Oracle® Retail EFTLink

Core Configuration Guide

Release 25.0.0
G34820-02
August 2025

Oracle Retail EFTLink Core Configuration Guide, Release 25.0.0

G34820-02

Primary Author: Bernadette Goodman

Contributors: Sean Hamill

Contents

6 PayPal

7 Six Pay

8 Solve Connect

9 Tender Retail

10 Verifone Ocius Sentinel

List of Tables

Send Us Your Comments

Oracle® Retail EFTLink Core Configuration Guide, Release 25.0.0

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 Help Center (OHC) website at https://docs.oracle.com/en/industries/retail/index.html. 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 website at http://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

Related Documents

For more information, see the following documents in the Oracle Retail EFTLink 25.0.0 documentation set:

- *Oracle Retail EFTLink Release Notes*
- *Oracle Retail EFTLink Framework Advanced Features Guide*
- *Oracle Retail EFTLink Framework Installation and Configuration Guide*
- *Oracle Retail EFTLink Security Guide*
- *Oracle Retail EFTLink Rest API Guide*
- *Oracle Retail EFTLink Xstore Compatibility Guide*
- *Oracle Retail EFTLink Validated Partners Guide*
- *Oracle Retail EFTLink Validated OPI Partners 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, 25.0.0) or a later patch release (for example, 25.0.1). If you are installing the base release and 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 at the Oracle Help Center (OHC) website (https://docs.oracle.com/en/industries/retail/index.html), 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 at the Oracle Help Center at the following URL:

https://docs.oracle.com/en/industries/retail/index.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 F123456-02 is an updated version of a document with part number F123456-01.

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

Oracle Retail Documentation at the Oracle Help Center

Oracle Retail product documentation is available on the following website:

https://docs.oracle.com/en/industries/retail/index.html

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

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.

Convention	Meaning
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1 Introduction

After installing EFTLink (instructions can be found in the *Oracle Retail EFTLink Framework Installation and Configuration Guide*), you will need to configure the specific core with the required settings to allow the POS to communicate with the selected EFT system.

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

This guide consists of separate chapters for each available core; go to the pertinent section for each core to be installed. The following cores are supported:

- AJB FIPay
- Cayan
- Oracle Payment Interface (OPI)
- Pay by Link (PBL)
- PayPal
- Six Pay
- Solve Connect
- Tender Retail
- Verifone Ocius Sentinel
- Verifone Point (US)
- WorldPay

Miscellaneous Data Disclaimer

EFTLink along with some selected Cores, has the ability for additional data to be sent and received in a field called `<MiscellaneousData>`.

This can be used by System Implementers (SIs) and Payment Service Providers (PSPs) to pass additional data in the messages between Xstore and the Payment Providers, using custom code.

Typically, this is used to add directives which we can trigger different payment workflows. However, it can also be used to capture additional payment data for downstream processing for the Retailer's to use for reconciliation or financial purposes.

Under no circumstances should any PCI or potentially sensitive PII data be placed in this field. Oracle will not be responsible for any issues caused by integration changes made by SIs, Retailers and Payment Providers, that enable sensitive data to be added into this field.

2

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 and Configuration 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 20.0.

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.
- `Lang<CC>_<Core>.properties` – Language translation file, for further information see the Language section below.
- `AJBComm.jar` – API supplied by AJB to allow communication to the terminal.

Third Party

The following file is required for core operation, but is 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

The translation files for this core should not require alteration, but if necessary then this can be accomplished by amending the relevant `Lang<CC>_<Core>.properties` within the base eftlink folder.

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

```
EftlinkConfig.properties
```

```
DisplayLanguage = EN
```

Supported country codes are CN, DE, EN, ES, FR, IT, JP, NL, PT, RU and SV.

Core Classname

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

```
EPSCore0 = manito.eft.ajb.FIPayCore
```

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.

Table 2-1 AJB FIPay - Key Settings

Setting	Description	Example
ip.address	Terminal address IP address	`ip.address = IP ADDRESS`
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.

Table 2-2 AJB FIPay - Secondary Settings

Setting	Description	Default	Example
ip.port	IP port number.	24900	`ip.port = 24900`

Table 2-2 (Cont.) AJB FIPay - Secondary Settings

Setting	Description	Default	Example
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`
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.	true	`combine.receipt = true`
combine.receipt.suppress.lines	When combine receipt is true, set which line to suppress.	N/A	`combine.receipt.suppress.lines = 1,2,3,4`
token.expiry.date	Specifies whether an expiry date will be included when performing tokenized refunds.	false	`token.expiry.date = false`
combine.receipt.suppress.strings	When combine receipt is true, set what line to suppress when strings are matched.	N/A	`combine.receipt.suppress.strings = DATE,DCC Not Available`

Table 2-2 (Cont.) AJB FIPay - Secondary Settings

Setting	Description	Default	Example

giftcard.handler	Gift card handler's, fully qualified class name. Possible values are:
	`manito.eft.ajb.giftcard.StandardFiPayGiftCardHandler`
	`manito.eft.ajb.giftcard.SVSGiftCardHandler, manito.eft.ajb.giftcard.ValueLinkGiftCardHandler`
		N/A	`giftcard.handler = manito.eft.ajb.giftcard.StandardFiPayGiftCardHandler`

giftcard.provider	Gift card provider's, 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`	N/A	`giftcard.provider =`
swipe.fallback.keyed	Failure of card swipe during payment can be configured to result in fallback to keyed.	False	`swipe.fallback.keyed=false`
authcode.minlength	Specify minimum length of the authorization code which will be accepted.	1	`authcode.minlength = 1`

void.header.n
[where n is >0]
	Specify several header lines to include on void receipts.	None	`void.header.1 = **********`
			`void.header.2 = ** VOID **`
			`void.header.3 = **********`

void.footer.n
[where n is >0
	Specify a number of footer lines to include on void receipts.	None	`void.footer.1 = **********`
			`void.footer.2 = ** VOID **`
			`void.footer.3 = **********`

authcode.attempts	Specify the number of attempts the operator is allowed when entering the authorization code, and characters entered are less than specified by authcode.minlength.	1	`authcode.attempts = 1`
ReceiptSignatureTriggerText	Specifies the text line on the receipt used to identify signature capture. Line will be replaced using TXT_SIGNATURE_CAPTURED.	x_____	`ReceiptSignatureTriggerText = x_____`
enable.check.payment	Enable check payment functionality.	false	`enable.check.payment = true`

Table 2-2 (Cont.) AJB FIPay - Secondary Settings

Setting	Description	Default	Example
display.language	Specify the language the PED uses to display the prompt (150) message.	0(English)	`display.language = 0`
customer.question.form.name	The name of the customer's question/verification form.	FI_YESNO	`customer.question.form.name = FI_YESNO`
customer.question.max.text.per.line	The maximum number of characters of each line for the customer question form.	24	`customer.question.max.text.per.line = 24`
display.message.form.id	The name of the form used to display custom messages like marketing ones.	FI_MSGONLY	`display.message.form.id = FI_MSGONLY`
display.message.max.text.per.line	The maximum number of characters for the custom message above.	24	`display.message.max.text.per.line=24`
capture.numeric.form.id	The name of the form used to capture generic numeric data from the PIN pad.	FI_GETDATA	`capture.numeric.form.id=FI_GETDATA`
display.message.duration	The duration in seconds to display the prompt/marketing message in the PED.	30	`display.message.duration = 30`
capture.numeric.max.text.per.line	The maximum characters for each line in the message to capture numeric data like phone number from the PIN pad.	42	`capture.numeric.max.text.per.line=42`
capture.phone.number.maxlength	The maximum characters of phone number to capture from the PIN pad.	10	`capture.phone.number.maxlength=10`
timeout.reversal.max.attempt	The maximum number of attempts our core sends a reversal to FIPay in a case for timeouts.	3	`timeout.reversal.max.attempt=3`
maxDisplayLines	The maximum number of lines the device will only display up to.	117	`maxDisplayLines =117`

Supported Functions

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

Table 2-3 AJB FIPay - Supported Functions

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.

Table 2-3 (Cont.) AJB FIPay - Supported Functions

Function	Description
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.
Check Payment	EFTLink sends check payment requests to AJB FIPay. Please note that offline authorization is not permitted for check payments. This type of payment is not permitted in the SAF queue of the AJB FIPay solution.
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.
Custom form for displaying a message	Sends a request to the terminal that displays the message text passed by the POS. The core sends a success or a failure flag back to the POS.
Custom form for customer question/verification	Sends a request to the terminal with a question/verification message. The customer selects either the Yes or No button. The core sends 'Y' or 'N' as part of the response to the POS.
Custom form for capturing phone number	Sends a request to the terminal triggering a phone number capture. The customer keys in their phone number and hit submit. The core sends the captured phone number to the POS.
Custom form for capturing signature	Sends a request to the terminal triggering a signature capture.

3
Cayan

This Cayan implementation is for use with Genius terminals in the US, with communication based on a web service 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 20.0.

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 Cayan core 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 start up 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.public.jks.properties` – for cayan.public.jks files created with newer versions of the EFTLink software, this file contains the information required to open the cayan.public.jks keystore
- `cayan.secure` – storage file for the random encryption key that is used to protect merchant information
- `cayan.secure.properties` – for cayan.secure files created with newer versions of the EFTLink software, this file contains the information required to open the cayan.secure keystore

JRE

The Cayan core is no longer limited to running JRE 1.8 as the required components for the Cayan core which were removed from Java are now included in the EFTLink distribution.

Account Information Entry

At initial software start up, 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 Supported 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-cayan.bat -g [<keystore name> <properties> <certificate> <dyanamicProperties> {<Colon-Separated List of Properties>} <keygenType> <cipherType> <keySize> <iterations>].`

  For example, `encrypt-cayan.bat -g`
- 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 key size 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/`

Linux Systems

> **Note**
>
> You may be required to give script file(s) execution rights. This can be accomplished by opening a terminal window and typing:
>
> `sudo chmod +x <PathToFile>`
>
> for example, `sudo chmod +x /opt/eftlink/encrypt-cayan.sh`

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

- Type at the command prompt: `sudo ./encrypt-cayan.sh -g [<keystore name> <properties> <certificate> <dyanamicProperties> {<Colon-Separated List of Properties>} <keygen type> <cipher type> <key size> <iterations>].`

  For example, `sudo ./encrypt-cayan.sh -g`
- 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 key size 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/`

Language

The translation files for this core should not require alteration, but if necessary then this can be accomplished by amending the relevant `Lang<CC>_<Core>.properties` within the base eftlink folder.

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

```
EftlinkConfig.properties
```

```
DisplayLanguage = EN
```

Supported country codes are CN, DE, EN, ES, FR, IT, JP, NL, PT, RU and SV.

Core Classname

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

```
EPSCore0 = manito.eft.cayan.CayanCore
```

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.

Table 3-1 Cayan - Key Settings

Setting	Description	Example
Terminal address	IP of Genius terminal.	`ced.ip = IP ADDRESS`
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.

Table 3-2 Cayan - Secondary Settings

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 =`

Table 3-2 (Cont.) Cayan - Secondary Settings

Setting	Description	Default	Example
Status Checks	Perform periodic status checks during a transaction.	false	`ced.statuschecks = false`
Status Check On Demand	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`
Admin menu	Specifies the admin menu configuration.	NA	`AdminMenu =`
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`
Signature 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`
Auto Report	Not USED.	false	`auto.report = false`
Terminal Response Timeout	Timeout used when waiting for terminal to become idle at start of order.	10	`ced.wait.idle.timeout=10`
Proxy Timeout	Timeout to connect in seconds to Cayan web service.	5	`cayan.service.connect.timeout=5`
Proxy Host	Host name to use as a proxy.	none	`https.proxyHost=adc-proxy.example.com`
Proxy Port	Port to use when using a proxy.	none	`https.proxyPort=80`

Table 3-2 (Cont.) Cayan - Secondary Settings

Setting	Description	Default	Example
Allow Duplicate in Request	Specify the value for the AllowDuplicate field in the StageTransaction Request.	false	`Allowduplicate=false`
Line Display Maximum Length	Specify maximum number of characters per line on the line display	35	`ced.item.linelength = 35`
Accept button label	Specify the label of the Agree or Accept button in a customer question/verification custom form.	YES	`CustQuestionYesLabel=YES`
Decline button label	Specify the label of the decline button in a customer question/ verification custom form.	NO	`CustQuestionNoLabel=NO`
Mask Customer Input	Specify whether to mask the customer's input in the PED for custom form.	false	`MaskCustomerInput=false`
Customer Input Max Length	Maximum number of characters when capturing data from the CED.	30	`GetCustomerInputMaxLength=30`
Phone Number Max Length	Maximum number of characters for phone capture.	10	`GetPhoneNumberMaxLength=10`
Customer Input Guidance Text Max Length	Maximum length of additional guidance text explaining what information the customer should enter.	144	`GetCustomerInputGuidanceTextMaxLength=144`
Customer Input Label Max Length	Maximum length of the label above the text entry box on the Genius device.	36	`GetCustomerInputLabelMaxLength`
Cancellable Input Types	A comma separated list of input types for custom forms that are cancellable.	SIGNATURE	`CancellableInputTypes = SIGNATURE`
Line Item Display Version	Specify the version of the routines used to update the line display on the cayan device. The latest version 4 includes the tenders on the line display. 1 Original implementation redraw all lines 2 Handle discounted items without redrawing all lines 3 Redraw on discount due to synchronization problems 4 As 3, with added display of tenders	4	ced.item.update.mode=4
truststore.ProtocolsWhiteList	Restricts the protocols which are permissible in the connection between EFTLink and EPS devices. Default allows for TLS 1.3 security.	TLSv1.3,TLSv1.2	truststore.ProtocolsWhiteList = TLSv1.3,TLSv1.2

Table 3-2 (Cont.) Cayan - Secondary Settings

Setting	Description	Default	Example
truststore.CipherWhiteList	Restricts the ciphers which are permissible in the connection between EFTLink and EPS devices. The whitelist only includes ciphers which are approved under Oracle Approved Technologies: Security Protocols	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_AES_128_GCM_SHA256,TLS_AES_256_GCM_SHA384,TLS_CHACHA20_POLY1305_SHA256,TLS_AES_128_CCM_SHA256,TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256,TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256,TLS_ECDHE_ECDSA_WITH_AES_256_CCM,TLS_ECDHE_ECDSA_WITH_AES_128_CCM	truststore.CipherWhiteList = `TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_AES_128_GCM_SHA256,TLS_AES_256_GCM_SHA384,TLS_CHACHA20_POLY1305_SHA256,TLS_AES_128_CCM_SHA256,TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256,TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256,TLS_ECDHE_ECDSA_WITH_AES_256_CCM,TLS_ECDHE_ECDSA_WITH_AES_128_CCM`
truststore.CipherBlackList	Restricts the ciphers which are not permissible in the connection between EFTLink and EPS devices. The blacklist only includes ciphers which are not approved under Oracle Approved Technologies: Security Protocols	SSL_.*,TLS_EMPTY_.*,.*_SHA,.*_3DES_.*,.*_DES_.*,.*_WITH_NULL_.*,.*_anon_.*,.*EXPORT.*,.*LOW.*,.*MD5.*,.*DES.*,.*RC2.*,.*RC4.*,.*PSK.*,TLS_DH_.*,TLS_DHE_.*	truststore.CipherBlackList = `SSL_.*,TLS_EMPTY_.*,.*_SHA,.*_3DES_.*,.*_DES_.*,.*_WITH_NULL_.*,.*_anon_.*,.*EXPORT.*,.*LOW.*,.*MD5.*,.*DES.*,.*RC2.*,.*RC4.*,.*PSK.*,TLS_DH_.*,TLS_DHE_.*`

Administration Functions

The terminal has some administration/maintenance functions. These are normally invoked from a dedicated EFT Maintenance button.

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 set up 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 Cayan core 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.

Table 3-3 Cayan - Administration Functions

Functions	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.

Table 3-4 Cayan - Supported Functions

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.

Table 3-4 (Cont.) Cayan - Supported Functions

Function	Description
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.
Custom form for customer question/verification	Sends a request to the terminal with a question/verification message. The customer selects either the Yes or No button. The core sends 'Y' or 'N' as part of the response to the POS.
Custom form for capturing phone number	Sends a request to the terminal triggering a phone number capture. The customer keys in their phone number and selects Submit. The core sends the captured phone number to the POS.
Custom form for capturing date	Sends a request to the terminal to capture a date, for example a birth date. The customer keys in their birth date and selects Submit. The core sends the captured date to the POS.
Custom form for signature capture	Sends a request to the terminal to capture signature. The customer signs and selects Accept. The core sends the decoded signature to the POS.

4

Oracle Payment Interface (OPI)

This document covers EFTLink Integration with Oracle Payment Interface (OPI) Payment Systems. It should be read in conjunction with the *Oracle Retail EFTLink Framework Installation and Configuration Guide*.

> ⓘ **Note**
>
> To avoid confusion references to OPI Retail or similar phrasing refers to Oracle Payment Interface and not Open Payment Initiative.

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 OPI interface requires a minimum EFTLink version of 20.0.

System Architecture

EFTLink connects to the OPI using only a secure HTTPS connection (using HTTP POST) and uses a Transport Layer Security (TLS) protocol version 1.2 or higher, additional restrictions to protocol and ciphers can be applied via dedicated configuration.

> ⓘ **Note**
>
> **IMPORTANT**
>
> **Please note that starting from version 24, strict validation of EPS device SSL certificates will be enforced using a dedicated truststore.**

The OPI terminal certificate chain, with root or intermediate, should be provided by the OPI partner and imported on the EFTLink truststore; also note they may be different for production and test/stage environments.

Refer to the *Oracle Retail EFTLink Framework and Configuration Installation Guide*, Securing Communication to EPS Devices by Importing Trusted Xertificates section to set up the truststore with the required trusted certificates.

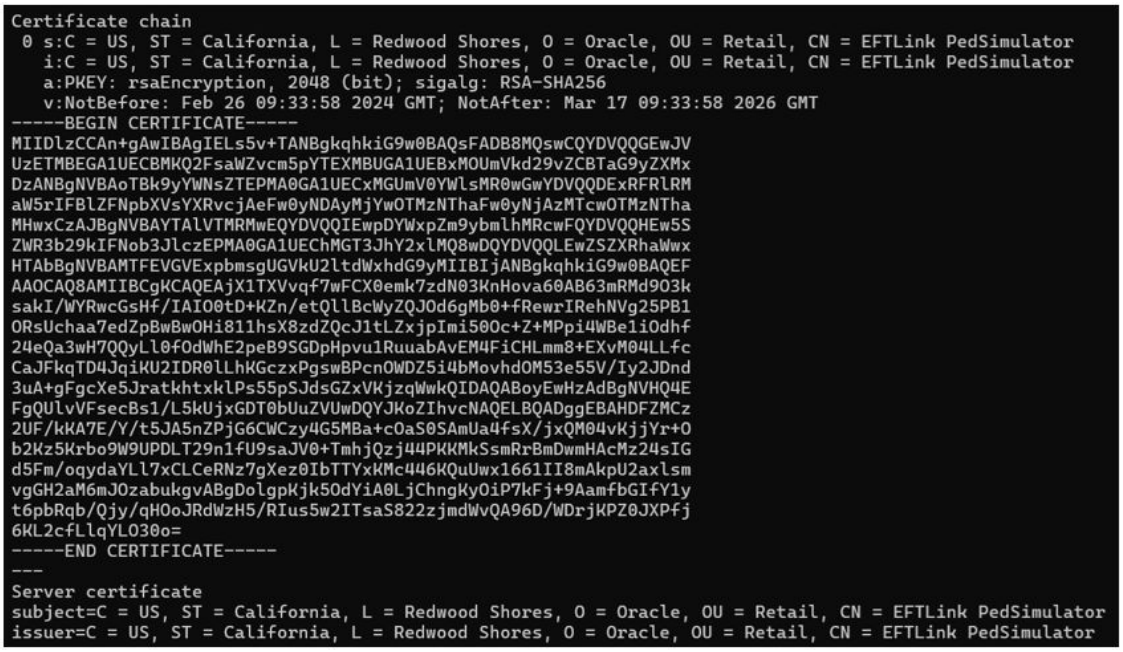
Also note that the core default configuration for protocols and ciphers may need changing for some OPI implementations. Please verify and eventually update the following configuration keys held within the opiretail.properties file:

- *truststore.ProtocolsWhiteList*
- *truststore.CipherWhiteList*
- *truststore.CipherBlackList*

The above property keys should be updated with the device supported values by contacting the PSP vendor or by using tools to inspect the SSL connection such as:

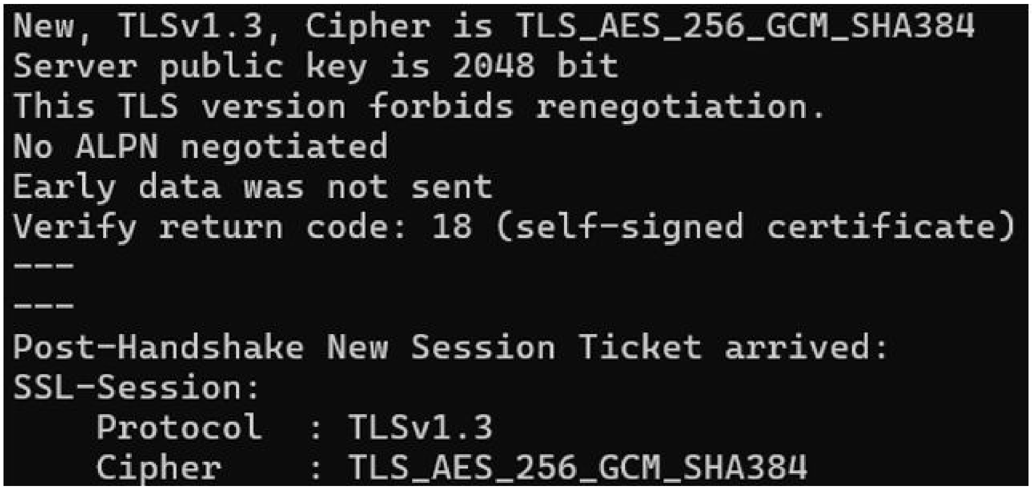
openssl s_client -showcerts -connect <terminal ip:port>

Example of a terminal certificate details by inspecting with openssl command:



```
Certificate chain
 0 s:C = US, ST = California, L = Redwood Shores, O = Oracle, OU = Retail, CN = EFTLink PedSimulator
   i:C = US, ST = California, L = Redwood Shores, O = Oracle, OU = Retail, CN = EFTLink PedSimulator
   a:PKEY: rsaEncryption, 2048 (bit); sigalg: RSA-SHA256
   v:NotBefore: Feb 26 09:33:58 2024 GMT; NotAfter: Mar 17 09:33:58 2026 GMT
-----BEGIN CERTIFICATE-----
MIIDlzCCAn+gAwIBAgIELs5v+TANBgkqhkiG9w0BAQsFADB8MQswCQYDVQQGEwJV
UzETMBEGA1UECBMKQ2FsaWZvcm5pYTEXMBUGA1UEBxMOUmVkd29vZCBTaG9yZXMx
DzANBgNVBAoTBk9yYWNsZTEPMA0GA1UECxMGUmV0YWlsMR0wGwYDVQQDExRFRlRM
aW5rIFBlZFNpbXVsYXRvcjAeFw0yNDAyMjYwOTMzNThaFw0yNjAzMTcwOTMzNTha
MHwxCzAJBgNVBAYTAlVTMRMwEQYDVQQIEwpDYWxpZm9ybmlhMRcwFQYDVQQHEw5S
ZWR3b29kIFNob3JlczEPMA0GA1UEChMGT3JhY2xlMQ8wDQYDVQQLEwZSZXRhaWwx
HTAbBgNVBAMTFEVGVExpbmsgUGVkU2ltdWxhdG9yMIIBIjANBgkqhkiG9w0BAQEF
AAOCAQ8AMIIBCgKCAQEAjX1TXVvqf7wFCX0emk7zdN03KnHova60AB63mRMd9O3k
sakI/WYRwcGsHf/IAIO0tD+KZn/etQllBcWyZQJOd6gMb0+fRewrIRehNVg25PB1
ORsUchaa7edZpBwBwOHi811hsX8zdZQcJ1tLZxjpImi50Oc+Z+MPpi4WBe1iOdhf
24eQa3wH7QQyLl0fOdWhE2peB9SGDpHpvu1RuuabAvEM4FiCHLmm8+EXvM04LLfc
CaJFkqTD4JqiKU2IDR0lLhKGczxPgswBPcnOWDZ5i4bMovhdOM53e55V/Iy2JDnd
3uA+gFgcXe5JratkhtxklPs55pSJdsGZxVKjzqWwkQIDAQABoyEwHzAdBgNVHQ4E
FgQUlvVFsecBs1/L5kUjxGDT0bUuZVUwDQYJKoZIhvcNAQELBQADggEBAHDFZMCz
2UF/kKA7E/Y/t5JA5nZPjG6CWCzy4G5MBa+cOaS0SAmUa4fsX/jxQM04vKjjYr+O
b2Kz5Krbo9W9UPDLT29n1fU9saJV0+TmhjQzj44PKKMkSsmRrBmDwmHAcMz24sIG
d5Fm/oqydaYLl7xCLCeRNz7gXez0IbTTYxKMc446KQuUwx1661II8mAkpU2axlsm
vgGH2aM6mJOzabukgvABgDolgpKjk5OdYiA0LjChngKyOiP7kFj+9AamfbGIfY1y
t6pbRqb/Qjy/qHOoJRdWzH5/RIus5w2ITsaS822zjmdWvQA96D/WDrjKPZ0JXPfj
6KL2cfLlqYLO30o=
-----END CERTIFICATE-----
---
Server certificate
subject=C = US, ST = California, L = Redwood Shores, O = Oracle, OU = Retail, CN = EFTLink PedSimulator
issuer=C = US, ST = California, L = Redwood Shores, O = Oracle, OU = Retail, CN = EFTLink PedSimulator
```

Example of the TLS protocol and cipher used by a terminal by inspecting with openssl command:



```
New, TLSv1.3, Cipher is TLS_AES_256_GCM_SHA384
Server public key is 2048 bit
This TLS version forbids renegotiation.
No ALPN negotiated
Early data was not sent
Verify return code: 18 (self-signed certificate)
---
---
Post-Handshake New Session Ticket arrived:
SSL-Session:
    Protocol  : TLSv1.3
    Cipher    : TLS_AES_256_GCM_SHA384
```

Following these examples, the reported certificate or it's root should be imported on the EFTLink truststore. The protocol *TLSv1.3* should be present in the *truststore.ProtocolsWhiteList* and *TLS_AES_256_GCM_SHA384* in the *truststore.CipherWhiteList* configuration keys in opiretail.properties to allow EFTLink to connect to the terminal device.

Please always contact the OPI partner for any questions about production and test terminal certificates, TLS protocols, and ciphers.

> **Note**
>
> The certificates and ciphers used by OPI implementations may change over time. Please contact your OPI partner for details on how they manage this.

Fileset

The following files are used in the EFTLink folder:

- `cores/opiretail/opiretail.jar`
- `opiretail.properties` (optional, if not present defaults apply)
- `Lang<CC>_<Core>.properties` – Language translation file, for further information see Language.

Language

The translation files for this core should not require alteration, but if necessary then this can be accomplished by amending the relevant `Lang<CC>_<Core>.properties` within the base eftlink folder.

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

```
EftlinkConfig.properties
```

```
DisplayLanguage = EN
```

Supported country codes are CN, DE, EN, ES, FR, IT, JP, NL, PT, RU and SV.

Core Classname

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

```
EPSCore0= oracle.eftlink.opiretail.OPIRetailCore
```

Multiple Core Folder Structure Settings

The OPI Core can be configured to run multiple instances of itself. Therefore, each iteration is required to run under its own subfolder within the EFTLink installation directory. For example, `C:\EFTLink\EPSCore_OPI_0 and C:\EFTLink\EPSCore_OPI_1`.

This allows unique `opiretail.properties` files to be held and configured for each instance of the core.

EFTLink needs to be informed what the working folder names are and to achieve this, the following configuration changes are required within the `EFTLinkConfig.properties` file.

The property key `EPSCore<n>` value, can be configured to pass through parameters (comma separated) by declaring them after the cores full package name. For example: `EPSCore<n> =< Full package name><space><Parameter><comma><Parameter>…)`

A parameter in this case is a property key/value pair (colon separated) to specify the cores working folder. For example:

```
EPSCore0 = oracle.eftlink.opiretail.OPIRetailCore WorkingFolder:./ EPSCore_OPI_0
```

```
EPSCore1 = oracle.eftlink.opiretail.OPIRetailCore WorkingFolder:./ EPSCore_OPI_1
```

Please refer to **Multiple Core Feature** section within the *Oracle Retail EFTLink Framework Advanced Features Guide* for further information on multiple core use cases.

Configuration Settings

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

Key Settings

Settings that may be different for each POS.

Table 4-1 OPI - Key Settings

Setting	Description	Default	Example
EPSAddress	Specifies the host address of the EPS.	localhost	`EPSAddress = IP ADDRESS`
EPSPort	Specifies the host port of the EPS.	5007	`EPSPort = 5007`
EPSName	Specifies the name of the EPS. This is typically used to identify the PaymentProviderName during Tax Free processing.	Blank	`EPSName = [PaymentProviderName]` where the PaymentProviderName is any supported provider.
truststore.trustAllServers	Disable validation of certificates with the truststore on connection between EFTLink and OPI devices. Please note, disabling validation of the trusted certificate is never a recommended practice. Ciphers and protocols will still continue to be restricted as configured below.	false	truststore.trustAllServers = false
truststore.ProtocolsWhiteList	Restricts the protocols which are permissible in the connection between EFTLink and EPS devices. Default allows for TLS 1.3 or 1.2 security.	TLSv1.3,TLSv1.2	truststore.ProtocolsWhiteList = TLSv1.3,TLSv1.2

Table 4-1 (Cont.) OPI - Key Settings

Setting	Description	Default	Example
truststore.CipherWhiteList	Restricts the ciphers which are permissible in the connection between EFTLink and EPS devices. The whitelist only includes ciphers which are approved under Oracle Approved Technologies: Security Protocols	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_AES_128_GCM_SHA256,TLS_AES_256_GCM_SHA384,TLS_CHACHA20_POLY1305_SHA256,TLS_AES_128_CCM_SHA256,TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256,TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256,TLS_ECDHE_ECDSA_WITH_AES_256_CCM,TLS_ECDHE_ECDSA_WITH_AES_128_CCM	truststore.CipherWhiteList = TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_AES_128_GCM_SHA256,TLS_AES_256_GCM_SHA384,TLS_CHACHA20_POLY1305_SHA256,TLS_AES_128_CCM_SHA256,TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256,TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256,TLS_ECDHE_ECDSA_WITH_AES_256_CCM,TLS_ECDHE_ECDSA_WITH_AES_128_CCM
truststore.CipherBlackList	Restricts the ciphers which are not permissible in the connection between EFTLink and EPS devices. The blacklist only includes ciphers which are not approved under Oracle Approved Technologies: Security Protocols	SSL_.*,TLS_EMPTY_.*,.*_SHA,.*_3DES_.*,.*_DES_.*,.*_WITH_NULL_.*,.*_anon_.*,.*EXPORT.*,.*LOW.*,.*MD5.*,.*DES.*,.*RC2.*,.*RC4.*,.*PSK.*,TLS_DH_.*,TLS_DHE_.*	truststore.CipherBlackList = SSL_.*,TLS_EMPTY_.*,.*_SHA,.*_3DES_.*,.*_DES_.*,.*_WITH_NULL_.*,.*_anon_.*,.*EXPORT.*,.*LOW.*,.*MD5.*,.*DES.*,.*RC2.*,.*RC4.*,.*PSK.*,TLS_DH_.*,TLS_DHE_.*
DefaultOperatorId	Specifies a default operator id for POS systems that do not provide an operator id in the CardServiceRequest.	EFTLink OPI Operator	DefaultOperatorId = EFTLink OPI Operator

Table 4-1 (Cont.) OPI - Key Settings

Setting	Description	Default	Example
DefaultBaseCurrency	Specifies the default base currency to be used as part of the TransCurrency element in the OPI Retail messages.	GBP	`DefaultBaseCurrency = GBP`
DefaultCheckType	Specifies the default check type for check processing.	01	`DefaultCheckType = 01`
DefaultCheckName	Specifies the default check name for check processing.	Personal Check	`DefaultCheckName = Personal Check`
SiteId	Specifies the SiteId data which is required in every Oracle Payments Interface request.	null	`SiteId = Site`
ProxyInfo	Override the ProxyInfo value reported on every Oracle Payments Interface request.	null	`ProxyInfo = OPI Version 25.1`
POSInfo	Specifies the POSInfo data which is required for every Oracle Payments Interface request.	null	`POSInfo = Test POS`
PartialAuthEnabled	Specifies whether partial authorization is enabled.	false	`PartialAuthEnabled = true`
ElectronicSignatureEnabled	Specifies whether Electronic Signature processing is enabled.	true	`ElectronicSignatureEnabled = true`
GiftCardProcessingEnabled	Specifies whether Gift Card processing is enabled.	true	`GiftCardProcessingEnabled = true`
PersonalCheckProcessingEnabled	Specifies whether Personal Check/Cheque processing is enabled.	true	`PersonalCheckProcessingEnabled = true`
LineDisplayEnabled	Specifies whether Line Item Display processing is enabled.	false	`LineDisplayEnabled = true`
ElectronicJournal	Specifies whether to add journal attributes to print lines.	false	`ElectronicJournal = true`
CombinedReceipt	Specifies whether to defer printing of the EFT customer receipt and instead include within the standard POS customer receipt. CombinedReceiptFilter_X and DoNotCombineCustomerReceiptForDecline settings become effective when this setting is true.	false	`CombinedReceipt = true`

Table 4-1 (Cont.) OPI - Key Settings

Setting	Description	Default	Example
EWalletProcessingEnabled	Specifies whether to enable EWallet processing. EWalletIssuerIds setting becomes effective when this setting is true.	true	`EWalletProcessingEnabled = true`
InstallmentsEnabled	Specifies whether to enable installment payments functionality. MaxInstallmentsAllowed, DebitCreditSelectionPromptTimeout, NoOfInstallmentsPromptTimeout, NoOfInstallmentsPromptRetries settings become effective when this setting is true.	false	`InstallmentsEnabled = true`
LoyaltyBrandsEnable	Specifies whether to enable Loyalty Brands.	false	`LoyaltyBrandsEnabled = true`
AllowedLoyaltyBrands	A comma-separated list of allowed loyalty brands. Dependent on property LoyaltyBrandsEnabled.		`AllowedLoyaltyBrands = AppleVas,Google Smarttap`
CardAcquisitionEnabled	Specifies whether to enable Card Acquisition message processing prior to Sale/ Purchase message processing for 2-stage payment.	false	`CardAcquisitionEnabled = true`
RefRefundUseCardTokenEnabled	Specifies whether to use the Card Token to perform refund requests instead of the OriginalRRN/ AcquirerTransactionReference.	false	`RefRefundUseCardTokenEnabled = false`
TokenizeAnonymousCardsEnabled	Specifies whether to always tokenize the card regardless of the existence of a CustomerId on the Payment CardServiceRequest.	false	`TokenizeAnonymousCardsEnabled = true`

Table 4-1   (Cont.) OPI - Key Settings

Setting	Description	Default	Example
QuickChipEnabled	Specifies whether to enable Quick Chip functionality to allow the customer to pre-authorize their card prior to sale tendering. QuickChipAverageTransactionValue setting becomes effective when this setting is true. Warning: This setting will not be in effect unless LineDisplayEnabled is true and a value greater than 0 is set in the QuickChipAverageTransactionValue.	false	`QuickChipEnabled = false`
UseLegacyTokenLogicEnabled	Specifies whether to use legacy token logic processing. This ensures the RRN is set on the CardServiceResponse.CardValue.Token object so that linked refunds will use the CardServiceRequest.CardValue.Token for processing as the OriginalRRN. Warning: this setting disables/disallows the ability to pay by stored card token/ vault card token.	false	`UseLegacyTokenLogicEnabled = true`
PayByLinkExpiresAtHours	Defines how many hours a generated URL should be considered valid.	24	`PayByLinkExpiresAtHours = 24`
LegacyXstoreCopyCardTokenToOrigAltTransRef	Enable when using Xstore pre V19.0.1 and Cardless Reference Refund required when moving from POS Jni Core to OPI Core. When process a Referenced Refund, it will copy CSRequest.Card.Token to OPI Refund Request OrigAltTransRef if CSRequest.TransactionReference is not set.	false	`LegacyXstoreCopyCardTokenToOrigAltTransRef = true`

Table 4-1 (Cont.) OPI - Key Settings

Setting	Description	Default	Example
CustomFormsUseSeparateTextLines	Define whether the core will move the separate TextLines in the DeviceRequest from PromptText to PromptText2, PromptText3, PromptText4 & PromptText5 Note, when obtaining Signature, these additional fields will be Title2, Title3, Title4 & Title5	false	`CustomFormsUseSeparateTextLines = true`
TerminalMessagesEnabled	Define whether the core will connect to PED to receives terminale messages. PSP terminal should support this functionality.	False	`TerminalMessagesEnabled=false`
TerminalMessagesEndpoint	When enabled, PED wss endpoint used to receive terminal message. PSP vendor should provide the address.	Empty	`TerminalMessagesEndpoint=wss://<terminal_ip:port>/path`
TerminalMessagesConnectionTimeoutMillis	When enabled, connection timeout in milliseconds to TerminalMessagesEndpoint.	3000	`TerminalMessagesConnectionTimeoutMillis=3000`
EPSTerminalID	Specify the unique name for EPS Terminal Id.	Empty	`EPSTerminalID = EPS1`
EnableSettingOfTerminalId	Whether to use the EPS terminal Id.	False	`EnableSettingOfTerminalId = false`

Secondary Settings

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

Table 4-2 OPI - Secondary Settings

Setting	Description	Default	Example
RequestResponseTimeout	Specifies the timeout when sending / receiving messages to / from the Oracle Payments Interface in seconds.	180	`RequestResponseTimeout = 200`
DetectReceiptSignatur eString	Specifies the text to find in the print data returned from the Oracle Payments Interface response in order to determine whether a signature check prompt is required for the request.	Signature	`DetectReceiptSignatureString = # Signature #`
ValidateMessaging	Specifies whether to validate all requests / responses against their respective XSDs.	false	`ValidateMessaging = true`

Table 4-2 (Cont.) OPI - Secondary Settings

Setting	Description	Default	Example
MaintenanceTimeout	Specifies the timeout for the core maintenance menu in seconds.	60	`MaintenanceTimeout = 30`
SignatureCheckTimeout	Specifies the timeout for the Signature Check prompt when required in seconds.	30	`SignatureCheckTimeout = 15`
BusyErrorText	Specifies the error text when the device is busy.	Device error retry	`BusyErrorText = Busy Device Error`
ReadResponseBuffer	Specifies a minimum buffer amount to allocate space in memory as a rough approximation of the expected content length of an OPI Retail Response in bytes.	1024	`ReadResponseBuffer = 1024`
MaxLineItemTextLength	Specifies the max length of an Item Description in characters. This is used to truncate the length of the item description in case the description of a product is too long during line item display on the pin entry device (PED) and on S/P request payload.	64	`MaxLineItemTextLength = 64`
CombinedReceiptFilter_X	Specifies custom filtering of information on the customer receipt. Replace X with a number between 0 and 100. A maximum of 100 filters are allowed and <blank> checks if empty lines should be suppressed.	blank	`CombinedReceiptFilter_0 = <blank>`
SuppressMerchantCopyForDecline	Specifies whether to suppress the printing of the merchant receipt transactions are declined.	false	`SuppressMerchantCopyForDecline = false`
DoNotCombineCustomerReceiptForDecline	Specifies whether to combine the customer receipt with the POS receipt when transactions are declined.	false	`DoNotCombineCustomerReceiptForDecline = false`
CardAcquisitionPromptTimeout	Specifies the timeout for the CardInserted DeviceRequest prompt on the POS for the specified BinRange and CountryCode sent from the TransactionResponse in seconds.	1200	`CardAcquisitionPromptTimeout = 1200`
GetCustomerVerificationAcceptLabel	Specifies the text label of the accept button for the Get Customer Verification custom form.	Yes	`GetCustomerVerificationAcceptLabel = No`

Table 4-2 (Cont.) OPI - Secondary Settings

Setting	Description	Default	Example
GetCustomerVerificationDeclineLabel	Specifies the text label of the decline button for the Get Customer Verification custom form.	No	`GetCustomerVerificationDeclineLabel = No`
DisplayMessageDuration	Specifies the timeout duration of the display message custom form in seconds.	30	`DisplayMessageDuration = 60`
GetPhoneNumberUseMaxLength	Specifies whether to use max length instead of the regex for phone number capture custom form.	true	`GetPhoneNumberUseMaxLength = true`
GetPhoneNumberMaxLength	Specifies the maximum number of digits for the phone number capture custom form.	10	`GetPhoneNumberMaxLength = 12`
GetPhoneNumberRegex	Specifies the regular expression for the phone number capture custom form.	\d{3}-\d{3}-\d{4}	`GetPhoneNumberRegex = \d{3}-\d{3}-\d{4}`
GetPhoneUseType63	Whether to use the transaction type 63 GetAlphaNumeric instead of the GetNumericField for capture Phone from PED.	false	`GetPhoneUseType63 = false`
GetSSNUseMaxLength	Specifies whether to use max length instead of the regex for social security number capture custom form.	false	`GetSSNUseMaxLength = true`
GetSSNMaxLength	Specifies the max length of the social security number capture custom form.	9	`GetSSNMaxLength = 9`
GetSSNRegex	Specifies the regular expression for the social security number capture custom form.	\d{3}-\d{2}-\d{4}	`GetSSNRegex = \d{3}-\d{2}-\d{4}`
GetEmailAddressMaxLength	Specifies the max length of the email address capture custom form.	50	`GetEmailAddressMaxLength = 100`
GetDriverLicenseMaxLength	Specifies the max length of the driver license capture custom form.	20	`GetDriverLicenseMaxLength = 20`
GetNumericFieldUseMaxLength	Specifies whether to use max length instead of a regex for the numeric field custom form.	true	`GetNumericFieldUseMaxLength = true`
GetNumericFieldMaxLength	Specifies the max length of the numeric field capture custom form.	50	`GetNumericFieldMaxLength = 50`
GetNumericFieldRegex	Specifies the regular expression for the numeric field custom form. Default is empty.		`GetNumericFieldRegex = \d{3}-\d{2}-\d{4}`

Table 4-2 (Cont.) OPI - Secondary Settings

Setting	Description	Default	Example
GetAlphanumericFieldUseMaxLength	Specifies whether to use max length instead of a regex for the alpha numeric field custom form.	true	`GetAlphanumericFieldUseMaxLength= true`
GetAlphaNumericFieldMaxLength	Specifies the max length of the alpha numeric field capture custom form.	50	`GetAlphaNumericFieldMaxLength = 50`
GetAlphaNumericFieldRegex	Specifies the regular expression for the alpha numeric field custom form. Default is empty.		`GetAlphaNumericFieldRegex = \d{3}-\d{2}-\d{4}`
DisplayQRCodeButtonLabel	Specifies the label of the button on the QR code custom form.	Done	`DisplayQRCodeButtonLabel = Cancel`
GetPhoneNumberGuidanceText	Specifies the guidance text when capturing a phone number. Default is empty.		`GetPhoneNumberGuidanceText = Please input your phone number below in the form of XXX-XXX-XXXX (replace X with a number between 0-9)`
GetEmailAddressGuidanceText	Specifies the guidance text when capturing an email address. Default is empty.		`GetEmailAddressGuidanceText = Please input your email address below, for example; test@oracleretail.com.`
GetSSNGuidanceText	Specifies the guidance text when capturing a social security number. Default is empty.		`GetSSNGuidanceText = Please input your social security number below, for example; 078-05-1120`
GetDateGuidanceText	Specifies the guidance text when capturing a date of birth. Default is empty.		`GetDateGuidanceText = Please input your date of birth below, for example; 1971-01-01 (YYYY-MM-DD)`
GetDriverLicenseGuidanceText	Specifies the guidance text when capturing a driver license number. Default is empty.		`GetDriverLicenseGuidanceText = Please input your driving license below, for example; 123 456 789`
GetNumericFieldGuidanceText	Specifies the guidance text when capturing generic numeric data. Default is empty.		`GetNumericFieldGuidanceText = Please input your loyalty number below, for example; 123-456-7891`

Table 4-2 (Cont.) OPI - Secondary Settings

Setting	Description	Default	Example
GetAlphanumericFieldGuidanceText	Specifies the guidance text when capturing generic alpha-numeric data. Default is empty.		`GetAlphanumericFieldGuidanceText = Please enter your name below incl. middle names, for example; Joe M Bloggs`
MaxInstallmentsAllowed	Specifies the maximum number of installments allowed per transaction. If the entered value on the installments prompt exceeds the MaxInstallmentsAllowed value, the installments prompt will retry until the configured NoOfInstallmentsPromptRetries amount is reached.	24	`MaxInstallmentsAllowed = 30`
NoOfInstallmentsPromptRetries	Specifies the number of installments prompt retries.	3	`NoOfInstallmentsPromptRetries = 1`
NoOfInstallmentsPromptTimeout	Specifies the timeout in seconds of the number of installments prompt.	600	`NoOfInstallmentsPromptTimeout = 60`
DebitCreditSelectionPromptTimeout	Specifies the timeout in seconds of the prompt between Debit and Credit card type.	600	`DebitCreditSelectionPromptTimeout = 60`
GiftCardPinEntryOnPOSEnabled	Specifies whether to enable processing of Gift Card Pins from the POS. Do not use this setting in conjunction with GiftCardPinEntryOnPEDEnabled=true as GiftCardPinEntryOnPOSEnabled will take precedence.	false	`GiftCardPinEntryOnPOSEnabled = true`
GiftCardPinEntryOnPEDEnabled	Specifies whether to request a PIN for the supplied Gift Card on the PED. Do not use this setting in conjunction with GiftCardPinEntryOnPOSEnabled=true as GiftCardPinEntryOnPOSEnabled will take precedence.	false	`GiftCardPinEntryOnPEDEnabled = true`
GiftCardPinEntryTypes	Specifies which OPI Retail Gift Card Transactions should apply Gift Card PIN processing. This is a comma-delimited string, and the values must map to the OPI Retail TransType equivalents.	27,28,29,30	`GiftCardPinEntryTypes = 27,28,29,30`

Table 4-2   (Cont.) OPI - Secondary Settings

Setting	Description	Default	Example
GiftCardPinEntryMinimumLength	Specifies the minimum length of the PIN to be entered on the POS.	4	`GiftCardPinEntryMinimumLength = 3`
GiftCardPinEntryMaximumLength	Specifies the maximum length of the PIN to be entered on the POS.	4	`GiftCardPinEntryMaximumLength = 6`
GiftCardPinEntryRetries	Specifies the maximum amount of retries to attempt on the POS if the default maximum length of the PIN in the OPI specification is exceeded. This property may be used in the event that the GiftCardPinEntryMinimumLength and GiftCardPinEntryMaximumLength settings are not set.	3	`GiftCardPinEntryRetries = 5`
GiftCardProviders	Specifies the gift card provider to use, when a single entry is configured. If a list of providers is specified, the cashier will be prompted to select the provider from the list. If no GiftCardProvider is specified, then this entry will not be used, and no value is passed to the EPS in the ProviderId element of the TransactionRequest. This is a comma-delimited string.	blank	`GiftCardProviders = SVS, GIVEX`
GiftCardProvidersPromptTimeou	Specifies the timeout in seconds to be used for the Gift Card Provider selection prompt.	1200	`GiftCardProvidersPromptTimeout==1200`
EnforceCardNameSetByOPI	If true, the card name returned in the CardServiceResponse adheres to the mappings specified by the OPI spec and is not overridden by any mapping performed within the cardrange.xml.	false	`EnforceCardNameSetByOPI = true`
QuickChipAverageTransactionValue	Specifies the average quickchip transaction value. This value determines the PED prompts in the transaction (for example NFC).	20	`QuickChipAverageTransactionValue = 20`
InvoicePaymentPromptForCardOrCashEnabled	Specifies whether to prompt for Card or Cash at the POS for Invoice Payment transactions.	false	`InvoicePaymentPromptForCardOrCashEnabled = true`

Table 4-2 (Cont.) OPI - Secondary Settings

Setting	Description	Default	Example
InvoicePaymentPromptForCardOrCashTimeout	Specifies the timeout in seconds of the prompt for Card or Cash for Invoice Payments transactions.	600	`InvoicePaymentPromptForCardOrCashTimeout = 720`
InvoicePaymentPromptScanBarcodeRetries	Specifies the number of Scan Barcode prompt retries.	3	`InvoicePaymentPromptScanBarcodeRetries = 6`
CellPhoneRechargePromptForCardOrCashEnabled	Specifies whether to prompt for Card or Cash at the POS for Cell Phone Recharge transactions.	false	`CellPhoneRechargePromptForCardOrCashEnabled = true`
CellPhoneRechargePromptForCardOrCashTimeout	Specifies the timeout in seconds of the prompt for Card or Cash for Cell Phone Recharge transactions.	600	`CellPhoneRechargePromptForCardOrCashTimeout = 720`
MerchantReference	Unique merchant reference.	N/A	`MerchantReference=Merchant A`
MerchantReferenceFormat	Specify the format of the merchant reference - replaces static value with a dynamically generated value using several substitutions. See Merchant Reference Formats.	R (use existing static merchant ref)	`MerchantReferenceFormat=R-dddddddddd-SSSSSS.WW WWWW.YYYYMMDD.hhmmss.TTTTTT.qq`
CapCancellation	Determines response for the getCapCancellation request.	true	`CapCancellation = false`
CapReversal	Determines response for the getCapReversal request	true	`CapReversal = false`
MerchantReferenceSpecialChar	Specify a character that is to be passed through 'as is' in addition to the ones already mentioned.	N/A	`MerchantReferenceSpecialChar = K`
PrintDataLogMasking	When enabled force masking on log of receipt information on <PrintData> tag	false	`PrintDataLogMasking = false`
AlternativePrintDataOrder	If true, then the OPICore expects the customer receipt copy first and the merchant receipt copy second inside the PrintData field within the transaction response.	false	`AlternativePrintDataOrder = true`
SaleTransactionRequestLineItemEnabled	If false, this will suppress the LineItem Field within the S/P Transaction request.	true	SaleTransactionRequestLineItemEnabled = false
UsePspPaymentBrandForCardCircuit	If true, and a value exists in the pspPaymentBrand field, then this value is mapped to the CardCiruit field in the cardServiceResponse.	false	UsePspPaymentBrandForCardCircuit = true

Table 4-2 (Cont.) OPI - Secondary Settings

Setting	Description	Default	Example

AdminMenu0.<X>	Specifies the ability to customize the Admin Menu. Replace X with a value between 0 - 2.	AdminMenu0.1 = TXT_RECONCILIATION, Reconciliation
		AdminMenu0.2 = TXT_CANCEL, Cancel
			AdminMenu0.1 = TXT_RECONCILIATION, Reconciliation
			AdminMenu0.2 = TXT_CANCEL, Cancel

IssuerId_<31 - 60>	Allows the modification to the default IssuerId values set in the OPI Spec.	Defaults match the values set by the OPI Spec	Issuerid31 = Bluebird

Merchant Reference Formats

For the merchant reference format, the following substitutions are available:

Table 4-3 Merchant Reference Formats

Component	Description	Example	Format
R	Use existing Merchant Reference	R	
S	StoreID	SSSSSS	min 3, max 10 chars, left 0 filled
W	WorkStation id	WWWWWW	min 3, max 20 chars, left 0 filled
YY	Year	YY or YYYY	extracted from POSTimeStamp
MM	Month	MM	extracted from POSTimeStamp
DD	Day	DD	extracted from POSTimeStamp
hh	Hour	hh	extracted from POSTimeStamp
mm	Minute	mm	extracted from POSTimeStamp
ss	Second	ss	extracted from POSTimeStamp
T	Transaction number	TTTTTT	min 3, max 20 chars, left 0 filled
d	Transaction date	dddddddddd	must be 10 chars

The following special characters are also allowed:

minus -
underscore _
period .

Example format:

```
R-dddddddddd-SSSSSS_WWWWWW.YYYYMMDD.hhmmss.TTTTTT.qq
```

Administration Functions

The terminal has some administration/maintenance functions. These are normally invoked from a dedicated EFT Maintenance button.

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

Table 4-4 OPI - Administration Functions

Function	Description
Day End	Print an end day report and close the current day. Manual alternative to automated ReconciliationWithClosure.

Supported Functions

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

Table 4-5 OPI - Supported Functions

Function	Description
Payment/Payment with Loyalty	EFTLink sends payment request to the OPI EPS. The OPI EPS will return a response message with formatted receipt strings for merchant and/or customer receipts. In an event of referral or communication failure where authorization cannot be obtained online then a prompt for authorization code will appear; authorization code must be obtained via telephone. After the manual authorization process is complete, a Sales Completion transaction is sent to finalize the original sale/purchase. In the event of a communication failure between EFTLink and the OPI EPS an initial Transaction Inquiry request is sent to the OPI EPS to determine if communication is back up. If communication is still down, the cashier must decide whether to retry or decline the transaction. In the event of a retry; a Transaction Inquiry message is sent and if a response is returned from the OPI EPS, the Transaction Inquiry response is used to finalize the sale/purchase. In the event of a decline, the OPI will initially attempt to reverse the transaction, however if communication is still down, the OPI will simply fail the transaction altogether and the POS will return to the tender selection screen.
Payment by Installments	The POS initiates a credit/debit tender, and the cashier is prompted as to whether the customer is paying by a Debit or Credit Card. If the customer is paying by Credit Card, the cashier will choose the number of installments to apply (if the customer decides to pay in installments). EFTLink receives the request from the POS and sends the payment request to the OPI EPS. The customer will proceed with the payment, and this will allow the customer to be able to spread out the cost of the purchase over a number of payments.
Payment by Card Token	EFTLink sends a payment request to the OPI EPS with the (Stored) card token value set in the request. The OPI EPS utilizes the card token value to process the sale without customer interaction required.

Table 4-5 (Cont.) OPI - Supported Functions

Function	Description
Payment by Quick Chip	EFTLink receives a Card Acquisition request from the POS at the start of the transaction. EFTLink sends this Card Acquisition request to the OPI EPS. This allows the customer to interact with the pin entry device while the POS operator can continue to ring items through the till. When the POS tenders' credit/debit at the end of the transaction; EFTLink will send a payment request with the Card Acquisition details associated with the Card Acquisition at the beginning of the transaction. The OPI EPS will process the card details in the payment request without customer interaction required.
Check Payment	EFTLink sends payment request to the OPI EPS. The OPI EPS will return a response message with formatted receipt strings for merchant and/or customer receipts. The Transaction Inquiry scenario outlined in the Payment/Payment with Loyalty section also applies to this transaction type.
Refund	EFTLink sends refund requests to the OPI EPS. The OPI EPS will refund a transaction with the specified amount. The Transaction Inquiry scenario outlined in the Payment/Payment with Loyalty section also applies to this transaction type.
Refund with Card Token	EFTLink sends a refund request to the OPI EPS with the (Stored) card token value set in the request. The OPI EPS utilizes the card token value to process the refund without customer interaction required.
Reversal	EFTLink sends reversal requests to the OPI EPS. The OPI EPS will reverse a transaction specified by the original transaction reference. The Transaction Inquiry scenario outlined in the Payment/Payment with Loyalty section also applies to this transaction type.
Sale State Notifications	EFTLink sends line items through to the OPI EPS so that the customer display can be updated on the terminal in line with the POS.
Cancel Current Transaction	POS sends an abort request to EFTLink and if a transaction is cancellable, it is cancelled.
Read Non-PCI Card	EFTLink sends a card swipe request to the OPI EPS to receive data for non-pci cards. The full pan is returned in clear text, unencrypted and without tokenization. PCI cards will return a blank PAN.
Electronic Signature capture on PED	EFTLink sends a purchase, refund, svc payment, svc unload (cashout) or check authorization request to the OPI EPS. The signature can be captured on the pin entry device for the request provided and this signature will be confirmed by the POS operator and processed accordingly. The Transaction Inquiry scenario outlined in the Payment/Payment with Loyalty section also allows prompts for signature capture (if configured on the pin entry device). The Manual Authorization scenario outlined in the Payment/Payment with Loyalty section also allows prompts for signature capture (if configured on the pin entry device).

Table 4-5 (Cont.) OPI - Supported Functions

Function	Description
SVC Payment	EFTLink sends a gift or merchandise credit card payment request to the OPI EPS. 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. The Transaction Inquiry scenario outlined in the Payment/Payment with Loyalty section also applies to this transaction type.
SVC Activate	EFTLink sends a gift or merchandise credit card activation request to the OPI EPS. The Transaction Inquiry scenario outlined in the Payment/Payment with Loyalty section also applies to this transaction type.
SVC Add Value	EFTLink sends a gift or merchandise credit card add value request to the OPI EPS. This will only add value to an account that has been activated. The Transaction Inquiry scenario outlined in the Payment/Payment with Loyalty section also applies to this transaction type.
SVC Balance Enquiry	EFTLink sends a gift or merchandise credit card balance enquiry request to the OPI EPS.
SVC Unload (Cashout)	EFTLink sends a gift or merchandise credit card cash out request to the OPI EPS. 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. The Transaction Inquiry scenario outlined in the Payment/Payment with Loyalty section also applies to this transaction type.
SVC Reversal	EFTLink sends a gift or merchandise credit card activate/add value/ payment to the OPI EPS which is voided, or post voided and the original transaction actions are reversed. The Transaction Inquiry scenario outlined in the Payment/Payment with Loyalty section also applies to this transaction type.
Custom form for customer question/verification	EFTLink sends a request to the OPI EPS with a question/verification message. The customer selects either the Yes or No button. The core sends 'Y' or 'N' as part of the response to the POS.
Custom form for capturing phone number	EFTLink sends a request to the OPI EPS triggering a phone number capture. The customer keys in their phone number and selects submit. The core sends the captured phone number to the POS.
Custom form for capturing date	EFTLink sends a request to the OPI EPS to capture a date, for example a birth date. The customer keys in their birth date and selects submit. The core sends the captured date to the POS.
Custom form for signature capture	EFTLink sends a request to the OPI EPS to capture signature. The customer signs and selects Accept. The core sends the decoded signature to the POS.

Table 4-5 (Cont.) OPI - Supported Functions

Function	Description
Custom form for any alphanumeric data capture	EFTLink sends a request to the OPI EPS to capture any data which could be alphanumeric. A prompt is displayed regarding the type of data expected. The customer keys in the relevant data and selects submit. The core sends the data back to the POS.
Custom form for a survey or donation selection	EFTLink sends a request to the OPI EPS to capture data in the form of either buttons or radio buttons. The customer can choose between a list of buttons or radio buttons which have different responses or amounts and selects submit. The core sends the value of the button pressed back to the POS.
Custom form for displaying a message	EFTLink sends a request to the OPI EPS to display a message to the customer. The message times out after a configurable amount of time.
Custom form for displaying a scannable QR code	EFTLink sends a request to the OPI EPS to display a scannable QR code to the customer. The message times out after a configurable amount of time.
E-Wallet payments for example, WeChat Pay and AliPay	**Flow 1** - Customer initiated transaction via E-Wallet button press on the PED. EFTLink sends a standard sale/purchase request to the OPI EPS. The customer selects the button to pay via their E-Wallet (as opposed to the usual chip and pin, swipe and other card payment methods) on the PED. The OPI EPS returns a response containing the E-Wallet data. EFTLink feeds this data back to the POS to complete the transaction. **Flow 2** - Cashier initiated transaction via E-Wallet tenders on the POS. POS tenders to pay the transaction via E-Wallet tender. EFTLink sends a sale/purchase message to the OPI EPS, specifying that the PaymentMethod is E-Wallet. The OPI EPS displays a QR code which the customer scans with their E-Wallet device (typically a mobile phone). The transaction is confirmed on the OPI EPS and the WalletAuthorizationData is returned via EFTLink to the POS to complete the transaction. **Flow 3** - Cashier initiated transaction via E-Wallet tenders on the POS - Customer QR code scanned on the POS. POS tenders to pay the transaction via E-Wallet tender. Cashier scans customer's E-Wallet QR code. EFTLink sends a sale/purchase message to the OPI EPS, specifying that the PaymentMethod is E-Wallet and the WalletId of the customer's E-Wallet. The transaction is confirmed on the OPI EPS and the WalletAuthorizationData is returned via EFTLink to the POS to complete the transaction.

Table 4-5 (Cont.) OPI - Supported Functions

Function	Description
Two-stage payments for example, tax-free shopping	EFTLink sends a Card Acquisition request to the OPI EPS prior to a sale/purchase request. The OPI EPS reads the customer's card details and responds with the BIN range and Country Code of the customer's card to determine (on the POS) whether this customer is eligible for tax free shopping. If so, the POS processes the tax-free refund and modifies the total transaction amount with the tax removed and sends the new amount to EFTLink. EFTLink sends a sale/purchase request with the new modified amount and the original transaction reference to the OPI EPS. The OPI EPS looks up the original card details from the transaction reference and charges the card with the modified amount to complete the transaction.
Invoice payments	The POS adds a non-merch item “Invoice Payment” which initiates an Invoice Payment transaction. The cashier is prompted as to whether the customer would like to pay by “Cash” or “Card” (If configured). The cashier is further prompted to “Please scan the invoice barcode”. The cashier may key or scan the invoice barcode. EFTLink receives the request from the POS and sends the “Invoice Payment” request to the OPI EPS. The customer will proceed with the payment at this point. If the customer selected to pay by “Cash” during the initial prompt on the POS, the OPI EPS will simply clear the invoice as paid and send back the successful response. The following prompts are skippable: 1. The customer will input on the PED the amount of the invoice they would like to pay in whichever currency the PED is configured. If the customer selected to pay by “Card” during the initial prompt on the POS, the PED will prompt as to whether or not the customer would like to pay by “Cash”, “Credit” or “Debit”. The customer may choose how to pay at this point. If the customer selects “Cash”, the invoice will be cleared as paid. If the customer selects “Credit” or “Debit” card, the usual steps will take place for a typical Card transaction. Invoice payments are not reversible at this time. However, if a card payment was made, the cashier may choose to reverse as a standard card payment. It is assumed that the merchant will have the necessary measures in place to avoid issues with “Cash” payments (in terms of a customer not having enough cash to pay for the transaction and so on). The Transaction Inquiry scenario outlined in the Payment/Payment with Loyalty section also applies to this transaction type. .

Table 4-5   (Cont.) OPI - Supported Functions

Function	Description
Cell phone recharge (Top-up)	The POS adds a non-merch item “Cell Phone Recharge” which initiates a Cell Phone Recharge transaction. The cashier is prompted as to whether the customer would like to pay by “Cash” or “Card” (If configured). EFTLink receives the request from the POS and sends the “Cell Phone Recharge” request to the OPI EPS. The customer will proceed with the payment at this point. If the customer selected to pay by “Cash” during the initial prompt on the POS, the OPI EPS will simply clear the invoice as paid and send back the successful response. The following prompts are skippable: 1. The customer will be prompted by the PED to select which Service Provider they are with. 2. The customer will be prompted to select a RechargeAmount which they would like to recharge (top-up) their phone by. 3. The customer will be prompted by the PED to input their phone number. If the customer selected to pay by “Card” during the initial prompt on the POS, the PED will prompt as to whether the customer would like to pay by “Cash”, “Credit” or “Debit”. The customer may choose how to pay at this point. If the customer selects “Cash”, the invoice will be cleared as paid. If the customer selects “Credit” or “Debit” card, the usual steps will take place for a typical Card transaction. Cell phone recharge payments are not reversible. However, if a card payment was made, the cashier may choose to reverse as a standard card payment. It is assumed that the merchant will have the necessary measures in place to avoid issues with “Cash” payments (in terms of a customer not having enough cash to pay for the transaction and so on). The Transaction Inquiry scenario outlined in the Payment/Payment with Loyalty section also applies to this transaction type.

Table 4-5 (Cont.) OPI - Supported Functions

Function	Description
PLCC (Private Label Credit Cards)	Support for PLCC has been added to the OPI core. PLCC Account Lookup – this is used to look up the account details for a PLCC. This includes the credit balance and remaining credit, and the card token in addition to cardholder information. This can be used to check the balance of the card prior to performing an account payment or tender transaction. The PLCC can be presented to the PED. If the PLCC token is saved against the customer, the saved card token can be used in CNP scenarios. PLCC Payment On Account – this transaction type can be used to make a payment against the balance on the PLCC. The payment transaction can be initiated by the POS and can usually include most types of tender except Credit cards. Some tenders such as Debit cards would be processed by the PED. This would usually be proceeded by an Account Lookup to determine the outstanding balance. PLCC Application Request – this is used to perform a new PLCC application request. Some of the required detail is collected on the POS and sent to EFTLink within the request message, but the PED may also collect some data directly from the applicant. Note that PLCC application process and data is very dependent on the PSP and the finance company offering the PLCC, and may require custom extensions on the POS. Tender by PLCC – In order to allow the POS to tender explicitly by PLCC, a new eWallet Payment Method “61” “PLCC” has been added, which the POS can set on the Sale/Purchase request message, instructing the PED to only allow tender by PLCC. Tender by Debit card – Since PLCC Payment On Account cannot be tendered by credit card, a new eWallet PaymentMethod “62” “Debit Card” has been added, allowing the POS to instruct the PED to only allow tender by Debit Card.
Card Acquisition	Support for card acquisition has been added to the OPI core. The OPI core will now accept an explicit card acquisition request from the POS. the customer will be prompted to present their payment card to the PED so the details of their card (tokens) can be pre-read. This can be useful in several scenarios such as looking up the sales transactions that used that same card, when shopper requests to return items, but does not have the original receipt, thus allowing for a referenced refund, rather than an un-referenced (blind) refund. The following fields are returned to the POS (as determined by PSP) none of which are in scope of PCI DSS - CardAlias, CustomerId, CardHolderEmail, CardToken.

Table 4-5 (Cont.) OPI - Supported Functions

Function	Description
Pre-authorization transactions	Support for card pre-authorization transactions has been added to the OPI core. Authorization Transaction – this message type can be used to request an authorization for funds (without capture/settlement). This can be used for gaining authorization for POS transactions that are not being immediately picked up or delivered to a customer, such as pre-sales and so on, and can be for either payment cards presented to the PED, or with a saved Card Token. The POS can then request completion (capture) of the authorization when the goods are picked up or dispatched, using the Sales Completion transaction. Authorization Release Transaction – this message can be used to release or cancel an existing authorization. Top-UpAuthorization Transaction – this message can be used to increase the amount of an existing authorization. It is possible that multiple authorizations are performed during the lifecycle of a POS transaction or order.
PED Status Messages	Support for PED status messages have been added to the OPI core. The OPI core has configurations that can be enabled, that will instruct EFTLink to connect to an endpoint running on the OPI implementation to establish a websocket channel, and will be listening on this websocket, for Terminal Status messages during the TransactionRequest operation. This is a “fire and forget” message sent from the PED to the POS, and there is no response message. These messages would usually be the same/similar as those messages that are displayed on the PED to instruct/inform the shopper, for example to remove their card.
PSP Cloud Host Endpoint	Support for a cloud host endpoint has been added to the OPI core. This allows a PSP to offer their OPI implementation in the cloud, rather than locally, on-premises. In order to enable this all Retail OPI request messages have a new field, RequestTerminalId, which allows the PSP OPI implementation to route the transaction to the appropriate PED.
Payment Query	Support for Payment Query has been added to the OPI core. The OPI Retail core supports a PaymentQuery request message. This allows the POS to send this message to query the status of the last payment request message. The OPI Retail core supports a PaymentQuery request message.
PSPPaymentBrand for the CardCircuit	The OPI Retail core allows the PSPPaymentBrand field to be returned directly to the POS in the CardCircuit response field (Xstore POS uses the CardCircuit attribute to map to a tender). Currently the IssuerID returned by the OPI payment partner is used via the CardRange.xml file to determine the value of the CardCircuit attribute. However, some payment types such as eWallets do not have a specific IssuerID, meaning that either a generic IssuerID is used, or one of the “Reserve-XX” needs to be configured. This feature is enabled via the OPI Retail configuration value, UsePspPaymentBrandForCardCircuit.

Table 4-5 (Cont.) OPI - Supported Functions

Function	Description
Reference Refund Enhancements	The OPI Retail core now has the capability to include the OPI partner name in the validation that it performs to prevent referenced refunds being sent to different OPI implementations. The EPSName property will be appended to the PaymentProviderName returned to the POS. When the referenced refund transaction is attempted, EFTLink will check that the OPI implementation in use has the same EPSName as the one specified in the reference refund request data.
Card Acquisition Enhancements	The OPI Retail core has added three new fields to the response data sent to the POS following a CardAcquisition request: FundingSourceType, PSPPaymentBrand and PSPPaymentVariant. The POS can then use these fields in the request date sent to tax free shopping providers.

5

Pay by Link (PBL)

This chapter covers EFTLink Integration with Pay by Link Payment Systems. It should be read in conjunction with the *Oracle Retail EFTLink Framework Installation and Configuration 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.

Important Note

Adyen are no longer accepting new customers for the *Customer Present* Pay By Link feature. Please discuss this directly with Adyen if further clarification is necessary.

Minimum Version

The Pay by Link interface requires a minimum EFTLink version of 21.0.0.

System Architecture

EFTLink utilizes the Pay by Link core in order to connect to either Adyen's Checkout Service (using the Adyen Checkout API) for Customer Not Present transactions or Adyen's Terminal for Customer Present transactions (using the Adyen Terminal API). Both connections are established using HTTPS / TLS from EFTLink to Endpoint.

Please note that starting from version 24, validation of EPS device or cloud endpoint SSL certificates will be enforced using a dedicated truststore. To set up the truststore with the required trusted certificates, refer to the *Oracle Retail EFTLink Framework Installation and Configuration Guide*, Securing Communication to EPS Devices by Importing Trusted Certificates section.

Also note that the core default configuration for protocols and ciphers may be too restrictive for some EPS terminals. Please verify and update the configuration with the device supported values by contacting the PSP vendor or by using tools to inspect the SSL connection such as:

openssl s_client -showcerts -connect <terminal/cloud URL endpoint>

Fileset

The following files are used in the EFTLink folder:

- `cores/paybylink/paybylink.jar`
- `paybylink.properties` –(optional, if not present defaults apply)
- `Lang<CC>_<Core>.properties` – Language translation file, for further information see Language.

Language

The translation files for this core should not require alteration, but if necessary then this can be accomplished by amending the relevant `Lang<CC>_<Core>.properties` within the base eftlink folder.

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

```
EftlinkConfig.properties
```

```
DisplayLanguage = EN
```

Supported country codes are CN, DE, EN, ES, FR, IT, JP, NL, PT, RU and SV.

Core Classname

The Pay by Link core is not a standalone primary core. This core should be configured in a multi-core environment. The following example needs to be set manually in the `EftlinkConfig.properties` file. In this example there are two cores. The primary one is the OPI Retail core and the secondary core is the Pay by Link core. The PayByLinkCore setting corresponds to the EPSCore for which you would like to use Pay by Link:

```
NumEPSCores = 2

EPSCore0 = oracle.eftlink.opiretail.OPIRetailCore

EPSCore1 = oracle.eftlink.paybylink.PayByLinkCore

PayByLinkCore = 1
```

Keystore

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

Open a terminal window and change directory to where the script file resides.

For Windows: Type `encrypt-paybylink.bat –k [<keystore name> <properties file>]`.

For example, `encrypt-paybylink.bat –k`

For Linux: Type `encrypt-paybylink.sh –k [<keystore name> <properties file>]`.

For Example, `./sudo encrypt-paybylink.sh –k`

Keystore file will be generated and stored in the data directory. If the keystore name and the properties file names are not specified, then the default values (`paybylink.keystore`, `paybylink.properties`) will be used.

> **Note**
>
> When creating the data directory for the keystore, access to the folder is restricted automatically. Therefore, the same user account as will be used to run the application should be used during this process. Use of the administrator account at this point may lead to the folder being inaccessible to the EFTLink application at runtime.

Encryption

The following settings within the `paybylink.properties` file needs to be encrypted.

- `adyen.paybylink.apikey`

If using a terminal (customer present functionality), the following properties also need to be encrypted:

- `adyen.paybylink.cp.key.identifier`
- `adyen.paybylink.cp.key.passphrase`
- `adyen.paybylink.cp.key.version`

To achieve this, the following steps must be followed:

To encrypt a value: Open a terminal window and change directory to where the script file resides.

For Windows: Type `encrypt-paybylink.bat –e <keystore name> <properties file> <value>`.

For example, `encrypt-paybylink.bat –e`

***For Linux**: Type `sudo ./encrypt–paybylink.sh –e [<keystore name> <properties file> <value>]`.

For example, `sudo ./encrypt-paybylink.sh –e`

The user will be presented with prompts to provide the value(s) which are to be encrypted. Once entered the corresponding properties keys will be automatically updated with the encrypted values.

> **Note**
>
> If the keystore name, properties file and unencrypted text is included as arguments then the encrypted value and initialization vector will be outputted to the console which must be copied and pasted to *relevant property key* in `paybylink.properties`. This process then needs to be repeated for every value that is required to be encrypted.

To re-encrypt; Open a terminal window and change directory to where the script file resides.

For Windows: Type `encrypt-paybylink.bat –r [<keystore name> <properties file> <keygen type> <cipher type> <key size> <iterations>]`.

For example, `encrypt-paybylink.bat –r`

***For Linux**: Type `sudo ./encrypt-paybylink.sh –r [<keystore name> <properties file> <keygen type> <cipher type> <key size> <iterations>]`.

For example, `sudo ./encrypt-paybylink.sh -r`

The key values to be re-encrypted will be taken from the properties file, re-encrypted and the properties file will be automatically updated.

* You may be required to give script file(s) execution rights for example, `chmod +x <PathToFile>`

> **Note**
>
> 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/.

Configuration Settings

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

Key Settings

Settings that may be different for each POS.

Table 5-1 Pay by Link - Key Settings

Setting	Description	Default	Example
keystore	Specifies the paybylink keystore name. Mandatory setting.	paybylink.keystore	`keystore = default.keystore`
http.connect.timeout.seconds	Specifies how long to wait to establish a connection before timing out; in seconds.	30	`http.connect.timeout.seconds = 180`
adyen.merchant.account	Specifies the Adyen merchant account to associate transactions with. Mandatory setting.	N/A	`adyen.merchant.account = Oracle`
adyen.paybylink.apikey	Specifies the (Encrypted) Adyen API Key to use to perform transactions. Mandatory setting.	N/A	
adyen.paybylink.apikey.iv	Specifies the (Encrypted) Adyen API Key's initialization vector. Mandatory setting.	N/A	
adyen.paybylink.cnp.endpoint	Specifies the endpoint for customer not present transactions/communications. Mandatory setting.	https://checkout-test.adyen.com/v67	`https://checkout-test.adyen.com/v66`

Table 5-1 (Cont.) Pay by Link - Key Settings

Setting	Description	Default	Example
abort.timeout.seconds	Specifies how long to attempt to wait for a cancellable transaction to be available to be cancelled otherwise the AbortRequest will be ignored; in seconds.	30	`abort.timeout.seconds = 180`

Secondary Settings

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

Table 5-2 Pay By Link - Secondary Settings

Setting	Description	Default	Example
maintenance.menu.timeout.seconds	Specifies the admin menu timeout; in seconds.	6000	`maintenance.menu.timeout.seconds = 10000`
adyen.paybylink.cnp.response.timeout.seconds	Specifies how long to wait for a response from the customer not present endpoint; in seconds.	600	`adyen.paybylink.cnp.response.timeout.seconds = 500`
adyen.paybylink.cnp.default.country.code	Specifies the default country code to use in customer not present transactions when the country code has not been specified in the original request. This is in ISO 3166-1 alpha-2 format.	N/A	`adyen.paybylink.cnp.default.country.code = GB`
adyen.paybylink.cnp.default.shopper.locale	Specifies the default shopper locale to use in customer not present transactions when the shopper locale has not been specified in the original request.	N/A	`adyen.paybylink.cnp.default.shopper.locale = en-GB`
adyen.paybylink.cnp.default.link.expiry.hours	Specifies the default link expiry of links created for customer not present transactions; in hours.	24	`adyen.paybylink.cnp.default.expiry.hours = 72`
adyen.paybylink.cp.endpoint	Specifies the endpoint for customer present transactions/ communications.	N/A	`adyen.paybylink.cp.endpoint = https://m400-123456789.test.terminal.adyen.com:8443/nexo`
adyen.paybylink.cp.response.timeout.seconds	Specifies how long to wait for a response from the customer present endpoint; in seconds.	300	`adyen.paybylink.cp.response.timeout.seconds = 600`

Table 5-2 (Cont.) Pay By Link - Secondary Settings

Setting	Description	Default	Example
adyen.paybylink.cp.sale.id	Specifies the sale id to send in customer present transactions. This is typically your unique cash register identifier. Mandatory setting if `adyen.paybylink.cp.endpoint` has been set.	N/A	`adyen.paybylink.cp.sale.id = TestPOS`
adyen.paybylink.cp.POI.id	Specifies the poi id to send in customer present transactions. This is your unique identifier of the terminal, in the format [device model]-[serial number]. Mandatory setting if `adyen.paybylink.cp.endpoint` has been set.	N/A	`adyen.paybylink.cp.POI.id = M400-123456789`
adyen.paybylink.cp.protocol.version	Specifies the version of Adyen's Terminal API to send in customer present transactions. Mandatory setting if `adyen.paybylink.cp.endpoint` has been set.	3.0	`adyen.paybylink.cp.protocol.version = 3.0`
adyen.paybylink.cp.adyen.crypto.version	Specifies the Adyen crypto version to send in customer present transactions. Mandatory setting if `adyen.paybylink.cp.endpoint` has been set.	N/A	`adyen.paybylink.cp.adyen.crypto.version = 1`
adyen.paybylink.cp.key.identifier	Specifies the (Encrypted) Adyen Key Identifier to use to perform customer present transactions. Mandatory setting if `adyen.paybylink.cp.endpoint` has been set.	N/A	`adyen.paybylink.cp.key.identifier = 3212f4323c917fbdf52c5a2436ed3c16`
adyen.paybylink.cp.key.identifier.iv	Specifies the (Encrypted) Adyen Key Identifier's initialization vector. Mandatory setting if `adyen.paybylink.cp.endpoint` has been set.	N/A	`adyen.paybylink.cp.key.identifier.iv = a62a76826f81dcaf3d9a55c534ea94ef`
adyen.paybylink.cp.key.passphrase	Specifies the (Encrypted) Adyen Key Passphrase to use to perform customer present transactions. Mandatory setting if `adyen.paybylink.cp.endpoint` has been set.	N/A	`adyen.paybylink.cp.key.passphrase = be9ff6c0cf49e436db59e7b395ae9862`

Table 5-2 (Cont.) Pay By Link - Secondary Settings

Setting	Description	Default	Example
adyen.paybylink.cp.key.passphrase.iv	Specifies the (Encrypted) Adyen Key Passphrase's initialization vector. Mandatory setting if `adyen.paybylink.cp.endpoint` has been set.	N/A	`adyen.paybylink.cp.key.passphrase.iv = 273f3ee5fecc001dfe52ea69e0d19b2f`
adyen.paybylink.cp.key.version	Specifies the (Encrypted) Adyen Key Version to use to perform customer present transactions. Mandatory setting if `adyen.paybylink.cp.endpoint` has been set.	N/A	`adyen.paybylink.cp.key.version = d9d8a033dd2ab3495953fff3d599a073`
adyen.paybylink.cp.key.version.iv	Specifies the (Encrypted) Adyen Key Version's initialization vector. Mandatory setting if `adyen.paybylink.cp.endpoint` has been set.	N/A	`adyen.paybylink.cp.key.version.iv = 9cc3564f43bcfa0ae4d9c456f66b217a`
crypto.keygenType	Specifies the keygen type to use for the internal encryption processing.	AES	`crypto.keygenType = AES`
crypto.cipherType	Specifies the cipher type to use for the internal encryption processing.	AES/GCM/PKCS5Padding	`crypto.cipherType = AES/GCM/PKCS5Padding`
crypto.keySize	Specifies the key size to use for the internal encryption processing.	128	`crypto.keySize = 256`
crypto.iterations	Specifies the number of iterations to use for the internal encryption processing.	100000	`crypto.iterations = 100000`
crypto.factoryinstance	Sets crypto factory instance for keystore password encryption.	PBKDF2WithHmacSHA512	`crypto.factoryinstance = PBKDF2WithHmacSHA512`
crypto.secretkeyspec	Sets crypto secret key spec for keystore password encryption.	AES	`crypto.secretkeyspec = AES`
crypto.keystoretype	Sets keystore type	JKS	crypto.keystoretype = JKS
crypto.digest	Sets digest for keystore password.	SHA-512	`crypto.digest = SHA-512`
crypto.hashbyteSize	Sets hash byte size for keystore password.	384	`crypto.hashbyteSize = 384`
merchant.reference	Specifies the unique merchant reference.	N/A	`merchant.reference = Merchant A`

Table 5-2 (Cont.) Pay By Link - Secondary Settings

Setting	Description	Default	Example
merchant.reference.format	Specify the format of the merchant reference - replaces static value with a dynamically generated value using several substitutions. See Merchant Reference Formats.	R (use existing static merchant ref)	`merchant.reference.format=R-dddddddddd-SSSSSS.WWWWWW.YYYYMMDD.hhmmss.TTTTTT.qq`
merchant.reference.special.characters	Specifies a character that is to be passed through 'as is' in addition to the ones already mentioned.	N/A	`merchant.reference.special.characters = K`
truststore.trustAllServers	Disable validation of certificates with the truststore on connection between EFTLink and EPS devices.	false	truststore.trustAllServers = false Disabling validation of the truststore is never a recommended practice.
truststore.ProtocolsWhiteList	Restricts the protocols which are permissible in the connection between EFTLink and EPS devices. Default allows for TLS 1.3 security.	TLSv1.3,TLSv1.2	truststore.ProtocolsWhiteList = TLSv1.3,TLSv1.2
truststore.CipherWhiteList	Restricts the ciphers which are permissible in the connection between EFTLink and EPS devices. The whitelist only includes ciphers which are approved under Oracle Approved Technologies: Security Protocols	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_AES_128_GCM_SHA256,TLS_AES_256_GCM_SHA384,TLS_CHACHA20_POLY1305_SHA256,TLS_AES_128_CCM_SHA256,TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384, TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256,TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256,TLS_ECDHE_ECDSA_WITH_AES_256_CCM,TLS_ECDHE_ECDSA_WITH_AES_128_CCM	truststore.CipherWhiteList = `TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_AES_128_GCM_SHA256,TLS_AES_256_GCM_SHA384,TLS_CHACHA20_POLY1305_SHA256,TLS_AES_128_CCM_SHA256, TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256,TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256,TLS_ECDHE_ECDSA_WITH_AES_256_CCM,TLS_ECDHE_ECDSA_WITH_AES_128_CCM`

Table 5-2 (Cont.) Pay By Link - Secondary Settings

Setting	Description	Default	Example
truststore.CipherBlackList	Restricts the ciphers which are not permissible in the connection between EFTLink and EPS devices. The blacklist only includes ciphers which are not approved under Oracle Approved Technologies: Security Protocols	SSL_.*,TLS_EMPTY_.*,.*_SHA,.*_3DES_.*,.*_DES_.*,.*_WITH_NULL_.*,.*_anon_.*,.*EXPORT.*,.*LOW.*,.*MD5.*,.*DES.*,.*RC2.*,.*RC4.*,.*PSK.*,TLS_DH_.*,TLS_DHE_.*	truststore.CipherBlackList = `SSL_.*,TLS_EMPTY_.*,.*_SHA,.*_3DES_.*,.*_DES_.*,.*_WITH_NULL_.*,.*_anon_.*,.*EXPORT.*,.*LOW.*,.*MD5.*,.*DES.*,.*RC2.*,.*RC4.*,.*PSK.*,TLS_DH_.*,TLS_DHE_.*`
truststore.useJVMTrustStore	Use JVM cacert truststore instead of EFTLink dedicated truststore.	False	truststore.useJVMTrustStore = false Disabling use of the EFTLink dedicated truststore is never a recommended practice.

Merchant Reference Formats

For the merchant reference format, the following substitutions are available:

Table 5-3 Merchant Reference Formats

Component	Description	Example	Format
R	Use existing Merchant Reference	R	
S	StoreID	SSSSSS	min 3, max 10 chars, left 0 filled
W	WorkStation id	WWWWWW	min 3, max 20 chars, left 0 filled
YY	Year	YY or YYYY	extracted from POSTimeStamp
MM	Month	MM	extracted from POSTimeStamp
DD	Day	DD	extracted from POSTimeStamp
hh	Hour	hh	extracted from POSTimeStamp
mm	Minute	mm	extracted from POSTimeStamp
ss	Second	ss	extracted from POSTimeStamp
T	Transaction number	TTTTTT	min 3, max 20 chars, left 0 filled
d	Transaction date	dddddddddd	must be 10 chars

The following special characters are also allowed:

minus -
underscore _
period .

Example format:

```
R-dddddddddd-SSSSSS_WWWWWW.YYYYMMDD.hhmmss.TTTTTT.qq
```

Supported Functions

Below is a list of supported functionalities of the interface to Pay by Link.

Important: Adyen are no longer accepting new customers for the *Customer Present* Pay By Link feature. Please discuss this directly with Adyen if further clarification is necessary.

Table 5-4 Pay By Link - Supported Functions

Function	Description
Customer not present - Payment/Payment with Loyalty	The POS initiates a customer not present pay by link tender. EFTLink sends the request to the Adyen Online Payments (Checkout API) endpoint. The Adyen Online Payments endpoint will create a payment link which EFTLink will send back to the POS. The customer will be referred to pay via the link generated by the Adyen Online Payments system. In the event of a communication failure between EFTLink and the Adyen Online Payments endpoint; the POS operator will be prompted by EFTLink as to whether they would like to Retry or Decline the transaction. Due to API Idempotency, if the operator chooses to Retry the transaction, an exact duplicate of the previous request is sent once again to the Adyen Online Payments endpoint. Should the connection timeout continue to occur, the POS operator will be continually asked whether they would like to Retry or Decline until they press the Decline button. If the operator presses the Decline button, EFTLink will simply return a failure result to the POS and the POS will return back to the tender screen.
Customer present – Payment/ Payment with Loyalty	The POS initiates a customer present pay by link tender. EFTLink sends the request to the Adyen In-Store (Terminal API) Payment Entry Device (PED). The PED will display a QR code which the customer will scan and enter their card details on the secure form generated by the Adyen payment gateway. Once the customer has completed the form and the payment has been processed, EFTLink will return a response including formatted merchant/customer receipts as required. In the event of a communication failure between EFTLink and the PED an initial Transaction Status request is sent from EFTLink to determine the status of the request. 1. If the initial Transaction Status check succeeds, the original response details will be sent from EFTLink to the POS. 2. If the initial Transaction Status check returns as in-progress, EFTLink will continue to query the status of the transaction until a valid response is returned. Once the final response has returned, EFTLink will send the original response details back to the POS. 3. If the initial Transaction Status check times out, the POS operator will be prompted by EFTLink as to whether they would like to Retry or Decline the transaction. If the operator chooses to Retry the transaction, EFTLink will attempt to re-send the previous Transaction Status check until a valid response is returned or until the cashier presses the Decline button when prompted after every check. If an in-progress response is returned, the in-progress scenario outlined above occurs. If a valid final response is returned, EFTLink will send the final response back to the POS and complete the transaction as usual.

Table 5-4 (Cont.) Pay By Link - Supported Functions

Function	Description
Customer not present – Link Query	The POS initiates a customer not present link query request with the link Id to check the status of a link. EFTLink sends the link query request to the Adyen Online Payments (Checkout API) endpoint. The Adyen Online Payments endpoint will respond with the status of the link. EFTLink will set the status of the link in the final response to the POS. The status of the link will be one of the following: • active • expired • completed • paid
Customer not present – Link Expiry	The POS initiates a customer not present link expiry request with the link Id to expire. EFTLink sends the link expiry request to the Adyen Online Payments (Checkout API) endpoint. The Adyen Online Payments endpoint will set the status of the link to expired in the response to EFTLink. EFTLink will send the final response to the POS with the new status set.
Reversal	Customer not present reversal is possible if the POS initiates a Post Void request to EFTLink following a customer not present pay by link tender. This will simply expire the link which was generated during the initial customer not present pay by link tender. Customer present reversals behave in the same way as a reversal for any standard transaction. EFTLink converts the reversal request into a Verified Refund request using the original transaction reference and sends this request to the Adyen In-Store (Terminal API) endpoint.
Verified Refund	All verified refund requests are handled by the Adyen Online Payments (Checkout API) endpoint. The POS initiates a Verified Refund request. EFTLink sends the refund request to the Adyen Online Payments endpoint. The Adyen Online Payments endpoint responds with a valid response. EFTLink sends the final response to the POS.
Time out handling	All transactions processed by the Pay by Link core are available for the timeout scenarios described under the “Customer not present - Payment/Payment with Loyalty” and “Customer present – Payment/ Payment with Loyalty” sections of this “Pay by Link – Supported Functions” table, refer to these sections for further information.

6 PayPal

This chapter covers EFTLink integration with PayPal.

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 and Configuration Guide*.

Minimum Version

The PayPal interface requires a minimum EFTLink version of 20.0.

System Architecture

PayPal is a REST service architecture. EFTLink connects to it using HTTP POST with JSON pay load.

Fileset

In addition to standard EFTLink files, PayPal uses:

- `cores/PayPal/PayPalCore.jar` – executable code for the PayPal EFTLink core.
- `paypal.properties` – configuration settings to specify which features are enabled and to define communication parameters for the interface with PayPal REST service.
- `paypal_dynamic.properties` - dynamic configuration file that contains the encrypted credentials needed for authenticating with PayPal.

Language

The translation files for this core should not require alteration, but if necessary then this can be accomplished by amending the relevant `Lang<CC>_<Core>.properties` within the base eftlink folder.

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

```
EftlinkConfig.properties
```

```
DisplayLanguage = EN
```

Supported country codes are CN, DE, EN, ES, FR, IT, JP, NL, PT, RU and SV.

Core Class Name

PayPal core is not a standalone primary core but an EWallet core. This is setup together with a different primary core in a multi core environment. The following example needs to be set manually in `EFTLinkConfig.properties`. In this example, there are 2 cores. The primary one is OPI Retail core and PayPal is the EWallet core.

```
NumEPSCores=2

EPSCore0 = oracle.eftlink.opiretail.OPIRetailCore

EPSCore1 = oracle.eftlink.paypal.PayPalCore

EwalletCore = 1
```

Merchant Account OnBoarding

Before processing PayPal transactions, you need to create two REST API apps (Location and Payment) in PayPal's portal. This process is described in the section **Setting up the API Caller Account** of the PayPal document *In-Store QR Code Integration for Direct Merchant Partners*. You need to get this from PayPal.

After creation, you should have the client ID and client secret credentials for each API. Send the client ID of both apps to your PayPal contact. PayPal sets up the permission for the accounts. Once permission is set up, you need to encrypt these credentials in file.

See the section below, Encrypting PayPal's Credentials, on how to accomplish this. Once encrypted, you can now enroll your store location. This is done by Xstore sending the login request to EFTLink. Once the store is enrolled successfully, you can process payments, reversal, and verified refunds.

Encrypting PayPal's Credentials

1. Assuming you installed EFTLink in C:\eftlink. Copy all files from C:\eftlink\cores\PayPal except the jar file to C:\eftlink.
2. Open a command terminal. Go to C:\eftlink directory. Make sure you have Java in your Windows system path. If not set it first.
3. First, create the key store file that will hold the encryption and decryption key. Execute the command below.

   The script creates a data directory on C:\eftlink with the paypal.keystore file in it.

   ```
   encrypt-paypal.bat -k
   ```
4. You can now encrypt the client ID and client secret of both the location and payment API. On the command terminal, execute the command below.

   This will ask you to enter the client ID and client secret of your location and payment API respectively.

   The script writes the encrypted values together with the initialization vector into paypal_dynamic.properties file.

   ```
   encrypt-paypal.bat -e
   ```

Note

When creating the data directory for the keystore, access to the folder is restricted automatically. Therefore, the same user account as will be used to run the application should be used during this process. Use of the administrator account at this point may lead to the folder being inaccessible to the EFTLink application at runtime.

Configuration Settings

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

Key Settings

Settings that may be different for each POS/PED.

Table 6-1 PayPal - Key Settings

Setting	Description	Default	Example
store.name	Name of the store.	NA	`store.name = My Shop`
store.id	Store ID	NA	`store.id = 101`
store.address.line1	Store's street address.	NA	`store.address.line1 = 1000 North St.`
store.address.city	Store's city address.	NA	`store.address.city = Cleveland`
store.address.state	Store's state address.	NA	`store.address.state = OH`
store.address.country	Store's country address.	NA	`store.address.country = US`
store.address.postalCode	Store's postal code.	NA	`store.address.postalCode = 44139`
store.latitude	Store's latitude location.	NA	`store.latitude = 32.5`
store.longitude	Store's longitude location.	NA	`store.longitude = -97.2`

Secondary Settings

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

Table 6-2 PayPal - Secondary Settings

Setting	Description	Default	Example
store.availability	Whether this location is currently open for business.	open	`store.availability= open`
store.tabType	The type of tab supported at this location.	none	`store.tabType = standard`
store.mobility	The mobile setting for this location.	fixed	`store.mobility = mobile`

Table 6-2 (Cont.) PayPal - Secondary Settings

Setting	Description	Default	Example
store.gratuityType	The type of gratuity that is accepted by this location.	none	`store.gratuityType = standard`
capture.endpoint	The capture service end point context.	/v2/retail/captures	`capture.endpoint = /v2/retail/captures`
token.endpoint	The token service end point context.	/v1/oauth2/token	`token.endpoint = /v1/oauth2/token`
location.endpoint	The location service end point context.	/retail/merchant/v1/locations	`location.endpoint = /retail/merchant/v1/locations`
cancel.endpoint	The end point context for cancelling payment or refund.	/v2/retail/cancel	`cancel.endpoint = /v2/retail/cancel`
base.api.url	Base URL of PayPal's services.	https://api.paypal.com	`base.api.url= https://api.paypal.com`
connect.timeout	Timeout in milliseconds when connecting to PayPal.	5000	`connect.timeout = 5000`
payment.read.timeout	Read timeout in milliseconds for PayPal's payment related services.(ex. capture.cancel,refund).	300000	`payment.read.timeoutut = 300000`
other.read.timeout	Read timeout in milliseconds for PayPal's services non-payment related service.(ex. token service).	120000	`other.read.timeout= 120000`
merchant.category.code	Merchant category code.	5965	`merchant.category.code = 5965`
qr.code.max.length	The maximum digits of the QR code.	18	`qr.code.max.length = 18`
crypto.keygenType	Sets the keygen algorithm type.	AES	`crypto.keygenType = AES`
crypto.cipherType	Sets the cipher algorithm type.	AES/GCM/PKCS5Padding	`crypto.cipherType = AES/GCM/PKCS5Padding`
crypto.keySize	Sets size of the keystore key. Note: 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/	128	`crypto.keySize = 128`
crypto.iterations	Sets the number of iterations.	100000	`crypto.iterations = 100000`

Table 6-2 (Cont.) PayPal - Secondary Settings

Setting	Description	Default	Example
crypto.factoryinstance	Sets crypto factory instance for keystore password encryption.	PBKDF2WithHmacSHA512	`crypto.factoryinstance = PBKDF2WithHmacSHA512`
crypto.secretkeyspec	Sets crypto secret key spec for keystore password encryption.	AES	`crypto.secretkeyspec = AES`
crypto.keystoretype	Sets keystore type	JKS	`crypto.keystoretype = JKS`
crypto.digest	Sets digest for keystore password.	SHA-512	`crypto.digest = SHA-512`
crypto.hashbyteSize	Sets hash byte size for keystore password.	384	`crypto.hashbyteSize = 384`

Administration Functions

PayPal core does not support administrative functions.

Supported Functions

Below is a list of supported functionalities of the interface to PayPal. Many functionalities are provided by PayPal. (Please refer to interface specification for details) but are not implemented because of the business requirement.

Table 6-3 PayPal - Supported Functions

Function	Description
Payment	Sends payment request to PayPal. The user initiates the payment by scanning a PayPal application generated QR code from the customer's mobile phone. If successful, appropriate receipts will be printed at the end of transaction.
Reversal	Sends reversal request to PayPal. This will reverse a transaction specified by the transaction ID, found on the receipt, which must be captured by the POS and pass on to EFTLink.
Refund	Sends refund request to PayPal. This will refund a transaction with specified amount.
Store Registration	This is accomplished by either Xstore v20 sending the login request or running the built in LocationService console application within the core.

7

Six Pay

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 and Configuration 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 Six Pay interface requires a minimum EFTLink version of 20.0.

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.
- `Lang<CC>_<Core>.properties` – Language translation file, for further information see Language.

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

Language

The translation files for this core should not require alteration, but if necessary then this can be accomplished by amending the relevant `Lang<CC>_<Core>.properties` within the base eftlink folder.

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

```
EftlinkConfig.properties
```

```
DisplayLanguage = EN
```

Supported country codes are CN, DE, EN, ES, FR, IT, JP, NL, PT, RU and SV.

Core Classname

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

```
EPSCore0 = manito.eft.sixpay.SixpayMPDOPIClient
```

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 Settings

Table 7-1 Six Pay - Key Settings

Setting	Description	Default	Example
SixpayServerIP	IP address of the store server running MPD.		`SixpayServerIP = IP ADDRESS`
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`

Table 7-1 (Cont.) Six Pay - Key Settings

Setting	Description	Default	Example
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.

Table 7-2 Six Pay - Optional Configuration Settings

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.	NA	`SixpayChannel1 = 20007`
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.	Signature	`SignatureCheckTag=sign`
SignatureCheckTimeout	Timeout for Signature OK? Question.	30 seconds	`SignatureCheckTimeout = 30`
ProcessTokenisedRefundReversalAsPayment	If true, process a tokenised refund reversal request as a payment request.	false	`ProcessTokenisedRefundReversalAsPayment = false`

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.

8 Solve Connect

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 and Configuration 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 Solve Connect interface requires a minimum EFTLink version of 20.0.

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`
- `Lang<CC>_<Core>.properties` – Language translation file, for further information see Language.

Language

The translation files for this core should not require alteration, but if necessary then this can be accomplished by amending the relevant `Lang<CC>_<Core>.properties` within the base eftlink folder.

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

```
EftlinkConfig.properties
```

```
DisplayLanguage = EN
```

Supported country codes are CN, DE, EN, ES, FR, IT, JP, NL, PT, RU and SV.

Core Classname

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

```
EPSCore0 = manito.eft.solveconnect.SolveConnectCore
```

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

Table 8-1 SolveConnectPOS.properties - Key Settings

Setting	Description	Default	Example
SourceID	The POS specific identifier, allocated by retailer, to be unique across the retailer's estate.	NA	`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`
PEDConnection	How the PED is connected: Serial/LAN	serial	`LAN`
PEDSerialNumber	Used as an identifier to target PED logon/logoff in a PED Pooling environment. Note: this setting is only applicable if the above key setting "PEDConnection" is set to Serial. When set to LAN the PED IP Address and Port is used as the identifier.	NA	`111-222-333`
AutoPEDLogoff	Determines whether the PED should be logged off after everything transaction. This is a required to be set to True when configuring a PED Pooling environment.	false	`true`

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

Table 8-2 SolveConnect.properties - Secondary Settings

Setting	Description	Default	Example
ServiceHost	Hostname or IP address of SolveConnect service.	NA	`ServiceHost = IP ADDRESS`
TransactionTimeoutPeriod	Number of seconds to allow a transaction to complete.	180	`TransactionTimeoutPeriod = 180`
CancellationTimeoutperiod	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`
TransactionReferenceScheme	The format and source of Store and Till-ID values. Recognized 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`

Table 8-2 (Cont.) SolveConnect.properties - Secondary Settings

Setting	Description	Default	Example

DCC Keywords	DCC keywords for extracting DCC from status message. There are no defaults.	NA	`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, for example, to 9999 if an open/background card read operation is required.	30	`CardSwipeTimeoutPeriod = 30`
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, that is enable for some conditions, disabled for others.	false	`SelectiveMerchantPrint = false`

Table 8-2 (Cont.) SolveConnect.properties - Secondary Settings

Setting	Description	Default	Example
MerchantPrint.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 (for example, MerchantPrint.present.pin) that can be set to "true" to re-enable merchant print for that attribute combination. **Note:** merchant print requiring signature will always be printed, it cannot be disabled. For example, to re-enable merchant print for CustomerNotPresent transactions: `MerchantPrint.not_present.not_performed = true` Re-enable merchant print for CustomerNotPresent transactions.	NA	`MerchantPrint.not_present.not_performed = true`
SelectiveCustomerPrint	Determine whether customer print is selective. For example, enabled for some conditions disabled for others.	false	`SelectiveCustomerPrint = false`

Table 8-2 (Cont.) SolveConnect.properties - Secondary Settings

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 (for example, CustomerPrint.present.pin) that can be set to "false" to disable customer print for that attribute combination. For example. to disable customer print for CustomerNotPresent transactions: `CustomerPrint.not_present.not_performed = true`	NA	`CustomerPrint.not_present.not_performed = true`
SignatureCheckReprintOption	Determine whether to include a "reprint" option when prompting operator for signature verification. 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.	false	`SignatureCheckReprintOption = true`
ManualAuthMinLength	Minimum input length required for Manual/Voice referral authorization code response.	0	`ManualAuthMinLength = 0`

Fixed Configuration Settings

There are several 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.

Table 8-3 SolveConnect - Supported Functions

Function	Description
Logon	Sends a PED Logon request to the Solve Connect client.
Logoff	Sends a PED Logoff request to the Solve Connect client.
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 gift card payment request to the terminal. Specified amount will be deducted from the gift card. Administration options to add balance and check balance is also supported.
Receipt Reprint	Reprint merchant/customer receipt.

9
Tender Retail

This document covers EFTLink Integration with Tender Retail Payment Systems. It should be read in conjunction with the *Oracle Retail EFTLink Framework Installation and Configuration 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 Tender Retail interface requires a minimum EFTLink version of 20.0.

System Architecture

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

> **Note**
>
> This document does not cover the install of the Tender Retail software.

Fileset

In addition to standard EFTLink files, Tender Retail uses:

- `cores/tenderretail/epstenderretail.jar`
- `tenderretail.properties`
- `Lang<CC>_<Core>.properties` – Language translation file, for further information see Language.
- `data/tenderretail.keystore` - keystore file is used to encrypt Givex user id and password within the `tenderretail.properties`. This file needs to be generated at installation but only if using Givex as the gift card provider. 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:

Open a terminal window and change directory to where the script file resides.

For Windows: Type `encrypt-tenderretail.bat -k [<keystore name> <properties file>]`.

For example, `encrypt-tenderretail.bat -k`

For Linux: Type `encrypt-tenderretail.sh -k [<keystore name> <properties file>]`.

For Example, `sudo ./encrypt-tenderretail.sh -k`

Keystore file will be generated and stored in the data directory. If the keystore name and the properties file names are not specified, then the default values (`tenderretail.keystore`, `tenderretail.properties`) will be used.

> **Note**
>
> When creating the data directory for the keystore, access to the folder is restricted automatically. Therefore, the same user account as will be used to run the application should be used during this process. Use of the administrator account at this point may lead to the folder being inaccessible to the EFTLink application at runtime.

Encryption

If using Givex for the SVC payment then the following settings within the `tenderretail.properties` file need to be encrypted:

- `user.id`
- `user.pin`

To achieve this, the following steps must be followed:

To encrypt a value: Open a terminal window and change directory to where the script file resides.

For Windows: Type `encrypt-tenderretail.bat -e <keystore name> <properties file> <value>`.

For example, `encrypt-tenderretail.bat -e`

***For Linux**: Type `sudo ./encrypt–tenderretail.sh -e [<keystore name> <properties file> <value>]`.

For example, `sudo ./encrypt-tenderretail.sh -e`

The user will be presented with prompts to provide the value(s) which are to be encrypted. Once entered the corresponding properties keys will be automatically updated with the encrypted values.

> **Note**
>
> If the keystore name, properties file and unencrypted text is included as arguments then the encrypted value and initialization vector will be outputted to the console which must be copied and pasted to relevant property key within `tenderretail.properties`. This process then needs to be repeated for every value that is required to be encrypted.

To re-encrypt; Open a terminal window and change directory to where the script file resides.

For Windows: Type `encrypt-tenderretail.bat –r [<keystore name> <properties file> <keygen type> <cipher type> <key size> <iterations>]`.

For example, `encrypt-tenderretail.bat –r`

***For Linux**: Type `sudo ./encrypt-tenderretail.sh –r [<keystore name> <properties file> <keygen type> <cipher type> <key size> <iterations>]`.

For example, `sudo ./encrypt-tenderretail.sh -r`

The key values to be re-encrypted will be taken from the properties file, re-encrypted and the properties file will be automatically updated.

* You may be required to give script file(s) execution rights for example, `chmod +x <PathToFile>`

> **Note**
>
> 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/.

PED Initialization

It is possible to send a Ped Initialization request to the tender retail application via the command line using arguments without EFTLink running.

To achieve this, EFTLink must not be running, and the tender retail application must be running and accepting requests.

> **Note**
>
> A windows batch file is provided called `initializePed.bat`, which requires to be run from the root folder of EFTLink. If the batch file is run without supplying any arguments it will attempt to gather the mandatory and secondary arguments from the `tenderretail.properties` file.

Table 9-1 Mandatory Arguments

Argument	Description	Example
-i	Instructs EFTLink to make a ped initialization request. If the terminal id or any of the secondary arguments are not provided then the core will attempt to get them from the tenderretail.properties file.	`java manito.eft.tenderretail.Main -i`
terminalId	Sets terminal.id.	`java manito.eft.tenderretail.Main -i 300`

Table 9-2 Secondary Arguments

Argument	Description	Example
hostName	Sets mcm.host.name.	`java manito.eft.tenderretail.Main -i 300 localhost`
hostPort	Sets mcm.host.port.	`java manito.eft.tenderretail.Main -i 300 localhost 3858`
timeout_1	Sets connection.timeout.1.	`java manito.eft.tenderretail.Main -i 300 localhost 3858 120000`
timeout_2	Sets connection.timeout.2.	`java manito.eft.tenderretail.Main -i 300 localhost 3858 120000 2000`
ackInterval	Sets do.positive.ack.	`java manito.eft.tenderretail.Main -i 300 localhost 3858 120000 2000 5000`
checkServerOnStartup	Sets check.mcm.server.on.startup.	`java manito.eft.tenderretail.Main -i 300 localhost 3858 120000 2000 5000 true`

Language

The translation files for this core should not require alteration, but if necessary then this can be accomplished by amending the relevant `Lang<CC>_<Core>.properties` within the base eftlink folder.

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

```
EftlinkConfig.properties
```

```
DisplayLanguage = EN
```

Supported country codes are CN, DE, EN, ES, FR, IT, JP, NL, PT, RU and SV.

Core Classname

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

```
EPSCore0=manito.eft.tenderretail.TenderRetailCore
```

Configuration Settings

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

Key Settings

Settings that may be different for all POS.

Table 9-3 Tender Retail - Key Settings

Setting	Description	Default	Example
terminal.id	The bank supplied ID number specific to the station processing the transaction as configured in the Tender Retail Merchant Connect Multi Credit/Debit server. Only one terminal is supported.	None	`terminal.id = 300`
wallet.terminal.id	The e-wallet supplied ID number specific to the station processing the transaction as configured in the Tender Retail Merchant Connect Multi.	None	wallet.terminal.id = 007
wallet.host	Which wallet host to use. Values are Citcon or Alipay. Important! This setting must match the configuration of Tender Retail's application. Misalignment of the configuration could cause erroneous transaction results.	Citcon	wallet.host = Alipay

Secondary Settings

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

Table 9-4 Tender Retail - Secondary Settings

Setting	Description	Default	Example
mcm.host.folder	The path to the folder where the Tender Retail's application is installed.	\MerchantConnectMulti	`mcm.host.folder = C:\MerchantConnectMulti`
mcm.host.port	The socket port for making payment requests.	3858	`mcm.host.port = 3858`
mcm.host.name	The file location of where Tender Retail's application is installed.	localhost	`mcm.host.name = localhost`
check.mcm.server.on.startup	Defines whether Tender Retail's application should be contacted initially when starting the EFT adapter.	true	`check.mcm.server.on.startup = true`
communications.type	Sets the communications type.	ip	`communications.type = ip`

Table 9-4 (Cont.) Tender Retail - Secondary Settings

Setting	Description	Default	Example
connection.timeout.1	This parameter configures the time until when the Tender Retail application is expected to respond to a request. This includes the time needed to swipe the card at the PIN Pad terminal (if required) and to contact the financial institution.	120000	`connection.timeout.1 = 120000`
connection.timeout.2	This parameter determines the interval in which Tender Retail's application is expected to continue delivering data once it began to deliver something.	2000	`connection.timeout.2 = 2000`
timeout.saf	This parameter determines the time Tender Retail's application must empty its store and forward queue. Note: If there is a communication problem with the eft transaction processor or if there are many transactions in the queue, it may take quite a while to process the queue.	600000	`timeout.saf`
do.positive.ack	Determines whether the responses are acknowledged. The value specifies the interval in which Tender Retail's application expects the acknowledgements.	5000	`do.positive.ack = 5000`
currency.code	Used in conjunction with SVS Gift card to set the currency code.	840	`currency.code = 840`
currency.symbol	Line display currency symbol.	$	`currency.symbol = $`
Use.tokens	Determines whether Tender Retail's application is set to use tokens. This setting must match the configuration of Tender Retail's application.	true	`use.tokens = true`
combine.receipts	Turn on/off POS combine receipt.	false	`combine.receipts = false`
combine.receipts.suppress.lines	When combine.receipt is true, sets which line number to suppress.	NA	`combine.receipts.suppress.lines=1,3,5`
combine.receipts.suppress.strings	When combine.receipt is true, sets which line to suppress when strings are matched.	NA	`combine.receipts.suppress.strings = Date,Time`

Table 9-4 (Cont.) Tender Retail - Secondary Settings

Setting	Description	Default	Example
exclude.combine.receipts.Strings	Sets what line to suppress when strings are matched.	NA	`exclude.combine.receipts.Strings = signature,welcome,ball`

Void.header.n
[where n is >0]
	Specify the number of header lines to include on void receipts (maximum of 3 supported lines).	None	`void.header.1 = **********`
			`void.header.2 = ** VOID **`
			`void.header.3 = **********`

Void.footer.n
[where n is >0]
	Specify the number of footer lines to include on void receipts.	None	`void.footer.1 = **********`
			`void.footer.2 = ** VOID **`
			`void.footer.3 = **********`

electronic.signature	Enable the extracting of electronic signature from the device for display/approval on the POS.	false	`Electronic.signature=true`
electronic.signature.timeout	Sets the time out for “Amount of time to wait after a user completes signing”.	25	`electronic.signature.timeout = 60`
Socket.file.encoding	Overrides what character set to be used while reading the response on the socket from the Tender Retail client. The default character set on a windows jvm is cp1252, if any other charset is specified by the jvm (-Dfile.encoding) the socket.file.encoding below will automatically be set to ISO-8859-1. Setting the character set below will override any defaults.		`socket.file.encoding=cp1252`
perform.card.range.lookup	If true, EFTLink will use its mapping file CardRange.xml to determine the card scheme name.	false	`perform.card.range.lookup=false`
suppress.additional.message.prompt	If true, suppresses the message prompt on a failed transaction.	false	`suppress.additional.message.prompt = false`
exclude.additional.message.prompt.by.response.code	Comma delimited exception list if suppress.additional.message.prompt is true.		`exclude.additional.message.prompt.by.response.code =`

Table 9-4 (Cont.) Tender Retail - Secondary Settings

Setting	Description	Default	Example
tr.fields.used.for.wallet.trans.ref	Comma delimited list of tender retail fields that map to the CardServiceResponse’s AcquirerTransactionReference field for wallet transactions.	REF	`tr.fields.used.for.wallet.trans.ref = REF`
tr.fields.used.for.gc.trans.ref	Comma delimited list of tender retail fields that map to the CardServiceResponse’s AcquirerTransactionReference field for gift card transactions.	INV,REF	`tr.fields.used.for.gc.trans.ref = INV,REF`
tr.fields.used.for.emv.trans.ref	Comma delimited list of tender retail fields that map to the CardServiceResponse’s AcquirerTransactionReference field for EMV transactions.	INV,REF	`tr.fields.used.for.emv.trans.ref = INV,REF`
suppress.additional.message.prompt.for.gift.card	If true, suppresses the message prompt on a failed gift card transaction.	false	`suppress.additional.message.prompt.for.gift.card = false`
exclude.additional.message.prompt.for.gift.card.by.response.code	Comma delimited exception list if suppress.additional.message.prompt.for.gift.card is true.		`exclude.additional.message.prompt.for.gift.card.by.response.code =`
add.response.properties.to.misc.data	Specify tender retail response property codes (comma delimited) to be sent through miscellaneous data within the card service response.	SEQ,ISO,FBK,DSP	`add.response.properties.to.misc.data = SEQ,ISO,FBK,DSP`
manual.auth.input.type.alphanumeric	Sets the input validation to alphanumeric for entering the authorization code.	false	`manual.auth.input.type.alphanumeric = false`
manual.auth.minimum.length	Sets the minimum length for an authorization code.	6	`manual.auth.minimum.length = 6`
manual.auth.maximum.length	Sets the maximum length for an authorization code.	6	`manual.auth.maximum.length = 6`
manual.auth.retrys	Sets the number of retry attempts for entering an authorization code.	3	`manual.auth.retrys = 3`
swipe.fallback	Turns on/off the ability to prompt for manual keyed entry.	false	`swipe.fallback = false`
svc.card.type	Sets the gift card provider. Valid values are SVS or Givex.	svs	`svc.card.type = svs`
svs.pin.entry	Turn on/off the prompting for pin for SVS Gift Cards.	false	`svs.pin.entry = false`
svs.pin.entry.on.ped	Turn on/off the prompting for pin capture on the ped. if false the pin capture is done on the POS.	false	`svs.pin.entry.on.ped = false`
svs.pin.entry.types	Determines which tender types require pin capture.	83,A3,D3	`svs.pin.entry.types = 83,A3,D3`

Table 9-4 (Cont.) Tender Retail - Secondary Settings

Setting	Description	Default	Example
svs.pin.minimum.length	Sets the minimum length for a pin.	4	`svs.pin.minimum.length = 4`
svs.pin.maximum.length	Sets the maximum length for a pin.	4	`svs.pin.maximum.length = 4`
svs.pin.retrys	Sets the number of retry attempts	3	`svs.pin.retrys = 3`
givex.user.id	User id provided by Givex.		`givex.user.id = [encrypted user id string]`
givex.user.id.iv	Encrypted password initialization vector.		`givex.user.id.iv = [encrypted user id iv string]`
givex.password	Password provided by Givex.		`givex.password = [encrypted password string]`
givex.password.iv	Encrypted password initialization vector.		`givex.password.iv = [encrypted password iv string]`
givex.allow.partial.tender	Specify whether to allow partial tendering of gift cards.	true	`givex.allow.partial.tender = true`
givex.pin.entry	Turn on/off the prompting for security pin for Givex Cards.	false	`givex.pin.entry= false`
givex.pin.entry.types	Sets which tender types require security pin capture.	73,83	`givex.pin.entry.types = 73,83`
givex.pin.minimum.length	Sets the minimum required length for a Givex security pin.	6	`givex.pin.minimum.length = 6`
givex.pin.maximum.length	Sets the maximum required length for a Givex security pin.	6	`givex.pin.maximum.length = 6`
givex.pin.retrys	Sets the maximum number of retries allowed.	3	`givex.pin.retrys = 3`
crypto.keygenType	Sets keygen algorithm type.	AES	`crypto.keygenType = AES`
crypto.cipherType	Sets cipher algorithm type.	AES/CBC/PKCS5Padding	`crypto.cipherType = AES/CBC/PKCS5Padding`
crypto.keySize	Sets size of the key store.	128	`crypto.keySize = 128`
crypto.iterations	Sets number of iterations.	100000	`crypto.iterations = 100000`
crypto.factoryinstance	Sets crypto factory instance for keystore password encryption.	PBKDF2WithHmacSHA512	`crypto.factoryinstance = PBKDF2WithHmacSHA512`
crypto.secretkeyspec	Sets crypto secret key spec for keystore password encryption.	AES	`crypto.secretkeyspec = AES`

Table 9-4 (Cont.) Tender Retail - Secondary Settings

Setting	Description	Default	Example
crypto.keystoretype	Sets keystore type.	JKS	crypto.keystoretype = JKS
crypto.digest	Sets digest for keystore password.	SHA-512	`crypto.digest = SHA-512`
crypto.hashbyteSize	Sets hash byte size for keystore password.	384	`crypto.hashbyteSize = 384`

AdminMenu0.X	Specifies the ability to customize the Admin Menu. Replace X with a value between 0 - 4.	AdminMenu0.1 = TXT_DAY_END, Day End
		AdminMenu0.2 = TXT_PINPAD_INITIALIZATION, PINPad Initialization
		AdminMenu0.3 = TXT_SIGNATURE_CAPTURE, Signature Capture
		AdminMenu0.4 = TXT_CANCEL, Cancel
			`AdminMenu0.1 = TXT_DAY_END, Day End`
			`AdminMenu0.2 = TXT_CANCEL, Cancel`

line.display.enabled	Enables line item display request to the PED.	false	`line.display.enabled = true`
wait.for.line.display	If set to true, the core will pause upon receiving a payment request if there are still sale line item requests being processed, until all of these requests have been completed.	false	`wait.for.line.display = true`
wait.for.line.display.timeout	The amount of time to wait for a sale line item request to be completed is dependent on the wait.for.line.display setting. A value of 0 means the wait is indefinite.	0	`wait.for.line.display.timeout`
max.line.items.processed	The maximum number of line items to be sent upon the request. If the number of line items is greater than the specified value, then all except the most recent line items are stripped from the request.	9	`max.line.items.processed = 9`
cnp.request.type	Sets the tender retail transaction type for card not present transaction. Possible values known are EU, E0, or 00 (however any value will be sent so long as the value length equals two).	EU	`cnp.request.type = EU`

Administration Functions

The terminal has some administration/maintenance functions. These can only be invoked from a dedicated EFT Maintenance menu button.

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

Table 9-5 Tender Retail - Administration Functions

Function	Description
Day End	Print a day report and close the current day. Manual alternative to automated reconciliation with closure.
PINPad Initialization	Sends a PINPad Initialization request.
Signature Capture	Sends request to test the signature capture functionality on the device.

Supported Functions

Below is a list of supported functionalities of the interface to Tender Retail.

Table 9-6 Tender Retail - Supported Functions

Function	Description
Sale State Notifications	Sends line items through to the device so the customer display can be updated in line with the POS.
Payment	Sends payment request to Tender Retail's application. The client will return a response message with formatted receipt strings for customer and/or merchant receipts. In an event of referral or communication failure where authorization cannot be obtained online then a prompt for authorization code will appear; authorization code must be obtained via telephone. Appropriate receipts will be printed at the end of transaction.
Reversal	Sends reversal requests to Tender Retail's application. The client 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 requests to Tender Retail's application. The client will refund a transaction with specified amount.
Tokenized Refund	Sends refund requests to Tender Retail's application. The client software will refund a transaction with specified token id.
SVC Redeem	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 Redeem Reversal	Sends a gift or merchandise credit card payment reversal request to the terminal.

Table 9-6 (Cont.) Tender Retail - Supported Functions

Function	Description

SVC Activate	Sends a gift or merchandise credit card Issuance request to Tender Retail's MCM software.EFTLink by default for SVS Gift card sends an Issuance message for RequestType=CardActivate. To process an activate message it will be necessary to send through via a card service request a MiscellaneousData element GiftCardType=Activate.For example,
	`<?xml version="1.0" encoding="UTF-8"?>`
	`<CardServiceRequest RequestType="CardActivate" ApplicationSender="POSSIM" WorkstationID="1" RequestID="1" RequestSubType="PresetValue">`
	`<POSdata LanguageCode="en">`
	`<POSTimeStamp>2019-05-13T16:50:03</POSTimeStamp>`
	`<TransactionNumber>479</TransactionNumber>`
	`<TransactionDay>2019-05-13</TransactionDay>`
	`</POSdata>`
	`<TotalAmount Currency="GBP">50.00</TotalAmount>`
	`<SaleItem ItemID="_1">`
	`<ProductCode>25</ProductCode>`
	`<Amount>50.00</Amount>`
	`<UnitMeasure>EA</UnitMeasure>`
	`<UnitPrice>50.00</UnitPrice>`
	`<Quantity>1</Quantity>`
	`<TaxCode>A</TaxCode>`
	`<TaxRate>0.00</TaxRate>`
	`<AdditionalProductCode>12345678</AdditionalProductCode>`
	`<AdditionalProductInfo>SVC Card</AdditionalProductInfo>`
	`</SaleItem>`
	`<MiscellaneousData>GiftCardType=Activate</MiscellaneousData>`
	`</CardServiceRequest>`

SVC Activate Reversal	Sends a gift or merchandise credit card activation request to Tender Retail's MCM software.
SVC Add Value	Sends a gift or merchandise credit card add value request to Tender Retail's MCM software. The card must be first activated before using this function.
SVC Add Reversal	Sends a gift or merchandise credit card payment request to Tender Retail's MCM software. Optionally a void will be activated in the core to avoid presenting the card for the reversal.
SVC Balance Enquiry	Sends a gift or merchandise credit card balance enquiry request to Tender Retail's MCM software.
SVC Unload (Cash Out)	Sends a gift or merchandise credit card cash out request to the Tender Retail's MCM software. 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.

Table 9-6 (Cont.) Tender Retail - Supported Functions

Function		Description

Read Non-PCI Card		EFTLink sends a card swipe request to receive data for non-pci cards. The full pan is returned in clear text, unencrypted and without tokenization.
		PCI cards will return a blank PAN.
		As EFTLink doesn't currently have a direct mapping for requesting for a Token. As a work around to request for a token then send RequestType=CardSwipe within the CardServiceRequest. Include the string "GetToken" within the Card Value element and map the Card Type to either of the following value (taken from the tender retail technical specification document)
		1 - Master Card
		2 - Visa
		3 - American Express
		4 - Discover
		6 - Enroute/Diners
		7 - JCB
		< - Debit
		0 - Unknown
		For example,
		`<?xml version="1.0" encoding="UTF-8"?>`
		`<CardServiceRequest RequestType="CardSwipe" ApplicationSender="MICROS" WorkstationID="1" RequestID="1">`
		`<POSdata LanguageCode="en">`
		`<POSTimeStamp>2019-05-10T16:30:20</POSTimeStamp>`
		`</POSdata>`
		`<CardValue CardType="0">`
		`<Token>GetToken</Token>`
		`</CardValue>`
		`</CardServiceRequest>`

Ewallet (Flow 3) Citcon / Alipay	Payment	Sends payment request to Tender Retail's application. The client will return a response message with formatted receipt strings for customer and/or merchant receipts. Appropriate receipts will be printed at the end of transaction.
	Refund	Sends refund requests to Tender Retail's application. The client will refund a transaction with specified amount.
	Cancel	Sends void/correction request for Payments or Refunds. Please note this is only supported for MCM direct to Alipay. MCM to Citon to Alipay is not supported.

10

Verifone Ocius Sentinel

This document covers EFTLink Integration with Ocius Sentinel Payment Systems. It should be read in conjunction with the *Oracle Retail EFTLink Framework Installation and Configuration 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 Verifone Ocius Sentinel interface requires a minimum EFTLink version of 20.0.

System Architecture

EFTLink connects to the Ocius Sentinel application using a proprietary socket protocol. Normally the Ocius Sentinel application, which is configured to run in a screenless state, 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 and Configuration 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.

Table 10-1 Verifone Ocius Sentinel - Key Settings

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=[encrypted user ID]`
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=[encrypted user pin]`
manager.pin	The manager 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.	`manager.pin[encrypted manager pin]`
user.id.iv	User ID initialization vector, as provided when encrypting User ID.	`user.id.iv=[encrypted user ID.iv]`
user.pin.iv	User pin initialization vector, as provided when encrypting User pin.	`user.pin.iv=[encrypted user pin.iv]`

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.

Table 10-2 Verifone Ocius Sentinel - Optional Configuration Settings

Setting	Description	Default
ip.address	The IP address of the Ocius Sentinel software.	
ip.port	The IP port of the terminal.	`25000`
progress.ip.port	The progress IP port of the terminal.	25001
ocius.payment.application.on.device	Identifies whether the providers client is running on the Pin Pad	false
crypto.keygenType	Sets keygen algorithm type.	AES
crypto.cipherType	Sets cipher algorithm type.	AES/CBC/PKCS5Padding
crypto.keySize	Sets size of the key store.	128
crypto.iterations	Sets number of iterations.	100000
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
account.id.iv	When used with encrypted account.id, specifies Account ID initialization vector, as provided when encrypting account.id.	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.	NA
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.	

Table 10-2 (Cont.) Verifone Ocius Sentinel - Optional Configuration Settings

Setting	Description	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.	Report.txt
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.	Please Tear Merchant Receipt
tear.customer.receipt.text	The text to be displayed at the POS when prompting the operator to remove the customer receipt from the printer.	Please Tear Customer Receipt
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`
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.	`true`
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.	NA

Table 10-2 (Cont.) Verifone Ocius Sentinel - Optional Configuration Settings

Setting	Description	Default
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).	NA
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.	NA
encrypted.passwords	user.id, user.pin, account.id and transax.account.id must be encryped using the encryption utility. See Password Encryption.	NA
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/ wait.record.header.cnp	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/ wait.record.body.cmp	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/ wait.record.footer.cnp	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/ wait.record.timeout.cnp	This is the time in seconds for the PED to wait for the card details to be presented.	0 (no timeout)

wait.record.capture.method s/
wait.record.capture.methods.cn ps
	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`

Table 10-2 (Cont.) Verifone Ocius Sentinel - Optional Configuration Settings

Setting	Description	Default
wait.record.fallback.methods.cnps	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`
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. When used with encrypted transax.account.id, specifies Transax Account ID initialization vector, as provided when encrypting transaxaccount.id.	NA
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`

Table 10-2 (Cont.) Verifone Ocius Sentinel - Optional Configuration Settings

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.	NA
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.	Amount:
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.	false
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`

Table 10-2 (Cont.) Verifone Ocius Sentinel - Optional Configuration Settings

Setting	Description	Default
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`
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.	0
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 proximity. The delay is only applied if needed and the default interval is 1000ms.	`1000`
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.	`0`
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.	true
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`

Table 10-2 (Cont.) Verifone Ocius Sentinel - Optional Configuration Settings

Setting	Description	Default
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`
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.`

Table 10-2 (Cont.) Verifone Ocius Sentinel - Optional Configuration Settings

Setting	Description	Default
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.	NA
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`

Table 10-2 (Cont.) Verifone Ocius Sentinel - Optional Configuration Settings

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`
send.ocius.update.to.pos	Whether to display the status update from Ocius to the POS or not.	`false`
Partial.Payment.Accept.Action.Id	The action ID to be sent back to Ocius for accepting a partial payment. For older Ocius API's this value may need to be changed to 65.	2
Partial.Payment.Reject.Action.Id	The action ID to be sent back to Ocius for rejecting a partial payment. For older Ocius API's this value may need to be changed to 66.	12

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 several 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 can supply raw receipt data in XML form rather than as formatted text. The directory path where Ocius writes these files should be setup in Ocius and points to the working directory of EFTLink, example C:\eftlink. 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 center 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 example of a template file:

Example 10-1   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 center 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 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:

Example 10-2   ocius_receipt.properties

```
template_customer_keyed_swiped.txt=<ReceiptType>Customer<CaptureMethod>SWIPED
template_merchant_keyed_swiped.txt=<ReceiptType>Merchant<CaptureMethod>SWIPED
template_signature.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:

template_customer_contactless.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:

Open a terminal window and change directory to where the script file resides.

For Windows: Type `encrypt-ocius-sentinel.bat -k [<keystore name> <properties file>]`.

For example, `encrypt-ocius-sentinel.bat -k`

For Linux: Type `encrypt-ocius-sentinel.sh -k [<keystore name> <properties file>]`.

For example, `sudo ./encrypt-ocius-sentinel.sh -k`

Keystore file will be generated and stored in the data directory. If the keystore name and the properties file names are not specified, then the default values (`ocius-sentinel.keystore`, `ocius-sentinel.properties`) will be used.

> **Note**
>
> When creating the data directory for the keystore, access to the folder is restricted automatically. Therefore, the same user account as will be used to run the application should be used during this process. Use of the administrator account at this point may lead to the folder being inaccessible to the EFTLink application at runtime.

Encryption

The following settings within the `ocius.properties` file need to be encrypted:

- `user.id`
- `user.pin`
- `account.id`
- `transax.account.id`
- `manager.pin`

To achieve this, the following steps must be followed:

To encrypt a value: Open a terminal window and change directory to where the script file resides.

For Windows: Type `encrypt-ocius-sentinel.bat -e <keystore name> <properties file> <value>`.

For example, `encrypt-ocius-sentinel.bat -e`

***For Linux**: Type `encrypt–ocius-sentinel.sh -e [<keystore name> <properties file> <value>]`.

For example, `sudo ./encrypt-ocius-sentinel.sh –e`

The user will be presented with prompts to provide the value(s) which are to be encrypted. Once entered the corresponding properties keys will be automatically updated with the encrypted values.

> **Note**
>
> If the keystore name, properties file and unencrypted text is included as arguments then the encrypted value and initialization vector will be outputted to the console which must be copied and pasted to relevant property key within `ocius-sentinel.properties`. This process then needs to be repeated for every value that is required to be encrypted.

To re-encrypt; Open a terminal window and change directory to where the script file resides.

For Windows: Type `encrypt-ocius-sentinel.bat –r [<keystore name> <properties file> <keygen type> <cipher type> <key size> <iterations>]`.

For example, `encrypt-ocius-sentinel.bat –r`

***For Linux**: Type `encrypt-ocius-sentinel.sh –r [<keystore name> <properties file> <keygen type> <cipher type> <key size> <iterations>]`.

For example, `sudo ./encrypt-ocius-sentinel.sh -r`

The key values to be re-encrypted will be taken from the properties file, re-encrypted and the properties file will be automatically updated.

* You may be required to give script file(s) execution rights for example, `chmod +x <PathToFile>`

> **Note**
>
> 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/.

Administration Functions

The terminal has some administration/maintenance functions. These can only be invoked from a dedicated EFT Maintenance menu button.

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

Table 10-3 Ocius Sentinel - Administration Functions

Function	Description
Customer receipt reprint	Prints the last customer receipt.

Supported Functions

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

Table 10-4 Ocius Sentinel- Supported Functions

Function	Description
Logon	Sends a PED Logon request to the Ocius Sentinel client.
Logoff	Sends a PED Logoff request to the Ocius Sentinel client.
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.
Refund	Sends refund request to the terminal. This will refund a transaction with specified amount.
Card Read	EFTLink sends a card swipe request to receive data for non-pci cards. The full pan is returned in clear text, unencrypted and without tokenization. PCI cards will return a blank PAN.
X Reports (reconciliation without closure)	Print a report showing the sales, returns, voids, and other register activity that occurred on the register from the beginning of a register shift until the present moment.
Z reports (reconciliation with closure)	Print a day report and close the current day. Manual alternative to automated reconciliation with closure.
SVC Payment (VX820 only)	EFTLink sends a gift or merchandise credit card payment request to the OPI EPS. 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. The Transaction Inquiry scenario outlined in the Payment/Payment with Loyalty section also applies to this transaction type.
SVC Activate (VX820 only)	EFTLink sends a gift or merchandise credit card activation request to the OPI EPS. The Transaction Inquiry scenario outlined in the Payment/Payment with Loyalty section also applies to this transaction type.
SVC Add Value (VX820 only)	EFTLink sends a gift or merchandise credit card add value request to the OPI EPS. This will only add value to an account that has been activated. The Transaction Inquiry scenario outlined in the Payment/Payment with Loyalty section also applies to this transaction type.
SVC Balance Enquiry (VX820 only)	EFTLink sends a gift or merchandise credit card balance enquiry request to the OPI EPS.

Table 10-4 (Cont.) Ocius Sentinel- Supported Functions

Function	Description
SVC Unload (VX820 only)	EFTLink sends a gift or merchandise credit card cash out request to the OPI EPS. 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. The Transaction Inquiry scenario outlined in the Payment/Payment with Loyalty section also applies to this transaction type.

11 Verifone Point (US)

This chapter covers EFTLink integration with Verifone Point.

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 and Configuration 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 20.0.

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`.

Language

The translation files for this core should not require alteration, but if necessary then this can be accomplished by amending the relevant `Lang<CC>_<Core>.properties` within the base eftlink folder.

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

```
EftlinkConfig.properties
```

```
DisplayLanguage = EN
```

Supported country codes are CN, DE, EN, ES, FR, IT, JP, NL, PT, RU and SV.

Core Classname

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

```
EPSCore0 = manito.eft.pointus.PointUSCore
```

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.

Table 11-1 Verifone Point (US) - Key Settings

Setting	Description	Default	Example
TerminalIP	IP of Mx915/M400/ M400/P200/E285 terminal.	NA	`TerminalIP =`

Secondary Settings

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

Table 11-2 Verifone Point (US) - Secondary Settings

Setting	Description	Default	Example
TerminalPort	Port number.	5015	`TerminalPort = 5015`
MaintenanceTimeout	Timeout on maintenance menu selection. Timeout is specified in seconds.	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	The Connection timeout. Timeout is specified in seconds.	10	`ConnectionTimeout = 10`

Table 11-2 (Cont.) Verifone Point (US) - Secondary Settings

Setting	Description	Default	Example
ConnectionIssueRetryAttempts	When exceeding the ResponseTimeout. How many attempts should the associate be given to re-establish communications before returning failure.	1	`ConnectionIssueRetryAttempts = 3`
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 authorization 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`
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.	false	`ShowEMVTagsOnVoucher = false`
EmbeddedPrintFilter_<n>	Print filters to allow voucher to be trimmed when embedded in the POS receipt for example, to remove header/footer lines starting with text in this list will be skipped. Maximum filter is 99.	null	`EmbeddedPrintFilter_1 =<blank>`

Table 11-2   (Cont.) Verifone Point (US) - Secondary Settings

Setting	Description	Default	Example
EnableTrack2ForCardSwipe	Specifies whether track2 is to be returned for certain card types. Also requires changes to range xml to prevent masking of numbers.	false	`EnableTrack2ForCardSwipe=false`
SwipeFallbackToKeyed	Specifies whether failure of card swipe during payment will result in fallback to keyed entry on the PED.	false	`SwipeFallbackToKeyed=false`
MaxRetrySendDeviceCommand	Number of retry to send the device command in a situation where the device is busy during the first attempt.	2	`MaxRetrySendDeviceCommand=2`
RetrySendDeviceCommandDelay	The delay in milliseconds in every retry of sending the device command.	3000	`RetrySendDeviceCommandDelay=3000`
DisplayMessageTimeout	The number of seconds to display the message in the PIN pad for the device command, DISPLAY_MESSAGE. The timeout value specified can be overriden within the DeviceRequest <?xml version='1.0' encoding='UTF-8'?> <DeviceRequest RequestType="Output" ApplicationSender="MICROS" WorkstationID="1" RequestID="5"> <Output OutDeviceTarget="PinPad" **TimeOut="10"**> <TextLine>Black Friday Sale</TextLine> </Output> </DeviceRequest>	10	`DisplayMessageTimeout = 30`
DisplayQRCodeDoneButtonLabel	The label of the button in the command to display QR code in the terminal.	Done	`DisplayQRCodeDoneButtonLabel = Done`
MaxNumberOfSurveyTextLine	Maximum number of lines of text in the survey command.	5	`MaxNumberOfSurveyTextLine = 5`
MaxSurveyLineMessageLength	Maximum number of characters for each line in the survey message.	50	`MaxSurveyLineMesageLength = 50`
MaxDonationLineMessageLength	Maximum number of characters for each line in the donation message.	50	`MaxDonationLineMessageLength = 50`
MaxNumberOfDonationTextLine	Maximum number of lines of text in the donation command.	5	`MaxNumberOfDonationTextLine = 5`
MaxNumberOfDonationChoices	Maximum number of choices for the donation command.	5	`MaxNumberOfDonationChoices = 5`
MaxNumberOfCustomerButtonChoices	Maximum number of choices in the customer buttons command.	6	`MaxNumberOfCustomerButtonChoices = 6`
MaxCustomerButtonsLineMessageLength	Maximum number of characters for each line in the customer buttons message.	45	`MaxCustomerButtonsLineMessageLength = 45`

Table 11-2 (Cont.) Verifone Point (US) - Secondary Settings

Setting	Description	Default	Example
MaxNumberOfCustomerButtonsTextLine	Maximum number of lines of the message for the customer buttons command.	5	`MaxNumberOfCustomerButtonsTextLine = 5`
MaxQRCodeDataLength	Maximum number of characters for the QR code data.	200	`MaxQRCodeDataLength = 200`
MaxQRCodeMessageLengthWithTransaction	The maximum number of characters of the message in the display QR code command in an active transaction.	72	`MaxQRCodeMessageLengthWithTransaction = 72`
MaxQRCodeMessageLengthWithOutTransaction	The maximum number of characters of the message in the display QR code command when there is no active transaction.	150	`MaxQRCodeMessageLengthWithOutTransaction = 150`
MaxDisplayTextLength	The maximum number of characters of the message to display for the DISPLAY_MESSAGE command.	150	`MaxDisplayTextLength = 150`
TokenExpiryDate	Enables the core to pass the card's expiration date to the POS. As such, this will be passed to PointUS for verified refund.	true	`TokenExpiryDate = true`
TenderLineItemIdAddend	The number that gets added to the tender line sequence to become the unique LINE_ITEM_ID.	1000	`TenderLineItemIdAddend = 1000`
MCLabelCounterMaxValue	The maximum value of the counter in the PED for a given Mac label.	4294967295	`MCLabelCounterMaxValue = 4294967295`
WaitTimeForLineDisplayToFinish	The wait time interval in milliseconds given to the line item manager thread to finish processing before the core sends the payment request to Point.	5000	`WaitTimeForLineDisplayToFinish = 5000`
SignatureLineIndicator	The indicator used as the signature line in the merchant receipt where customer will sign in case the device used does not support electronic signature.	x	`SignatureLineIndicator = x`
MerchantReceiptIndicator	An indicator for the merchant copy of the receipt.	MERCHANT COPY	`MerchantReceiptIndicator = MERCHANT COPY`
CustomerReceiptIndicator	An indicator for the customer copy of the receipt.	CUSTOMER COPY	`CustomerReceiptIndicator = CUSTOMER COPY`
PrintMerchantReceipt	Whether to print the merchant receipt or not.	true	`PrintMerchantReceipt = true`
MerchantID	Merchant ID required for "Setup Device Parameters" function in EFTLink Admin functions in Xstore back office.		`MerchantID= 12345678`
TerminalID	Terminal ID required for required for "Setup Device Parameters" function in EFTLink Admin functions in Xstore back office.		`TerminalID=002`

Table 11-2 (Cont.) Verifone Point (US) - Secondary Settings

Setting	Description	Default	Example
Lane	Lane required for "Setup Device Parameters" function in EFTLink Admin functions in Xstore back office.		`Lane=002`
HostIndicator	Host Indicator required for "Setup Device Parameters" function in EFTLink Admin functions in Xstore back office.		`HostIndicator=VNTV`
RestrictToDebitCredit	Restrict payment capture command to credit debit payment types.	false	`RestrictToDebitCredit=true`
RemoveOfferLineItem	Specify whether to include the OFFER line item when removing the parent or main item. Enable this for Engage devices such as the M400 that will not remove the child OFFER line automatically when the parent item is removed. Consult Verifone to determine which devices behave this way.	false	`RemoveOfferLineItem = true`
IgnoreSignatureCapture	Determine whether to ignore the capture of electronic signature. This is applicable for Engage devices such as the E285 which does not support the 3BA format and signature capture cannot be disabled on the device.	false	`IgnoreSignatureCapture=true`
EnforceGiftTenderType	Specifies whether a giftcard type action will enforce a GIFT tendertype on PED device.	false	`EnforceGiftTenderType = false`
ReceiptAPMType	Whether the additional receipt data should be used. 0 — do not print additional receipt data. 1 — only print additional receipt data with APM transactions. 2 — append additional receipt data to standard receipt.	0	`ReceiptAPMType = 2`
AddResponseFieldsToMiscData	The PointUS fields to be returned within Miscellaneous Data.		`AddResponseFieldsToMiscData = BANK_USERDATA`
RestartSessionOnVoids	Performs a session stop and start before sending the void/reversal request.	false	`RestartSessionOnVoids = true`
AddCtroutdToCreditRefunds	If true and the Payment Type is Credit, then add CTROUTD to the Credit/ Refund request.	false	`AddCtroutdToCreditRefunds = true`
IncludeBankUserDataWithCardToken	If true and the BANK_USERDATA and CARD_TOKEN is included in the Payment response then store them together within the CardServiceResponse Card Object.	true	`IncludeBankUserDataWithCardToken = true`

Table 11-2 (Cont.) Verifone Point (US) - Secondary Settings

Setting	Description	Default	Example
VoiceAuthorizationResultCodes	A comma-separated list of RESULT_CODE indicators to trigger for voice authorization.	59024,59074	`VoiceAuthorizationResultCodes = 59024,59074`
FullPacketEncryption	Enables Full Packet Encryption where all messages will be encrypted with the AES 128. Note: REGISTER_ENCRYPTION and STATUS requests are not encrypted.	false	`FullPacketEncryption = true`
PayloadKeyPairAlgorithm	The name of the secret-key algorithm used for encrypting/decrypting the payload. This setting requires FullPacketEncryption to be set to true. Must match the PointUS Algorithm.	AES	`PayloadKeyPairAlgorithm = AES`
PayloadKeyPairCipher	The cipher initialization used for encrypting/decrypting the payload. This setting requires FullPacketEncryption to be set to true. Must match the PointUS Algorithm.	false	`PayloadKeyPairCipher = AES/CBC/PKCS5PADDING`
crypto.keygenType	Sets keygen algorithm type.	AES	`crypto.keygenType = AES`
crypto.cipherType	Sets cipher algorithm type.	AES/GCM/PKCS5Padding	`crypto.cipherType = AES/CBC/PKCS5Padding`
crypto.keySize	Sets size of the keystore. Note: When the 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/	128	`crypto.keySize = 128`
crypto.iterations	Sets number of iterations.	100000	`crypto.iterations = 10000`
crypto.factoryinstance	Sets crypto factory instance for keystore password encryption.	PBKDF2WithHmacSHA512	`crypto.factoryinstance = PBKDF2WithHmacSHA512`
crypto.secretkeyspec	Sets crypto secret key spec for keystore password encryption.	AES	`crypto.secretkeyspec = AES`
crypto.keystoretype	Sets keystore type.	JKS	`crypto.keystoretype = JKS`
crypto.digest	Sets digest for keystore password.	SHA-512	`crypto.digest = SHA-512`

Table 11-2 (Cont.) Verifone Point (US) - Secondary Settings

Setting	Description	Default	Example
crypto.hashbyteSize	Sets hash byte size for keystore password.	384	`crypto.hashbyteSize = 384`

Administration Functions

The terminal has some administration/maintenance functions. These can only be invoked from a dedicated EFT Maintenance menu button.

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

Table 11-3 Verifone Point (US) - Administration 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.
Un-registration	This operation removes a pairing.
Test MAC	This operation tests that the terminal is accessible and that a pairing 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 PointUS. Many functionalities are provided by PointUS, such as Loyalty, Cashback and so on. (Please refer to interface specification for details) but are not implemented because of the business requirement.

Table 11-4 Verifone Point (US) - Supported Functions

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.

Table 11-4 (Cont.) Verifone Point (US) - Supported Functions

Function	Description
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.
Custom form for displaying a message	Sends a request to the terminal that displays the message text passed by the POS. The core sends a success or a failure flag back to the POS.
Custom form for customer question/verification	Sends a request to the terminal with a question/verification message. The customer selects either the Yes or No button. The core sends 'Y' or 'N' as part of the response to the POS.
Custom form for capturing phone number	Sends a request to the terminal triggering a phone number capture. The customer keys in their phone number and hit submit. The core sends the captured phone number to the POS.
Custom form for signature capture	Sends a request to the terminal to capture signature. The customer signs and hit accept. The core sends the decoded signature to the POS.
Custom form for capturing email address	Sends a request to the terminal triggering an email address capture. The customer keys in their email address in the virtual keyboard and selects the Enter key. The core sends the captured email address to the POS.
Custom form for customer survey	Sends a request to the terminal to present a survey. The number of choices could be from 1 to 5 or 1 to 10.
Custom form for charity donation	Sends a request to the terminal asking if the customer wants to donate to a charity. The customer selects a button for their choice. The core sends the selected amount as presented in the terminal back to the POS.

Table 11-4 (Cont.) Verifone Point (US) - Supported Functions

Function	Description
Custom form for customer buttons selection	Sends a request to the terminal to present a list of regular and bigger buttons as choices. The maximum number of choices is 5. The label of each button is set by the POS. The sixth label is used as cancel. The customer selects a button. The core sends the label of the selected button back to the POS.
Custom form for QR code display	Sends a request to the terminal to display QR code. The terminal generates a QR image corresponding to payload data sent from the POS and displays the image on screen with appropriate text and/or button label.
Custom form for cancelling QR code display	Sends a request to the terminal to clear the QR code image in the PED. The PED goes back to the previous screen afterwards.
E-Wallet Payments	Supports Alipay/Klarna/WeChat/PayPal/Venmo Flow 1 - Customer initiated transaction via E-Wallet button press on the PED. EFTLink sends a capture request. The customer selects the button to pay via their E-Wallet (as opposed to the usual chip and pin, swipe and other card payment methods) on the PED. The provider returns a response containing the E-Wallet data. EFTLink feeds this data back to the POS to complete the transaction. Flow 2 - Cashier initiated transaction via E-Wallet tenders on the POS. POS tenders to pay the transaction via E-Wallet tender. EFTLink sends a sale/purchase message to the Provider, specifying that the PaymentMethod is E-Wallet. The OPI EPS displays a QR code which the customer scans with their E-Wallet device (typically a mobile phone). The transaction is confirmed on the PED and the WalletAuthorizationData is returned via EFTLink to the POS to complete the transaction.
E-Wallet Refunds	Supports Alipay/Klarna/WeChat/PayPal/Venmo Sends a refund request which includes the token to the provider.
Full Packet Encryption	Supports encryption of request/response messages being sent/received between the EFTLink and PointUS. StatusCommand and the RegisterEncryptionCommand are not subject to encryption.
Dynamic Currency Conversion (DCC)	Supports DCC on Payment request.

12

WorldPay

This chapter covers EFTLink Integration with WorldPay Payment Systems. It should be read in conjunction with the *Oracle Retail EFTLink Framework Installation and Configuration 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 WorldPay interface requires a minimum EFTLink version of 20.0.

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 *Oracle Retail EFTLink Framework Installation and Configuration 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.bat 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. All available settings are listed below.

> **Note**
>
> The software was previously called YesPay.

Table 12-1 WorldPay - Configuration Settings

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`

Table 12-1 (Cont.) WorldPay - Configuration Settings

Setting	Description	Default	Example
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 \YESEFT\properties folder.	en_GB	`language = en_GB`
signature.reprint.prompt	The text to display when asking if a signature receipt should be reprinted. This message will only be shown if the operator answers no, when asked to confirm that the signature is acceptable 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?`

Table 12-1 (Cont.) WorldPay - Configuration Settings

Setting	Description	Default	Example
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, for example "GBP" and so on.	£	`currency.symbol = £`
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 **`
address.prompt	This is the text to display for the address prompt.	Please enter first address line	`address.prompt=Please enter first address line`
postcode.prompt	This is the text to display for the postcode prompt	Please enter postcode	`postcode.prompt=Please enter postcode`
disable.cashback.prompt	If true, then cashback will not be offered by the Point-of-Sale terminal.	false	`disable.cashback.prompt = true`

Supported Functions

Below is a list of supported functionalities of the interface to WorldPay

Table 12-2 WorldPay - Supported Functions

<table>
<tr><th>Function</th><th>Description</th></tr>
<tr><td>Payment</td><td>Sends payment request to WorldPay application. The client will return a response message with formatted receipt strings for customer and/or merchant receipts.
Appropriate receipts will be printed at the end of transaction.</td></tr>
<tr><td>Cashback</td><td>If the WorldPay client (IPC) is enabled for cashback, then EFTLink will prompt the associate if cashback is required. EFTLink can suppress the cashback request by enabling the property "disable.cashback.prompt" in the core properties file.
The offering of cashback can also be suppressed via the CardServiceRequest.
If suppress.cashback=true is added to the MiscellaneousData element then cashback will be suppressed.
Example:<pre>
<?xml version="1.0" encoding="UTF-8"?>
<CardServiceRequest RequestType="CardPayment"
ApplicationSender="XSTORE" WorkstationID="1" RequestID="3">
 <POSdata LanguageCode="eng">
  <POSTimeStamp>2020-01-14T11:33:39</POSTimeStamp>
  <TransactionNumber>39</TransactionNumber>
  <StoreID>101</StoreID>
 </POSdata>
    <MiscellaneousData>suppress.cashback=true</MiscellaneousData>
 <TotalAmount Currency="USD">10.00</TotalAmount>
 <SaleItem ItemID="_1">
 <ProductCode>0</ProductCode>
 <Department>NP</Department>
 <Amount OriginalAmount="10.00">10.00</Amount>
 <UnitPrice>10.00</UnitPrice>
 <Quantity>1</Quantity>
 <TaxCode>0</TaxCode>
 <TaxRate>0.00</TaxRate>
 <AdditionalProductCode>1026</AdditionalProductCode>
 <AdditionalProductInfo>Non Phys   Item</AdditionalProductInfo>
 </SaleItem>
 <PaymentProviderName />
</CardServiceRequest>
</pre></td></tr>
</table>

Table 12-2 (Cont.) WorldPay - Supported Functions

<table>
<tr><th>Function</th><th>Description</th></tr>
<tr><td>Reversal</td><td>Reversal requests require the card payment reference, PAN and card expiry date from the original transaction. 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.
<pre>
<?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>
</pre></td></tr>
<tr><td>Refund</td><td>Sends refund requests to the WorldPay application. The client will refund a transaction with specified amount.</td></tr>
</table>

Table 12-2 (Cont.) WorldPay - Supported Functions

Function	Description

Tokenized Refund	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>`

X reports (reconciliation)	Offers the ability to print a reconciliation report. This is an online function only. If the network is not available, then the transaction will be canceled, and no report data will be returned in response.
Z reports (reconciliation with closure)	Offer the ability to print a reconciliation with closure report. This is an online function only, If the network is not available, then the transaction will be canceled, and no report data will be returned in response.

Integration Notes

This section describes key points for the WorldPay integration.

WorldPay Configuration

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

Online/Offline Indication

In a card payment response, the miscellaneous data field will indicate whether the authorization 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 Device ID 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.

Signature Print Notification

If the core is configured to notify the POS of a signature print, then a device event will be generated as shown below. The POS should examine the Event Type 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 Event Type="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" />
```