectively re-enabled based on a combination of the transaction attributes returned by SolveConnect. Note - this is the opposite way round to selective customer print. The attributes used are: TRANSACTION:customer present, not_present, internet CARDHOLDER_RESULT:verification pin, signature, pin_and_signature, on_device, not_performed, failed, unknown These attributes are formed into a dot-separated property name (e.g. MerchantPrint.present.pin) that can be set to "true" to reenable merchant print for that attribute combination. Note - merchant print requiring signature will always be printed, it cannot be disabled. e.g. to re-enable merchant print for CustomerNotPresent transactions: `MerchantPrint.not_present.not_performed = true` Re-enable merchant print for CustomerNotPresent transactions.		`MerchantPrint.not_present.not_performed = true`
SelectiveCustomerPrint	Determine whether customer print is selective i.e. enabled for some conditions disabled for others.	false	`SelectiveCustomerPrint = false`

Setting	Description	Default	Example
CustomerPrint.not_present.not_performed	In selective mode, customer print is enabled by default, but it can be selectively disabled based on a combination of the transaction attributes returned by SolveConnect. Note - this is the opposite way round to selective merchant print. The attributes used are: TRANSACTION:customer present, not_present, internet CARDHOLDER_RESULT:verification pin, signature, pin_and_signature, on_device, not_performed, failed, unknown These attributes are formed into a dot-separated property name (e.g. CustomerPrint.present.pin) that can be set to "false" to disable customer print for that attribute combination. e.g. to disable customer print for CustomerNotPresent transactions: `CustomerPrint.not_present.not_performed = true`		`CustomerPrint.not_present.not_performed = true`
SignatureCheckReprintOption	Determine whether to include a “reprint” option when prompting operator for signature verification. Default false. Caution – if set true, the display request will be sent as a menu selection rather than a yes/no and this will affect the way it is presented to the operator.		`SignatureCheckReprintOption = true`
ManualAuthMinLength	Minimum input length required for Manual/Voice referral authorization code response.	0	`ManualAuthMinLength = 0`

Fixed Configuration Settings

There are a number of fixed configuration settings in `SolveConnect.properties` that are commented in the property file. These are advanced options for development use.

Supported Functions

The following operations are supported by this implementation of the SolveConnect interface.

Function	Description
Payment	Sends payment request to the terminal. Terminal will return a response message with formatted receipt strings for customer and/or merchant receipts. In an event of referral where authorization cannot be obtained online then a prompt for authorization code will appear; authorization code must be obtained via telephone and entered here. If successful, appropriate receipts will be printed at the end of transaction.
Reversal	Sends reversal request to the terminal. This will reverse a transaction specified by the transaction number, found on the receipt, which must be captured by the POS and passed on to EFTLink.
Refund	Sends refund request to the terminal. This will refund a transaction with specified amount.
GiftCard	Sends giftcard payment request to the terminal. Specified amount will be deducted from the giftcard. Administration options to add balance and check balance is also supported.
Receipt Reprint	Reprint merchant/customer receipt.

11 TransaxEFT

EFTLink General

This document assumes static EFTLink configuration. When deploying with a POS that supports dynamic configuration, all property settings referred to below should be set on the POS, and not directly into local property files. It should be read in conjunction with the *Oracle Retail EFTLink Framework Installation Guide*.

Minimum Version

The Transax interface requires a minimum EFTLink version of v15.0.

System Architecture

EFTLink connects to FIS TransaxEFT software running on the same PC as the POS via TCP/IP sockets as an OPI client.

> **Note**: This document does not cover installation of FIS TransaxEFT software.

Fileset

In addition to standard EFTLink files, TransaxEFT uses:

- `cores/TransaxEFT/transaxeftcore.jar` – executable code for the TransaxEFT core.
- `TransaxEFT.properties` – configuration settings to specify which features are enabled and to define communication parameters for the interface with the EFT payment system.

Translation

There is an English translation file in transaxeftcore.jar, that should not need to be modified, but if a translation needs to be changed, it can be extracted to the base eftlink folder.

LangEN_TransaxEFT.properties

The file in use follows the language setting for EFTLink itself, defined in `EftlinkConfig.properties`, so the only possible setting is EN; the default.

Example

```
DisplayLanguage = EN
```

Core Classname

The following should have been set in the `EftlinkConfig.properties` file by installcore.bat or installcore.sh

```
 EPSCore0 = manito.eft.transaxeft.TransaxEFTOPIClient
```

Configuration Settings

Settings are defined in `TransaxEFT.properties`.

Key Settings

These are no settings that must be set differently per POS.

Secondary Settings

These settings should not need adjustment, but are defined here.

Setting	Description	Default	Example
TransaxEFTChannel 0	The TCP port on which the Core sends requests and device responses to TransaxEFT.	8900	`TransaxEFTChannel0 = 8900`
TransaxEFTChannel 1	The TCP port on which the Core listens for responses and device requests from TransaxEFT.	9900	`TransaxEFTChannel1 = 9900`
ReceiptFormatFile	Name and path of the receipt XSLT translation file used to format the receipt text.	`transaxeft\\AccreditationReceipt.xsl`	`ReceiptFormatFile = transaxeft\\AccreditationReceipt.xsl`
BalanceEnquiryFormatFile	Name and path of the card balance enquiry XSLT translation file.	`transaxeft\\Balance.xsl`	`BalanceEnquiryFormatFile = transaxeft\\Balance.xsl`
ReconciliationFormatFile	Name and path of the reconciliation report XSLT translation file.	`transaxeft\\Reconciliation.xsl`	`ReconciliationFormatFile = transaxeft\\Reconciliation.xsl`
ReceiptTextPassThroughEnabled	Only enable when TransaxEFT provides pre-formatted plain text receipt lines rather than name/value pairs.	false	`ReceiptTextPassThroughEnabled = false`
TransaxEFTResponseTimeout	Extend the period we will wait for a CardServiceResponse message after sending our request.	120	`TransaxEFTResponseTimeout = 120`
TransaxEFTOperatorRecoverySupported	Do not allow the operator to specify the success or failure of a transaction.	false	`TransaxEFTOperatorRecoverySupported = false`
MaintenanceTimeout	Specifies the maximum number of seconds to wait for the operator to select an administration menu option. If no option is selected the maintenance function is completed and control passes back to the POS.	60	`MaintenanceTimeout = 60`
MaintMenuOptReceiptReprintEnabled	Enable/disable the maintenance menu option TXT_REPRINT. Set to false if, besides the maintenance menu, the POS has its own means of requesting a ticket reprint. Set to true for Power POS.	true	`MaintMenuOptReceiptReprintEnabled = false`

Setting	Description	Default	Example
MaintMenuOptEODEnabled	Enable/disable the TXT_RECONCILIATION_WITH_CLOSURE menu option. Set to false if the POS has its own means of requesting reconciliation with closure.	true	`MaintMenuOptEODEnabled = true`
MaintMenuOptPEDTestEnabled	Enable/disable the menu option TXT_PED_TEST. Set to false for Power POS as it cannot display multiple lines of text received in cashier display and cashier input device requests.	false	`MaintMenuOptPEDTestEnabled = false`
MaintMenuOptPEDTestPrintedEnabled	Enable/disable the menu option TXT_PED_TEST_PRINTER_OUTPUT. Set to true for Power POS.	true	`MaintMenuOptPEDTestPrintedEnabled = true`
MaintMenuOptEODQueryEnabled	Enable/disable the menu option TXT_QUERY_LAST_RECONCILIATION. Set to false for Power POS as it cannot display multiple lines of text received in cashier display and cashier input device requests.	false	`MaintMenuOptEODQueryEnabled = false`
MaintMenuOptEODQueryPrintedEnabled	Enable/disable the menu option TXT_QUERY_LAST_RECONCILIATION_PRINTER_OUTPUT.	true	`MaintMenuOptEODQueryPrintedEnabled = true`

Fixed Settings

There are a small number of settings included in `TransaxEFT.properties` that should not be changed. All settings not documented here fall into that category.

Supported Functions

Below is a list of supported functionalities of the interface to Transax.

Function	Description
Payment	EFTLink sends payment requests to Transax. Transax will return a response message with formatted receipt strings for customer and/or merchant receipts. If successful, appropriate receipts will be printed at the end of transaction.
Refund	EFTLink sends refund requests to Transax. This will refund a transaction with specified amount.
Reconciliation / Settlement	Prints a settlement report at day end.
Receipt Reprint	Reprint merchant/customer receipt.
Loyalty Balance Enquiry	Check the balance of a loyalty card.

Maintenance Options

The Transax interface has some administration/maintenance operations. These are normally invoked from a dedicated “EFT Maintenance” button at the POS, though the content of the display screen then presented to the operator is controlled by MaintMenu options enabled above.

- Reprint last receipts
- Reconciliation with Closure (EFT EOD)
- Test PED (Display results)
- Test PED (Print results)
- Test connection to Authorisation Host
- Test printer
- Query reconciliation figures (Display results)
- Query reconciliation figures (Print results)

12

VeriFone Ocius Sentinel

General Information

Overview

This document covers EFTLink Integration with Ocius Sentinel Payment Systems. It should be read in conjunction with the *Oracle Retail EFTLink Framework Installation Guide*.

System Architecture

EFTLink connects to the Ocius Sentinel application using a proprietary socket protocol. Normally the Ocius Sentinel application, which is configured to run screenlessly, is installed on the same PC as the POS application.

> **Note**: This document does not cover the installation of the Ocius Sentinel application itself.

Fileset

In addition to standard EFTLink files:

- `Cores/OciusSentinel/ociussentinelcore.jar`
- `ocius.properties`
- `ocius_receipt.properties` (only if using XML receipt data, can be auto-deployed, see XML Receipts).
- receipt template files (only if using XML receipt data, can be auto-deployed , see XML receipts).

Language

There are no translation files in ociussentinelcore.jar

Ocius Sentinel is deployed in the UK, so the language set in the EFTLink framework should be English, which is the default.

See the *Oracle Retail EFTLink Framework Installation Guide*, EFTLink General Information section, Translation sub-section.

```
EftlinkConfig.properties
DisplayLanguage = EN
```

Core Classname

The following should have been set in `eftlinkconfig.properties` by installcore.bat or installcore.sh

```
EPSCore0 = manito.eft.ocius_sentinel.OciusSentinelCore
```

Configuration Settings

The core is configured via properties contained in the `ocius.properties` file, which is copied from cores/OciusSentinel folder to the root eftlink folder by `installcore.bat` or `installcore.sh`.

Key Settings

These must be set. Since these two properties must be encrypted by default, see Password Encryption

Setting	Description	Example
user.id	The user ID to send to the terminal when logging on. The ID is allocated by the Ocius Sentinel, and needs to be encrypted for default configuration.	`user.id=89eb96f2dfed02384e99fb7f8bfea610`
user.pin	The user PIN to send to the terminal when logging on. The PIN is allocated by the Ocius Sentinel, and needs to be encrypted for default configuration.	`user.pin=89eb96f2dfed02384e99fb7f8bfea610`

Optional Configuration Settings

There are a large number of optional settings that usually do not need to be set or modified, but for completeness they are defined here. In the property file all are commented with default values or empty.

Setting	Description	Default
ip.address	The IP address of the Ocius Sentinel software. The default is 127.0.0.1, which will work as long as the Ocius Sentinel software is installed on the POS PC.	127.0.0.1
ip.port	The IP port of the terminal.	25000
terminal.menu.configuration	The menu configuration to send to the terminal when logging on. The default is * which enables all menus. See the Ocius Manual for more details.	*
account.id	The account ID to send with each transaction. This option is used in some deployments, and Verifone would indicate the value to use.	blank
auto.logon	If this is set true then the core will log on to the terminal automatically when it receives a transaction (if the POS has not already sent a logon command).	true
pause.before.auto.logon	The number of milliseconds to wait before issuing an automatic logon command to Sentinel. This is to allow for an issue with Sentinel which causes it to occasionally reject or lose messages which are sent too soon after a previous communication.	1000
auto.logon.pause	The number of milliseconds to wait after an auto logon before sending a transaction. The pause should be for several seconds.	

Setting	Description	Default
merchant.receipt.path	The folder where Ocius Sentinel is to place the merchant receipt. If undefined (commented or blank value) the file would be expected at the root of the same drive, which is where Ocius Sentinel puts the receipt by default.	
merchant.receipt.filename	The name that Ocius Sentinel will use for the merchant receipt. Default is Receipt1.txt, it can be modified in the Ocius Sentinel application, and if so the name used should be entered here.	Receipt1.txt
customer.receipt.path	The folder where Ocius Sentinel is to place the customer receipt. This is only relevant if xml. If undefined (commented or blank value) the file would be expected at the root of the same drive, which is where Ocius Sentinel puts the receipt by default.	
customer.receipt.filename	The name that Ocius Sentinel is to use for the customer receipt. Default is Receipt2.txt. This can be modified in the Ocius Sentinel application, and if so, the name used must be entered here.	Receipt2.txt
report.path	The folder where Ocius Sentinel is to place the report file.	
report.filename	The name that Ocius Sentinel is to use for the report file.	
progress.ip.port	The port that the core listens on for status messages from Ocius Sentinel.	25001
tear.merchant.receipt.text	The text to be displayed at the POS when prompting the operator to remove the merchant receipt from the printer.	
tear.customer.receipt.text	The text to be displayed at the POS when prompting the operator to remove the customer receipt from the printer.	
strip.receipt.carriage.returns	Ocius Sentinel delivers receipts with lines terminated by both carriage return and linefeed characters. If this option is set true then the carriage return characters will be removed.	false
max.cashback.length	The maximum length permitted for a cashback amount.	5
duplicate.receipt.title	An extra title to add to the top of a receipt which is reprinted in response to the “Re-print/Continue” message.	`*** Duplicate Receipt ***\n` where the \n indicates a linefeed. Leave blank to suppress this title.
suppress.merchant.receipt	Whether to suppress printing of the merchant receipt so only a customer copy is provided.	false
offer.reprint	Whether to display the “Re-print/Continue” dialogue after printing a receipt.	true

Setting	Description	Default
defer.customer.receipt	If true this will cause the customer receipt to be sent as part of the final CardServiceResponse when payment processing is complete.	false
account.on.file.mode	This may be set to an integer from 0 to 4 inclusive. Values are defined in the Ocius Sentinel integration guide v1.5 as follows: 0 – Not Set 1 – Do Not Register (the default) 2 – Register 3 – Register Only 4 – Register, decline transaction if registration fails.	
card.read.mode	This may be set to 0, 1 or 2 and defines what type of card is to be read when the core receives a card read request: 0 - Non EFT card 1 – EFT card 2 – Automatic based on the EFTLink background flag set by the POS, background=true reads a non-EFT card, otherwise an EFT card is expected (this is the default behaviour for this setting).	
remove.card.after.read	If true this should cause Ocius Sentinel to prompt for the card to be removed after a card read. In practice it has been found that Sentinel ignores this setting.	
encrypted.passwords	user.id, user.pin, account.id and transax.account.id must be encrypted using the encryption utility see Password Encrption.	
auto.confirm.licence.key	If true (the default), then there will be an automatic response to the `LicenceDetailConfirmation` status from Ocius Sentinel.	true
card.wait.mode	If true the core will send CARDWAIT records, otherwise it will operate in standard mode.	false
wait.record.header	This is the header text to display on the PED when it prompts for the card details to be presented.	The default is for the section to be left blank.
wait.record.body	This is the body text to display on the PED when it prompts for the card details to be presented.	The default is for the section to be left blank.
wait.record.footer	This is the footer text to display on the PED when it prompts for the card details to be presented.	The default is for the section to be left blank.
wait.record.timeout	This is the time in seconds for the PED to wait for the card details to be presented.	0 (no timeout)

wait.record.capture.methods	This is a hex bitmap of the capture methods that the PED is to allow.
	The hex bitmap is comprised of the following hex values:
	`Keyed = 01`
	`Swipe = 02`
	`ICC = 04`
	`Reserved = 08`
		The default is for the core to leave this blank, in which case Sentinel will apply the following default:
		`ICC + Swipe + Keyed = 07`

Setting	Description	Default

wait.record.fallback.methods	This is a hex bitmap of the fallback methods that the PED is to allow.
	The hex bitmap is comprised of the following hex values:
	`Fallback from ICC to Swipe = 01`
	`Fallback from Swipe to Key = 02`
		The default is for the core to leave this blank, in which case Sentinel will apply the following default:
		`Fallback from ICC to Swipe + Fallback from Swipe to Key = 03`

auto.offline	If true the core will automatically instruct Ocius Sentinel to work offline if the remote server is unavailable.	false
reference	This setting configures the customer reference generated by the core. It may contain any text except commas but the following case-sensitive keywords will be substituted with corresponding data: Date: the transaction date provided by the POS in the form YYMMDD Time: the transaction time provided by the POS in the form HHMMSS Transnum: the transaction number provided by the POS User: the operator ID provided by the POS when it logged on to EFTLink Pos: the POS ID provided by the POS when it logged on to EFTLink	`date transnum user pos`
CARDWAIT with CNP	If a card swipe request is issued with the CNP flag set then an alternative set of wait record parameters will be sent to the PED. These have the same names as the wait record properties already defined but with `.cnp` appended, for example: `wait.record.capture.methods.cnp = 1` The primary purpose of this is to allow the PED to be forced into keyed only mode in a customer-not-present (telesales) scenario. The definitions and default settings for the alternative parameters are the same as the standard parameters.	
simple.cnp.enabled	For telesales if a card has been keyed via a previous card swipe and customer address capture is not required as part of the subsequent transaction then this setting should be set true. **Note:** In this mode <CNP>true</CNP> is added to the XML receipt data for telesales.	`false`
transax.account.id	The account ID to use for Transax transactions.	
transax.types.requiring.card	The Transax transaction types which require card entry at the PED. This may be any combination of the letters A, B, M, O or P without spaces or separators.	`P`

Setting	Description	Default

transax.declined.operator.message	If a Transax payment is declined or otherwise fails this optional setting can be used to provide an acknowledgeable message to bring the failure to the attention of the operator. The default value is blank (no message will be displayed). If required the value may be static text. For example:
	`transax.declined.operator.message=Transax Payment Void`
	or it may be used to display one of the fields of a Transax XML receipt. For example:.
	`transax.declined.operator.message=<Message>`	

auto.confirm.auth.code	If this is set true then Ocius Sentinel status 20 (Confirm Auth Code) will be answered automatically.	
voice.referral.amount.text	This defines the label shown against the transaction amount in the voice referral prompt. If the POS already displays the amount elsewhere on the screen then `voice.referral.amount.text` may be set to blank to exclude it from the message sent by the core.	
voice.referral.compact.dialogue	If true the two stage referral dialogue where the operator must first confirm that the authorization has been accepted before entering the authorization code will be reduced to a single dialogue where the operator may immediately enter an authorization code or blank to cancel.	
signature.verification.reprint.option	By default the signature verification dialogue offers two options to confirm or reject the signature. If this setting has a value a third option will be displayed which will cause the signature slip to be reprinted. The value should be the text to be displayed, for example `Reprint`. The default is blank which disables this option. **Note**: `offer.reprint` provides a more general purpose reprint mechanism.	
defer.void.receipts	If true then void customer receipts will not be printed immediately but will be embedded in the final response from the core. Applies only in XML mode.	`false`
suppress.final.declined.message	If the POS displays its own declined message on receiving a payment failure response from the core then this setting may be used to suppress any similar display message from the core.	`false`
suppress.cnp.signature.receipt	If true then the signature receipt will be suppressed for telesales transactions when `simple.cnp.enabled` is true. Applies only for XML based receipts.	`true`
auto.translate.status.messages	Indicates whether the core should translate status messages according to the recommendations in the Ocius Sentinel Integration Guide. If false then status messages can still be translated.	`false`
space.out.status.messages	Indicates whether status text from Ocius Sentinel should be spaced out for display, for example `ExpiryDateRequired` becomes `Expiry Date Required`.	`true`

Setting	Description	Default
ped.unavailable.retry.pause	If status message 55 (PEDUnavailable) is received this setting specifies the number of milliseconds to wait before requesting Ocius Sentinel to retry.	The default is 0 (zero) which disables handling of status message 55.
legacy.printing	Enables file-based printing if set to true, otherwise socket-based printing will be used.	`true`
cancel.card.wait.delay	When `card.wait.mode=true` this setting defines the minimum interval in milliseconds between a card swipe request from the POS and a cancellation of the card swipe (abort). This is to allow for a limitation in Ocius Sentinel which cannot cope with the two messages being sent in close proximity. The delay is only applied if needed and the default interval is 1000ms.	`1000ms`
max.login.ready.wait	After a processing a login request from the POS this is the maximum time to wait in milliseconds for a Ready status from Ocius Sentinel before returning a login success response to the POS. If this setting is zero then the wait will be indefinite.	`zero`
await.ready.after.transaction	The default behavior for the core is to wait for Ocius Sentinel to complete all necessary actions after a payment including having the customer remove the card from the PED before responding to the POS with the result. To allow the transaction to complete at the POS without waiting for card removal set `await.ready.after.transaction=false`.	
store.merchant.receipt	If true the merchant receipt will not be printed but will be sent to the POS to be stored in an electronic audit journal (where the POS supports this capability).	`false`
use.ocius.card.text	If true EFTLink will use the card scheme name provided by Ocius Sentinel rather than performing a look-up in its Card Range File.	`false`
separate.receipt.lines	If true the deferred (embedded) customer receipt will be sent as separate lines rather than as a single block of text containing line breaks. This is to cater for POS systems which have a limit to the length of continuous text that they can accept.	`false`
auto.logoff	If the response to a logon request to Ocius Sentinel indicates that a user is already logged in then this setting will cause the core to send a logoff followed by another logon.	`false`
deploy.default.templates	If true then a default set of receipt templates will be created by EFTLink if they do not already exist in the EFTLink folder at start up. Applies only when XML receipts are in use.	`false`
dummy.void.receipts	If true then the core will generate a dummy success response and receipt for a payment refund request without any interaction with Ocius Sentinel.	`false`
fixed.receipt.merchant.text	When using Ocius Sentinel's preformatted receipts (as opposed to XML based receipts) this defines the text within the receipt which identifies it as a merchant receipt.	`MERCHANT COPY`
fixed.receipt.customer.text	When using Ocius Sentinel's preformatted receipts (as opposed to XML based receipts) this defines the text within the receipt which identifies it as a customer receipt.	`CARDHOLDER COPY`

Setting	Description	Default
fixed.receipt.signature.text	When using Ocius Sentinel's preformatted receipts (as opposed to XML based receipts) this defines the text within the receipt which identifies it as a signature receipt.	`Please Sign Below.`
fixed.receipt.void.text	When using Ocius Sentinel's preformatted receipts (as opposed to XML based receipts) this defines the text within the receipt which identifies it as a void receipt.	`VOID`
fixed.receipt.declined.text	When using Ocius Sentinel's preformatted receipts (as opposed to XML based receipts) this defines the text within the receipt which identifies it as a declined receipt.	`DECLINED`
download.retry.limit	As part of the login process Ocius Sentinel may detect and attempt to download a software update. It is possible at this stage for Sentinel to send status 75 (Download Still Being Prepared) in which case this setting defines the number of times to retry the software download.	1 which indicates unlimited retries.
cancel.download.on.failure	If a software download fails due to reaching the retry limit, this setting defines whether a download cancellation command should be sent to Ocius Sentinel in order to allow the POS to login and proceed with sales operations. If no cancellation command is sent then the operator will need to interact with the (Windows) Ocius Sentinel application manually in order to cancel the download or attempt further retries.	`true`
ocius.sentinel.exe.path	After a successful software download Ocius Sentinel will send status 58 (Restart After Software Update) indicating that it needs to be restarted. In response to this the core will send a message instructing Ocius Sentinel to shut down and will then re-launch the application by running an executable file, the location of which is defined by this setting.	`\Program Files\Verifone\Ocius Sentinel\OciusSentinel.exe`
ocius.sentinel.restart.pause	When restarting Ocius Sentinel after a software download this setting defines the delay in milliseconds between instructing Sentinel to shut down and restarting it.	`3000`
offline.reconnect.retry.limit	When Ocius Sentinel reports that it is offline from the remote server this setting can be used to configure a number of connection retries. A value of -1 indicates unlimited retries. If a connection still cannot be established after the required number of retries then the `auto.offline` setting applies.	`0`
gift.card.type	Defines the type of gift card supported by the core where 0 = Park Retail (the default) 1 = SVS **Note**: The POS may override this setting to specify the gift card type in its request message.	
report.card.events	If true then the core will send `DeviceEvent` messages to the POS when a card is inserted into or removed from the PED. This is determined from status messages sent to the core by Ocius Sentinel.	`false`
print.dcc.quote	If true then the core will print a DCC currency conversion quote at the point when the customer is asked to make a DCC decision at the PED.	`true`

Setting	Description	Default

keystore.name	The name of the keystore file containing the key for decrypting passwords.
	Since the keystore file will be created in the cores/OciusSentinel folder, the property can either include the relative path, or the keystore file can be copied to the base EFTLink folder.
	Example with path
	`keystore.name = cores/OciusSentinel/ocius.keystore`
	Example where the keystore file has been copied to the base EFTLink folder
	`keystore.name = myfile.dat`
		`ocius.keystore`.

Translating and Suppressing Status Messages

Status messages sent by Ocius Sentinel for display at the POS can be translated or suppressed by adding entries to `ocius.properties`. Each message is identified by a number and the Ocius Sentinel integration guide lists all the possible messages.

As an example, status message 1 displays the text `Enter Gratuity`. To change this to "Enter Tip" the following entry can be added to `ocius.properties`:

```
status.1=Enter Tip
```

To suppress this message leave the text blank (nothing after the equal sign) as follows:

```
status.1=
```

Overriding Other Text Messages

There are a number of other messages and prompts which are provided by the core itself and these are also configurable. The settings in `ocius.properties` are listed below with their defaults:

- `confirm.auth.code.prompt=Confirm Transaction?`
- `confirm.auth.code.yes.option=Yes - Confirm Txn`
- `confirm.auth.code.no.option=No - Decline Txn`
- `voice.referral.prompt=Call Auth Centre`
- `voice.referral.tel.text=Tel:`
- `voice.referral.mid.text=MID:`
- `voice.referral.tid.text=TID:`
- `voice.referral.amount.text=Amount: £`
- `voice.referral.trailing.text=`
- `voice.referral.yes.option=Authorise`
- `voice.referral.no.option=Abort`
- `voice.referral.auth.entry.prompt=Enter Auth Code (or blank to cancel)`
- `signature.verification.prompt=Valid Signature?`
- `signature.verification.yes.option=Yes – Confirm Txn`
- `signature.verification.no.option=No – Decline Txn`
- `signature.verification.reprint.option=`
- `cashback.prompt=Please enter cashback amount`
- `declined.card.removal.prefix.text= Declined –`
- `svs.partial.payment.title=PARTIAL PAYMENT ONLY`

- `svs.requested.amount.text=Requested £`
- `svs.available.amount.text=Available £`
- `svs.outstanding.amount.text=Outstanding £`
- `svs.partial.payment.yes.option=Continue`
- `svs.partial.payment.no.option=Cancel`

Positioning Dialogue Options

For POS systems which support this it is possible to specify the position or order of some dialogue options using index numbers. The index should be an integer with value 1 or higher. The maximum index number allowed and the interpretation of the number will depend upon the implementation at the POS, for example in the case of Retail-J there are 8 button positions available down the right-hand side of the screen so the index numbers would range from 1 to 8.

The following settings are available:

```
confirm.auth.code.yes.position
confirm.auth.code.no.position
voice.referral.yes.position
voice.referral.no.position
signature.verification.yes.position
signature.verification.no.position
signature.verification.reprint.position
svs.partial.payment.yes.position
svs.partial.payment.no.position
```

XML Receipts

Ocius Sentinel is able to supply raw receipt data in XML form rather than as formatted text. There are a considerable number of data fields available in this way (see the latest Ocius Sentinel Integration Guide for a full list). Here is an example of an XML signature receipt received by the core from Sentinel:

```
<VoucherDetails>
  <TrainingMode>false</TrainingMode>
  <ReceiptType>Signature</ReceiptType>
  <Header>B &amp; Q</Header>
  <PTID>PW001654</PTID>
  <TID>04380001</TID>
  <MID>21249872</MID>
  <MkTransactionID>1552313</MkTransactionID>
  <TxnDateTime>2010-12-06 20:40:37.845 CET</TxnDateTime>
  <CardScheme>Visa</CardScheme>
  <PAN>*********2222</PAN>
  <ExpiryDate>12/12</ExpiryDate>
  <TxnType>Sale</TxnType>
  <CaptureMethod>SWIPED</CaptureMethod>
  <CustomerPresent>true</CustomerPresent>
  <ECommerce>false</ECommerce>
  <ContAuth>false</ContAuth>
  <AccountOnFile>false</AccountOnFile>
  <PinEntered>false</PinEntered>
  <CreditDebitMessage>Please debit my account</CreditDebitMessage>
  <CurrencySymbol>£</CurrencySymbol>
  <CurrencyAbbreviation>GBP</CurrencyAbbreviation>
  <Amount>1.00</Amount>
  <Total>1.00</Total>
  <CVM>Please Sign Below</CVM>
  <KeepText1>Please Keep This Receipt</KeepText1>
  <KeepText2>For your Records</KeepText2>
  <EFTSN>0508</EFTSN>
  <AuthCode>789DE</AuthCode>
  <Reference>101206 61 1 1</Reference>
  <Footer>B &amp; Q</Footer>
  <GratuityBoxRequired>false</GratuityBoxRequired>
  <ExtendedReceipt>false</ExtendedReceipt>
  <DisableCurrencySymbol>false</DisableCurrencySymbol>
  <AuthOnly>false</AuthOnly>
  <CardSchemePrintText></CardSchemePrintText>
  <PrintAttempts>1</PrintAttempts>
  <ContactlessMSD>false</ContactlessMSD>
  <TokenRegistrationResult>NotSet</TokenRegistrationResult>
  <TokenRegistrationOnly>false</TokenRegistrationOnly>
</VoucherDetails>
```

In XML mode the core must be configured to convert the XML data into formatted text receipts. Formatting is achieved using template files in which free text and XML fields can be positioned and left, right or centre justified as required. Any number of templates can be created and you would typically expect to have seven or more, one for each of the merchant, signature, customer, merchant void, customer void, merchant declined and customer declined receipts, and further templates for any extended functionality (for example gift cards). Below is an example of a template file:

customer_template.txt

```
<WIDTH=36>
<CENTRE>Customer Test Template
------------------------------------
Card Sale<RIGHT><Total>
<PAN>
------------------------------------
Card   : <CardScheme>
Number : <PAN><RIGHT><CaptureMethod>
AID        : <AID>
App Date   : <AppEff>
Cryptogram : <CID>/<AC>
Auth Code  : <AuthCode>
Merchant ID: <MID>
Terminal ID: <TID>
------------------------------------
<CreditDebitMessage>
<CENTRE><CVM>
```

In the template, XML element names are specified in angled brackets like this <CVM> and each will be substituted with the actual value supplied by Sentinel. There are four special directives used for formatting which are:

- `<WIDTH=nn>` This specifies the maximum width of the receipt in columns.
- `<CENTRE>` This will centre any text which appears after it on the same line.
- `<RIGHT>` This will right-justify any text which appears after it on the same line.
- `<SUPPRESS>` The receipt will not be printed.

Note: All of the above directives must be uppercase to be recognized.

In order to decide which template to use for a receipt the core will read a file called `ocius_receipt.properties` in which templates can be selected by looking for one or more values in the XML data. This file contains entries in the form

```
template-filename=<XML-element-1>required-value<XML-element-2>required-value
```

If all of the XML elements listed on the line have the specified value then that template file will be used. Below is an example file:

ocius_receipt.properties

```
customer_template.txt=<ReceiptType>Customer
merchant_template.txt=<ReceiptType>Merchant
signature_template.txt=<ReceiptType>Signature
```

When looking for a match templates are checked in the order that they appear in `ocius_receipt.properties`. If no matching template is found then the core will return the entire XML data in place of a formatted receipt. If a template appears which does not specify any XML fields to match on (nothing after the equal sign) then that template will always be treated as a match.

It is also possible to match partial values using one or more of the flags [PREFIX], [SUFFIX] or [CONTAINS] followed by the partial text to match. For example:

```
contactless_template.txt= <ReceiptType>Customer<CaptureMethod>[SUFFIX]CONTACTLESS
```

The above will match when ReceiptType has the fixed value `Customer` and `CaptureMethod` is any text followed by `CONTACTLESS`.

Keystore

The encryption key must be generated and stored in a keystore. To achieve this, the following steps must be followed:

Windows Operating Systems

- Open a command prompt, and change directory to the eftlink location.
- Type : `encrypt.bat -k <keystore name> <properties file>`

  For example, `encrypt.bat -k ocius.keystore ocius.properties.`

Keystore file will be generated and stored in the data directory.

Password Encryption

Default configuration requires user.id, user.pin and, where used, account.id and transax.account.id to be encrypted in `ocius.properties`.

user.id, user.pin, account.id and transax.account.id is allocated or configured in the Ocius Sentinel software itself, and varies from site to site.

To achieve this, the following steps must be followed:

Windows Operating Systems

To encrypt a password; open a command prompt and change directory to eftlink's location.

- Type `encrypt.bat –e <keystore name> <properties file> <password>.`

For example, `encrypt.bat –e ocius.keystore ocius.properties[`followed by the required password as a final parameter].

- Password and initialization vector will be outputted to the console.

Copy and paste it to the appropriate property in `ocius.properties`.

To re-encrypt a password (or multiple passwords) with new encryption settings; open a command prompt and change directory to eftlink's location.

- Type `encrypt.bat –r <keystore name> <properties> <encrypted passwords colon separated> <previous initialization vectors colon separated> <keygen type> <cipher type> <key size> <iterations>.`

  For example, `encrypt.bat –r ocius.keystore ocius.properties [Encrypted password1: Encrypted password2] [Encrypted password iv1: Encrypted password iv2] AES AES/CBC/PKCS5Padding 128 10000.`
- Re-encryption uses existing crypto settings in the properties file to decrypt the password. Once the password is decrypted, a new keystore file is generated using the new crypto parameters specified at the command line and the new encrypted password / initialization vector is generated.
- When using AES algorithm with a keysize that is greater than 128, you may get java.security.InvalidKeyException: Illegal key size or default parameters. If so, Additional Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files will need to be downloaded and extracted to %JAVA_HOME%/jre/lib/security/

Supported Functions

The following operations are supported by this implementation of the Ocius Sentinel interface.

- Logon and logoff (at the beginning and end of a shift or trading period)
- Sale
- Refund
- Card Read (for non-EFT cards only)
- X reports (reconciliation)
- Z reports (reconciliation with closure)
- Customer receipt re-print (via maintenance menu)
- SVS gift cards

13

VeriFone Point Scandinavia

General Information

This document covers EFTLink Integration with VeriFone Point Scandinavia Payment Systems. It should be read in conjunction with the *EFTLink Framework Installation Guide*.

> **Note**: VeriFone Point Scandinavia was formerly known as SteriaPay.

Disambiguation

This VeriFone Point implementation is for use with the VeriFone Point Scandinavia Payment System, formerly known as SteriaPAY. There is also a Point US implementation, which is unrelated.

EFTLink General

This document assumes static EFTLink configuration. When deploying with a POS that supports dynamic configuration, all property settings referred to below should be set on the POS, and not directly into local property files.

System Architecture

EFTLink connects directly to the payment system using a proprietary socket protocol.

Fileset

Verifone Point Scandinavia uses:

Oracle files

```
cores/SteriaPay/steriapaycore.jar
steriapay.properties
```

VeriFone Point files

Obtain PayPoint.jar from Point Scandinavia, and place in the same folder (cores/SteriaPay) as `steriapaycore.jar`

```
PayPoint.jar
```

> **Note**: There is a UK interface within EFTLink to PayPoint, an online payment service provider, which uses a completely different paypoint.jar. Although that interface, and the associated jar file are not included in this release, the similarity in name may cause confusion, which is outside of Oracle’s control.

Language

There are no language specific translation files within steriapay.jar

Core Classname

The following should have been set in EftLinkconfig.properties by installcore.bat or installcore.sh

```
EPSCore0 = manito.eft.steriapay.SteriaPayCore
```

Configuration Settings

Configuration settings should be defined in `steriapay.properties`.

This will be copied from `cores/SteriaPay` to the base eftlink folder by `installcore.bat` or `installcore.sh`

The available settings are listed below (none of these are compulsory so the file may be absent, and the default `steriapay,properties` only has `com.port=127.0.0.1`). Each setting should be placed on a separate line in the file in form name=value. The setting names are all lower case.

Setting	Description	Default	Example
com.port	The com port of the terminal if serial communications are being used (in the form `COMn` where n is the number of the port) or the IP address of the terminal for TCP/IP communications (in the form `n.n.n.n`).	COM1	
baud.rate	The baud rate to use to communicate to the terminal for serial communications.	115200	
verify.signature	When a signature is required, this specifies whether it should be verified as authentic.	false	
signature.verification.question	When signature verification is required, this is the text of the question. **Note**: The terminal will deliver the signature receipt after approving the transaction. Therefore if the operator answers no to the signature verification question the core must request a reversal. For this reason after the operator answers no, both approval and reversal receipts will be printed.	Signature Ok?	
print.two.chip.card.receipts	The terminal delivers one set of receipt text for a chip card transaction. This setting configures the core to print two copies of a chip card receipt (one each for merchant and customer).	false	
receipt.one.title	The title of the first receipt to be printed.	no title printed	`receipt.one.title=*** MERCHANT RECEIPT ***\n` **Note**: The character sequence `\n` denotes a line break.

Setting	Description	Default	Example
receipt.two.title	The title of the second receipt to be printed. Note: If print.two.chip.card.receipts=false then use this setting to specify a title not receipt.one.title.	no title printed	`receipt.two.title=*** CUSTOMER RECEIPT ***\n`
signature.receipt.title	The title of the signature receipt.	same as receipt.one.title	
receipt.header	The receipt header. This will print at the top of the receipt above the title.	no header printed	`receipt.header=Retailer’s Name\nRetailer’s Address\nTel: +44 1234 123456\n`
receipt.footer	The receipt footer. This will print at the bottom of the receipt. **Note**: At the time of writing the current version of the java PayPoint software delivers some receipts with a paper cut message printed at the bottom, which cannot be changed or removed via PayPoint’s own configuration. Since the core will insert the receipt footer below this it will be necessary to use the receipt.text.to.remove setting below to remove the paper cut message.	no footer printed	`receipt.footer=\nThank you for choosing\n     Retailer’s name.`
receipt.text.to.remove	Allows a section of the receipt text provided by the terminal to be removed. This can be used to remove the paper cut message if required. The text to be removed should be specified exactly as shown on the receipt. The character sequence \n can be included to remove preceding or trailing line breaks.	nothing removed	`receipt.text.to.remove=\n....PAPER CUT...`
reference.title	This setting defines a label to print at the beginning of the line containing the reference field. **Note**: A label called `REF:` is already used elsewhere in the SteriaPay receipt.	`POS`	
remove.blank.lines.from.top.of.steriapay.receipt	The terminal may deliver receipts containing blank lines at the top. If adding a header and/or title it may be useful to remove these, which can be achieved using this true/false setting.	`false`	

Other Information

Supported Functions

The following additional operations are supported by this implementation of the Verifone Point Scandinavia interface:

- Logon and logoff
- Sale (with cashback)
- Refund
- Reversal
- Reconciliation with closure
- Print stored reports (via the EFTLink maintenance menu function)
- Print last receipt (via the EFTLink maintenance menu function)

14 VeriFone Point (US)

EFTLink General

This document assumes static EFTLink configuration. When deploying with a POS that supports dynamic configuration, all property settings referred to below should be set on the POS, and not directly into local property files. It should be read in conjunction with the *Oracle Retail EFTLink Framework Installation Guide*.

Disambiguation

This Point implementation is for use with Mx915 terminals in the US, with communication based on a socket/XML protocol. There is also a Point implementation in Norway, which is unrelated.

Minimum Version

The Point interface requires a minimum EFTLink version of v1.1.124.

System Architecture

VeriFone Point is deployed as an intelligent terminal. EFTLink connects directly to the terminal using a proprietary socket/XML protocol.

Fileset

In addition to standard EFTLink files, PointUS uses:

- `cores/pointus/pointuscore.jar` – executable code for the PointUS EFTLink core.
- `pointus.properties` – configuration settings to specify which features are enabled and to define communication parameters for the interface with the EFT terminal.

> **Note** – If the POS supports dynamic configuration, properties can be set there instead of in `pointus.properties`

Core Classname

manito.eft. pointus. PointUSCore

This should be set as `EPSCore<x>` in `eftlinkconfig.properties`.

Configuration Settings

The full set of configuration properties is defined and commented in `pointus.properties`.

Key Settings

Settings that may be different for each POS/PED.

Setting	Description	Default	Example
TerminalIP	IP of Mx915 terminal.		`TerminalIP =`

Secondary Settings

These settings are normally correct at their default values, but can be overridden if necessary.

Setting	Description	Default	Example
Terminal address	Port number	5015	`TerminalPort = 5015`
MaintenanceTimeout		60	`MaintenanceTimeout = 60`
ResponseTimeout	Time allowed in seconds for the transaction to complete at the terminal. This needs to be long enough to cover all customer interaction and host authorization.	120	`ResponseTimeout = 120`
ConnectionTimeout		10	`ConnectionTimeout = 10`
ValidateLoyaltyData	When a loyalty card swipe is requested, the customer may identify themselves by entering a phone number rather than swiping a card. If loyalty cards are suitably defined in the card range file and tagged as "Loyalty", this can be checked. Option to enable validation of loyalty data to try to differentiate between card numbers and phone numbers.	false	`ValidateLoyaltyData = false`
SignatureCheckFloorLimit	floor limit for swiped credit transactions	0.00	`SignatureCheckFloorLimit = 0.00`
SignatureCheckTimeout	timeout on waiting for signature	30	`SignatureCheckTimeout = 30`
EmbeddedPrinting	whether customer printout is to be buffered and included in the POS authorisation response such that it can be embedded # in the POS receipt	false	`EmbeddedPrinting = false`
MaxLineItems	maximum number of line items allowed in a single message	10	`MaxLineItems = 10`

Setting	Description	Default	Example
MaxLineItemUpdates	maximum number of individual line item update messages it is practical to send before it becomes too cumbersome and slow	20	`MaxLineItemUpdates = 20`
MaxLineItemTextLength	maximum line item description length allowed by comms protocol or visible on terminal	17	`MaxLineItemTextLength = 17`
ShowlineItemsOnVoucher	whether to show line items on the EFT voucher	false	`ShowlineItemsOnVoucher = false`
ReceiptLineItemStartTag	key word/phrase that identifies the start of line item summary on EFT voucher	QTY DESCRIPTION	`ReceiptLineItemStartTag = QTY DESCRIPTION`
ReceiptLineItemEndTag	key word/phrase that identifies the end of line item summary on EFT voucher	Total:	`ReceiptLineItemEndTag = Total:`
ShowEMVTagsOnVoucher	whether to show diagnostic EMV tags on the EFT voucher, default false	false	`ShowEMVTagsOnVoucher = false`
EmbeddedPrintFilter	print filters to allow voucher to be trimmed when embedded in the POS receipt e.g. to remove header/footer lines starting with text in this list will be skipped	null	`EmbeddedPrintFilter`

Administration Functions

The terminal has some administration/maintenance functions. These are normally invoked from a dedicated “EFT Maintenance” button, but if this is not available, they could be accessed by an engineer using the EFTLink built-in test harness.

EFTLink uses DeviceProxy messages to display input prompts on the POS to manage these functions.

Function	Description
Terminal-POS Pairing	The terminal has to be paired with a specific POS, by entering a code.
Registration	This operation displays a 4-digit number on the POS that must then be typed into the terminal to complete the pairing.
Unregistration	This operation removes a pairing.
Test MAC	This operation tests that the terminal is accessible and that a pairing in in pace.
Day Report	Print a non-closing day report (summary)
Day End	Print a day report and close the current day. Manual alternative to automated ReconciliationWithClosure.
Last Transaction	Print details of the last transaction at the terminal.

Supported Functions

Below is a list of supported functionalities of the interface to Merchant Link. Many functionalities are provided by PointUS, such as Loyalty, Cashback etc. (please refer to interface specification for details) but are not implemented because of the business requirement.

Function	Description
Payment	Sends payment request to the terminal. Terminal will return a response message with unformatted receipt strings for customer and/or merchant receipts. In an event of referral where authorization cannot be obtained online then a prompt for authorization code will appear; authorization code must be obtained via telephone and entered here. If successful, appropriate receipts will be printed at the end of transaction.
Reversal	Sends reversal request to the terminal. This will reverse a transaction specified by the transaction number, found on the receipt, which must be captured by the POS and pass on to EFTLink.
Refund	Sends refund request to the terminal. This will refund a transaction with specified amount.
Reconciliation / Settlement	This is supported directly by the terminal via TCP/IP request.
Sale State Notifications	Sends line items through to the device so the customer display can be updated in line with the POS.
SVC Payment	Sends a Gift or Merchandise credit card payment request to the terminal. If there are not enough funds available, only the funds available will be deducted. The POS client will have to settle the transaction with another tender in this scenario.
SVC Activate	Sends a Gift or Merchandise credit card activation request to the terminal.
SVC Deactivate	Sends a Gift or Merchandise credit card deactivation request to the terminal. The account is disabled after this as the request is intended to be used for lost or stolen cards. It is not possible to use the card or account once this request has been issued and accepted.
SVC Add Value	Sends a Gift or Merchandise credit card add value request to the terminal. This will only add value to an account that has been activated.
SVC Balance Enquiry	Sends a Gift or Merchandise credit card balance enquiry request to the terminal.
SVC Unload (Cashout)	Sends a Gift or Merchandise credit card cash out request to the terminal. All funds are deducted from the account and the cash back amount is returned to the POS. The account is not deactivated as part of this process.

15

WorldPay

General Information

Overview

This document covers EFTLink Integration with WorldPay Payment Systems. It should be read in conjunction with the *Oracle Retail EFTLink Framework Installation Guide*.

System Architecture

EFTLink connects to the WorldPay application that is installed on the same PC as the POS, using a proprietary socket protocol. The WorldPay application must be started.

Note: This document does not cover the install of the WorldPay software.

Fileset

The following files are used in the EFTLink folder:

`cores/WorldPay/worldpaycore.jar`

`worldpay.properties` (optional, if not present defaults apply)

Language

There are no translation files in `worldpaycore.jar`. EFTLink Framework should be set to default English. See the *EFTLink Framework Installation Guide*, EFTLink General Information, Translation section:

```
EftlinkConfig.properties
DisplayLanguage = EN
```

Core Classname

The following should have been set in the `EftlinkConfig.properties` file by installcore.bar or installcore.sh

```
 EPSCore0=manito.eft.worldpay.WorldPayCore
```

Configuration Settings

The core is configured via settings inserted into the `worldpay.properties` file located in the chosen EFTLink folder. If the default port numbers are used within WorldPay's software configuration then this file does not need to be present as the core will work without it. The available settings are listed below.

Note: The software was previously called YesPay.

Setting	Description	Default	Example
yeseft.folder	The path to the folder where the WorldPay software is installed. Worldpay is normally installed in a folder at the root of the C: drive of the PC called YESEFT.	\YESEFT	`yeseft.folder = \YESEFT`
request.port	The socket port for making payment requests	10000	`request.port = 10000`
receipt.port	The socket port for receiving receipts.	20000	`receipt.port = 20000`
message.port	The socket port for receiving status messages and dialogue requests.	8000	`message.port = 8000`
perform.card.range.lookup	If true, EFTLink will use its mapping file CardRange.xml to determine the card scheme name based on information returned by WorldPay. Otherwise it will return the text provided by WorldPay.	false	`perform.card.range.lookup = false`
embed.customer.receipt	If true, EFTLink will return the customer receipt to the POS to be included in its own receipt rather than printing it separately. Note: not all POS systems may support this feature.	false	`embed.customer.receipt = false`
suppress.merchant.receipt	If true, EFTLink will discard the merchant receipt.	false	`suppress.merchant.receipt = false`
store.merchant.receipt	If true, EFTLink will return the merchant receipt to the POS to be added to the electronic journal rather than printing it separately. This setting is overridden by suppress.merchant.receipt. Note: not all POS systems may support this feature.	false	`store.merchant.receipt = false`
language	The language code for translating responses from WorldPay on the message port. The translations are taken from WorldPay files in the WorldPay folder. The default value is "`en_GB`", and references part of the filename provided by WorldPay. `JVTMessageBundle_en_GB.properties` in C:\YESEFT\properties folder.	`en_GB`	`language = en_GB`

Setting	Description	Default	Example
signature.reprint.prompt	The text to display when asking if a signature receipt should be reprinted. This text will only be shown if the operator answers no when asked to confirm signature ok for a previous print.	Blank, meaning reprint will not be offered.	`signature.reprint.prompt =`
notify.signature.print	If true the POS will be notified that a signature receipt has been printed. This is for the business case where the signed receipt must be stored in the cash drawer and therefore the POS needs to know to open the drawer. **Note**: An additional setting is required in `EftlinkConfig.properties` to enable this function: `DeviceEvents=true`	true	`notify.signature.print = true`
mid.text	The title to display for the merchant ID in voice referrals.	MID:	`mid.txt = MID:`
tel.text	The title to display for the telephone numbers in voice referrals.	Tel:	`tel.txt = Tel:`
auth.prompt	The text to display for the authorization code entry prompts in voice referrals.	Enter Auth Code (or blank to cancel)	`auth.prompt = Enter Auth Code (or blank to cancel)`
max.auth.code.length	The maximum length allowed for an entered authorization code.	9	`max.auth.code.length = 9`
cashback.prompt	The text to display for the cashback prompt.	Cashback required?	`cashback.prompt = Cashback required?`
cashback.amount.prompt	The text to display for the cashback amount prompt	Please enter cashback amount.	`cashback.amount.prompt = Please enter cashback amount`
min.cashback	This is the minimum cashback amount allowed.	Blank (no minimum amount).	`min.cashback =`
max.cashback	This is the maximum cashback amount allowed.	Blank (no maximum amount).	`max.cashback = 100`
max.cashback.length	This is the maximum length allowed for an entered cashback amount.	5	`max.cashback.length = 5`
currency.symbol	The currency symbol to use when displaying cashback limits to the operator. This can be any text required, e.g. "GBP" and so on.	£	`currency.symbol = £`

Setting	Description	Default	Example
cnp.prompt	This is the text to display for the customer not present prompt.	CNP confirmation	`cnp.prompt = CNP confirmation`
response.timeout	The timeout in milliseconds to wait for a response from WorldPay after sending a request. It is recommended that this be left disabled (indefinite) and leave the timeout to WorldPay.	0 (indefinite).	`response.timeout = 0`
print.x.report	Whether to print an X report on reconciliation.	false	`print.x.report = false`
print.z.report	Whether to print a Z report on reconciliation with closure.	false	`print.z.report = false`
x.report.title	The title for X reports.	`** EFT X REPORT **`	`x.report.title=** EFT X REPORT **`
z.report.title	The title for Z reports.	`** EFT Z REPORT **`	`z.report.title=** EFT Z REPORT **`

Other Information

Additional Functions

The following additional operations are supported by this implementation of the WorldPay interface:

- Sale
- Refund
- Refund with token
- Reversal
- X reports (reconciliation)
- Z reports (reconciliation with closure)

Integration Notes

WorldPay configuration

The WorldPay software must be configured to use its socket interface on all three ports (request, receipt and message) respectively. Within the WorldPay (YESEFT) configuration utility the relevant tabs are Interfacing, Receipt and HostEvt.

Online/Offline indication

In a card payment response the miscellaneous data field will indicate whether the authorisation was online, offline or manual (voice referral). The format will be `{Status=xxx}` where `xxx` is one of `ONLINE`, `OFFLINE` or `MANUAL`.

Device ID

The terminal number will be returned in the DeviceID element of the EFTLink login response (if the WorldPay software is running at the point of login) and with each card payment response thereafter. An example login response is provided below.

```
<?xml version="1.0" encoding="UTF-8"?>
<ServiceResponse RequestType="Login" ApplicationSender="POSSIM" WorkstationID="1"
RequestID="2" OverallResult="Success">
  <Terminal DeviceID="12345678" />
</ServiceResponse>
```

Note: The `Terminal Device ID` should be the pertinent one for the terminal being connected.

Refund with token

To perform refunds via token both the token and the card payment reference from the original sale must be provided in the refund request, please see below for an example of a payment response from EFTLink showing these fields.

```
<?xml version="1.0" encoding="UTF-8"?>
<CardServiceResponse RequestType="CardPaymentLoyaltyAward"
ApplicationSender="POSSIM" WorkstationID="1" RequestID="4"
OverallResult="Success">
  <Terminal TerminalID="22980092" DeviceID="0081226814" MerchantID="6818780"
STAN="345" />
  <Tender>
    <TotalAmount Currency="GBP">56.00</TotalAmount>
    <Authorization AcquirerID="UNKNOWN" TimeStamp="2015-04-29T12:45:31"
ApprovalCode="947265" CardType="3" Tender="0108" CardPAN="476173******0119"
ExpiryDate="1251" CardCircuit="VISA CREDIT" TransactionReference="PGTR740971038"
/>
  </Tender>
  <CardValue CardType="3" Tender="0108" LoyaltyEligible="true">
    <CardPAN>476173******0119</CardPAN>
    <EndDate>1251</EndDate>
    <CardCircuit>VISA CREDIT</CardCircuit>
    <Hash>1CCF57529637C314FBE9C6544BF10E3D16FE20B8</Hash>
    <Token>533173099D9A95649</Token>
    <TransactionReference>PGTR740971038</TransactionReference>
  </CardValue>
  <MiscellaneousData>{Status=ONLINE}</MiscellaneousData>
</CardServiceResponse>
```

Below is an example of a subsequent refund request from the POS.

```
<?xml version="1.0" encoding=" UTF-8"?>
<CardServiceRequest RequestType="PaymentRefund" ApplicationSender=" POSSIM "
WorkstationID="1" RequestID="5">
  <POSdata LanguageCode="en" SpooledPrint="false">
     <POSTimeStamp>2015-04-29T12:46:31</POSTimeStamp>
     <TransactionNumber>920</TransactionNumber>
  </POSdata>
  <TotalAmount Currency="GBP">56.00</TotalAmount>
  <CardValue>
     <Token>533173099D9A95649</Token>
     <TransactionReference> PGTR740971038</TransactionReference>
  </CardValue>
</CardServiceRequest>
```

Note: The `Token` and `Transaction Reference` in the above statement are demonstration values only.

Reversal

Reversal requests require the card payment reference, PAN and card expiry date from the original transaction. Additionally a reversal should carry the same transaction number as the transaction it is cancelling. Below is an example reversal request with the necessary data fields highlighted.

```
<?xml version="1.0" encoding="UTF-8"?>
<CardServiceRequest RequestType="PaymentReversal" ApplicationSender="POSSIM"
WorkstationID="1" RequestID="9" RequestSubType="OperatorReversal">
  <POSdata LanguageCode="en">
    <POSTimeStamp>2015-06-09T11:48:29</POSTimeStamp>
    <TransactionNumber>401</TransactionNumber>
  </POSdata>
  <OriginalTransaction TerminalID="22980092" STAN="401" TimeStamp="2015-06-
09T11:48:27" RequestType="CardPaymentLoyaltyAward" ApprovalCode="956872"
MiscellaneousData="{Status=ONLINE}" />
  <TotalAmount Currency="GBP">15.00</TotalAmount>
  <CardValue CardType="3" Tender="0108" LoyaltyEligible="true">
    <CardPAN>476173******0119</CardPAN>
    <EndDate>1263</EndDate>
    <CardCircuit>VISA CREDIT</CardCircuit>
    <Hash>52FDA2337F840BEE654353EA1D1F54FB5EFC2E98</Hash>
    <Token>533173099D9A95649</Token>
    <TransactionReference>PGTR327632569</TransactionReference>
  </CardValue>
</CardServiceRequest>
```

Note: The `Transaction Number`, `Card Pan`, `End Date` and `Transaction Reference` in the above statement are demonstration values only.

Signature Print Notification

If the core is configured to notify the POS of a signature print (see section 0) then a device event will be generated as shown below. The POS should examine the EventType field to determine that this is a signature print notification.

```
<?xml version="1.0" encoding="UTF-8"?>
<DeviceRequest ApplicationSender="MICROS" WorkstationID="1" RequestID="5.11"
RequestType="Event">
  <Event EventType="SIGNATURE" />
</DeviceRequest>
```

The POS should acknowledge the device event as in the following example.

```
<?xml version="1.0" encoding="UTF-8"?>
<DeviceResponse RequestType="Event" ApplicationSender="MICROS" WorkstationID="1"
RequestID="5.11" OverallResult="Success" />
```